

**STANDARDS PROJECT**

**Draft Standard for Information Technology —  
Portable Operating System Interface (POSIX)  
Part 2:  
Shell and Utilities — Amendment**

Sponsor  
**Portable Application Standards Committee**  
of the  
**IEEE Computer Society**

**Work Item Number: JTC 1 22.41**

**Abstract:** P1003.2b is part of the POSIX series of standards for applications and user interfaces to open systems. It consists of modifications and clarifications to ISO/IEC 9945-2: 1993 (IEEE Std 1003.2-1992), including support for symbolic links, a new archive/interchange format, and other modifications and clarifications prompted by ISO/IEC balloting.

**Keywords:** API, application portability, data processing, open systems, operating system, portable application, POSIX, shell and utilities, user portability

**P1003.2b / D12**  
**June 1999**

*This is an unapproved IEEE Standards Draft, subject to change. Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities. Permission is also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position.*

*Other entities seeking permission to reproduce this document for standardization or other activities, or to reproduce portions of this document for these or other uses, must contact the IEEE Standards Department for the appropriate license. Use of information contained in this unapproved draft is at your own risk.*

IEEE Standards Department  
Copyright and Permissions  
445 Hoes Lane, P.O. Box 1331  
Piscataway, NJ 08855-1331, USA  
+1 (732) 562-3800  
+1 (732) 562-1571 [FAX]  
*June 1999*

*SH XXXXX*

## 1 *Editor's Notes*

2 This section will not appear in the final document. It is used for editorial com-  
3 ments concerning this draft.

4 This is the second recirculation balloting draft of P1003.2b. Please see the ballot- C  
5 ing instructions in Annex I. See the Change History later in these notes for a C  
6 summary of the nontrivial changes from the last working group meeting. This C  
7 draft uses small numbers in the right margin in lieu of change bars. Diff marks C  
8 "c" denote changes from Draft 11 to Draft 12. Diff marks "B" denote changes from C  
9 Draft 10 to Draft 11. Editorial changes such as typos, grammatical errors or C  
10 changes, changes in cross references, and removal of editorial notes are not diff- B  
11 marked. Please note that it is not always feasible to get the diff marks exactly B  
12 right; they will sometimes start or end a line too soon. B

13 This draft attempts to fully document the authorization sources of all changes C  
14 being made to IEEE Std 1003.2-1992. Thus, all interpretation requests and inter- C  
15 national balloting comments resulting in changes are cited explicitly. However, C  
16 there is a large collection of changes related to the addition of symbolic link sup- C  
17 port that are not specifically cited; it was felt that these changes are so obvious in C  
18 identification that no specific citations were required. See the Introduction (page B  
19 v) for a list of authorized changes. B

20 This draft modifies IEEE Std 1003.2-1992, which is technically identical to  
21 ISO/IEC 9945-2: 1993. (However, note that there are very minor editorial and line  
22 number differences between these two documents.) You can purchase the stan-  
23 dard by contacting:

24 IEEE Publications  
25 P.O. Box 1331  
26 445 Hoes Lane  
27 Piscataway, NJ 08855-1331  
28 1 (800) 678-IEEE  
29 +1 (732) 562-3800 (outside US)

30 Since portions of this standard are meant to be modifications of the base POSIX.2  
31 standard, the draft headings have been set up to match the affected clauses and  
32 still go into the table of contents. Therefore, there are gaps in the clause numbers  
33 of some sections.

### 34 *POSIX.2b Change History* C

35  
36 This section is provided to track major changes between drafts.

37	Draft 12	[June 1999] Second IEEE recirculation draft.	C
38		— Changes incorporated from Draft 11 ballot resolution, includ- C	
39		ing substantial rework for ex, more, vi, and cd. C	
40	Draft 11	[March 1995] First IEEE recirculation draft.	B

- 41 — Major changes to the charmap format to accommodate ISO/IEC B  
42 10646 and to move character width information from B  
43 **LC\_CTYPE**. (The latter change also affected REs, `localedef` B  
44 and `tr`.) B
- 45 — Major revisions to `ex`, `more`, and `vi`. B
- 46 — A number of `pax` changes to address Canadian concerns about B  
47 the effects of invalid pathnames in `cpio` and `tar` archives, B  
48 and other balloting resolution issues. B
- 49 — Miscellaneous utility changes to address balloting comments B  
50 and interpretation requests. B
- 51 **Draft 10** [June 1994] First IEEE balloting draft. This draft includes the B  
52 working group input from the April 1994 meeting.
- 53 — The subclauses on BREs and EREs Matching Multiple Charac- B  
54 ters (2.8.3.3 and 2.8.4.3) were updated.
- 55 — The synopses of utilities dealing with the [ `-h` | `-R` ] and [ `-H` B  
56 | `-L` ] options were cleaned up.
- 57 — Th effect of SIGQUIT on `ed` was specified.
- 58 — The `pax` list-mode format in 4.48.3.1 was changed B  
59 significantly, based on a proposal from David Korn.
- 60 — A number of terminology changes were made in `sed`.
- 61 — The `xargs -E` option was changed.
- 62 — Escaping in `csplit` REs was specified.
- 63 — A security hole in `ex` (and `vi`) initialization was plugged. The B  
64 meaning of `\l et al` was clarified. The indentation behavior  
65 using `eof` was clarified. The `beautify` option was deleted.
- 66 — References to `{POSIX2_C_BIND}` were deleted from `c89`.
- 67 **Draft 9** [February 1994] This draft includes the working group input from B  
68 the January 1994 meeting.
- 69 — The reorganization of standards with the APIs transferring to B  
70 P1003.1a caused changes primarily in Sections 1 and 2, and  
71 the deletion of Section 7 and Annex B.
- 72 — The new `pax` format was changed significantly, based on a B  
73 proposal from Hal Jespersen.
- 74 — The symbolic link interfaces were changed significantly, based B  
75 on a proposal from Keith Bostic.
- 76 — The `file` command added support for the traditional *magic* B  
77 file. Thanks to Keith for this big addition.
- 78 — Miscellaneous minor changes to `dd`, `ed`, `ex`, `sed`, `tr`, and B  
79 `write`.

- 80 Draft 8 [December 1993] This draft includes the working group input  
81 from the October 1993 meeting.
- 82 — Miscellaneous minor changes to `ed`, `ex`, `find`, `patch`, `test`,  
83 `uudecode`, `uuencode`, `vi`, `xargs`, and the rationale for `sys-`  
84 `tem()`.
- 85 Draft 7 [October 1993] This draft includes the working group input from  
86 the April and July 1993 meetings.
- 87 — A number of the Annex H changes were addressed.
- 88 Draft 6 [March 1993] This draft includes the working group input from  
89 the January 1993 meeting.
- 90 — Mods to the `date` and `pax` commands.
- 91 — Minor mods to `LC_CTYPE (2.5)`, `tr`
- 92 Draft 5 [December 1992] This draft includes the working group input  
93 from the October 1992 meeting.
- 94 — Modifications based on Japanese proposals for state-  
95 dependent encoding, character width definitions, and era  
96 date/time formats.
- 97 — Minor mods to `iconv`, `pax`, and `sed`.
- 98 Draft 4 [August 1992] This draft includes the working group input from  
99 the April and July 1992 meetings.
- 100 — Integration of the WG15 requirements (POSIX.2/D12 Annex H)  
101 for enhancements. Although many of these are currently  
102 placeholders for promised proposals from Japan and Den-  
103 mark, there are substantive additions as follows:
- 104 — Locale definition (2.5) has a new `LC_CTYPE charclass` key-  
105 word.
- 106 — The `date` utility has added field widths.
- 107 — The `pax` format has been updated, based on work by Mark  
108 Brown and David Rowley, to include support for the 10646  
109 UTF canonical form.
- 110 — The `uuencode` utility has added an option for the Internet  
111 Base64 format.
- 112 — The `uudecode` utility has added a `-o` option to override the  
113 output pathname.
- 114 — A new `iconv` utility has been added to convert codesets.
- 115 Draft 3 [February 1992] Miscellaneous minor changes to the `pax` format,  
116 provided by Mark Brown. Symbolic link material added, based on  
117 initial proposals from Dawn Burnett, as modified by the working  
118 group.

119  
120  
121

Draft 2

[December 1991] Miscellaneous minor changes to the `pax` format, provided by Mark Brown. Limited online access provided as part of an IEEE Computer Society experiment.

122  
123

Draft 1

[September 1991] Conversion of `pax` formatting from P1003.1b Draft 5 and `cpio` and `pax` from IEEE Std 1003.1-1990.

**IEEE Standards** documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board  
445 Hoes Lane  
P.O. Box 1331  
Piscataway, NJ 08855-1331  
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.
---

# Contents

	PAGE
Introduction . . . . .	v
Section 1: Revisions to General . . . . .	1
1.1 Scope . . . . .	1
1.2 Normative References . . . . .	2
1.3 Conformance . . . . .	2
1.4 Test Methods . . . . .	3
Section 2: Revisions to Terminology and General Requirements . . . . .	5
2.2.2 General Terms . . . . .	5
2.2.3 Abbreviations . . . . .	6
2.3 Built-In Utilities . . . . .	6
2.4 Character Set . . . . .	7
2.4.2 State-Dependent Character Encodings . . . . .	10
2.5 Locale . . . . .	11
2.6 Environment Variables . . . . .	14
2.8 Regular Expression Notation . . . . .	15
2.10 Utility Conventions . . . . .	17
2.10.2 Utility Syntax Guidelines . . . . .	17
2.13 Configuration Values . . . . .	17
2.13.3 Pathname Variable Values . . . . .	19
Section 3: Revisions to Shell Command Language . . . . .	21
3.13 Pattern Matching Notation . . . . .	23
Section 4: Revisions to Execution Environment Utilities . . . . .	27
4.1 awk – Pattern scanning and processing language . . . . .	27
4.3 bc – Arbitrary-precision arithmetic language . . . . .	35
4.5 cd – Change working directory . . . . .	37
4.6 chgrp – Change file group ownership . . . . .	40
4.7 chmod – Change file modes . . . . .	41
4.8 chown – Change file ownership . . . . .	42
4.13 cp – Copy files . . . . .	43
4.14 cut – Cut out selected fields of each line of a file . . . . .	45
4.15 date – Write the date and time . . . . .	46
4.16 dd – Convert and copy a file . . . . .	46
4.17 diff – Compare two files . . . . .	47
4.20 ed – Edit text . . . . .	49
4.22 expr – Evaluate arguments as an expression . . . . .	54
4.24 find – Find files . . . . .	54
4.26 getconf – Get configuration values . . . . .	56

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.



4.33	ln – Link files	57
4.35	localedef – Define locale environment	58
4.39	ls – List directory contents	60
4.40	mailx – Process Messages	62
4.41	mkdir – Make directories	68
4.42	mkfifo – Make Make FIFO special files	68
4.43	mv – Move files	69
4.45	od – Dump files in various formats	70
4.48	pax – Portable archive interchange	71
4.51	pwd – Return working directory name	100
4.53	rm – Remove directory entries	101
4.55	sed – Stream editor	102
4.56	sh – Shell, the standard command language interpreter	109
4.62	test – Evaluate expression	110
4.64	tr – Translate characters	111
4.72	xargs – Construct argument list(s) and invoke utility	113
4.73	iconv – Convert file codesets	115
Section 5: Revisions to User Portability Utilities Option		119
5.2	at – Execute commands at a later time	119
5.3	batch – Execute commands at a later time	120
5.6	csplit – Split files based on context	121
5.7	ctags – Create a tags file	122
5.9	du – Estimate file usage	123
5.10	ex – Text editor	124
5.11	expand – Convert tabs to spaces	179
5.14	file – Determine file type	180
5.18	more – Display files on a page-by-page basis	185
5.22	patch – Apply changes to files	197
5.24	renice – Set system scheduling priorities of running processes	198
5.32	unexpand – Convert spaces to tabs	199
5.33	uudecode – Decode a binary file	200
5.34	uuencode – Encode a binary file	201
5.35	vi – Screen-oriented (visual) display editor	205
5.37	write – Write to another user	253
Section 6: Revisions to Software Development Utilities Option		255
6.1	ar – Create and maintain library archives	255
6.2	make – Maintain, update, and regenerate groups of programs	256
Section 7: Revisions to Language-Independent System Services		263
Annex A (normative) Revisions to C Language Development Utilities		
Option		265
A.1	c89 – Compile Standard C programs	265
A.3	yacc – Yet another compiler compiler	266
Annex B (normative) Revisions to C Language Bindings Option		267

Annex C (normative) Revisions to FORTRAN Development and Runtime Utilities Options . . . . .	269
Annex D (informative) Revisions to Bibliography . . . . .	271
Annex E (informative) Revisions to Rationale and Notes . . . . .	273
E.4.48 <code>pax</code> – Portable archive interchange . . . . .	278
E.5.10 <code>ex</code> – Text editor . . . . .	292
E.5.18 <code>more</code> — Display files on a page-by-page basis . . . . .	327
E.5.35 <code>vi</code> – Screen-oriented (visual) display editor . . . . .	329
Annex F (informative) Revisions to Portability Considerations . . . . .	351
Annex G (informative) Revisions to Sample National Profile . . . . .	353
Annex H (informative) Balloting Instructions . . . . .	355
Alphabetic Topical Index . . . . .	365

FIGURES

Figure 4-1 – <code>pax</code> Format Archive Example . . . . .	87
--	----

TABLES

Table 2-100 – Pathname Variable Values . . . . .	19
Table 4-100 – <code>ustar</code> Header Block . . . . .	93
Table 4-101 – <code>ustar mode</code> Field . . . . .	94
Table 4-102 – Octet-Oriented <code>cpio</code> Archive Entry . . . . .	97
Table 4-103 – <code>cpio c_mode</code> File Modes . . . . .	98
Table 4-104 – <code>cpio c_mode</code> File Types . . . . .	98
Table 5-100 – <code>uuencode</code> Base64 Values . . . . .	203
Table E-100 – Historical Practice for Symbolic Links . . . . .	277

# Introduction

[This introduction is not a normative part of P1003.2b, Draft Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment, but is included for information only.]

1 This amendment to ISO/IEC 9945-2: 1993 (IEEE Std 1003.2-1992) was developed to  
2 address issues associated with the harmonization of the IEEE standard and the  
3 ISO/IEC International Standard. When the Draft International Standard was  
4 approved, ISO/IEC JTC 1/SC22/WG15 listed specific areas in which enhancements  
5 should be evaluated. Furthermore, it was realized that such a large standard  
6 would encounter various problems (interpretations, clarifications, elimination of  
7 ambiguities, conflicts with test suites, etc.) as it was implemented. Therefore,  
8 this amendment work was authorized with the following goals.<sup>1)</sup>

- 9 (1) Resolve international comments on ISO/IEC 9945-2: 1993. (See Annex H  
10 of that International Standard for a specific list of these areas.)
- 11 (2) Resolve issues resulting from requests for interpretation of IEEE Std  
12 1003.2-1992.
- 13 (3) Improve the clarity, accuracy, and precision of the language in IEEE Std  
14 1003.2-1992, correcting deficiencies found in implementing systems, test  
15 suites, or applications based on the documents.
- 16 (4) Resolve issues identified by IEEE working groups producing functional  
17 standards (profiles) that desire finer granularity in groupings of optional  
18 utilities and features.
- 19 (5) Incorporate interfaces associated with new facilities being produced by  
20 the P1003.1a project, such as symbolic links.
- 21 (6) Assume responsibility for definition of file interchange and archiving for-  
22 mats from P1003.1. This would involve movement of the current section  
23 10 in IEEE Std 1003.1-1990 and the proposed new format from P1003.1a  
24 to the clause in P1003.2 that describes the `pax` utility.

---

25 1) These goals are paraphrased from the IEEE P1003.2b Project Authorization Request (PAR).

26 **Related Standards Activities**

27 Activities to extend this standard to address additional requirements are in pro-  
28 gress, and similar efforts can be anticipated in the future.

29 The following areas are under active consideration at this time, or are expected to  
30 become active in the near future:<sup>2)</sup>

- 31 (1) Shell and Utility facilities 1
- 32 (2) Verification testing methods
- 33 (3) Realtime facilities 1
- 34 (4) Network interface facilities
- 35 (5) System Administration
- 36 (6) Profiles describing application- or user-specific combinations of Open Sys-  
37 tems standards for: supercomputing, multiprocessor, and batch exten-  
38 sions; transaction processing; realtime systems; and multiuser systems  
39 based on historical models
- 40 (7) Services for reliable, available and serviceable systems 1

41 Extensions are approved as “amendments” or “revisions” to this document, follow-  
42 ing the IEEE and ISO/IEC Procedures.

43 Approved amendments are published separately until the full document is  
44 reprinted and such amendments are incorporated in their proper positions.

45 If you have interest in participating in the Portable Application Standards Com-  
46 mitte (PASC) working groups addressing these issues, please send your name,  
47 address, and phone number to the Secretary, IEEE Standards Board, Institute of  
48 Electrical and Electronics Engineers, Inc., P.O. Box 1331, 445 Hoes Lane, Piscata-  
49 way, NJ 08855-1331, and ask to have this forwarded to the chairperson of the  
50 appropriate PASC working group. If you have interest in participating in this  
51 work at the international level, contact your ISO/IEC national body.

---

2) A *Standards Status Report* that lists all current IEEE Computer Society standards projects is available from the IEEE Computer Society, 1730 Massachusetts Avenue NW, Washington, DC 20036-1903; Telephone: +1 202 371-0101; FAX: +1 202 728-9614. Working drafts of POSIX standards under development are also available from this office.

This amendment to IEEE Std 1003.2-1992 was prepared by the Shell and Utilities Working Group, sponsored by the Portable Application Standards Committee of the IEEE Computer Society. At the time this standard was approved, the membership of the Shell and Utilities Working Group was as follows:

*Editor's Note: The full membership list will be provided in a future draft.*

**Portable Application Standards Committee**

Chair: Lowell Johnson  
Vice-Chair: Joe Gwinn  
Functional Chairs: Jay Ashford  
Andrew Josey  
Curtis Royster  
Secretary: Nicholas Stoughton

**Shell and Utilities Workin Working Group**

Chair: Donald W. Cragun  
Secretary: Nicholas Stoughton  
Past Secretaries: Dave Grindeland (1992)  
Dawn Burnett (1993)  
Jeff Zado (1994)  
Editor: Hal Jespersen

**Technical Reviewers**

*Keith Bostic                      Mark Funkenhauser      David Korn*  
*Donald W. Cragun              Andrew Hume              Nick Stoughton*  
*List incomplete ...*

The following persons provided valuable input during the balloting period:

John Q. Public                      Jane Doe                      John Q. Public  
Jane Doe                              John Q. Public                      John Q. Public

The following persons were members of the P1003.2b balloting group that approved the standard for submission to the IEEE Standards Board:

John Q. Public                      John Q. Public                      Jane Doe

When the IEEE Standards Board approved this standard on *<date to be provided>*, it had the following membership:

(to be pasted in by IEEE)

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.

# Draft Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment

## Section 1: Revisions to General

### 1 1.1 Scope

2 ⇒ **1.1 Scope.** *Update references to POSIX.1-1990 to be the version amended by*  
3 *P1003.1a.*

4 **Rationale:** The P1003.1 and P1003.2 working groups have agreed that the  
5 P1003.1a and P1003.2b drafts will be submitted to the IEEE Standards Board for  
6 approval at the same time.

7 ⇒ **1.1 Scope.** *At the beginning of the eleventh paragraph, delete the following:*

8 Portions of this standard comprise optional language bindings to system ser-  
9 vice interfaces. (See, for example, the C-Language Bindings Option in Annex  
10 B.)

11 **Rationale:** The P1003.1 and P1003.2 working groups have agreed that all C-  
12 language APIs will be transferred into the P1003.1a amendment.

13 ⇒ **1.1 Scope.** *Delete the twelfth paragraph, which reads:*

14 For language interfaces, or functions, this standard has been defined  
15 exclusively at the source-code level. The objective is that a conforming port-  
16 able application source program can be translated to execute on a conforming  
17 implementation. This standard assumes that the source program may need to  
18 be retranslated to produce target code for a new environment prior to execu-  
19 tion in that environment.

## 20 **1.2 Normative References**

21 ⇒ **1.2 Normative References.** *Update the reference to POSIX.1 {8} to represent  
22 the version including the IEEE Std 1003.1a-199x and IEEE Std 1003.1b-1993  
23 amendments (and 1003.1c if it is approved in time).*

24 ⇒ **1.2 Normative References.** *Add the following entry to the Normative Refer-  
25 ences clause:*

26 {10} ISO/IEC 10646-1: 1993, *Information technology—Universal Multiple-  
27 Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multil-  
28 ingual Plane.*

## 29 **1.3 Conformance**

30 ⇒ **1.3 Conformance.** *Delete all references to the C-Language Bindings Option  
31 and the {POSIX2\_C\_BIND} symbol from 1.3 and all of its subclauses.*

32 ⇒ **1.3.1.1 Requirements.** *Change item (3) to:*

33 (3) The system may provide additional or enhanced utilities or facilities not  
34 required by this standard. Nonstandard extensions should be identified  
35 as such in the system documentation. Nonstandard extensions, when  
36 used, may change the behavior of utilities or facilities defined by this  
37 standard. In such cases, the conformance document of the implementa-  
38 tion (see 2.2.1.3) shall define an execution environment (i.e., shall provide  
39 general operating instructions) in which an application can be run with  
40 the behavior specified by this standard. In no case shall such an environ-  
41 ment require modification of a Strictly Conforming POSIX.2 Application.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



42           **Rationale:** Since Annex B is gone, all references to “functions” have to be  
43 removed.

#### 44   **1.4 Test Methods**

45   ⇒ **1.4 Test Methods.** *Change the entire clause to:*

B

46           The test methods for this standard are described in P2003.2 {Bxx}.

47           *Editor's Note: This will be updated to indicate a revised P2003.2 (if and when a*  
48           *PAR is authorized).*



## Section 2: Revisions to Terminology and General Requirements

1 *Editor's Note: The following material on symbolic links is related to* B  
 2 *P1003.1a/D12; the definition is from that draft verbatim. All of the symbolic link* B  
 3 *material in this and later sections is contingent on P1003.1a being approved before*  
 4 *P1003.2b. The P1003.1 and P1003.2 working groups have agreed that the*  
 5 *P1003.1a and P1003.2b drafts will be submitted to the IEEE Standards Board for*  
 6 *approval at the same time. When P1003.1a is approved, a number of the P1003.2*  
 7 *definitions copied from POSIX.1 {8} will be updated automatically. See also the*  
 8 *considerable rationale text about symbolic links being added to Annex E.*

### 9 2.2.2 General Terms

10 ⇒ **2.2.2.35 collation sequence.** *Change the second paragraph of this definition* B  
 11 *(the one beginning with "The character order, ... ") to:* B

12 The collation sequence is used for sorting and is determined from the collating B  
 13 weights assigned to each collating element. In the absence of weights, the col- B  
 14 lation sequence is also the *collating element order* (see 2.2.2.201). B

15 **Rationale:** This change is the result of interpretation requests PASC 1003.2-92 B  
 16 #27 and #40 submitted for IEEE Std 1003.2-1992. B

17 ⇒ **2.2.2.87 hard link.** *Replace the definition with the following:*

18 The relationship between two directory entries that represent the same file;  
 19 the result of an execution of the `ln` utility (without the `-s` option) or the  
 20 POSIX.1 {8} `link()` function.

21 ⇒ **2.2.2.165 source code.** *Change the second and third paragraphs to:*

22 When dealing with an ISO/IEC conforming programming language, source code  
 23 is input to a compiler conforming to that ISO/IEC standard.

24 **Rationale:** Since Annex B is gone, all references to C-Language Binding Option  
25 have to be removed.

26 ⇒ **2.2.2 General Terms.** *Modify the contents of subclause 2.2.2, General Terms,*  
27 *to add the following definitions in the correct sorted order [disregarding the*  
28 *subclause numbers shown here].*

29 **2.2.2.201 collating element order:** The relative order of collating elements as B  
30 determined by the setting of the LC\_COLLATE category in the current locale. B

31 The collating element order is used in range expressions in REs (see 2.8) and is B  
32 determined by the order in which collating elements are specified between B  
33 `order_start` and `order_end` keywords in the LC\_COLLATE category. B

34 **2.2.2.202 symbolic link:** A type of file that contains a string whose length is B  
35 less than or equal to `{SYMLINK_MAX}`. B

36 The string in the file is interposed into a pathname being resolved, when the file B  
37 is encountered during pathname resolution, to create a new pathname. B  
38 [P1003.1a/D12] B

### 39 **2.2.3 Abbreviations**

40 ⇒ **2.2.3 Abbreviations.** *Modify the contents of subclause 2.2.3, Abbreviations,*  
41 *to add the following definition in the correct sorted order [disregarding the sub-*  
42 *clause numbers shown here].*

43 **2.2.3.201 UTF8:** The File-System Safe Universal Translation Format defined in  
44 Annex N of ISO/IEC 10646 {10}, as amended by ISO/IEC JTC 1/SC2/WG2 N993.

## 45 **2.3 Built-In Utilities** B

46 ⇒ **2.3 Built-In Utilities.** *In Table 2-3, add the `pwd` utility in the proper sorted* C  
47 *order.* C

48 **Rationale:** Changes to the `pwd` utility in this draft require it to affect the C  
 49 environment variable **PWD**, so it must become a shell built-in. C

50 ⇒ **2.3 Built-In Utilities.** *Delete the final paragraph in this subclause (the one B*  
 51 *beginning “Since exec-able versions . . . ”).* B

52 **Rationale:** As part of a general cleanup to remove references to the now-deleted B  
 53 Chapter 7, this paragraph was removed because it is little more than rationale B  
 54 and duplicates material in the previous paragraph. B

## 55 2.4 Character Set

56 ⇒ **2.4 Character Set.** *Replace the paragraph and following dashed list that*  
 57 *begins “The current version of this standard does not address fully”, with:*

58 State-dependent character encodings are described in 2.4.2.

59 ⇒ **2.4.1 Character Set Description File.** *Change the second paragraph (the B*  
 60 *one beginning “Each character set . . . ”) to:* B

61 Each character set description file, except those that use ISO/IEC 10646 {10} B  
 62 position values as the encoding values, shall define characteristics for the B  
 63 coded character set and the encoding for the characters specified in Table 2-4, B  
 64 and may define encoding for additional characters supported by the implemen- B  
 65 tation. Other information about the coded character set may also be in the B  
 66 file. Coded character set character values shall be defined using symbolic B  
 67 character names followed by character encoding values. B

68 ⇒ **2.4.1 Character Set Description File.** *Change the two consecutive para-* B  
 69 *graph that begin “The encoding part . . . ” and “Decimal constants . . . ” to:* B

70 The encoding part shall be expressed as one (for single-byte character values) B  
 71 or more concatenated decimal, octal, or hexadecimal constants in the following B  
 72 formats: B

73 "%cd%2d", <escape\_char>, <decimal byte value> B

74 "%cx%2x", <escape\_char>, <hexadecimal byte value> B

75 "%c%2o", <escape\_char>, <octal byte value> B

76 C

77 Decimal constants shall be represented by two or three decimal digits, pre- B  
 78 ceded by the escape character and the lowercase letter d; for example, \d05, B  
 79 \d97, or \d143. Hexadecimal constants shall be represented by two hexade- B  
 80 cimal digits, preceded by the escape character and the lowercase letter x; for B  
 81 example, \x05, \x61, or \x8f. Octal constants shall be represented by two or B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

82 three octal digits, preceded by the escape character; for example, \05, \141, or B  
 83 \217. In a portable charmap file, each constant shall represent an 8 b byte. B  
 84 Implementations supporting other byte sizes may allow constants to represent B  
 85 values larger than those that can be represented in 8 b bytes, and to allow B  
 86 additional digits in constants. When constants are concatenated for multibyte B  
 87 character values, they shall be of the same type and interpreted in byte order B  
 88 from first to last with the least significant byte of the multibyte character B  
 89 specified by the last constant. The manner in which these constants are B  
 90 represented in the character stored in the system is implementation defined. B  
 91 Omitting bytes from a multibyte character definition produces undefined B  
 92 results. B

93 ⇒ **2.4.1 Character Set Description File.** *Add a new paragraph preceding the* B  
 94 *one that consists of “The comment is optional.”* B

95 In lines defining ranges of symbolic names that also use ISO/IEC 10646 {10} B  
 96 position constant values, the conversion to the target codeset encoding value B  
 97 shall be performed before assignment of encoding values to symbolic names. B

98 *Editor’s Note: The following rationale will be added to E.2.4.1, but is kept here for* B  
 99 *this draft:* B

100 *(Rationale text deleted in Draft 12.)* B

101 ⇒ **2.4.1 Character Set Description File.** *Delete the final paragraph in the* B  
 102 *subclause, which was:*

103 For interpretation of the dollar sign and the number sign, see 2.2.2.45 and B  
 104 2.2.2.110.

105 **Rationale:** This change satisfies the following corrigendum request from ISO/IEC B  
 106 9945-2: 1993 Annex H.2:

107 (2) The final paragraph of 2.4.1 implies that there are special interpretations B  
 108 of the dollar sign and number sign characters described in 2.2.2, but no B  
 109 text appears in 2.2.2.45 or 2.2.2.110 to explain these interpretations.

110 ⇒ **2.4.1 Character Set Description File.** *Add the following text to the end of* B  
 111 *the subclause:* B

112 The following declarations can follow the character set mapping definitions B  
 113 (after the END CHARMAP statement). Each shall consist of the keyword shown B  
 114 in the following list, starting in column 1, followed by the value(s) to be associ- B  
 115 ated to the keyword, as defined below. B

116	WIDTH	An unsigned positive integer value defining the column	B
117		width (see 2.2.2.36) for the coded character set	B
118		specified in Table 2-4 and Table 2-5. Coded character	B
119		set character values shall be defined using symbolic	B
120		character names followed by column width values.	B

121 Defining a character with more than one `WIDTH` pro- B  
 122 duces undefined results. The `END WIDTH` keyword shall B  
 123 be used to terminate the `WIDTH` definitions. B

124 C

125 `WIDTH_DEFAULT` An unsigned positive integer value defining the default B  
 126 column width for any printable character not listed by C  
 127 one of the `WIDTH` keywords. If no `WIDTH_DEFAULT` key- B  
 128 word is included in the charmap, the default character B  
 129 width shall be 1. B

130 *Example:* B

131 After the `END CHARMAP` statement, a syntax for a width definition would be: B

```
132     WIDTH                               B
133     <NUL>...<IS1>                       -1      C
134     <A>                                   1        B
135     <B>                                   1        B
136     <C>...<Z>                             1        B
137     ...                                   B
138     <fool>...<foon>                       2        B
139     ...                                   B
140     END WIDTH                             B
```

141 The code point values represented by the symbols `<A>` and `<B>` are assigned a B  
 142 width of 1. Also, the code point values `<C>` to `<Z>` inclusive (`<C>`, `<D>`, `<E>`, B  
 143 `<F>`, `<G>`, `<H>`, `<I>`, and so on) are assigned a width of 1. B

144 In this example, `<A>...<Z>` would have required fewer lines, but the alterna- B  
 145 tive was given to demonstrate flexibility. B

146 The keyword `WIDTH_DEFAULT` can be added as appropriate. All nonprintable C  
 147 characters shall have a width of `-1`. C

148 **Rationale:** This change satisfies the following requirement from ISO/IEC 9945- B  
 149 2: 1993 Annex H.1: B

150 (9) The definition of column position (see 2.2.2.36) relies on the B  
 151 implementation's knowledge of the integral width of the characters. The B  
 152 charmap (2.4) or `LC_CTYPE` (2.5.2.1) locale definitions should be B  
 153 enhanced to allow application specification of these widths. B

154 The character "width" information was first considered for inclusion under B  
 155 `LC_CTYPE` but was moved because it is more closely associated with the informa- B  
 156 tion in the charmap than information in the locale source (cultural conventions B  
 157 information). Concerns were raised that formalizing this type of information is B  
 158 moving the locale source definition from the codeset independent entity that it B  
 159 was designed to be to a repository of codeset specific information. A similar issue B  
 160 occurred with the `<code_set_name>`, `<mb_cur_max>`, and `<mb_cur_min>` infor- B  
 161 mation, which was resolved to reside in the charmap definition. B

162 The width definition was added to the POSIX.2b standard with the intent that the B  
 163 functions `wcswidth()` and/or `wcwidth()` [currently specified in the X/Open B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

164 Common Application Environment Specification (X/Open CAE), System Interfaces  
 165 and Headers, Version 2] be the mechanism to retrieve the character width  
 166 information. B  
B  
B

167 ⇒ **2.4 Character Set.** *Add the following new subclause:*

#### 168 **2.4.2 State-Dependent Character Encodings**

169 This subclause addresses the use of state-dependent character encodings (i.e.,  
 170 those in which the encoding of a character is dependent on one or more shift codes  
 171 that may precede it).

172 A single-shift encoding (where each character not in the initial shift state is pre-  
 173 ceded by a shift code) can be defined in the charmap format if each shift-  
 174 code/character sequence is considered a multibyte character, defined using the  
 175 concatenated-constant format described in 2.4.1. If the implementation supports  
 176 a character encoding of this type, all of the standard utilities shall support it.

177 A locking-shift encoding (where the state of the character is determined by a shift  
 178 code that may affect more than the single character following it) could be defined  
 179 with an extension to the charmap format described in 2.4.1. If the implementa- B  
 180 tion supports a character encoding of this type, any of the standard utilities that  
 181 describe character (versus byte) or text-file manipulation shall have the following  
 182 characteristics:

183 (1) The utility shall process the statefully encoded data as a concatenation of  
 184 state-independent characters. The presence of redundant locking shifts  
 185 shall not affect the comparison of two statefully encoded strings.

186 (2) A utility that divides, truncates, or extracts substrings from statefully  
 187 encoded data contain locking shifts at the beginning or end of the result- B  
 188 ing data, if appropriate, to retain correct state information.

#### 189 **State-Dependent Character Encodings Rationale**

190 A requirement was considered that would force utilities to eliminate any redun-  
 191 dant locking shifts, but this was left as a quality of implementation issue.

192 **Rationale:** This change satisfies the following requirement from ISO/IEC 9945-  
 193 2: 1993 Annex H.1:

194 (8) The support of state-dependent (shift encoding) character sets should be  
 195 addressed fully. See descriptions of these in 2.4. If such character encod-  
 196 ings are supported, it is expected that this will impact 2.4 (charmap), 2.5  
 197 (locale definition), 2.8 (regular expressions), and the comm, cut, diff,  
 198 grep, head, join, paste, and tail utilities.



199 **2.5 Locale**

200

B

201 ⇒ **2.5 Locale.** *Change the second paragraph (the one following the list of*  
 202 *environment variable names) to:*

203 Conforming implementations shall implement the standard utilities so that  
 204 their behavior is based on the current locale, as defined in the Environment  
 205 Variables subclause for each utility.

206 **Rationale:** Since Annex B is gone, all references to it and to “functions” have to  
 207 be removed.

208 ⇒ **2.5.2 Locale Definition.** *In the numbered list, change the first sentence of*  
 209 *item (2) to:*

210 (2) A character in the portable character set can be represented by the char-  
 211 acter itself, in which case the value of the character is implementation  
 212 defined. (Implementations may allow other characters to be represented  
 213 as themselves, but such locale definitions are not portable.)

214 **Rationale:** This change was made in response to a Japanese ballot comment to  
 215 ISO/IEC 9945-2: 1993.

216 ⇒ **2.5.2.1 LC\_CTYPE.** *Add the following keyword items between the items*  
 217 *labeled blank and toupper:*

218	<code>charclass</code>	Define one or more locale-specific character class names as strings separated by semicolons. Each named character class can then be defined subsequently in the LC_CTYPE definition. A character class name shall consist of at least one and no more than {CHARCLASS_NAME_MAX} bytes of alphanumeric characters from the portable filename character set. The first character cannot be a digit. The name cannot match any of the LC_CTYPE keywords defined in this standard. Future revisions of this standard will not specify any LC_CTYPE keywords containing uppercase letters.	B B B B B B B
219			
220			
221			
222			
223			
224			
225			
226			
227			

228 *charclass-name*

229 Define characters to be classified as belonging to the named  
 230 locale-specific character class. In the POSIX Locale, the  
 231 locale-specific named character classes need not exist.

232 If a class name is defined by a `charclass` keyword, but no  
 233 characters are subsequently assigned to it, this is not an  
 234 error; it shall represent a class without any characters  
 235 belonging to it.

236 The *charclass-name* can be used in regular expression and  
 237 shell pattern-matching bracket expressions, and by the `tr` B  
 238 utility. B

239 **Rationale:** This addition was adopted from XPG4 {B49} to satisfy the following  
 240 requirement from ISO/IEC 9945-2: 1993 Annex H.1:

241 (3) The LC\_CTYPE (2.5.2.1) locale definition should be enhanced to allow  
 242 user-specified additional character classes, similar in concept to the C  
 243 C Standard {7} Multibyte Support Extension (MSE) *is\_wctype()* function. C  
 C

244 The symbolic constant {CHARCLASS\_NAME\_MAX} was adopted from the XPG4 C  
 245 {B49}. Application portability is enhanced by the use of symbolic constants. C

246 ⇒ **2.5.2.1 LC\_CTYPE.** *Add the following keyword items between the items*  
 247 *labeled digit and space:*

248 `alnum` Define characters to be classified as letters and numeric  
 249 digits. Only the characters specified for the `alpha` and  
 250 `digit` keywords shall be specified. Characters specified for  
 251 the keywords `alpha` and `digit` are automatically included  
 252 in this class.

253 **Rationale:** The `alnum` keyword was added to correct an oversight in POSIX.2-  
 254 1992; it was clearly implied by the POSIX Locale table, but since it was mentioned  
 255 only in a comment field, it was considered not normative.

256 B

257 ⇒ **2.5.2.2.4 Collation Sequence.** *Remove the following sentence from the*  
 258 *second paragraph:*

259 The NUL character shall compare lower than any other character.

260 **Rationale:** This change partially satisfies the following requirement from  
 261 ISO/IEC 9945-2: 1993 Annex H.1:

262 (7) The specific encoding and collation requirements for the character NUL  
 263 should be removed.

264 The specific encoding was retained because the C Standard {7} requires it.

265 ⇒ **2.5.2.3 LC\_MONETARY.** *In Table 2-9, add the following after the entry for*  
 266 *int\_frac\_digits:*

267 `frac_digits` -1

268 **Rationale:** This change satisfies the following corrigendum request from ISO/IEC  
269 9945-2: 1993 Annex H.2:

270 (3) Table 2-9, listing the LC\_MONETARY Category Definition in the POSIX  
271 Locale, omits the value to be assigned to `frac_digits`.

272 ⇒ **2.5.2.5 LC\_TIME.** *Add new keywords in between `era_d_fmt` and `alt_`* B  
273 *digits:* B

274 `era_d_t_fmt` The format of the date and time in alternate era notation, B  
275 corresponding to the `%Ec` field descriptor. B

276 `era_t_fmt` The format of the time in alternate era notation, B  
277 corresponding to the `%EX` field descriptor. B

278 **Rationale:** This change was to correct an oversight in ISO/IEC 9945-2: 1993,  
279 pointed out by Japan. It is identical to an extension in XPG4 {B49}.

280 ⇒ **2.5.2.5 LC\_TIME.** *In Table 2-11, change the lines defining `t_fmt_ampm` to:*

```
281 # Appropriate 12 h time representation (%r) "%I:%M:%S %p"
282 t_fmt_ampm "<percent-sign><I><colon><percent-sign><M><colon>\
283 <percent-sign><S><space><percent_sign><p>"
```

284 **Rationale:** This change satisfies the following corrigendum request from ISO/IEC  
285 9945-2: 1993 Annex H.2:

286 (5) Table 2-11, listing the LC\_TIME Category Definition in the POSIX Locale,  
287 contains the following entry:

```
288 # Appropriate 12 h time representation (%r) "%I:%M:%S %p"
289 t_fmt_ampm "<percent-sign><I><colon><percent-sign><M><colon>\
290 <percent-sign><S> <percent_sign><p>"
```

291 It is unclear whether there is a space between `<S>` and `<percent_`  
292 `sign>` (which should have been represented as `<space>` to match the  
293 other entries) or whether this is a typographical error.

294 ⇒ **2.5.3.1 Locale Lexical Conventions.** *Add the following token description:*

295 CHARCLASS A string of alphanumeric characters from the portable  
296 character set, the first of which shall not be a digit, consist-  
297 ing of at least one and at most {CHARCLASS\_NAME\_MAX} B  
298 bytes, and optionally surrounded by double-quotes. B

299 **Rationale:** See the 2.5.2.1 changes.

300 ⇒ **2.5.3.2 Locale Grammar.** *Modify the `ctype_keyword` and `charclass_`*  
 301 *keyword descriptions as follows:*

```

302     ctype_keyword           : charclass_keyword charclass_list EOL
303                           | charwidth_keyword charclass_list EOL
304                           | defwidth_keyword defwidth_value EOL
305                           | charconv_keyword charconv_list EOL
306                           | 'charclass' charclass_namelist EOL
307                           ;
308     charclass_namelist     : charclass_namelist ';' CHARCLASS
309                           | CHARCLASS
310                           ;
311     charclass_keyword      : 'upper' | 'lower' | 'alpha' | 'digit'
312                           | 'alnum' | 'xdigit' | 'space' | 'print'
313                           | 'graph' | 'blank' | 'cntrl' | 'punct'
314                           | CHARCLASS
315                           ;
316

```

B

317 ⇒ **2.5.3.2 Locale Grammar.** *In the `time_keyword_opt` description, add*  
 318 *'era\_d\_t\_fmt' and 'era\_t\_fmt' as alternatives to the four existing entries.*

B

319 **Rationale:** See the 2.5.2.1 changes.

## 320 2.6 Environment Variables

B

321 ⇒ **2.6 Environment Variables.** *In the fourth paragraph, change the sentence*  
 322 *"See 7.2 and 3.12 for methods of accessing these variables." to:*

B

B

323 See the `getenv()` function in POSIX.1 {8} and 3.12 for methods of accessing  
 324 these variables.

B

B

325 **Rationale:** This change is part of a general cleanup to remove references to the  
 326 now-deleted Chapter 7.

B

B

327 ⇒ **2.6 Environment Variables.** *Add the following variable in proper sorted*  
 328 *order:*

B

B

329 **PWD** This variable shall represent an absolute pathname of  
 330 the current working directory. It shall not contain any  
 331 filename components of dot or dot-dot. The value is set  
 332 by the `cd` utility.

B

B

B

B

333 **2.8 Regular Expression Notation**

334 ⇒ **2.8.1 RE Introduction.** *Delete the final sentence in this subclause: “Both* B  
 335 *BREs and EREs are supported by the RE Matching interface in 7.3.”* B

336 **Rationale:** As part of a general cleanup to remove references to the now-deleted B  
 337 Chapter 7, this sentence was removed because it was little more than rationale. B

338 ⇒ **2.8.3.2 RE Bracket Expression.** *Change the first paragraph of item (7) to:*

339 (7) A *range expression* represents the set of collating elements that fall B  
 340 between two elements in the collating element order (see 2.2.2.201) of the C  
 341 current locale, inclusive. A range expression shall be expressed as the C  
 342 starting point and the ending point separated by a hyphen (-). B

343 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B  
 344 #27 submitted for IEEE Std 1003.2-1992. B

345 ⇒ **2.8.3.3 BREs Matching Multiple Characters.** *In the numbered list, change*  
 346 *item (3) to:*

347 (3) The *backreference expression*  $\backslash n$  shall match the same (possibly empty)  
 348 string of characters as was matched by a subexpression enclosed between  
 349  $\backslash ($  and  $\backslash )$  preceding the  $\backslash n$ . The character  $n$  shall be a digit from 1  
 350 through 9, specifying the  $n$ -th subexpression [the one that begins with  
 351 the  $n$ -th  $\backslash ($  and ends with the corresponding paired  $\backslash )$ ]. The expression  
 352 is invalid if less than  $n$  subexpressions precede the  $\backslash n$ . For example, the  
 353 expression  $\wedge \backslash (.*) \backslash 1 \$$  matches a line consisting of two adjacent  
 354 appearances of the same string, and the expression  $\backslash (a \backslash ) * \backslash 1$  fails to  
 355 match  $a$ . When the referenced subexpression matched more than one  
 356 string, the backreferenced expression shall refer to the last matched  
 357 string. If the subexpression referenced by the backreference matches  
 358 more than one string because of an asterisk (\*) or an interval expression  
 359 [see item (5)], the backreference shall match the last (rightmost) of these  
 360 strings.

361 **Rationale:** The changes to 2.8.3.3 and 2.8.4.3 remove an unspecified or ambigu-  
 362 ous behavior in POSIX.2, aligning it with the requirements specified for the  
 363 *regcomp()* expression, and is the result of interpretation request PASC 1003.2-92  
 364 #43 submitted for IEEE Std 1003.2-1992.

365 ⇒ **2.8.3.3 BREs Matching Multiple Characters.** *At the end of the subclause,*  
 366 *add a new paragraph:*

367 A subexpression repeated by an asterisk (\*) or an interval expression shall not  
 368 match a null expression unless this is the only match for the repetition or it is  
 369 necessary to satisfy the exact or minimum number of occurrences for the inter-  
 370 val expression.

371 ⇒ **2.8.4.3 EREs Matching Multiple Characters.** *At the end of the subclause,*  
 372 *add a new paragraph:*

373 An ERE matching a single character repeated by an \*, ?, or an interval expres- B  
 374 sion shall not match a null expression unless this is the only match for the  
 375 repetition or it is necessary to satisfy the exact or minimum number of  
 376 occurrences for the interval expression.

377 ⇒ **2.8.5.2 RE and Bracket Expression Grammar.** *In the section of the gram-*  
 378 *mar for the nondupl\_RE nonterminal, remove the third line:*

379 | Back\_open\_paren Back\_close\_paren

380 **Rationale:** This change is the result of interpretation request PASC 1003.2-92  
 381 #43 submitted for IEEE Std 1003.2-1992. Although the grammar required sup-  
 382 port for null subexpressions, subclause 2.8.3.3 does not describe the meaning of,  
 383 and historical practice did not support, this construct.

#### 384 **2.9.1.4 File Read, Write, and Creation**

385 ⇒ **2.9.1.4 File Read, Write, and Creation.** *In the first numbered list, change*  
 386 *item (3) to:*

387 (3) If the file is a regular file, the permission bits are set to

388 S\_IROTH | S\_IWOTH | S\_IRGRP | S\_IWGRP | S\_IRUSR | S\_IWUSR

389 (see Section 5.6.1.2 of POSIX.1 {8}), except that the bits specified by the  
 390 file mode creation mask of the process are cleared.

391 If the file is a directory, the permission bits are set to

392 S\_IRWXU | S\_IRWXG | S\_IRWXO

393 (see Section 5.6.1.2 of POSIX.1 {8}), except that the bits specified by the  
 394 file mode creation mask of the process are cleared.

395 **Rationale:** This change is required to match historical practice and is the result  
396 of interpretation request PASC 1003.2-92 #18 submitted for IEEE Std 1003.2-1992.

397 ⇒ **2.9.1.4 File Read, Write, and Creation.** *In the first numbered list, change* B  
398 *item (6) to:* B

399 (6) If the file is a symbolic link, the effect shall be undefined unless the B  
400 {POSIX2\_SYMLINKS} variable is in effect for the directory in which the B  
401 symbolic link would be created. B

402 (7) Unless otherwise specified, the file created shall be a regular file. B

403 *Editor's Note: The following rationale will be added to E.2.9.1.8, but is kept here*  
404 *for this draft:*

405 **Pathname Resolution Rationale.** *(This subclause is not a part of P1003.2b)*

406 P1003.1a now includes symbolic links in pathname resolution and a number of  
407 concepts are automatically inherited in POSIX.2 by this inclusion. The large  
408 majority of standard utilities resolve pathnames and operate on files without spe-  
409 cial arrangements for symbolic links. Because of the global POSIX.1 {8} inheri-  
410 tance, this entails very few modifications to utility descriptions.

411 **2.10 Utility Conventions** B

412 **2.10.2 Utility Syntax Guidelines** B

413 ⇒ **2.10.2 Utility Syntax Guidelines.** *Change the first paragraph to:* B

414 The following guidelines are established for the naming of utilities and for the B  
415 specification of options, option-arguments, and operands. The *getopt()* func- B  
416 tion in POSIX.1 {8} assists utilities in handling options and operands that con- B  
417 form to these guidelines. B

418 **Rationale:** This change is part of a general cleanup to remove references to the B  
419 now-deleted Chapter 7. All of the applicable functions are now in POSIX.1-199x, B  
420 the version created by the currently balloting P1003.1a. B

421 **2.13 Configuration Values**

422 ⇒ **2.13.1 Symbolic Limits.** *Change the second paragraph (the one beginning*  
 423 *“The values specified in Table 2-17 ... ”) to:*

424 The values specified in Table 2-17 represent the lowest values conforming  
 425 implementations shall provide and, consequently, the largest values on which  
 426 an application can rely without further enquiries, as described below. These  
 427 values shall be accessible to applications via the `getconf` utility (see 4.26).

428 ⇒ **2.13.1 Symbolic Limits.** *Change the fourth paragraph (the one beginning*  
 429 *“The functions in 7.8.2 ... ”) to:*

430 The `getconf` utility shall return the value of each symbol on each specific  
 431 implementation. The value so retrieved shall be the largest, or most liberal,  
 432 value that shall be available throughout the session lifetime, as determined at  
 433 session creation.

434 ⇒ **2.13.1 Symbolic Limits.** *Add a new symbol to Table 2-17:* B

Name	Description	Value
{POSIX2_CHARCLASS_NAME_MAX}	The maximum number of bytes in a character class name.	14

435 B  
 436 B  
 437 B

438 ⇒ **2.13.1 Symbolic Limits.** *Add a new symbol to Table 2-18:* B

Name	Description	Minimum Value
{CHARCLASS_NAME_MAX}	The maximum number of bytes in a character class name.	{POSIX2_CHARCLASS_NAME_MAX}

439 B  
 440 B  
 441 B  
 442 B

443 ⇒ **2.13.2 Symbolic Constants for Portability Specifications.** *Change the*  
 444 *first paragraph to:*

445 Table 2-19 lists symbols that can be used by the application to determine  
 446 which optional facilities are present on the implementation. The `getconf`  
 447 utility can be used to retrieve the value of each symbol on each specific imple-  
 448 mentation.



449 ⇒ **2.13.2 Symbolic Constants for Portability Specifications.** *Delete the*  
 450 *Table 2-19 entry for {POSIX2\_C\_BIND}.*

451 **Rationale:** The preceding four changes are related to the removal of Annex B.

### 452 2.13.3 Pathname Variable Values

453 ⇒ **2.13.3 Pathname Variable Values.** *Add a new subclause, 2.13.3, Pathname*  
 454 *Variable Values, as follows:*

455 The values in Table 2-100 may be constants within an implementation or may  
 456 vary from one pathname to another.

457 **Table 2-100 – Pathname Variable Values**

Name	Description
{POSIX2_SYMLINKS}	When referring to a directory, the system supports the creation of symbolic links within that directory; for nondirectory files, the meaning of {POSIX2_SYMLINKS} is undefined.

462 **Symbolic Constants Rationale.** *(This subclause is not a part of P1003.2b)*

463 The {POSIX2\_SYMLINKS} variable indicates that the underlying operating system  
 464 supports the creation of symbolic links in specific directories. Many of the  
 465 POSIX.2 utilities that deal with symbolic links do not depend on this value. For  
 466 example, a utility that follows symbolic links (or does not, as the case may be) will  
 467 only be affected by a symbolic link if it encounters one. Presumably, a file system  
 468 that does not support symbolic links will not contain any. This variable does  
 469 affect such utilities as `ln -s` and `pax` that attempt to create symbolic links.

470 {POSIX2\_SYMLINKS} was developed even though there is no comparable  
 471 configuration value in P1003.1a. Since POSIX.2 does not depend on a fully con-  
 472 forming POSIX.1 {8} system underneath, the developers of the standard wished to  
 473 allow systems in which this was an optional feature, perhaps on a file system  
 474 basis.



## Section 3: Revisions to Shell Command Language

1	⇒ <b>3.1 Shell Introduction.</b> <i>Change the first paragraph to:</i>	B
2	The shell is a command language interpreter. This section describes the syn-	B
3	tax of that command language as it is used by the <code>sh</code> utility and the	B
4	POSIX.1 {8} <code>system()</code> and <code>popen()</code> functions.	B
5	<b>Rationale:</b> This and the following change are part of a general cleanup to remove	B
6	references to the now-deleted Chapter 7. All of the applicable functions are now	B
7	in POSIX.1-199x, the version created by the currently balloting P1003.1a.	B
8	⇒ <b>3.1 Shell Introduction.</b> <i>Change the first numbered item to:</i>	B
9	(1) Reads its input from a file (see <code>sh</code> in 4.56), from the <code>-c</code> option, or from	B
10	the POSIX.1 {8} <code>system()</code> or <code>popen()</code> functions. If the first line of a file of	B
11	shell commands starts with the characters <code>#!</code> , the results are unspecified.	B
12	⇒ <b>3.2.3 Double Quotes.</b> <i>Change the description of backslash to:</i>	B
13	\ The backslash shall retain its special meaning as an escape character	B
14	(see 3.2.1) only when followed by one of the following characters when	B
15	considered special:	B
16	\$ \ " \ <newline>	B
17	<b>Rationale:</b> This change is the result of interpretation request PASC 1003.2-92	B
18	#102 submitted for IEEE Std 1003.2-1992.	B
19	⇒ <b>3.5.3 Environment Variables.</b> <i>Change the description of ENV to:</i>	B
20	<b>ENV</b> This variable, when and only when an interactive shell is	C
21	invoked, shall be subjected to parameter expansion (see	B
22	3.6.2) by the shell, and the resulting value shall be used	B
23	as a pathname of a file containing shell commands to	B
24	execute in the current environment. The file need not be	B
25	executable. If the expanded value of <b>ENV</b> is not an abso-	B
26	lute pathname, the results are unspecified. <b>ENV</b> shall be	B
27	ignored if the real and effective user IDs or real and effec-	B
28	tive group IDs of the user are different.	C

29 ⇒ **3.5.3 Environment Variables.** *Add a new variable in the proper sorted* C  
 30 *order:* C

31 **PWD** This variable shall be set by the shell to be an absolute C  
 32 pathname of the current working directory, containing no C  
 33 components of type symbolic link, no components that C  
 34 are dot, and no components that are dot-dot when the C  
 35 shell is initialized. If an application sets or unsets the C  
 36 value of **PWD**, the behaviors of the `cd` and `pwd` utilities C  
 37 are unspecified. C

38 *Editor's Note: The following rationale will be added to E.3.5.3, but is kept here* B  
 39 *with Environment Variables for this draft:* B

40 **Environment Variables Rationale.** *(This subclause is not a part of P1003.2b)* B

41 A previous version of this standard did not specify whether **ENV** file processing C  
 42 was performed by noninteractive shells. Historical practice supports **ENV** pro- C  
 43 cessing only for interactive shells, and this is what the standard now requires. C

44 ⇒ **3.9.4.3 case Conditional Construct.** *In the first paragraph, replace the*  
 45 *sentence "The compound-list for each list of patterns shall be " terminated with*  
 46 *;;" with:*

47 The *compound-list* for each list of patterns, with the possible exception of the  
 48 last, shall be terminated with `;;`.

49 **Rationale:** This change is the result of interpretation request PASC 1003.2-92  
 50 #46 submitted for IEEE Std 1003.2-1992.

51 ⇒ **3.9.4.3 case Conditional Construct.** *Replace the synopsis of the case*  
 52 *statement with:*

```
53     case word in
54         [([pattern]|[pattern])... )  compound-list ; ; ]...
55         [([pattern]|[pattern])... )  compound-list]
56     esac
```

57 ⇒ **3.10.2 Shell Grammar Rules.** *Replace the rules for case\_clause, case\_-*  
 58 *list, and case\_item with:*

```
59     case_clause : Case WORD linebreak in linebreak case_list      Esac
60                | Case WORD linebreak in linebreak case_list_ns Esac
61                | Case WORD linebreak in linebreak                  Esac
62                ;
```

```

63     case_list_ns : case_list case_item_ns
64                 |
65                 ;
66     case_list   : case_list case_item
67                 |
68                 ;
69     case_item_ns : pattern ')' linebreak linebreak
70                 | pattern ')' compound_list linebreak
71                 | '(' pattern ')' linebreak linebreak
72                 | '(' pattern ')' compound_list linebreak
73                 ;
74     case_item   : pattern ')' linebreak DSEMI linebreak
75                 | pattern ')' compound_list DSEMI linebreak
76                 | '(' pattern ')' linebreak DSEMI linebreak
77                 | '(' pattern ')' compound_list DSEMI linebreak
78                 ;

```

79 **Rationale:** This change is required to match historical practice and is the result  
80 of interpretation request PASC 1003.2-92 #46 submitted for IEEE Std 1003.2-1992.  
81 The case construct in 3.9.4.3 was incorrectly described in 1992 as requiring a  
82 minimum of two compound lists, when the grammar and historical practice  
83 allowed the `case_item` to be omitted. The grammar did not allow the historical  
84 practice of omitting the final `;` (that was already described in 3.9.4.3).

### 85 3.13 Pattern Matching Notation

86 ⇒ **3.13 Pattern Matching Notation.** *At the end of the first paragraph, change*  
87 *“... the description of RE notation.” to:*

88 ... the description of RE notation, modified to include backslash escape  
89 processing.

90 **Rationale:** This change, and the following in 3.13.1, are required to match histor-  
91 ical practice and are the result of interpretation request PASC 1003.2-92 #21 sub-  
92 mitted for IEEE Std 1003.2-1992.

93 ⇒ **3.13.1 Patterns Matching a Single Character.** *At the end of the first para-*  
94 *graph, add:*

95 A `<backslash>` character shall escape the following character. The escaping  
96 `<backslash>` shall be discarded.

97 ⇒ **3.14.11 set – Set/unset options and positional parameters.** (This B  
 98 *change should be read only in conjunction with the following change.*) *Change* B  
 99 *the Synopsis to:* B

100 set [-abCefmnuvx] [-o option]... [argument ...] B  
 101 set [+abCefmnuvx] [+o option]... [argument ...] B  
 102 set -- [argument ...] B  
 103 set -o B  
 104 set +o B  
 105 *Obsolescent version:* B  
 106 set - [argument ...] B

107 ⇒ **3.14.11 set – Set/unset options and positional parameters.** *Add the fol-* B  
 108 *lowing after the description of the -n option:* B

109 -o Write the current settings of the options to standard output in B  
 110 an unspecified format. B  
 111 +o Write the current option settings to standard output in a for- B  
 112 mat that is suitable for reinput to the shell as commands that B  
 113 achieve the same options settings. B

114 ⇒ **3.13 set – Set/unset options and positional parameters.** *Change the* B  
 115 *description of the -x option to:* B

116 -x Write to standard error a trace for each command after the B  
 117 shell expands the command and before it executes it. It is B  
 118 unspecified whether the command that turns tracing off is B  
 119 traced. B

120 *Editor's Note: The following rationale will be added to E.3.14.11, but is kept here*  
 121 *with set for this draft:*

122 **set Rationale.** (This subclause is not a part of P1003.2b)

123 Historical implementations are inconsistent in the format used for -o option  
 124 status reporting. The +o format without an option-argument was added to allow  
 125 portable access to the options that can be saved and then later restored using, for  
 126 instance, a dot script.

127 Historically, sh did trace the command set +x, but ksh did not. B

128  
129

**Rationale:** The preceding three changes are the result of interpretation requests PASC 1003.2-92 #79 and #99 submitted for IEEE Std 1003.2-1992.

B  
B





## Section 4: Revisions to Execution Environment Utilities

### 1 4.1 awk – Pattern scanning and processing language

2 **Rationale:** The changes to `awk` are the result of interpretation requests PASC B  
3 1003.2-92 #91 and #107 submitted for IEEE Std 1003.2-1992. B

4 ⇒ **4.1.4 awk Operands.** *In the description of the assignment operand, change B  
5 the fourth and fifth sentences (the ones beginning “The variable shall be B  
6 assigned ... ” and “If that value is considered a numeric string ... ”) to B*

7 The variable shall be assigned the value of that `STRING` token and, if appropri- B  
8 ate, shall be considered a *numeric string* (see 4.1.7.2). B

9 ⇒ **4.1.5.1 awk Standard Input.** *Add to the end of the paragraph:* B

10 If the `awk` program contains no actions and no patterns, but is otherwise a B  
11 valid `awk` program, standard input and any *file* operands shall not be read and B  
12 `awk` shall exit with a return status of zero. B

13 ⇒ **4.1.7.1 awk Overall Program Structure.** *Change the second paragraph (the B  
14 one beginning “A missing pattern ... ”) to:* B

15 A missing pattern shall match any record of input, and a missing action shall B  
16 be equivalent to B

17 { print } B

18 If the `awk` program contains no actions and no patterns, but is otherwise a B  
19 valid `awk` program, standard input and any *file* operands shall not be read and B  
20 `awk` shall exit with a return status of zero. B

21 ⇒ **4.1.7.1 awk Overall Program Structure.** *Change the last paragraph to:* B

22 Execution of the `awk` program shall start by first executing the actions associ- B  
23 ated with all `BEGIN` patterns in the order they occur in the program. Then B  
24 each *file* operand (or standard input if no files were specified) shall be pro- B  
25 cesses in turn by reading data from the file until a record separator is seen B  
26 (<newline> by default). Before the first reference to a field in the record is C  
27 evaluated, the record shall be split into fields, according to the rules in 4.1.7.4, C  
28 using the value of `FS` that was current at the time the record was read. Each C  
29 pattern in the program then shall be evaluated in the order of occurrence, and B

30 the action associated with each pattern that matches the current record exe- B  
 31 cuted. The action for a matching pattern shall be executed before evaluating B  
 32 subsequent patterns. Finally, the actions associated with all END patterns B  
 33 shall be executed in the order they occur in the program. B

34 ⇒ **4.1.7.2 awk Expressions.** *Change the fourth paragraph (the one beginning B  
 35 “A string value shall be converted to a numeric value ...”) and the following B  
 36 dashed list to the following text. In the paragraph following the list, change “in B  
 37 the above steps” to “in the preceding description”.* B

38 A string value shall be considered a *numeric string* if it comes from one of the B  
 39 following: B

40 — Field variables B

41 — Input from the `getline` function B

42 — FILENAME B

43 — ARGV array elements B

44 — ENVIRON array elements B

45 — Array elements created by the `split` function C

46 — A command-line variable assignment B

47 — Variable assignment from another *numeric string* variable B

48 and after all the following conversions have been applied, the resulting string C  
 49 would lexically be recognized as a NUMBER token as described by the lexical C  
 50 conventions in 4.1.7.8: C

51 — All leading and trailing blanks are discarded C

52 — If the first non-`<blank>` character is + or -, it is discarded C

53 — Changing each occurrence of the decimal point character from the current C  
 54 locale to a period C

55 ⇒ **4.1.7.2 awk Expressions.** *Change the final paragraph to:* B

56 Comparisons (with the `<`, `<=`, `!=`, `==`, `>`, and `>=` operators) shall be made B  
 57 numerically if both operands are numeric, if one is numeric and the other has B  
 58 a string value that is a *numeric string*, or if one is numeric and the other has B  
 59 the *uninitialized value*. Otherwise, operands shall be converted to strings as B  
 60 required, and a string comparison shall be made using the locale-specific colla- B  
 61 tion sequence. The value of the comparison expression shall be 1 if the rela- B  
 62 tion is true, or zero if the relation is false. B

63 ⇒ **4.1.7.3 awk Variable and Special Variables.** *Change the first paragraph* B  
 64 *(which currently contains four lines of text across a page break) to:* B

65 Variables can be used in an `awk` program by referencing them. With the B  
 66 exception of function parameters (see 4.1.7.6.2.4), they are not explicitly B  
 67 declared. Function parameter names shall be local to the function; all other B  
 68 variable names shall be global. The same name shall not be used as both a B  
 69 function parameter name and as the name of a function or a special `awk` vari- B  
 70 able. The same name shall not be used both as a variable name with global B  
 71 scope and as the name of a function. The same name shall not be used within B  
 72 the same scope both as a scalar variable and as an array. Uninitialized vari- B  
 73 ables, including scalar variables, array elements, and field variables shall have B  
 74 an *uninitialized value*. An *uninitialized value* shall have both a numeric value B  
 75 of zero and a string value of the empty string. Evaluation of variables with an B  
 76 *uninitialized value*, to either string or numeric, shall be determined by the B  
 77 context in which they are used. B

78 ⇒ **4.1.7.3 awk Variable and Special Variables.** *Change the second paragraph* B  
 79 *(the one beginning “Field variables shall be designated . . .”) to:* B

80 Field variables shall be designated by a `$` followed by a number or numeric B  
 81 expression. The effect of the field number *expression* evaluating to anything B  
 82 other than a nonnegative integer is unspecified; uninitialized variables or B  
 83 string values need not be converted to numeric values in this context. New B  
 84 field variables can be created by assigning a value to them. References to B  
 85 nonexistent fields (i.e., fields after `$NF`), shall evaluate to the *uninitialized* B  
 86 *value*. Such references shall not create new fields. However, assigning to a B  
 87 nonexistent field [e.g., `$(NF+2) = 5`] shall increase the value of `NF`; create any B  
 88 intervening fields with the *uninitialized value*; and cause the value of `$0` to be B  
 89 recomputed, with the fields being separated by the value of `OFS`. Each field B  
 90 variable shall have a string value or an *uninitialized value* when created. B  
 91 Field variables shall have the *uninitialized value* when created from `$0` using B  
 92 `FS` and the variable does not contain any characters. If appropriate, the field B  
 93 variable shall be considered a *numeric string* (see 4.1.7.2). B

94 ⇒ **4.1.7.3 awk Variable and Special Variables.** *In the first paragraph of the* B  
 95 `ENVIRON` *description, change the sentence “If the value of an environment vari-* B  
 96 *able is considered a numeric string . . .” to* B

97 If appropriate, the environment variable shall be considered a *numeric string* B  
 98 (see 4.1.7.2). B

- 99 ⇒ **4.1.7.3 awk Variable and Special Variables.** *Change the description of* OFS *B*  
 100 *to:* *B*
- 101 OFS The print statement output field separator; <space> by default. B
- 102 ⇒ **4.1.7.4 awk Regular Expressions.** *Change the final sentence in the first* B  
 103 *paragraph to:* B
- 104 Using a slash character within an ERE token requires the escaping shown in B  
 105 Table 4-2. B
- 106 ⇒ **4.1.7.4 awk Regular Expressions.** *Add a new item (1) to the numbered list,* B  
 107 *changing the existing items to (2) and (3):* B
- 108 (1) If FS is a null string, the behavior is unspecified. B
- 109 ⇒ **4.1.7.4 awk Regular Expressions.** *Change the first paragraph that follows* B  
 110 *the numbered list (which begins “Except in the gsub, ...”) to:* B
- 111 Except for the ~ and !~ operators, and in the gsub, match, split, and sub B  
 112 built-in functions, ERE matching shall be based on input records; i.e., record B  
 113 separator characters (the first character of the value of the variable RS, <new- B  
 114 line> by default) cannot be embedded in the expression, and no expression B  
 115 shall match the record separator character. If the record separator is not B  
 116 <newline>, <newline> characters embedded in the expression can be B  
 117 matched. For the ~ and !~ operators, and in those four built-in functions, ERE B  
 118 matching shall be based on text strings; i.e., any character (including <new- B  
 119 line> and the record separator) can be embedded in the pattern, and an B  
 120 appropriate pattern shall match any character. However, in all awk ERE B  
 121 matching, the use of one or more NUL characters in the pattern, input record, B  
 122 or text string produces undefined results. B
- 123 ⇒ **4.1.7.6.1 awk Output Statements.** *Change the first sentence of the second* B  
 124 *paragraph to:* B
- 125 In all cases, the *expression* shall be evaluated to produce a string that is used B  
 126 as a pathname into which to write (for > or >>) or as a command to be executed B  
 127 (for |). B

128	⇒ <b>4.1.7.6.2.1 awk Arithmetic Functions.</b> <i>Change the description of atan2 to:</i>	B
129	atan2( <i>y</i> , <i>x</i> )	Return arctangent of <i>y/x</i> in radians in the range $-\pi$ to
130		$\pi$ .
131	⇒ <b>4.1.7.6.2.2 awk String Functions.</b> <i>Change the description of split to:</i>	B
132	split( <i>s</i> , <i>a</i> [, <i>fs</i> ])	Split the string <i>s</i> into array elements <i>a</i> [1], <i>a</i> [2], ... ,
133		<i>a</i> [ <i>n</i> ], and return <i>n</i> . All elements of the array shall be
134		deleted before the split is performed. The separation
135		shall be done with the ERE <i>fs</i> or with the field separa-
136		tor FS if <i>fs</i> is not given. Each array element shall have
137		a string value when created and, if appropriate, the
138		array element shall be considered a <i>numeric string</i> (see
139		4.1.7.2). The effect of a null string as the value of <i>fs</i> is
140		unspecified.
141	⇒ <b>4.1.7.6.2.2 awk String Functions.</b> <i>Change the description of sub to:</i>	B
142	sub( <i>ERE</i> , <i>repl</i> [, <i>in</i> ])	B
143		Substitute the string <i>repl</i> in place of the first instance
144		of the extended regular expression <i>ERE</i> in string <i>in</i>
145		and return the number of substitutions. An amper-
146		sand (&) appearing in the string <i>repl</i> shall be replaced
147		by the string from <i>in</i> that matches the ERE. An
148		ampersand preceded with a backslash (\) shall be
149		interpreted as the literal ampersand character. Any
150		other occurrence of a backslash (e.g., preceding any
151		other character) shall be treated as a literal backslash
152		character. [Note that if <i>repl</i> is a string literal (the lex-
153		ical token STRING, see 4.1.7.8), the handling of the
154		ampersand character occurs after any lexical process-
155		ing, including any lexical backslash escape sequence
156		processing.] If <i>in</i> is specified and it is not an <i>lvalue</i>
157		(see 4.1.7.2), the behavior is undefined. If <i>in</i> is omit-
158		ted, awk shall use the current record (§0) in its place.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 159 ⇒ **4.1.7.6.2.2 awk String Functions.** *Change the description of `substr` to:* B
- 160       `substr(s, m[, n])` B
- 161               Return the at most *n*-character substring of *s* that B
- 162               begins at position *m*, numbering from 1. If *n* is miss- B
- 163               ing, or if *n* specifies more characters than are left in B
- 164               the string, the length of the substring shall be limited B
- 165               by the length of the string *s*. B
- 166 ⇒ **4.1.7.6.2.3 awk Input/Output and General Functions.** *In the description B*
- 167       *of expression | `getline [var]`, change “file” to “stream”.* B
- 168 ⇒ **4.1.7.6.2.3 awk Input/Output and General Functions.** *Change the B*
- 169       *description of `getline var` to:* B
- 170       `getline var`       Set variable *var* to the next input record from the B
- 171               current input file and, if appropriate, *var* shall be con- B
- 172               sidered a *numeric string* (see 4.1.7.2). This form of B
- 173               `getline` shall set the `FNR` and `NR` variables. B
- 174 ⇒ **4.1.7.6.2.3 awk Input/Output and General Functions.** *In the first para- B*
- 175       *graph of the description of expression | `getline [var]`, change the last sen- B*
- 176       *tence (the one beginning with “If *var* is missing ...”) to:* B
- 177       If *var* is missing, `$0` and `NF` shall be set; otherwise, *var* shall be set and, if B
- 178       appropriate, it shall be considered a *numeric string* (see 4.1.7.2). B
- 179 ⇒ **4.1.7.6.2.3 awk Input/Output and General Functions.** *In the description B*
- 180       *of `getline [var] < expression`, change “full pathname” to “pathname”.* B
- 181 ⇒ **4.1.7.6.2.3 awk Input/Output and General Functions.** *In the first para- B*
- 182       *graph of the description of `getline [var] < expression`, change the last sen- B*
- 183       *tence (the one beginning with “If *var* is missing ...”) to:* B
- 184       If *var* is missing, `$0` and `NF` shall be set; otherwise, *var* shall be set and, if B
- 185       appropriate, it shall be considered a *numeric string* (see 4.1.7.2). B

- 186 ⇒ **4.1.7.6.2.4 awk User-Defined Functions.** *Change the first paragraph to:* B
- 187 The awk language provides for user-defined functions. Such functions can be B  
188 defined as B
- 189 `function name([parameter,...]) { statements }` C
- 190 ⇒ **4.1.7.6.2.4 awk User-Defined Functions.** *Change the third paragraph (the B  
191 one beginning "Function arguments ... ") to:* B
- 192 Function parameters, if present, can be either scalars or arrays; the behavior B  
193 is undefined if an array name is passed as a parameter that the function uses B  
194 as a scalar, or if a scalar expression is passed as a parameter that the function B  
195 uses as an array. Function parameters shall be passed by value if scalar and B  
196 by reference if array name. B
- 197 ⇒ **4.1.7.6.2.4 awk User-Defined Functions.** *In the fourth paragraph, change B  
198 the third sentence (the one beginning "If fewer arguments are supplied ... ") to:* B
- 199 If fewer arguments are supplied in a function call than are in the function B  
200 definition, the extra parameters that are used in the function body as scalars B  
201 shall evaluate to the *uninitialized value* until they are otherwise initialized, B  
202 and the extra parameters that are used in the function body as arrays shall be B  
203 treated as uninitialized arrays where each element evaluates to the *uninitial-* B  
204 *ized value* until otherwise initialized. B
- 205 ⇒ **4.1.7.8 awk Lexical Conventions.** *In item (6), change the fifth sentence from B  
206 "An ERE constant shall be terminated by the first unescaped occurrence of the B  
207 slash character after the one that begins the string constant." to:* B
- 208 An ERE constant shall be terminated by the first unescaped occurrence of the B  
209 slash character after the one that begins the ERE constant. B
- 210 *Editor's Note: The following rationale will be added to E.4.1, but is kept here with B  
211 awk for this draft:* B
- 212 **awk Rationale.** *(This subclause is not a part of P1003.2b)* B
- 213 In `sub` and `gsub`, if `repl` is a string literal (the lexical token `STRING`, see 4.1.7.8), B  
214 then two consecutive backslash characters should be used in the string to ensure B  
215 a single backslash will precede the ampersand when the resultant string is B  
216 passed to the function. [For example, to specify one literal ampersand in the B  
217 replacement string, use `gsub(ERE, "\\&")`.] B
- 218 Historically the only special character in the `repl` argument of `sub` and `gsub` B  
219 string functions was the ampersand (&) character and preceding it with the B  
220 backslash character was used to turn off its special meaning. B
- 221 The description in the 1992 standard introduced behavior such that the backslash B  
222 character was another special character and it was unspecified whether there B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

223 were any other special characters. This description introduced several portability B  
 224 problems, some of which are described below, and so it has been replaced with the B  
 225 more historical description. Some of the problems include: B

226 — Historically, to create the replacement string, a script could use B  
 227 `gsub(ERE, "\\&")`, but with the 1992 wording, it was necessary to use B  
 228 `gsub(ERE, "\\&&")`. Backslash characters are doubled here because all B  
 229 string literals are subject to lexical analysis, which would reduce each pair B  
 230 of backslash characters to a single backslash before being passed to `gsub`. B

231 — Since it was unspecified what the special characters were, for portable B  
 232 scripts to guarantee that characters are printed literally, each character B  
 233 had to be preceded with a backslash. (For examples, a portable script had B  
 234 to use `gsub(ERE, "\\h\\i")` to produce a replacement string of `hi`.) B

235 The description for comparisons in the 1992 version of 4.1.7.2 did not properly B  
 236 describe historical practice because of the way numeric strings are compared as B  
 237 numbers. The current rules cause the following code: B

```
238     if (0 == "000")           B
239         print "strange, but true" B
240     else                       B
241         print "not true"      B
```

242 to do a numeric comparison, causing the `if` to succeed. It should be intuitively B  
 243 obvious that this is incorrect behavior, and indeed, no historical implementation B  
 244 of `awk` actually behaves this way. B

245 To fix this problem, the definition of *numeric string* was enhanced to include only B  
 246 those values obtained from specific circumstances (mostly external sources) where B  
 247 it is not possible to determine unambiguously whether the value is intended to be B  
 248 a string or a numeric. B

249 Variables that are assigned to a *numeric string* shall also be treated as a *numeric* B  
 250 *string*. (For example, the notion of a *numeric string* can be propagated across B  
 251 assignments.) In comparisons, all variables having the *uninitialized value* are to B  
 252 be treated as a numeric operand evaluating to the numeric value zero. B

253 Uninitialized variables includes all types of variables including scalars, array ele- B  
 254 ments, and fields. The definition of an *uninitialized value* in 4.1.7.3 is necessary B  
 255 to describe the value placed on uninitialized variables and on fields that are valid B  
 256 (e.g., `< $NF`) but have no characters in them and to describe how these variables B  
 257 are to be used in comparisons. A valid field, such as `$1`, that has no characters in B  
 258 it can be obtained by from an input line of `"\t\t"` when `FS="\t"`. Historically, B  
 259 the comparison (`$1 < 10`) was done numerically after evaluating `$1` to the value B  
 260 zero. B

261 The phrase "... also shall have the numeric value of the *numeric string*" was B  
 262 removed from several sections of the 1992 version because they specify an B  
 263 unnecessary implementation detail. It is not necessary for this standard to B  
 264 specify that these objects be assigned two different values. It is only necessary to B  
 265 specify that these objects may evaluate to two different values depending on con- B  
 266 text. B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



267 The description of numeric string processing is based on the behavior of the *atof()* C  
 268 function in the C Standard {7}. While it is not a requirement for an implementa- C  
 269 tion to use this function, many historical implementations of *awk* do. In the C  
 270 C Standard {7}, floating point constants use a period as a decimal point character C  
 271 for the language itself, independent of the current locale, but the *atof()* function C  
 272 and the associated *strtod()* function use the decimal point character of the current C  
 273 locale when converting strings to numeric values. Similarly in *awk*, floating point C  
 274 constants in an *awk* script use a period independent of the locale, but input C  
 275 strings use the decimal point character of the locale. C

### 276 4.3 *bc* – Arbitrary-precision arithmetic language B

277 ⇒ 4.3.7.1 *bc* Operations. Change the paragraph with the numbered list (the B  
 278 one beginning “For all values of *obase* . . . ”) to: B

279 For all values of *obase* specified by this standard, *bc* shall output numeric B  
 280 values by performing each of the following steps in order: B

281 (1) If the value is less than zero, *bc* shall write a hyphen (–) character. B

282 (2) Depending on the numeric value, *bc* shall write one of the following: B

283 — If the absolute value of the numeric value is greater than or equal to B  
 284 one, *bc* shall write the integer portion of the value as a series of digits B  
 285 appropriate to *obase* (as described below), most significant digit first. B  
 286 It shall write the most significant nonzero digit next, followed by each B  
 287 successively less significant digit. B

288 — If the absolute value of the numeric value is less than one but greater B  
 289 than zero and the scale of the numeric value is greater than zero, it is B  
 290 unspecified whether *bc* writes the character 0. B

291 — If the numeric value is zero, *bc* shall write the character 0. B

292 (3) If the scale of the value is greater than zero and the numeric value is not B  
 293 zero, *bc* shall write a period character, followed by a series of digits B  
 294 appropriate to *obase* (as described below) representing the most B  
 295 significant portion of the fractional part of the value. If *s* represents the B  
 296 scale of the value being written, the number of digits written shall be *s* if B  
 297 *obase* is 10, less than or equal to *s* if *obase* is greater than 10, or greater B  
 298 than or equal to *s* if *obase* is less than 10. For *obase* values other than B  
 299 10, this should be the number of digits needed to represent a precision of B  
 300  $10^s$ . B

301 ⇒ **4.3.7.1 bc Operations.** *Change the paragraph describing the return state-* B  
 302 *ments (the fourth last paragraph in the subclause) to:* B

303 The return statements [`return` and `return(expression)`] shall cause termi- B  
 304 nation of a function, popping of its auto variables, and specification of the B  
 305 result of the function. The first form shall be equivalent to `return(0)`. The B  
 306 value and scale of the result returned by the function shall be the value and B  
 307 scale of the expression returned. B

308 ⇒ **4.3.7.1 bc Operations.** *Change the last paragraph in the subclause (the one* B  
 309 *beginning “The scale of an invocation ...”) to:* B

310 The scale of the result returned by these functions shall be the value of the B  
 311 scale register at the time the function is invoked. The value of the scale B  
 312 register after these functions have completed their execution shall be the same B  
 313 value it had upon invocation. The behavior is undefined if any of these func- B  
 314 tions is invoked with an argument outside the domain of the mathematical B  
 315 function. B

316 **Rationale:** The preceding three changes are the result of interpretation request B  
 317 PASC 1003.2-92 #96 submitted for IEEE Std 1003.2-1992. B

318 ⇒ **4.3.7.2 bc Grammar.** *Change the definition of `argument_list` to:* B

```
319     argument_list : expression                B
320                   | argument_list ',' expression B
321                   | LETTER '[' ']' ',' argument_list C
322                   ;                          B
```

323 **Rationale:** The preceding change is the result of interpretation request PASC B  
 324 1003.2-92 #101 submitted for IEEE Std 1003.2-1992. B

325 *Editor’s Note: The following rationale will be added to E.4.3, but is kept here with* B  
 326 *bc for this draft:* B

327 **bc Rationale.** *(This subclause is not a part of P1003.2b)* B

328 Historical implementations of `bc` did not allow array parameters to be passed as B  
 329 the last parameter to a function. New implementations are encouraged to remove B  
 330 this restriction even though it is not required by the grammar. B

331 **4.5 cd – Change working directory**

332 *Editor's Note: Virtually all of this clause has been changed in Draft 11. To avoid* B  
 333 *clutter, it is not further diffmarked.* B

334 ⇒ **4.5.1 cd Synopsis.** *Modify the Synopsis to be:*

335 `cd [-L] [-P] [directory]` C

336 ⇒ **4.5.2 cd Description.** *Change the entire subclause to:*

337 The `cd` utility shall change the working directory of the current shell execution  
 338 environment (see 3.12) by executing the following steps in sequence. (In the C  
 339 following steps, the symbol *curpath* represents an intermediate value used to C  
 340 simplify the description of the algorithm used by `cd`. There is no requirement C  
 341 that *curpath* be made visible to the application.) C

342 (1) If no *directory* operand is given and the **HOME** environment variable is  
 343 empty or undefined, the default behavior is implementation defined and  
 344 no further steps shall be taken.

345 (2) If no *directory* operand is given and the **HOME** environment variable is  
 346 set to a nonempty value, the `cd` utility shall behave as if the directory  
 347 named in the **HOME** environment variable was specified as the *directory*  
 348 operand.

349 (3) If the operand begins with a slash, *curpath* shall be set to the operand. If C  
 350 the first component is dot or dot-dot, *curpath* shall be set to the **PWD** C  
 351 environment variable with a slash character and the operand appended. C  
 352 Otherwise, *curpath* shall be set as affected by the **CDPATH** environment C  
 353 variable. The `cd` utility shall construct a directory name to store in *cur-* C  
 354 *path* by appending a slash and the operand to each directory named in C  
 355 the **CDPATH** variable, in the order listed. The resulting value of *curpath* C  
 356 shall be the first of these strings that is a directory. If none of the result- C  
 357 ing strings represented a directory, *curpath* shall be set to the equivalent C  
 358 of the **PWD** environment variable with a slash character and the operand C  
 359 appended. C

360 (4) If *curpath* is being handled dot-dot physically, the `cd` utility shall per- C  
 361 form actions equivalent to the POSIX.1 {8} *chdir()* function, called with C  
 362 *curpath* as the *path* argument. If these actions succeed, the **PWD** C  
 363 environment variable shall be set to an absolute pathname for the C  
 364 current working directory and shall not contain filename components C  
 365 that, in the context of pathname resolution, refer to a file of type symbolic C  
 366 link. If there is insufficient permission on the new directory, or on any C  
 367 parent of that directory, to determine the current working directory, it is C  
 368 unspecified to what the **PWDenvironment** variable shall be set. If the C  
 369 actions equivalent to *chdir()* fail for any reason, the `cd` utility shall C  
 370 display an appropriate error message and not alter the **PWD** environ- C  
 371 ment variable. In either case, no further steps shall be taken. C

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 372 (5) The *curpath* value shall then be converted to canonical form as follows,  
 373 considering each component from beginning to end, in sequence:
- 374 (a) Dot components and any slashes that separate them from the next C  
 375 component shall be deleted. C
- 376 (b) For each dot-dot component, if there is a preceding component and C  
 377 it is neither root nor dot-dot, the preceding component, all slashes C  
 378 separating the preceding component from dot-dot, dot-dot, and all C  
 379 slashes separating dot-dot from the following component shall be C  
 380 deleted. C
- 381 (c) An implementation may further simplify *curpath* by removing any C  
 382 trailing slash characters that are not also leading slashes, replacing C  
 383 multiple non-leading consecutive slashes with a single slash, and C  
 384 replacing three or more leading slashes with a single slash. C
- 385 If as a result of this canonicalization the *curpath* variable is null, no C  
 386 further steps shall be taken. C
- 387 (6) The `cd` utility shall then perform actions equivalent to the POSIX.1 {8}  
 388 *chdir()* function called with *curpath* as the *path* argument. If these C  
 389 actions failed for any reason, the `cd` utility shall display an appropriate C  
 390 error message and no further steps shall be taken. The **PWD** environ- C  
 391 ment variable shall be set to *curpath*.

392 ⇒ **4.5.3 cd Options.** *Change the entire subclause to:*

393 The `cd` utility shall conform to the utility argument syntax guidelines  
 394 described in 2.10.2.

395 The following options shall be supported by the implementation:

- 396 `-L` Handle the operand dot-dot logically; see 4.5.2. C
- 397 `-P` Handle the operand dot-dot physically, resolving any filename C  
 398 components that refer to symbolic links. C

399 If both `-L` and `-P` options are specified, the last of these options shall be used C  
 400 and all others ignored. If neither `-L` nor `-P` is specified, the operand shall be C  
 401 handled dot-dot logically; see 4.5.2. C

402 ⇒ **4.5.4 cd Operands.** *Change the directory entry with:*

403 *directory* An absolute or relative pathname of the directory that shall  
 404 become the new working directory. The interpretation of a  
 405 relative pathname by `cd` depends on the `-L` option and the  
 406 **CDPATH** and **PWD** environment variables. If *directory* is `-`,  
 407 the results are implementation defined. If *directory* is an C  
 408 empty string, the results are unspecified. C

409 ⇒ **4.5.5.3 cd Environment Variables.** *Change the CDPATH entry to:*

410           **CDPATH**           This variable shall consist of a colon-separated list of  
411 pathnames that refer to directories. The `cd` utility  
412 shall use this list in its attempt to change the direc-  
413 tory, as described in 4.5.2. An empty string in place of  
414 a directory pathname represents the current directory.  
415 If **CDPATH** is not set, it shall be treated as if it were  
416 an empty string.

417 ⇒ **4.5.5.3 cd Environment Variables.** *Add the following entry in the proper*  
418 *sorted order:*

419           **PWD**           This variable shall be set as specified in 4.5.2. If an   C  
420 application sets or unsets the value of **PWD**, the   C  
421 behavior of `cd` is unspecified.                           C

422 *Editor's Note: The following rationale will be added to E.4.5, but is kept here with*  
423 *cd for this draft:*

424 **cd Rationale.** *(This subclause is not a part of P1003.2b)*

425 Some historical shells, such as the KornShell, took special actions when the direc-  
426 tory name contained a dot-dot component, selecting the logical parent of the direc-  
427 tory, rather than the actual parent directory; i.e., it moved up one level toward  
428 the / in the pathname, remembering what the user typed, rather than performing  
429 the equivalent of the POSIX.1 {8} call

```
430       chdir("../");
```

431 In such a shell, the following commands would not necessarily produce equivalent  
432 output for all directories:

```
433       cd .. && ls                       ls ..
```

434 This behavior is not permitted by default because it is not consistent with the  
435 definition of dot-dot in most historical practice; i.e., while this behavior has been  
436 optionally available in the KornShell, other shells have historically not supported  
437 this functionality.

438 The logical pathname is stored in the **PWD** environment variable when the `cd`  
439 utility completes and this value is used to construct the next directory name if `cd`  
440 is invoked with the `-L` option.

## 441 4.6 chgrp – Change file group ownership

442 ⇒ **4.6.1 chgrp Synopsis.** *Modify the Synopsis to be:*

443 `chgrp [ -R [ -H | -L ] ] [-h] group file ...` B

444 ⇒ **4.6.3 chgrp Options.** *Change the entire subclause to:*

445 The `chgrp` utility shall conform to the utility argument syntax guidelines  
446 described in 2.10.2.

447 The following options shall be supported by the implementation:

448     -h           If the system supports group IDs for symbolic links, for each  
449                   *file* operand that names a file of type symbolic link, `chgrp`  
450                   shall attempt to set the group ID of the symbolic link instead  
451                   of the file referenced by the symbolic link. If the system does  
452                   not support group IDs for symbolic links, for each *file* operand  
453                   that names a file of type symbolic link, `chgrp` shall do nothing  
454                   more with the current file and shall go on to any remaining  
455                   files. B

456     -H           If the `-R` option is specified and a symbolic link referencing a B  
457                   file of type directory is specified on the command line, `chgrp`  
458                   shall change the group of the directory referenced by the sym-  
459                   bolic link and all files in the file hierarchy below it.

460     -L           If the `-R` option is specified and a symbolic link referencing a B  
461                   file of type directory is specified on the command line or B  
462                   encountered during the traversal of a file hierarchy, `chgrp`  
463                   shall change the group of the directory referenced by the sym-  
464                   bolic link and all files in the file hierarchy below it.

465     -R           Recursively change file group IDs. For each *file* operand that  
466                   names a directory, `chgrp` shall change the group of the direc-  
467                   tory and all files in the file hierarchy below it. When a sym-  
468                   bolic link is specified on the command line or encountered dur-  
469                   ing the traversal of a file hierarchy, `chgrp` shall change the  
470                   group ID of the symbolic link if the system supports this opera-  
471                   tion. Unless the `-H` or `-L` options are specified, the `chgrp` util- B  
472                   ity shall not follow the symbolic link to any other part of the  
473                   file hierarchy.

474     Specifying more than one of the mutually exclusive options `-H` and `-L` shall not B  
475     be considered an error. The last option specified shall determine the behavior B  
476     of the utility. B

477 **4.7 chmod – Change file modes**478 ⇒ **4.7.1 chmod Synopsis.** *Modify the Synopsis to be:*479 `chmod [ -R [ -H | -L ] ] [-h] mode file ...` B480 ⇒ **4.7.3 chmod Options.** *Change the entire subclause to:*481 The `chmod` utility shall conform to the utility argument syntax guidelines  
482 described in 2.10.2.

483 The following options shall be supported by the implementation:

484 `-h` If the system supports permissions for symbolic links, for each  
485 *file* operand that names a file of type symbolic link, `chmod`  
486 shall attempt to set the permissions of the symbolic link  
487 instead of the file referenced by the symbolic link. If the sys-  
488 tem does not support permissions for symbolic links, for each  
489 *file* operand that names a file of type symbolic link, `chmod`  
490 shall do nothing more with the current file and shall go on to  
491 any remaining files. B

492 `-H` If the `-R` option is specified and a symbolic link referencing a B  
493 file of type directory is specified on the command line, `chmod` B  
494 shall change the file mode bits of the directory referenced by  
495 the symbolic link and all files in the file hierarchy below it.

496 `-L` If the `-R` option is specified and a symbolic link referencing a B  
497 file of type directory is specified on the command line or B  
498 encountered during the traversal of a file hierarchy, `chmod`  
499 shall change the file mode bits of the directory referenced by  
500 the symbolic link and all files in the file hierarchy below it.

501 `-R` Recursively change file mode bits. For each *file* operand that  
502 names a directory, `chmod` shall change the file mode bits of the  
503 directory and all files in the file hierarchy below it. When a  
504 symbolic link is specified on the command line or encountered  
505 during the traversal of a file hierarchy, `chmod` shall change  
506 the file mode bits of the symbolic link if the system supports  
507 this operation. Unless the `-H` or `-L` options are specified, the B  
508 `chmod` utility shall not follow the symbolic link to any other  
509 part of the file hierarchy.

510 Specifying more than one of the mutually exclusive options `-H` and `-L` shall not B  
511 be considered an error. The last option specified shall determine the behavior B  
512 of the utility. B

## 513 4.8 chown – Change file ownership

514 ⇒ **4.8.1 chown Synopsis.** *Modify the Synopsis to be:*

515 `chown [ -R [ -H | -L ] ] [-h] owner[:group] file...` B

516 ⇒ **4.8.3 chown Options.** *Change the entire subclause to:*

517 The `chown` utility shall conform to the utility argument syntax guidelines  
518 described in 2.10.2.

519 The following options shall be supported by the implementation:

520	-h	If the system supports user IDs for symbolic links, for each <i>file</i>	B
521		operand that names a file of type symbolic link, <code>chown</code> shall	B
522		attempt to set the user ID of the symbolic link. If the system	B
523		supports group IDs for symbolic links, and a group ID was	B
524		specified, for each <i>file</i> operand that names a file of type sym-	B
525		bolic link, <code>chown</code> shall attempt to set the group ID of the sym-	B
526		bolic link. By default, <code>chown</code> shall not attempt to set the user	B
527		ID or group ID of the file referenced by the symbolic link. If	B
528		the system does not support user or group IDs for symbolic	B
529		links, for each <i>file</i> operand that names a file of type symbolic	B
530		link, <code>chown</code> shall do nothing more with the current file and	
531		shall go on to any remaining files.	B
532	-H	If the <code>-R</code> option is specified and a symbolic link referencing a	B
533		file of type directory is specified on the command line, <code>chown</code>	B
534		shall change the user ID (and group ID, if specified) of the	B
535		directory referenced by the symbolic link and all files in the	
536		file hierarchy below it.	
537	-L	If the <code>-R</code> option is specified and a symbolic link referencing a	B
538		file of type directory is specified on the command line or	B
539		encountered during the traversal of a file hierarchy, <code>chown</code>	B
540		shall change the user ID (and group ID, if specified) of the	B
541		directory referenced by the symbolic link and all files in the	
542		file hierarchy below it.	
543	-R	Recursively change file user and group IDs. For each <i>file</i>	B
544		operand that names a directory, <code>chown</code> shall change the user	
545		ID (and group ID, if specified) of the directory and all files in	B
546		the file hierarchy below it. When a symbolic link is specified	
547		on the command line or encountered during the traversal of a	
548		file hierarchy, <code>chown</code> shall change the user ID (and group ID, if	B
549		specified) of the symbolic link if the system supports this	B
550		operation. Unless the <code>-H</code> or <code>-L</code> options are specified, the	B
551		<code>chown</code> utility shall not follow the symbolic link to any other	
552		part of the file hierarchy.	

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



553        Specifying more than one of the mutually exclusive options `-H` and `-L` shall not    B  
 554        be considered an error. The last option specified shall determine the behavior    B  
 555        of the utility.    B

## 556    **4.13 cp – Copy files**

557    ⇒ **4.13.1 cp Synopsis.** *Modify the Synopsis to be:*

558        `cp [-fip] source_file target_file`

559        `cp [-fip] source_file ... target`

560        `cp -R [-H | -L] [-fip] source_file ... target`

561        `cp -r [-H | -L] [-fip] source_file ... target`

562    ⇒ **4.13.2 cp Description.** *Change the second sentence of the first paragraph to:*

563        The `cp` utility shall copy the contents of *source\_file* (or, if *source\_file* is a file of  
 564        type symbolic link, the contents of the file referenced by *source\_file*) to the des-  
 565        tination path named by *target\_file*.

566    ⇒ **4.13.2 cp Description.** *Change the last sentence of the second paragraph to:*

567        The `cp` utility shall copy the contents of each *source\_file* (or, if *source\_file* is a  
 568        file of type symbolic link, the contents of the file referenced by *source\_file*) to  
 569        the destination path named by the concatenation of *target*, a slash character,  
 570        and the last component of *source\_file*.

571    ⇒ **4.13.2 cp Description.** *Change the seventh paragraph to:*

572        In the following description, the term *dest\_file* refers to the file named by the  
 573        destination path. The term *source\_file* refers to the file that is being copied,  
 574        whether specified as an operand or a file in a file hierarchy rooted in a  
 575        *source\_file* operand. If *source\_file* is a file of type symbolic link:

576        — If neither the `-R` nor `-r` options were specified, `cp` shall take actions based  
 577        on the type and contents of the file referenced by the link, and not by the  
 578        link itself.

579        — If the `-R` option was specified:

580        — If neither the `-H` nor `-L` options were specified, `cp` shall take actions  
 581        based on the file being of type symbolic link.    B

582        — If the `-H` option was specified, `cp` shall take actions based on the type  
 583        and contents of the file referenced by any symbolic link specified as a  
 584        *source\_file* operand.

- 585           — If the `-L` option was specified, `cp` shall take actions based on the type  
 586           and contents of the file referenced by any symbolic link specified as a  
 587           `source_file` operand or any symbolic links encountered during traversal  
 588           of a file hierarchy.
- 589           — If the `-r` option was specified, the behavior is implementation defined.

590   ⇒ **4.13.2 `cp` Description.** *In item (4b), add a subitem [3] at the end:*

- 591           [3] If `source_file` is a file of type symbolic link, the pathname contained in    B  
 592           `dest_file` shall be the same as the pathname contained in `source_file`.    B
- 593           If this fails for any reason, `cp` shall write a diagnostic message to  
 594           standard error, do nothing more with `source_file`, and go on to any  
 595           remaining files.

596   ⇒ **4.13.3 `cp` Options.** *Add the following options in the proper sorted order:*

- 597           `-H`           Take actions based on the type and contents of the file refer-  
 598           enced by any symbolic link specified as a `source_file` operand.
- 599           `-L`           Take actions based on the type and contents of the file refer-  
 600           enced by any symbolic link specified as a `source_file` operand  
 601           or any symbolic links encountered during traversal of a file  
 602           hierarchy.

603   ⇒ **4.13.3 `cp` Options.** *Add the following paragraph to the end of the subclause:*    B

- 604           Specifying more than one of the mutually exclusive options `-H` and `-L` shall not    B  
 605           be considered an error. The last option specified shall determine the behavior    B  
 606           of the utility.    B

607 **4.14 cut – Cut out selected fields of each line of a file**

608 ⇒ **4.14.3 cut Options.** *Change the last sentence of the second paragraph from*  
 609 *“The elements in list can be ... in any order.” to:*

610 The elements in list can be repeated, can overlap, and can be specified in any  
 611 order, but the bytes, characters, or fields selected shall be written in the order  
 612 of the input data. If an element appears in the selection list more than once, it  
 613 shall be written exactly once.

614 **Rationale:** This change is in response to P1003.2-N149. It represents historical  
 615 practice on all known systems. The original standard was ambiguous on the  
 616 nature of the output. Add the following example to E.4.14:

617 The *list* option-arguments are historically used to select the portions of the line to  
 618 be written, but do not affect the order of the data. For example,

619 `echo abcdefghi | cut -c6,2,4-7,1`

620 yields `abdefg`.

621 A proposal to enhance `cut` with the following option:

622     `-o`            Preserve the selected field order. When this option is specified,  
 623                    each byte, character, or field (or ranges of such) shall be written  
 624                    in the order specified by the *list* option-argument, even if this  
 625                    requires multiple outputs of the same bytes, characters, or fields.

626 was rejected because this type of enhancement is outside the scope of the  
 627 P1003.2b amendment.

628 **4.15 date – Write the date and time**

629 ⇒ **4.15.4.2 date Modified Field Descriptors.** *Add the following list item fol-*  
 630 *lowing %Ex:*

631       %EX    Alternate time representation of the locale.

632 **Rationale:** This change was to correct an oversight in ISO/IEC 9945-2:1993,  
 633 pointed out by Japan. It is identical to an extension in XPG4 {B49}.

634 **4.16 dd – Convert and copy a file**

635 ⇒ **4.16.2 dd Description.** *Change processing order step (4) to:*

636       (4) If the `swab` conversion is specified, each pair of input data bytes shall be  
 637       swapped. If there are an odd number of bytes in the input block, the last  
 638       byte in the input record shall not be swapped.

639 **Rationale:** This change is required to match historical practice and is the result  
 640 of interpretation requests PASC 1003.2-92 #03 and PASC 1003.2-92 #04 submitted  
 641 for IEEE Std 1003.2-1992.

642 ⇒ **4.16.5.4 dd Asynchronous Events.** *Change the entire subclause to:*

643       For SIGINT, the `dd` utility shall interrupt its current processing, write status  
 644       information to standard error, and exit as though terminated by SIGINT. It  
 645       shall take the standard action for all other signals; see 2.11.5.4.

646 **Rationale:** This change is required to match historical practice and is the result  
 647 of interpretation request PASC 1003.2-92 #06 submitted for IEEE Std 1003.2-1992.

648 **4.17 diff – Compare two files** B

649 *Editor's Note: This clause is new in Draft 11. To avoid clutter, it is not further* B  
 650 *diffmarked.* B

651 ⇒ **4.17.3 diff Options.** *Change the description of -b to:*

652        -b           Cause any amount of white space at the end of a line to be  
 653                    treated as a single <newline> (i.e., the white-space charac-  
 654                    ters preceding the <newline> are ignored) and other strings  
 655                    of white-space characters, not including <newline>s, to com-  
 656                    pare equally. The -b option shall not affect the comparison of  
 657                    files of type symbolic link.

658 ⇒ **4.17.3 diff Options.** *Change the description of -r to:*

659        -r           Apply diff recursively to files and directories of the same  
 660                    name when *file1* and *file2* are both directories. If a symbolic  
 661                    link is encountered during the traversal of the file hierarchy,  
 662                    the diff utility shall take actions based on the file being of  
 663                    type symbolic link, rather than based on the type of the file  
 664                    referenced by the symbolic link.

665 ⇒ **4.17.4 diff Operands.** *Change the second paragraph (the one beginning "If*  
 666 *both file1 and file2 ... ") to:*

667        If *file1* or *file2* is a symbolic link, the diff utility shall take actions based on  
 668        the type and contents of the file referenced by the symbolic link; e.g., if *file1* is  
 669        a symbolic link that references a file of type directory, diff shall behave as if  
 670        it were a file of type directory.

671        If both *file1* and *file2* are directories, diff shall not compare block special files,  
 672        character special files, or FIFO special files to any files and shall not compare  
 673        files of different types. The system documentation shall specify the behavior of  
 674        diff on implementation-specific file types not specified by POSIX.1 {8} when  
 675        found in directories. Further details are as specified in 4.17.6.1.1.

676 ⇒ **4.17.6.1.1 diff Directory Comparison Format.** *Change the fifth para-*  
 677 *graph from:* For each file common to the two directories, if the files are to be  
 678        compared and are identical, no output shall be written. If the two files differ,  
 679        the following format: shall be written:

680                "diff %s %s %s\n", <diff\_options>, <filename1>, <filename2>

681        where <diff\_options> are the options as specified on the command line.  
 682        Depending on these options, one of the following output formats shall be used  
 683        to write the differences.

684 *to:*

685 For each file common to the two directories, if the files are symbolic links and  
686 their contents differ, the following format shall be written in the POSIX Locale:

687 "Symbolic links: %s -> %s and %s -> %s\n", <filename1>,  
688 <filename1 contents>, <filename2>, <filename2 contents>

689 Otherwise, for each file common to the two directories, if the files are to be  
690 compared and are identical, no output shall be written. If the two files differ,  
691 the following format shall be written:

692 "diff %s %s %s\n", <diff\_options>, <filename1>, <filename2>

693 where <diff\_options> are the options as specified on the command line.  
694 Depending on these options, one of the following output formats shall be used  
695 to write the differences.

696 ⇒ **4.17.6.1.4 diff -c or -C Output Format.** *(This change should be read only*  
697 *in conjunction with the following change.) Delete the phrase:*

698 and a string of 15 asterisks:

699 "\*\*\*\*\*\n"

700 ⇒ **4.17.6.1.4 diff -c or -C Output Format.** *Change the line "First, the range*  
701 *of lines in file1 shall be written in the "following format:" to:*

702 First, a line shall be written in the following format:

703 "\*\*\*\*\*\n"

704 Next, the range of lines in *file1* shall be written in the following format:

705 **Rationale:** The two preceding changes are the result of interpretation request  
706 PASC 1003.2-92 #71 submitted for IEEE Std 1003.2-1992.

707	<b>4.20 ed – Edit text</b>	
708	<i>Editor's Note: All instances of RE have been changed to BRE without specific diff</i>	C
709	<i>marks. This was an editorial error and was not intended to deviate from the 1992</i>	C
710	<i>text.</i>	C
711	⇒ <b>4.20.5.4 ed Asynchronous Events.</b> <i>Add a new list item at the end of the list:</i>	C
712	SIGQUIT    The ed utility shall ignore this event.	C
713	<b>Rationale:</b> This change is to align with historical practice and is the result of	C
714	interpretation request PASC 1003.2-92 #7 submitted for IEEE Std 1003.2-1992.	C
715	⇒ <b>4.20.7.2 ed Addressing.</b> <i>Change the entire subclause to:</i>	B
716	Addressing in ed relates to the current line. Generally, the current line is the	B
717	last line affected by a command. The current line number is the address of the	B
718	current line. If the edit buffer is not empty, the initial value for the current	B
719	line shall be the last line in the edit buffer; otherwise, zero.	B
720	Addresses shall be constructed as follows:	B
721	(1) The period character (.) shall address the current line.	B
722	(2) The dollar-sign character (\$) shall address the last line of the edit buffer.	B
723	(3) The positive decimal number <i>n</i> shall address the <i>n</i> -th line of the edit	B
724	buffer.	B
725	(4) The apostrophe- <i>x</i> character pair (' <i>x</i> ') shall address the line marked with	B
726	the mark name character <i>x</i> , which shall be a lowercase letter from the	C
727	portable character set. It shall be an error if the character has not been	C
728	set to mark a line, or if the line that was marked is not currently present	B
729	in the edit buffer, or the mark has not been set. Lines can be marked	B
730	with the <i>k</i> command.	B
731	(5) A BRE (see 2.8.3) enclosed by slash characters (/) shall address the first	B
732	line found by searching forwards from the line following the current line	B
733	toward the end of the edit buffer and stopping at the first line containing	B
734	a string matching the BRE. The BRE consisting of a null BRE delimited	B
735	by a pair of slash characters shall address the next line containing the	B
736	last BRE encountered. In addition, the second slash can be omitted at	B
737	the end of a command line. Within the BRE, a backslash-slash pair (\/)	B
738	shall represent a literal slash instead of the BRE delimiter.	B
739	(6) A BRE enclosed by question-mark characters (?) shall address the first	B
740	line found by searching backwards from the line preceding the current	B
741	line toward the beginning of the edit buffer and stopping at the first line	B
742	containing a string matching the BRE. The BRE consisting of a null BRE	B
743	delimited by a pair of question-mark characters (??) shall address the	B
744	previous line containing the last BRE encountered. In addition, the	B
745	second question-mark can be omitted at the end of a command line.	B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

746	Within the BRE, a backslash-question-mark pair (\?) shall represent a	B														
747	literal question mark instead of the BRE delimiter.	B														
748	(7) A plus-sign (+) or hyphen character (-) followed by a decimal number	B														
749	shall address the current line plus or minus the number. A plus-sign or	B														
750	hyphen character not followed by a decimal number shall address the	B														
751	current line plus or minus 1.	B														
752	Addresses can be followed by zero or more address offsets, optionally <blank>	B														
753	separated. Address offsets are constructed as follows:	B														
754	— A plus-sign or hyphen character followed by a decimal number shall add or	B														
755	subtract, respectively, the indicated number of lines to or from the address.	B														
756	A plus-sign or hyphen character not followed by a decimal number shall	B														
757	add or subtract 1 to or from the address.	B														
758	— A decimal number shall add the indicated number of lines to the address.	B														
759	It shall not be an error for an intermediate address value to be less than zero or	B														
760	greater than the last line in the edit buffer. It shall be an error for the final	B														
761	address value to be less than zero or greater than the last line in the edit buffer.	B														
762	Commands accepts zero, one, or two addresses. If more than the required number	B														
763	of addresses are provided to a command that requires zero addresses, it shall be	B														
764	an error. Otherwise, if more than the required number of addresses are provided	B														
765	to a command, the addresses specified first shall be evaluated and then discarded	B														
766	until the maximum number of valid addresses remain, for the specified command.	B														
767	Addresses shall be separated from each other by a comma (,) or semicolon charac-	B														
768	ter (;). In the case of a semicolon separator, the current line (.) shall be set to the	B														
769	first address, and only then will the second address be calculated. This feature	B														
770	can be used to determine the starting line for forwards and backwards searches	B														
771	[see rules (5) and (6)].	B														
772	Addresses can be omitted on either side of the comma or semicolon separator, in	B														
773	which case the resulting address pairs shall be as follows:	B														
774	<table border="0" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: left;"><u>Specified</u></th> <th style="text-align: left;"><u>Resulting</u></th> </tr> </thead> <tbody> <tr> <td>,</td> <td>1 , \$</td> </tr> <tr> <td>, <i>addr</i></td> <td>1 , <i>addr</i></td> </tr> <tr> <td><i>addr</i> ,</td> <td><i>addr</i> , <i>addr</i></td> </tr> <tr> <td>;</td> <td>. ; \$</td> </tr> <tr> <td>; <i>addr</i></td> <td>. ; <i>addr</i></td> </tr> <tr> <td><i>addr</i> ;</td> <td><i>addr</i> ; <i>addr</i></td> </tr> </tbody> </table>	<u>Specified</u>	<u>Resulting</u>	,	1 , \$	, <i>addr</i>	1 , <i>addr</i>	<i>addr</i> ,	<i>addr</i> , <i>addr</i>	;	. ; \$	; <i>addr</i>	. ; <i>addr</i>	<i>addr</i> ;	<i>addr</i> ; <i>addr</i>	B
<u>Specified</u>	<u>Resulting</u>															
,	1 , \$															
, <i>addr</i>	1 , <i>addr</i>															
<i>addr</i> ,	<i>addr</i> , <i>addr</i>															
;	. ; \$															
; <i>addr</i>	. ; <i>addr</i>															
<i>addr</i> ;	<i>addr</i> ; <i>addr</i>															
775		B														
776		B														
777		B														
778		B														
779		B														
780		B														
781	Any <blank> characters included between addresses, address separators, or	B														
782	address offsets shall be ignored.	B														



783 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B  
 784 #XX submitted for IEEE Std 1003.2-1992. B

785 ⇒ **4.20.7.3 ed Commands.** *Replace the sixth paragraph (the one beginning “If*  
 786 *an end-of-file is detected . . . ”) with:*

787 If a terminal disconnect is detected: B

788 — If the buffer is not empty and has changed since the last write, the `ed` util- B  
 789 ity shall attempt to write a copy of the buffer to a file. First, the file named B  
 790 `ed.hup` in the current directory shall be used; if that fails, the file named B  
 791 `ed.hup` in the directory named by the **HOME** environment variable shall B  
 792 be used. B

793 — The `ed` utility shall not write the file to the currently remembered path- B  
 794 name or return to command mode, and shall terminate with a nonzero exit B  
 795 status. B

796 If an end-of-file is detected on standard input: B

797 — If the `ed` utility is in input mode, `ed` shall terminate input mode and return B  
 798 to command mode. It is unspecified if any partially entered lines, (i.e., B  
 799 input text without a terminating `<newline>` character) are discarded from B  
 800 the input text. B

801 — If the `ed` utility is in command mode, it shall act as if a `q` command had B  
 802 been entered. B

803 **Rationale:** This change is required to match historical practice and is the result B  
 804 of interpretation request PASC 1003.2-92 #36 submitted for IEEE Std 1003.2-1992. B

805 ⇒ **4.20.7.3.2 ed Change Command.** *Add a new sentence at the end of the para- C  
 806 graph:* C

807 Address 0 shall be valid for this command; it shall be interpreted as if address C  
 808 1 were specified. C

809 ⇒ **4.20.7.3.7 ed Global Command.** *Change the second sentence (the one begin- B  
 810 ning with “Then, for every such line, . . . ”) to:* B

811 Then, going sequentially from the beginning of the file to the end of the file, B  
 812 the given *command list* shall be executed for each marked line, with the B  
 813 current line number set to the address of that line. Any line modified by the B  
 814 command list shall be unmarked. B

- 815 ⇒ **4.20.7.3.8 ed Interactive Global Command.** B
- 816 **Rationale:** The preceding two changes are the result of interpretation request B  
 817 PASC 1003.2-92 #119 submitted for IEEE Std 1003.2-1992. B
- 818 ⇒ **4.20.7.3.11 ed Insert Command.** *Change the final sentence of the paragraph* C  
 819 *to:* C
- 820 Address 0 shall be valid for this command; it shall be interpreted as if address C  
 821 1 were specified. C
- 822 ⇒ **4.20.7.3.14 ed List Command.** *Replace the second sentence with:*
- 823 The characters listed in Table 2-16 (see 2.12), except for \n, shall be written as  
 824 the corresponding escape sequences.
- 825 **Rationale:** The exception for \n was added to avoid breaking historical practice  
 826 and is the result of interpretation request PASC 1003.2-92 #32 submitted for IEEE  
 827 Std 1003.2-1992.
- 828 ⇒ **4.20.7.3.14 ed List Command.** *In the second paragraph, change the sentence* C  
 829 *“The end of each line shall be marked with a \$.” to:* C
- 830 The end of each line shall be marked with a \$, and \$ characters within the C  
 831 text shall be written with a preceding backslash. C
- 832 *Editor’s Note: The following rationale will be added to E.4.20, but is kept here with* B  
 833 *ed for this draft:* B
- 834 **ed Rationale.** *(This subclause is not a part of P1003.2b)* B
- 835 It is difficult under some modes of some versions of historical operating system B  
 836 terminal drivers to distinguish between an end-of-file condition and terminal B  
 837 disconnect. POSIX.2 does not require implementations to distinguish between the B  
 838 two situations, which permits historical implementations of the ed utility on his- B  
 839 torical platforms to conform. Implementations are encouraged to distinguish B  
 840 between the two, if possible, and take appropriate action on terminal disconnect. B
- 841 Historically, ed accepted a zero address for the a and r commands in order to B  
 842 insert text at the start of the edit buffer. When the buffer was empty the com- B  
 843 mand “.=” returned zero. This standard requires conformance to historical B  
 844 practice. B
- 845 For consistency with the a and r commands and better user functionality, the i B  
 846 and c commands must also accept an address of 0, in which case 0i is treated as B  
 847 1i and likewise for the c command. B
- 848 All of the following are valid addresses: B
- 849 +++ Three lines after the current line B

850     /*pattern*/-           One line before the next occurrence of pattern     B  
 851     -2                    Two lines before the current line                    B  
 852     3 ---- 2             Line one (note the intermediate negative address)     B  
 853     1 2 3                 Line six   B

854     Any number of addresses can be provided to commands taking addresses; e.g.,     B  
 855     1,2,3,4,5p prints lines 4 and 5, because two is the greatest valid number of     B  
 856     addresses accepted by the `print` command. This, in combination with the semi-     B  
 857     colon delimiter, permits users to create commands based on ordered patterns in     B  
 858     the file. For example, the command `3;/foo/;+2p` will display the first line after     B  
 859     line 3 that contains the pattern `foo`, plus the next two lines. Note that the     B  
 860     address “3;” still must be evaluated before being discarded, because the search     B  
 861     origin for the `/foo/` command depends on this.                                     B

862     Historically, `ed` disallowed address chains, as discussed above, consisting solely of     B  
 863     comma or semicolon separators; e.g., “,,,” or “; ; ;” were considered an error.     B  
 864     For consistency of address specification, this restriction is removed. The following     B  
 865     table list some of the address forms now possible:                                 B

866	<u>Address</u>	<u>Addr1</u>	<u>Addr2</u>	<u>Status</u>	<u>Comment</u>	B
867	7,	7	7	historical		B
868	7,5,	5	5	historical		B
869	7,5,9	5	9	historical		B
870	7,9	7	9	historical		B
871	7,+	7	8	historical		B
872	,	1	\$	historical		B
873	,7	1	7	extension		B
874	,,	\$	\$	extension		B
875	,i	\$	\$	extension		B
876	7i	7	7	historical		B
877	7;5;	5	5	historical		B
878	7;5;9	5	9	historical		B
879	7;5,9	5	9	historical		B
880	7;\$;4	\$	4	historical	valid, but erroneous	B
881	7;9	7	9	historical		B
882	7i+	7	8	historical		B
883	i	.	\$	historical		B
884	i7	.	7	extension		B
885	;;	\$	\$	extension		B
886	i,	\$	\$	extension		B

887     Historically, values could be added to addresses by including them after one or     B  
 888     more <blank> characters; e.g., “3 - 5p” wrote the seventh line of the file, and     B  
 889     “/foo/ 5” was the same as `/foo/+5`. However, only absolute values could be     B  
 890     added; e.g., “5 /foo/” was an error. This standard requires conformance to his-     B  
 891     torical practice.   B

892 Historically, `ed` accepted the `^` character as an address, in which case it was B  
 893 identical to the hyphen character. This standard does not require or prohibit this B  
 894 behavior. B

## 895 **4.22 `expr` – Evaluate arguments as an expression** B

896 ⇒ **4.22.6.1 `expr` Standard Output.** *Change the contents of this subclause to:* B

897 The `expr` utility shall write the evaluation of the expression to standard out- B  
 898 put followed by a `<newline>` character. B

899 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B  
 900 #104 submitted for IEEE Std 1003.2-1992. B

901 ⇒ **4.22.7 `expr` Extended Description.** *Change the first row in Table 4-5 to:* B

Expression	Description
<code>expr1   expr2</code>	Returns the evaluation of <code>expr1</code> if it is neither null nor zero; otherwise, the evaluation of <code>expr2</code> if it is not null; otherwise, zero.

905 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B  
 906 #104 submitted for IEEE Std 1003.2-1992. B

## 907 **4.24 `find` – Find files**

908 ⇒ **4.24.1 `find` Synopsis.** *Change the Synopsis to:*

909 `find [-H | -L] path... [operand_expression...]`

910 ⇒ **4.24.2 `find` Description.** *Add at the end of the second paragraph:*

911 The `find` utility shall detect infinite loops; i.e., entering a previously visited B  
 912 directory that is an ancestor of the last file encountered. When it detects an B  
 913 infinite loop, `find` shall write a diagnostic message to standard error and shall  
 914 either recover its position in the hierarchy or terminate.

915 ⇒ **4.24.3 find Options.** *Change the entire subclause to:*

916 The `find` utility shall conform to the utility argument syntax guidelines  
917 described in 2.10.2.

918 The following options shall be supported by the implementation:

919        -H           Cause the file information and file type evaluated for each  
920                      symbolic link encountered on the command line to be those of  
921                      the file referenced by the link, and not the link itself. If the  
922                      referenced file does not exist, the file information and type  
923                      shall be for the link itself. File information for all symbolic  
924                      links not on the command line shall be that of the link itself.

925        -L           Cause the file information and file type evaluated for each  
926                      symbolic link to be those of the file referenced by the link, and  
927                      not the link itself. If the referenced file does not exist, the file  
928                      information and type shall be for the link itself.

929        Specifying more than one of the mutually exclusive options `-H` and `-L` shall not    C  
930        be considered an error. The last option specified shall determine the behavior    C  
931        of the utility.    C

932 *Editor's Note: The following rationale will be added to E.4.24, but is kept here with*  
933 *find for this draft:*

934 **find Rationale.** *(This subclause is not a part of P1003.2b)*

935 Historically, the `-L` option was implemented using the primary `-follow`. The `-H`  
936 and `-L` options were added for two reasons. First, they offer a finer granularity of  
937 control and consistency with other programs that walk file hierarchies. Second,  
938 the `-follow` primary always evaluated to true. As they were historically really  
939 global variables that took effect before the traversal began, some valid expres-  
940 sions had unexpected results. An example is the expression `-print -o -follow`.  
941 Because `-print` always evaluates to true, the standard order of evaluation  
942 implies that `-follow` would never be evaluated. This was never the case.

943 B

944 ⇒ **4.24.4 find Operands.** *Replace the `-atime`, `-ctime`, and `-mtime` descrip-*  
945 *tions with:*

946        -atime *n*       The primary shall evaluate as true if the file access time    B  
947    B  
948    B  
  B

949        -ctime *n*       The primary shall evaluate as true if the time of last    B  
950    B  
951    B  
952    B

953            `-mtime n`            The primary shall evaluate as true if the file modification  
 954                                    time subtracted from the initialization time, divided by  
 955                                    86 400 (with any remainder discarded), is *n*.            B

956    **Rationale:** This change is required to match historical practice and is the result  
 957    of interpretation request PASC 1003.2-92 #58 submitted for IEEE Std 1003.2-1992.

958    ⇒ **4.24.4 find Operands.** *Add the following primary in the proper sorted order:*

959            `-follow`            The primary always shall evaluate as true. If it occurs  
 960                                    anywhere in *operand\_expression*, it shall cause `find` to  
 961                                    evaluate the file information and file type for all symbolic  
 962                                    links (whether named on the command line or encoun-  
 963                                    tered in a file hierarchy) to be those of the file referenced  
 964                                    by the link, and not the link itself. If the referenced file  
 965                                    does not exist, the file information and type shall be for  
 966                                    the link itself. By default, `find` shall not follow symbolic  
 967                                    links. If any `-follow` primary is specified, it shall apply    C  
 968                                    to the entire expression even if the `-follow` primary    C  
 969                                    would not normally be evaluated.            C

970    ⇒ **4.24.4 find Operands.** *In the `-type c` description, add the character `l` (*ell*)*  
 971    *to represent a symbolic link.*

972    **4.26 getconf – Get configuration values**                                    B

973    ⇒ **4.26.4 getconf Operands.** *Change the first paragraph of the `system_var`*    B  
 974    *operand to:*    B

975            `system_var`    A name of a configuration variable or minimum value avail-    B  
 976                                    able from the *confstr()* or *sysconf()* functions in POSIX.1 {8}.    B

977    **Rationale:** The `getconf` changes are part of a general cleanup to remove refer-    B  
 978    ences to the now-deleted Chapter 7. All of the applicable functions are now in    B  
 979    POSIX.1-199x, the version created by the currently balloting P1003.1a.            B

980 ⇒ **4.26.6.1 getconf Standard Output.** *In the first paragraph, change the* B  
 981 *phrase “the function in 7.8.1 ” to:* B

982 the POSIX.1 {8} *confstr()* function B

### 983 **4.33 ln – Link files**

984 ⇒ **4.33.1 ln Synopsis.** *Modify the Synopsis to be:*

985 ln [-fs] *source\_file target\_file*

986 ln [-fs] *source\_file ... target\_dir*

987 ⇒ **4.33.2 ln Description.** *Change the first two paragraphs to:*

988 In the first synopsis form, the *ln* utility shall create a new directory entry  
 989 (link), or if the *-s* option is specified, a symbolic link, for the file specified by  
 990 the *source\_file* operand at the *destination* path specified by the *target\_file*  
 991 operand. This first synopsis form shall be assumed when the final operand  
 992 does not name an existing directory; if more than two operands are specified  
 993 and the final operand is not an existing directory, an error shall result.

994 In the second synopsis form, the *ln* utility shall create a new directory entry,  
 995 or if the *-s* option is specified, a symbolic link, for each file specified by a  
 996 *source\_file* operand at a *destination* path in the existing directory named by  
 997 *target\_dir*.

998 *Editor’s Note: The third paragraph of POSIX.2-1992 (If the last operand specifies*  
 999 *an existing file of a type not specified by POSIX.1 {8}, the behavior is implementa-*  
 1000 *tion defined.) is referring to the version of POSIX.1 {8} at the time the dot2b*  
 1001 *amendment is approved, not the 1990 version. Since dot2b and dot1a are proceed-*  
 1002 *ing in sync, this will be P1003.1a, which includes symlinks.*

1003 ⇒ **4.33.2 ln Description.** *In the fourth paragraph, change “The corresponding*  
 1004 *destination path ... ” to:*

1005 The corresponding *destination* path ...

1006 ⇒ **4.33.2 ln Description.** *Change item (2) to:*

1007 (2) If the *-s* option is specified, *ln* shall create a symbolic link named by the  
 1008 *destination* path and containing as its pathname *source\_file*. The *ln* util- B  
 1009 ity shall do nothing more with *source\_file* and shall go on to any remain- B  
 1010 ing files. B

1011 (3) If *source\_file* is a symbolic link, actions shall be performed equivalent to B  
 1012 the POSIX.1 {8} *link()* function using the object that *source\_file* references B

1013 as the *path1* argument and the *destination* path as the *path2* argument. B  
 1014 The `ln` utility shall do nothing more with *source\_file* and shall go on to B  
 1015 any remaining files. B

1016 (4) Actions shall be performed equivalent to the POSIX.1 {8} `link()` function B  
 1017 using *source\_file* as the *path1* argument and the *destination* path as the B  
 1018 *path2* argument.

1019 ⇒ **4.33.3 `ln` Options.** *Add the following option in the proper sorted order:*

1020 `-s` Create symbolic links instead of hard links.

1021 ⇒ **4.33.4 `ln` Operands.** *Replace the description of *source\_file* with:*

1022 *source\_file* A pathname of a file to be linked. If the `-s` option is specified,  
 1023 no restrictions on the type of file or on its existence shall be  
 1024 made. If the `-s` option is not specified, whether a directory  
 1025 can be linked is implementation defined.

1026 **4.35 `localedef` – Define locale environment** B

1027 *Editor's Note: This clause is new in Draft 11. To avoid clutter, it is not further* B  
 1028 *diffmarked.* B

1029 ⇒ **4.35.1 `localedef` Synopsis.** *Modify the Synopsis to be:*

1030 `localedef [-c] [-f charmap] [-i sourcefile] [-u code_set_name] name`

1031 ⇒ **4.35.3 `localedef` Options.** *Add the following option in the proper sorted*  
 1032 *order:*

1033 `-u code_set_name`  
 1034 Specify the name of a code set used as the target mapping of  
 1035 character symbols and collating element symbols whose encod-  
 1036 ing values are defined in terms of ISO/IEC 10646 {10} position  
 1037 constant values.



- 1038 ⇒ **4.35.7 localedef Extended Description.** *Change this subclause from*  
 1039 *“None.” to:*
- 1040 When the `-u` option is used, the `code_set_name` option-argument shall be inter-  
 1041 preted as an implementation-defined name of a code set to which the  
 1042 ISO/IEC 10646 {10} position constant values shall be converted via an  
 1043 implementation-defined method. Both ISO/IEC 10646 {10} position constant  
 1044 values and other formats (decimal, hexadecimal, or octal) shall be valid as  
 1045 encoding values within the charmap file. The code set represented by the  
 1046 implementation-defined name can be any codeset that is supported by the  
 1047 implementation.
- 1048 When conflicts occur between the charmap specification of `<code_set_name>`,  
 1049 `<mb_cur_max>`, or `<mb_cur_min>` and the implementation-defined interpre-  
 1050 tation of these respective items for the codeset represented by the `-u` option-  
 1051 argument `code_set_name`, the result is unspecified.
- 1052 When conflicts occur between the charmap encoding values specified for sym-  
 1053 bolic names of characters of the portable character set (Table 2-4) and the  
 1054 implementation-defined assignment of character encoding values, the result is  
 1055 unspecified.
- 1056 If a nonprintable character in the charmap has a width specified that is not `-1`, C  
 1057 `localedef` shall generate a warning. C
- 1058 ⇒ **4.35.9 localedef Consequences of Errors.** *Add a final list entry to the* C  
 1059 *dashed list of conditions for warning messages:* C
- 1060 — If a nonprintable character has a width specified other than `-1`. C

1061 **4.39 ls – List directory contents**1062 ⇒ **4.39.1 ls Synopsis.** *Modify the Synopsis to be:*1063 `ls [-CFRacdilqrtul] [-H | -L ] [file ... ]`1064 ⇒ **4.39.2 ls Description.** *Replace the entire subclause with:*

1065 For each operand that names a file of a type other than directory or symbolic  
 1066 link to a directory, `ls` shall write the name of the file as well as any requested,  
 1067 associated information. For each operand that names a file of type directory, B  
 1068 `ls` shall write the names of files contained within the directory as well as any B  
 1069 requested, associated information. If one of the `-d`, `-F`, or `-l` options are B  
 1070 specified, and one of the `-H` or `-L` options are not specified, for each operand B  
 1071 that names a file of type symbolic link to a directory, `ls` shall write the name B  
 1072 of the file as well as any requested, associated information. If none of the `-d`, B  
 1073 `-F`, or `-l` options are specified, or the `-H` or `-L` options are specified, for each B  
 1074 operand that names a file of type symbolic link to a directory, `ls` shall write B  
 1075 the names of files contained within the directory as well as any requested, B  
 1076 associated information. B

1077 If no operands are specified, `ls` shall write the contents of the current direc-  
 1078 tory. If more than one operand is specified, `ls` shall write nondirectory  
 1079 operands first; it shall sort directory and nondirectory operands separately  
 1080 according to the collating sequence in the current locale.

1081 The `ls` utility shall detect infinite loops; i.e., entering a previously visited B  
 1082 directory that is an ancestor of the last file encountered. When it detects an B  
 1083 infinite loop, `ls` shall write a diagnostic message to standard error and shall  
 1084 either recover its position in the hierarchy or terminate.

1085 ⇒ **4.39.3 ls Options.** *Replace the descriptions of the `-d`, `-F`, and `-l` options  
1086 with the following:*

1087 `-d` Do not follow symbolic links named as operands unless the `-H` B  
 1088 or `-L` options are specified. Do not treat directories differently B  
 1089 than other types of files. The use of `-d` with `-R` produces  
 1090 unspecified results.

1091 `-F` Do not follow symbolic links named as operands unless the `-H` B  
 1092 or `-L` options are specified. Write a slash (/) immediately B  
 1093 after each pathname that is a directory, an asterisk (\*) after  
 1094 each that is executable, a vertical bar (|) after each that is a  
 1095 FIFO, and an at-sign (@) after each that is a symbolic link.

1096            -l            (The letter ell.) Do not follow symbolic links named as  
 1097                            operands unless the -H or -L options are specified. Write out    B  
 1098                            in long format (see 4.39.6.1). When -l (ell) is specified, -l  
 1099                            (one) shall be assumed.

1100   ⇒ **4.39.3 ls Options.** *Add the following options in the proper sorted order:*

1101            -H            If a symbolic link referencing a file of type directory is  
 1102                            specified on the command line, ls shall evaluate the file infor-  
 1103                            mation and file type to be those of the file referenced by the  
 1104                            link, and not the link itself; however, ls shall write the name  
 1105                            of the link itself and not the file referenced by the link.

1106            -L            Evaluate the file information and file type for all symbolic  
 1107                            links (whether named on the command line or encountered in  
 1108                            a file hierarchy) to be those of the file referenced by the link,  
 1109                            and not the link itself; however, ls shall write the name of the  
 1110                            link itself and not the file referenced by the link. When -L is  
 1111                            used with -l, write the contents of symbolic links in the long  
 1112                            format (see 4.39.6.1).

1113   ⇒ **4.39.3 ls Options.** *Change the final paragraph in this subclause to:*            B

1114            Specifying more than one of the options in the following mutually exclusive    B  
 1115                            pairs shall not be considered an error: -C and -l (ell), -C and -l (one), -H and    B  
 1116                            -L, -C and -u. The last option specified in each pair shall determine the out-    B  
 1117                            put format.    B

1118   ⇒ **4.39.6.1 ls Standard Output.** *Replace the six-line description of -l (begin-*  
 1119                            *ning with "If the -l option is specified, ... ") with:*

1120            If the -l option is specified without -L, the following information shall be writ-  
 1121                            ten:

1122                            "%s %u %s %s %u %s %s\n", <file mode>, <number of links>,  
 1123                            <owner name>, <group name>, <number of bytes in the file>,  
 1124                            <date and time>, <pathname>

1125            If the file is a symbolic link, this information shall be about the link itself and  
 1126                            the <pathname> field shall be of the form:

1127                            "%s -> %s", <pathname of link>, <contents of link>

1128            If both -l and -L are specified, the following information shall be written:

1129                            "%s %u %s %s %u %s %s\n", <file mode>, <number of links>,  
 1130                            <owner name>, <group name>, <number of bytes in the file>,  
 1131                            <date and time>, <pathname of link>

1132            where all fields except <pathname of link> shall be for the file resolved from

1133 the symbolic link.

1134 In both of the preceding `-l` forms, if `<owner name>` or `<group name>` cannot be  
1135 determined, they shall be replaced with their associated numeric values using  
1136 the format "`%u`".

1137 ⇒ **4.39.6.1 `ls` Standard Output.** *Add the following to the list of `<entry type>`  
1138 characters:*

1139 `l (ell)` Symbolic link

1140 ⇒ **4.39.8 `ls` Exit Status.** *Change the zero exit status from "All files were writ-  
1141 ten successfully." to:*

1142 `0` Successful completion.

1143 **Rationale:** This change is in response to confusion about whether `ls` was sup-  
1144 posed to write to the files about which it was reporting. It is the result of  
1145 interpretation request PASC 1003.2-92 #39 submitted for IEEE Std 1003.2-1992.

#### 1146 **4.40 `mailx` – Process Messages**

1147 **Rationale:** The majority of changes to the `mailx` utility arise from interpretation C  
1148 requests submitted for IEEE Std 1003.2-1992. In particular, the changes here C  
1149 address interpretation requests PASC 1003.2-92 #10, 11, 103, 106, 108, 114, 115, C  
1150 122 and 129. Where a change is particularly relevant to an interpretation C  
1151 request, it is highlighted by additional in-line rationale. Where there is no addi- C  
1152 tional rationale given, the change has been caused by problems highlighted by the C  
1153 resolution of these interpretations. C

1154 ⇒ **4.40.5.3 `mailx` Environment Variables.** *In the description of the **LISTER**  
1155 variable, delete the sentence "The default value shall be unset."*

1156 **Rationale:** This change satisfies the following corrigendum request from ISO/IEC  
1157 9945-2: 1993 Annex H.2:

1158 (6) In the 4.40.5.3 description of the `mailx` **LISTER** variable, the sentence  
1159 "The default value shall be unset" may be redundant.

1160 ⇒ **4.40.7 mailx Extended Description.** *Change the second paragraph (the* B  
 1161 *one beginning with “When mailx is invoked ...”) to:* B

1162 When `mailx` is invoked using one of the Receive Mode synopsis forms, it shall B  
 1163 write a page of header-summary lines (if `-N` was not specified and there are B  
 1164 messages, see below), followed by a prompt indicating `mailx` can accept regu- B  
 1165 lar commands (see 4.40.7.2); this is termed *command mode*. The page of B  
 1166 header-summary lines shall contain the first new message if there are new B  
 1167 messages, or the first unread message if there are unread messages, or the B  
 1168 first message. When `mailx` is invoked using the Send Mode synopsis and B  
 1169 standard input is a terminal, if no subject is specified on the command line B  
 1170 and the `asksub` variable is set, a prompt for the subject shall be written. At B  
 1171 this point `mailx` is in *input mode*. This input mode is also entered when B  
 1172 using one of the Receive Mode synopsis forms and a reply or new message is B  
 1173 composed using the `reply`, `Reply`, or `mail` commands and standard input is a B  
 1174 terminal. When the message is typed and the end of message is encountered, B  
 1175 the message shall be passed to the mail delivery software. Commands can be B  
 1176 entered by beginning a line with the escape character [by default, tilde (~)] fol- B  
 1177 lowed by a single command letter and optional arguments. See 4.40.7.3 for a B  
 1178 summary of these commands. It is unspecified what effect these commands B  
 1179 will have if standard input is not a terminal when a message is entered using B  
 1180 either the Send Mode synopsis, or the Read Mode commands `reply`, `Reply`, or B  
 1181 `mail`. B

1182 **Rationale:** The preceding change is the result of interpretation request PASC C  
 1183 1003.2-92 #103, submitted for IEEE Std 1003.2-1992. C

1184 ⇒ **4.40.7 mailx Extended Description.** *Change the fifth paragraph (the one* B  
 1185 *beginning “If no command is specified ...”) to:* B

1186 If no *command* is specified in command mode, `next` shall be assumed. In B  
 1187 input mode, commands shall be recognized by the escape character, and lines B  
 1188 not treated as commands shall be taken as input for the message. B

1189 **Rationale:** The preceding change is the result of interpretation requests PASC C  
 1190 1003.2-92 #103 and 115, submitted for IEEE Std 1003.2-1992. C

1191 ⇒ **4.40.7 mailx Extended Description.** *In the seventh paragraph (the one* B  
 1192 *beginning “All messages have a state ...”), change the sentence “All messages* B  
 1193 *are in one of the following states:” to:* B

1194 When `mailx` is invoked using one of the Receive Mode synopsis forms, the B  
 1195 current message shall be the first new message, if there is a new message, or B  
 1196 the first unread message if there is an unread message, or the first message if B  
 1197 there are any messages, or unspecified if there are no messages in the mailbox. C  
 1198 Each command that takes an optional list of messages (*msglist*) or an optional B  
 1199 single message (*message*) on which to operate shall leave the current message B  
 1200 set to the highest-numbered message of the messages specified, unless the C  
 1201 command deletes messages, in which case the current message shall be set to C

1202 the first undeleted message (i.e., a message not in the deleted state) after the C  
 1203 highest-numbered message deleted by the command, if one exists, or the first C  
 1204 undeleted message before the highest-numbered message deleted by the com- C  
 1205 mand, if one exists, or to an unspecified value if there are no remaining C  
 1206 undeleted messages. All messages are in one of the following states: C  
 1207 C

1208 ⇒ **4.40.7 mailx Extended Description.** *Change the description of the deleted* C  
 1209 *state to:*

1210 *deleted* The message has been processed by one of the following com- C  
 1211 mands: delete, dp, dt. Messages in state *deleted* when C  
 1212 mailx quits shall be deleted. Deleted messages shall be C  
 1213 ignored until mailx quits or changes mailboxes or they are B  
 1214 specified to the undelete command; e.g., the message B  
 1215 specification */string* shall only search the subject lines of mes- B  
 1216 sages that have not yet been deleted, unless the command B  
 1217 operating on the list of messages is undelete. No deleted B  
 1218 message or deleted message header shall be displayed by any B  
 1219 mailx command other than undelete. B

1220 C

1221 ⇒ **4.40.7 mailx Extended Description.** *Add a description of the saved state:* C

1222 *saved* The message has been processed by one of the following com- C  
 1223 mands: save or write. If the current mailbox is the system C  
 1224 mailbox, and the internal variable keepsave is set, messages C  
 1225 in the state *saved* shall be saved to the file designated by the C  
 1226 **MBOX** variable (see 4.40.5.3). If the current mailbox is the C  
 1227 system mailbox, messages in the state *saved* shall be deleted C  
 1228 from the current mailbox, when the quit or file command is C  
 1229 used to exit the current mailbox. C

1230 ⇒ **4.40.7.1 mailx Internal Variables.** *Change the description of the keepsave* C  
 1231 *variable to:* C

1232 keepsave Keep the messages that have been saved from the system C  
 1233 mailbox into other files in the file designated by the variable C  
 1234 **MBOX**, instead of deleting them. The default shall be nokeep- C  
 1235 save. C

- 1236 ⇒ **4.40.7.2.5 mailx Delete messages.** *Change the paragraph following the* B  
 1237 *Synopsis to:* B
- 1238 Mark messages for deletion from the mailbox. The deletions shall not occur B  
 1239 until mailx quits (see 4.40.7.2.24) or changes mailboxes (see 4.40.7.2.10). If B  
 1240 autoprint is set and there are messages remaining after the delete com- B  
 1241 mand, the current message shall be written as described for the print com- B  
 1242 mand (see 4.40.7.2.23); otherwise, the mailx prompt shall be written. B
- 1243 **Rationale:** The preceding change is the result of interpretation requests PASC C  
 1244 1003.2-92 #129, submitted for IEEE Std 1003.2-1992. C
- 1245 ⇒ **4.40.7.2.7 mailx Delete messages and display.** *Change the paragraph fol-* B  
 1246 *lowing the Synopsis to:* B
- 1247 Delete the specified messages as described for the delete command, except B  
 1248 that the autoprint variable shall have no effect, and the current message B  
 1249 shall be written only if it was set to a message after the last message deleted B  
 1250 by the command. Otherwise, an informational message to the effect that there B  
 1251 are no further messages in the mailbox shall be written, followed by the mailx B  
 1252 prompt. B
- 1253 **Rationale:** The preceding change is the result of interpretation requests PASC C  
 1254 1003.2-92 #129, submitted for IEEE Std 1003.2-1992. C
- 1255 ⇒ **4.40.7.2.8 mailx Edit messages.** *Change the paragraph following the* B  
 1256 *Synopsis to:* B
- 1257 Edit the given messages. Each message shall be placed in a temporary file, C  
 1258 and the utility named by the EDITOR variable (see 4.40.5.3) shall be invoked C  
 1259 to edit each file in sequence. The default editor is unspecified. B
- 1260 **Rationale:** The preceding change is the result of interpretation requests PASC B  
 1261 1003.2-92 #108 submitted for IEEE Std 1003.2-1992. B
- 1262 ⇒ **4.40.7.2.11 mailx Display list of folders.** *Change the sentence following the*  
 1263 *synopsis to:*
- 1264 Write the names of the files in the directory set by the folder variable (see  
 1265 4.40.7.1). The command specified by the LISTER environment variable shall  
 1266 be used (see 4.40.5.3).
- 1267 **Rationale:** This change satisfies the following corrigendum request from ISO/IEC  
 1268 9945-2: 1993 Annex H.2:
- 1269 (7) In 4.40.7.2.11, the mailx folders command does not indicate how the  
 1270 value of the LISTER variable affects this command.

1271	⇒ <b>4.40.7.2.13 mailx Display header summary.</b> <i>Change the entire subclause</i>	C
1272	<i>to:</i>	C
1273	<i>Synopsis:</i> h[eaders] [message]	C
1274	Write the page of headers that includes the message specified. If the <i>message</i>	C
1275	argument is not specified, the current message shall not change. However, if	C
1276	the <i>message</i> argument is specified, the current message shall become the mes-	C
1277	sage that appears at the top of the page of headers that includes the message	C
1278	specified. The <code>screen</code> variable sets the number of headers per page. See also	C
1279	the <code>z</code> command.	C
1280	⇒ <b>4.40.7.2.20 mailx Process next specified message.</b> <i>Change the sentence</i>	B
1281	<i>following the synopsis to:</i>	B
1282	If the current message has not been written (e.g., by the <code>print</code> command)	B
1283	since <code>mailx</code> started or since any other message was the current message,	B
1284	behave as if the <code>print</code> command was entered. Otherwise, if there is a	B
1285	undelivered message after the current message, make it the current message	B
1286	and behave as if the <code>print</code> command was entered. Otherwise, an informa-	B
1287	tional message to the effect that there are no further messages in the mailbox	B
1288	shall be written, followed by the <code>mailx</code> prompt.	B
1289		C
1290	⇒ <b>4.40.7.2.28 mailx Save messages.</b> <i>Change the final sentence, “The message</i>	C
1291	<i>shall be deleted from the mailbox ... ” to:</i>	C
1292	The message shall be put in the state <i>saved</i> , and shall behave as specified in	C
1293	the description of the <i>saved</i> state when the current mailbox is exited by the	C
1294	<code>quit</code> or <code>file</code> command (see 4.40.7).	C
1295	⇒ <b>4.40.7.2.36 mailx Undelete messages.</b> <i>Change all of the subclause follow-</i>	B
1296	<i>ing the synopsis to:</i>	B
1297	Change the state of the specified messages from <i>deleted</i> to <i>read</i> . If <code>autoprint</code>	B
1298	is set, the last message of those restored shall be written. If <code>msglist</code> is not	B
1299	specified, the message shall be selected as follows:	B
1300	— If there are any deleted messages that follow the current message, the first	B
1301	of these shall be chosen.	B
1302	— Otherwise, the last deleted message that also precedes the current message	B
1303	shall be chosen.	B
1304		C



1305 ⇒ **4.40.7.2.38 mailx Edit message with full-screen editor.** *Change the*  
 1306 *paragraph following the synopsis to:*

1307 Edit the given messages with a screen editor. Each message shall be placed in C  
 1308 a temporary file, and the utility named by the **VISUAL** variable (see 4.40.5.3)  
 1309 shall be invoked to edit each file in sequence. The default editor shall be `vi`.

1310 **Rationale:** The preceding change is the result of interpretation requests PASC  
 1311 1003.2-92 #115 submitted for IEEE Std 1003.2-1992.

1312 *Editor's Note: The following rationale will be added to E.4.40, but is kept here with*  
 1313 *mailx for this draft:*

1314 **mailx Rationale.** *(This subclause is not a part of P1003.2b)*

1315 The intent of the wording for the `next` command is that if any command has  
 1316 already displayed the current message it should display a following message, but  
 1317 otherwise, it should display the current message. Consider the command  
 1318 sequence:

```
1319     next 3
1320     delete 3
1321     next
```

1322 where the `autoprint` option was not set. The normative text specifies that the  
 1323 second `next` command should display a message following the third message,  
 1324 because even though the current message has not been displayed since it was set  
 1325 by the `delete` command, it has been displayed since the current message was  
 1326 anything other than message number 3. This does not always match historical  
 1327 practice in some implementations, where the command `file address` followed by  
 1328 `next` (or the default command) would skip the message for which the user had  
 1329 searched.

1330	<b>4.41 mkdir – Make directories</b>	B
1331	⇒ <b>4.41.2 mkdir Description.</b> <i>Change item (2) to:</i>	B
1332	(2) The value of the bitwise inclusive OR of S_IRWXU, S_IRWXG, and	B
1333	S_IRWXO is used as the <i>mode</i> argument. (If the <i>-m</i> option is specified, the	B
1334	value of the <i>mkdir()</i> <i>mode</i> argument is unspecified, but the directory	B
1335	shall at no time have permissions less restrictive than the <i>-m mode</i>	B
1336	option-argument.)	B
1337	⇒ <b>4.41.3 mkdir Options.</b> <i>Change the description of <i>-m</i> to:</i>	B
1338	<i>-m mode</i> The file permission bits of the directory shall be set to the	B
1339	specified <i>mode</i> value. The <i>mode</i> option-argument shall be the	B
1340	same as the <i>mode</i> operand defined for the <i>chmod</i> utility (see	B
1341	4.7). In the <i>symbolic_mode</i> strings, the <i>op</i> characters + and -	B
1342	shall be interpreted relative to an assumed initial mode of	B
1343	<i>a=rwx</i> ; + shall add permissions to the default mode, - shall	B
1344	delete permissions from the default mode.	B
1345	<b>Rationale:</b> The preceding two changes are the result of interpretation request	B
1346	PASC 1003.2-92 #67 submitted for IEEE Std 1003.2-1992. Identical changes were	B
1347	made for <i>mkdir</i> and <i>mkfifo</i> .	B
1348	<b>4.42 mkfifo – Make Make FIFO special files</b>	C
1349	⇒ <b>4.42.2 mkfifo Description.</b> <i>Change item (2) to:</i>	C
1350	(2) The value of the bitwise inclusive OR of S_IRWXU, S_IRWXG, and	C
1351	S_IRWXO is used as the <i>mode</i> argument. (If the <i>-m</i> option is specified, the	C
1352	value of the <i>mkfifo()</i> <i>mode</i> argument is unspecified, but the FIFO shall at	C
1353	no time have permissions less restrictive than the <i>-m mode</i> option-	C
1354	argument.)	C
1355	⇒ <b>4.42.3 mkfifo Options.</b> <i>Change the description of <i>-m</i> to:</i>	C
1356	<i>-m mode</i> The file permission bits of the FIFO shall be set to the specified	C
1357	<i>mode</i> value. The <i>mode</i> option-argument shall be the same as	C
1358	the <i>mode</i> operand defined for the <i>chmod</i> utility (see 4.7). In	C
1359	the <i>symbolic_mode</i> strings, the <i>op</i> characters + and - shall be	C
1360	interpreted relative to an assumed initial mode of <i>a=rwx</i> ; +	C
1361	shall add permissions to the default mode, - shall delete per-	C
1362	missions from the default mode.	C

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1363 **Rationale:** The preceding two changes are the result of interpretation request C  
 1364 PASC 1003.2-92 #67 submitted for IEEE Std 1003.2-1992. Identical changes were C  
 1365 made for `mkdir` and `mkfifo`. C

#### 1366 **4.43 mv – Move files**

1367 ⇒ **4.43.2 mv Description.** *Replace the first two paragraphs of the Description*  
 1368 *with:*

1369 In the first synopsis form, the `mv` utility shall move the file named by the  
 1370 *source\_file* operand to the *destination* specified by the *target\_file*. This first  
 1371 synopsis form is assumed when the final operand does not name an existing  
 1372 directory and is not a symbolic link referring to an existing directory.

1373 In the second synopsis form, `mv` shall move each file named by a *source\_file*  
 1374 operand to a *destination* file in the existing directory named by the *target\_dir*  
 1375 operand, or referenced if *target\_dir* is a symbolic link referring to an existing  
 1376 directory. The *destination* path for each *source\_file* shall be the concatenation  
 1377 of the target directory, a single slash character, and the last pathname com-  
 1378 ponent of the *source\_file*. This second form is assumed when the final operand  
 1379 names an existing directory.

1380 ⇒ **4.43.2 mv Description.** *Replace the first sentence of item (5) with:*

1381 The file hierarchy rooted in *source\_file* shall be duplicated as a file hierarchy  
 1382 rooted in the destination path. If *source\_file* or any of the files below it in the  
 1383 hierarchy are symbolic links, the links themselves shall be duplicated, includ- B  
 1384 ing their contents, rather than any files to which they refer.

1385 *Editor's Note: The following rationale will be added to E.4.43, but is kept here with*  
 1386 *mv for this draft:*

1387 **mv Rationale.** *(This subclause is not a part of P1003.2b)*

1388 When `mv` is dealing with a single file system and *source\_file* is a symbolic link, the  
 1389 link itself is moved as a consequence of the dependence on the POSIX.1 {8}  
 1390 *rename()* functionality, per the Description. Across file systems, this has to be  
 1391 made explicit.

1392 **4.45 od – Dump files in various formats**1393 ⇒ **4.45.4 od Operands.** *Change the description of the file operand to:* B

1394 *file* A pathname of a file to be read. If no *file* operands are B  
 1395 specified, the standard input shall be used. If there are more B  
 1396 than two operands, none of the *-A*, *-j*, *-N*, or *-t* options is B  
 1397 specified, and either of the following are true: B

1398 — the first character of the last operand is a plus sign (+), or B

1399 — the first character of the second operand is numeric B

1400 then the results are unspecified. B

1401 ⇒ **4.45.7 od Extended Description.** *Replace the second sentence with:*

1402 If no output type is specified, the default output shall be as if *-t oS* had been  
 1403 specified.

1404 **Rationale:** The changes to *od* are required to match historical practice and are  
 1405 the result of interpretation requests PASC 1003.2-92 #47 and #95 submitted for B  
 1406 IEEE Std 1003.2-1992. B

1407 ⇒ **4.45.7 od Extended Description.** *Change the first dashed list item to:*

1408 — The default number of bytes transformed by output type specifiers *d*, *o*, *u*, C  
 1409 and *x* corresponds to the various C-language types, as follows: C

1410 — If the *c89* compiler is present on the system, these specifiers shall C  
 1411 correspond to the sizes used by default in that compiler. C

1412 — Otherwise, these sizes may vary among systems that conform to this C  
 1413 standard. For the type specifier characters *d*, *o*, *u*, and *x* the default C  
 1414 number of bytes shall correspond to the size of the basic integral data C  
 1415 type of the underlying implementation. For these specifier characters, C  
 1416 systems that conform to this standard shall support values of the C  
 1417 optional number of bytes to be converted corresponding to the number of C  
 1418 bytes in the C-language types *char*, *short*, *int*, and *long*. These C  
 1419 numbers can also be specified by an application as the characters *C*, *S*, C  
 1420 *I*, and *L*, respectively. The implementation shall also support the C  
 1421 values 1, 2, and 4, even if it provides no C-Language types of those sizes. C

1422 The byte order used when interpreting numeric values is implementation C  
 1423 defined, but shall correspond to the order in which a constant of the C  
 1424 corresponding type is stored in memory on the system. C

1425 *Editor's Note: The following rationale will be added to E.4.45, but is kept here with*  
 1426 *od for this draft:*

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

1427 **od Rationale.** (*This subclause is not a part of P1003.2b*)

1428 The original standard specified `-t o2` as the default when no output type was  
 1429 given. This was changed to `-t oS` (the length of a *short*) to accommodate a super-  
 1430 computer implementation that historically used 64 b as its default (and that  
 1431 defined *shorts* as 64 b). This change should not affect portable applications. The  
 1432 requirement to support lengths of 1, 2, and 4 was added at the same time to  
 1433 address an historical implementation that had no two-byte data types in its C  
 1434 compiler.

#### 1435 **4.48 pax – Portable archive interchange**

1436 Editor's Note: This note is a road map to the many changes in `pax` proposed by  
 1437 this draft. In Draft 11, the volume of changes became such that I chose to B  
 1438 integrate the changes in with the original `pax` text from the 1992 standard. All of B  
 1439 the `pax` rationale is now merged into E.4.48. In the merged normative and B  
 1440 rationale text, only the changes from Draft 11 onwards are diff-marked. As is B  
 1441 standard with recirculation ballots, only diff-marked text is subject to objections. B

- 1442 (1) Support has been added for symbolic links in the options and interchange  
 1443 formats.
- 1444 (2) A new format has been devised, based on extensions to `ustar`. This new  
 1445 format should satisfy the following requirement from ISO/IEC 9945-  
 1446 2:1993 Annex H.1: (13) The `pax` utility should provide a new file inter-  
 1447 change format, in addition to `cpio` and `ustar`, that allows extended  
 1448 characters in file, user, and group names. Rules should be given for the  
 1449 cases where an archived name cannot be represented by the local charac-  
 1450 ter set in the file system.
- 1451 (3) The descriptions of the `ustar` and `cpio` formats have been moved from  
 1452 Sections 10.1.1 and 10.1.2 of POSIX.1 {8}, but have been cleaned up in  
 1453 three areas:
  - 1454 (a) Rather than referring to a generic “reading or writing utility,” they  
 1455 refer directly to `pax`.
  - 1456 (b) Some instances in POSIX.1 where “byte” had not been expressed  
 1457 correctly as “octet” have been converted.
  - 1458 (c) The C-language header file orientation has been converted to a  
 1459 more tabular approach.

1460 This converted text is intended to have no normative difference from that  
 1461 in POSIX.1 {8}.
- 1462 (4) References to the “extended” `tar` and `cpio` formats derived from  
 1463 POSIX.1 {8} have been changed to remove the “extended” adjective  
 1464 because this could cause confusion with the extended `tar` header added  
 1465 in this revision. (All references to `tar` are actually to `ustar`).

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1466 (5) In Draft 11, the `-o invalid=` option was added to address Canadian B  
 1467 National Body concerns about overwriting existing files, expressed origi- B  
 1468 nally during international balloting on the `tar` and `cpio` formats in B  
 1469 POSIX.1 {8}. Also, various numeric fields were added to the extended B  
 1470 header record to allow for the cases where the original `ustar` format was B  
 1471 too small; this was prompted by communications from a group designing B  
 1472 support for very large files. B

#### 1473 4.48.1 Synopsis

1474 `pax [-cdnv] [-H | -L] [-f archive] [-s replstr]... [pattern ...]` B  
 1475 `pax -r [-cdiknuv] [-H | -L] [-f archive] [-o options]... [-p string]... B  
 1476 [-s replstr]... [pattern ...]` B  
 1477 `pax -w [-dituvX] [-H | -L] [-b blocksize] [ [-a] [-f archive] ] B  
 1478 [-o options]... [-s replstr]... [-x format] [file ...]` B  
 1479 `pax -r -w [-dikltuvX] [-H | -L] [-p string]... [-s replstr]... [file ...]` B  
 1480 `directory` B

#### 1481 4.48.2 Description

1482 The `pax` utility shall read, write, and write lists of the members of archive files  
 1483 and copy directory hierarchies. A variety of archive formats shall be supported;  
 1484 see 4.48.7. B

1485 The action to be taken depends on the presence of the `-r` and `-w` options. The  
 1486 four combinations of `-r` and `-w` are referred to as the four modes of operation: *list*,  
 1487 *read*, *write*, and *copy* modes, corresponding respectively to the four forms shown  
 1488 in 4.48.1.

1489 *list* In list mode (when neither the `-r` option nor the `-w` option is  
 1490 specified), `pax` shall write the names of the members of the archive  
 1491 file read from the standard input, with pathnames matching the  
 1492 specified patterns, to standard output. If a named file is of type  
 1493 directory, the file hierarchy rooted at that file shall be listed as well. B

1494 *read* In read mode (when `-r` is specified, but `-w` is not), `pax` shall extract  
 1495 the members of the archive file read from the standard input, with  
 1496 pathnames matching the specified patterns. If an extracted file is of  
 1497 type directory, the file hierarchy rooted at that file shall be extracted  
 1498 as well. The extracted files shall be created relative to the current  
 1499 file hierarchy.

1500 The ownership, access and modification times, and file mode of the  
 1501 restored files are discussed under the `-p` option.

1502 *write* In write mode (when `-w` is specified, but `-r` is not), `pax` shall write  
1503 the contents of the file operands to the standard output in an archive  
1504 format. If no *file* operands are specified, a list of files to copy, one per  
1505 line, shall be read from the standard input. A file of type directory  
1506 shall include all of the files in the file hierarchy rooted at the file.

1507 *copy* In copy mode (when both `-r` and `-w` are specified), `pax` shall copy the  
1508 *file* operands to the destination directory.

1509 If no *file* operands are specified, a list of files to copy, one per line,  
1510 shall be read from the standard input. A file of type directory shall  
1511 include all of the files in the file hierarchy rooted at the file.

1512 The effect of the copy shall be as if the copied files were written to an  
1513 archive file and then subsequently extracted, except that there may  
1514 be hard links between the original and the copied files. If the desti-  
1515 nation directory is a subdirectory of one of the files to be copied, the  
1516 results are unspecified. If the destination directory is a file of a type  
1517 not defined by POSIX.1 {8}, the results are implementation defined;  
1518 otherwise, it shall be an error for the file named by the *directory*  
1519 operand not to exist, not be writable by the user, or not be a file of  
1520 type directory.

1521 In read or copy modes, if intermediate directories are necessary to extract an  
1522 archive member, `pax` shall perform actions equivalent to the POSIX.1 {8} `mkdir()`  
1523 function, called with the following arguments:

- 1524 — The intermediate directory used as the *path* argument.
- 1525 — The value of the bitwise inclusive OR of `S_IRWXU`, `S_IRWXG`, and `S_IRWXO` C  
1526 as the *mode* argument. C

1527 If any specified *pattern* or *file* operands are not matched by at least one file or  
1528 archive member, `pax` shall write a diagnostic message to standard error for each  
1529 one that did not match and exit with a nonzero exit status.

1530 The archive formats described in 4.48.7 shall be automatically detected on input. B  
1531 The default output archive format shall be implementation defined.

1532 A single archive can span multiple files. The `pax` utility shall determine, in an  
1533 implementation-defined manner, what file to read or write as the next file.

1534 If the selected archive format supports the specification of linked files, it shall be  
1535 an error if these files cannot be linked when the archive is extracted. For archive B  
1536 formats that do not store file contents with each name that causes a hard link, if B  
1537 the file that contains the data is not extracted during this `pax` session, either the B  
1538 data shall be restored from the original file, or a diagnostic message shall be B  
1539 displayed with the name of a file that can be used to extract the data. B

1540 In traversing directories, `pax` shall detect infinite loops; i.e., entering a previously B  
1541 visited directory that is an ancestor of the last file visited. When it detects an B  
1542 infinite loop, `pax` shall write a diagnostic message to standard error and shall  
1543 terminate.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1544 **4.48.3 Options**

1545 The `pax` utility shall conform to the utility argument syntax guidelines described  
 1546 in 2.10.2, except that the order of presentation of the `-o`, `-p`, and `-s` options is  
 1547 significant. B  
B

1548 The following options shall be supported by the implementation:

1549 `-r` Read an archive file from standard input.

1550 `-w` Write files to the standard output in the specified archive format.

1551 `-a` Append files to the end of the archive. It is implementation  
 1552 defined which devices on the system support appending. Addi-  
 1553 tional file formats unspecified by this standard may impose res-  
 1554 trictions on appending.

1555 `-b` *blocksize*

1556 Block the output at a positive decimal integer number of bytes per  
 1557 write to the archive file. Devices and archive formats may impose  
 1558 restrictions on blocking. Blocking shall be automatically deter-  
 1559 mined on input. Conforming POSIX.2 applications shall not  
 1560 specify a *blocksize* value larger than 32 256 B. Default blocking  
 1561 when creating archives depends on the archive format. (See the  
 1562 `-x` option below.)

1563 `-c` Match all archive members except those specified by the *pattern*  
 1564 operands.

1565 `-d` Cause files of type directory being copied or archived or archive  
 1566 members of type directory being extracted or listed to match only B  
 1567 the file or archive member itself and not the file hierarchy rooted  
 1568 at the file.

1569 `-f` *archive* Specify the pathname of the input or output archive, overriding  
 1570 the default standard input (in list or read modes) or standard out-  
 1571 put (write mode).

1572 `-H` If a symbolic link referencing a file of type directory is specified on  
 1573 the command line, `pax` shall archive the file hierarchy rooted in  
 1574 the file referenced by the link, using the name of the link as the  
 1575 root of the file hierarchy. The default behavior shall be to archive  
 1576 the symbolic link itself.

1577 `-i` Interactively rename files or archive members. For each archive  
 1578 member matching a *pattern* operand or file matching a *file*  
 1579 operand, a prompt shall be written to the file `/dev/tty`. The  
 1580 prompt shall contain the name of the file or archive member, but  
 1581 the format is otherwise unspecified. A line shall then be read  
 1582 from `/dev/tty`. If this line is blank, the file or archive member  
 1583 shall be skipped. If this line consists of a single period, the file or  
 1584 archive member shall be processed with no modification to its  
 1585 name. Otherwise, its name shall be replaced with the contents of



1586 the line. The `pax` utility shall immediately exit with a nonzero  
 1587 exit status if end-of-file is encountered when reading a response  
 1588 or if `/dev/tty` cannot be opened for reading and writing.

1589 The results of extracting a hard link to a file that has been B  
 1590 renamed during extraction are unspecified.

1591 `-k` Prevent the overwriting of existing files.

1592 `-l` (The letter ell.) In copy mode, hard links shall be made between B  
 1593 the source and destination file hierarchies whenever possible.

1594 `-L` If a symbolic link referencing a file of type directory is specified on  
 1595 the command line or encountered during the traversal of a file  
 1596 hierarchy, `pax` shall archive the file hierarchy rooted in the file  
 1597 referenced by the link, using the name of the link as the root of  
 1598 the file hierarchy. The default behavior shall be to archive the  
 1599 symbolic link itself.

1600 `-n` Select the first archive member that matches each *pattern*  
 1601 operand. No more than one archive member shall be matched for  
 1602 each pattern (although members of type directory shall still  
 1603 match the file hierarchy rooted at that file).

1604 `-o options` Provide information to the implementation to modify the algo-  
 1605 rithm for extracting or writing files. The value of *options* shall  
 1606 consist of one or more comma-separated keywords of the form:

1607  $keyword[[[:]=value][, keyword[[[:]=value], \dots ]$

1608 Some keywords apply only to certain file formats, as indicated B  
 1609 with each description. Use of keywords that are inapplicable to B  
 1610 the file format being processed produces undefined results. B

1611 Keywords in the options argument shall be a string that would be C  
 1612 a valid portable filename as described in portable filename char- C  
 1613 acter set (see 2.2.2.131). C

1614 NOTE: Keywords are not expected to be filenames, merely to follow the same C  
 1615 character composition rules as portable filenames. C

1616 Keywords can be preceded with white space. The *value* field shall  
 1617 consist of zero or more characters; within *value*, the application  
 1618 shall precede any literal comma with a backslash, which shall be  
 1619 ignored, but preserves the comma as part of *value*. A comma as  
 1620 the final character, or a comma followed solely by white space as  
 1621 the final characters, in *options* shall be ignored. Multiple `-o`  
 1622 options can be specified; if keywords given to these multiple `-o`  
 1623 options conflict, the keywords and values appearing later in  
 1624 command-line sequence shall take precedence and the earlier  
 1625 shall be silently ignored. The following keyword values of *options*  
 1626 shall be supported for the file formats as indicated:

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

1627 `delete=pattern`  
 1628 (Applicable only to the `-x pax` format.) When used in  
 1629 write or copy mode, `pax` shall omit from extended  
 1630 header records that it produces any keywords matching  
 1631 the string *pattern*. When used in read or list mode, `pax`  
 1632 shall ignore any keywords matching the string *pattern*  
 1633 in the extended header records. In both cases, matching  
 1634 shall be performed using the pattern matching notation  
 1635 described in 3.13.1 and 3.13.2. For example,  
 1636 `-o delete=security.*`  
 1637 would suppress security-related information. See  
 1638 4.48.7.1.2 for extended header record keyword usage.

1639 `exthdr.name=string`  
 1640 (Applicable only to the `-x pax` format.) This keyword B  
 1641 allows user control over the name that is written into B  
 1642 the `ustar` header blocks for the extended header B  
 1643 records produced under the circumstances described in B  
 1644 4.48.7.1.1. The name shall be the contents of *string*,  
 1645 after the following character substitutions have been  
 1646 made:

<b><i>string</i> Includes</b>	<b>Replaced By</b>
<code>%d</code>	The directory name of the file, equivalent to the result of the <code>dirname</code> utility on the translated pathname.
<code>%f</code>	The filename of the file, equivalent to the result of the <code>basename</code> utility on the translated pathname.
<code>%%</code>	A <code>%</code> character.

1647  
 1648  
 1649  
 1650  
 1651  
 1652  
 1653  
 1654

1655 Any other `%` characters in *string* produce undefined  
 1656 results.

1657 If no `-o exthdr.name=string` is specified, `pax` shall  
 1658 use the following default value:  
 1659 `%d/PaxHeaders/%f`

1660 `globexthdr.name=string`  
 1661 (Applicable only to the `-x pax` format.) When used in  
 1662 write or copy mode with the appropriate options, `pax`  
 1663 creates global extended header records with `ustar`  
 1664 header blocks that will be treated as regular files by  
 1665 previous versions of `pax`. This keyword allows user con-  
 1666 trol over the name that is written into the `ustar`  
 1667 header blocks for global extended header records. The  
 1668 name shall be the contents of *string*, after the following  
 1669 character substitutions have been made:

1670	<b><u>string Includes</u></b>	<b><u>Replaced By</u></b>	
1671	%n	An integer that represents the sequence	
1672		number of the global extended header	
1673		record in the archive, starting at 1.	
1674	%%	A % character.	
1675	Any other % characters in <i>string</i> produce undefined		
1676	results.		
1677	If no <code>-o globexthdr.name=string</code> is specified, <code>pax</code>		
1678	shall use the following default value:		
1679	\$TMPDIR/GlobalHead.%n		
1680	where \$TMPDIR represents the value of the <b>TMPDIR</b>		
1681	environment variable. If <b>TMPDIR</b> is not set, <code>pax</code> shall		
1682	use <code>/tmp</code> .		
1683	<code>invalid=action</code>		B
1684	(Applicable only to the <code>-x pax</code> format.) This keyword		B
1685	allows user control over the action <code>pax</code> takes upon		B
1686	encountering values in an extended header record that,		B
1687	in read or copy mode, are invalid in the destination		B
1688	hierarchy or, in list mode, cannot be written in the		B
1689	codeset and current locale of the implementation. The		B
1690	following are invalid values that shall be recognized by		B
1691	<code>pax</code> :		B
1692	— In read or copy mode, a file name or link name that		B
1693	contains character encodings invalid in the destina-		B
1694	tion hierarchy. (For example, the name may contain		B
1695	embedded NULs.)		B
1696	— In read or copy mode, a file name or link name that		B
1697	is longer than the maximum allowed in the destina-		B
1698	tion hierarchy (for either a pathname component or		B
1699	the entire pathname).		B
1700	— In list mode, any character string value (file name,		B
1701	link name, user name, etc.) that cannot be written		B
1702	in the codeset and current locale of the		B
1703	implementation.		B
1704	The following mutually exclusive values of the <i>action</i>		B
1705	argument are supported:		B
1706	<code>bypass</code>		B
1707	In read or copy mode, <code>pax</code> shall bypass the file,		B
1708	causing no change to the destination hierarchy.		B
1709	In list mode, <code>pax</code> shall write all requested valid		B
1710	values for the file, but its method for writing		B
1711	invalid values is unspecified.		B

1712	rename	B
1713	In read or copy mode, <code>pax</code> shall act as if the <code>-i</code>	B
1714	option were in effect for each file with invalid file	B
1715	name or link name values, allowing the user to	B
1716	provide a replacement name interactively. In list	B
1717	mode, <code>pax</code> shall behave identically to the <code>bypass</code>	B
1718	action.	B
1719	UTF8	B
1720	When used in read, copy, or list mode and a file	B
1721	name, link name, owner name, or any other field	B
1722	in an extended header record cannot be	B
1723	translated from the <code>pax</code> UTF8 codeset format to	B
1724	the codeset and current locale of the implementa-	B
1725	tion, <code>pax</code> shall use the actual UTF8 encoding for	B
1726	the name.	B
1727	write	B
1728	In read or copy mode, <code>pax</code> shall write the file,	B
1729	translating or truncating the name, regardless of	B
1730	whether this may overwrite an existing file with	B
1731	a valid name. In list mode, <code>pax</code> shall behave	B
1732	identically to the <code>bypass</code> action.	B
1733	If no <code>-o invalid=</code> option is specified, <code>pax</code> shall act as	B
1734	if <code>-o invalid=bypass</code> were specified. Any overwrit-	B
1735	ing of existing files that may be allowed by the <code>-o</code>	B
1736	<code>invalid=</code> actions shall be subject to permission ( <code>-p</code> )	B
1737	and modification time ( <code>-u</code> ) restrictions, and shall be	B
1738	suppressed if the <code>-k</code> option is also specified.	B
1739	linkdata	
1740	(Applicable only to the <code>-x pax</code> format.) In write mode,	
1741	<code>pax</code> shall write the contents of a file to the archive even	
1742	when that file is merely a hard link to a file whose con-	
1743	tents have already been written to the archive.	
1744	listopt= <i>format</i>	
1745	This keyword specifies the output format of the table of	
1746	contents produced when the <code>-v</code> option is specified in list	
1747	mode. See 4.48.3.1. To avoid ambiguity, the	
1748	listopt= <i>format</i> shall be the only or final	
1749	keyword= <i>value</i> pair in a <code>-o</code> option-argument; all charac-	
1750	ters in the remainder of the option-argument shall be	
1751	considered part of the <i>format</i> string. When multiple	
1752	<code>-o listopt=<i>format</i></code> options are specified, the <i>format</i>	
1753	strings shall be considered a single, concatenated	
1754	string, evaluated in command-line order.	
1755	times	
1756	(Applicable only to the <code>-x pax</code> format.) When used in	

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1757 write or copy mode, `pax` shall include `atime`, `ctime`,  
 1758 and `mtime` extended header records for each file. See  
 1759 4.48.7.1.4.

1760 In addition to these keywords, if the `-x pax` format is specified,  
 1761 any of the keywords and values defined in 4.48.7.1.2, including  
 1762 implementation extensions, can be used in `-o` option-arguments,  
 1763 in either of two modes:

1764 *keyword=value*

1765 When used in write or copy mode, these keyword/value  
 1766 pairs shall be included at the beginning of the archive  
 1767 as *typeflag* `g` global extended header records. When  
 1768 used in read or list mode, these keyword/value pairs  
 1769 shall act as if they had been at the beginning of the  
 1770 archive as *typeflag* `g` global extended header records.

1771 *keyword:=value*

1772 When used in write or copy mode, these keyword/value  
 1773 pairs shall be included as records at the beginning of a  
 1774 *typeflag* `x` extended header for each file. (This is  
 1775 equivalent to the equal-sign form except that it creates  
 1776 no *typeflag* `g` global extended header records.) When  
 1777 used in read or list mode, these keyword/value pairs  
 1778 shall act as if they were included as records at the end  
 1779 of each extended header; thus, they shall override any  
 1780 global or file-specific extended header record keywords  
 1781 of the same names. For example, in the command

```
1782 pax -r -o "  
1783 gname:=mygroup,  
1784 " <archive
```

1785 the group name will be forced to a new value for all files  
 1786 read from the archive.

1787 The precedences of `-o` keywords over various fields in the archive  
 1788 are described in 4.48.7.1.3.

1789 `-p string` Specify one or more file characteristic options (privileges). The  
 1790 *string* option-argument shall be a string specifying file charac-  
 1791 teristics to be retained or discarded on extraction. The string  
 1792 shall consist of the specification characters `a`, `e`, `m`, `o`, and `p`,  
 1793 and/or other implementation-defined characters. Multiple  
 1794 characteristics can be concatenated within the same string, and  
 1795 multiple `-p` options can be specified. The meanings of the  
 1796 specification characters are as follows:

- 1797 `a` Do not preserve file access times.
- 1798 `e` Preserve the user ID, group ID, file mode bits (see  
 1799 2.2.2.71), access time, modification time, and any other  
 1800 implementation-defined file characteristics.

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

- 1801           m     Do not preserve file modification times.
- 1802           o     Preserve the user ID and group ID.
- 1803           p     Preserve the file mode bits. Other, implementation-  
1804                 defined file-mode attributes may be preserved.
- 1805           In the preceding list, “preserve” indicates that an attribute stored  
1806           in the archive shall be given to the extracted file, subject to the  
1807           permissions of the invoking process. The access and modification    B  
1808           times of the file shall be preserved unless otherwise specified with    B  
1809           the `-p` option or not stored in the archive. All attributes that are    B  
1810           not preserved shall be determined as part of the normal file crea-    B  
1811           tion action (see 2.9.1.4).    B
- 1812           If neither the `e` nor the `o` specification character is specified, or  
1813           the user ID and group ID are not preserved for any reason, `pax`  
1814           shall not set the `S_ISUID` and `S_ISGID` bits of the file mode.
- 1815           If the preservation of any of these items fails for any reason, `pax`  
1816           shall write a diagnostic message to standard error. Failure to  
1817           preserve these items shall affect the final exit status, but shall  
1818           not cause the extracted file to be deleted.
- 1819           If file-characteristic letters in any of the *string* option-arguments  
1820           are duplicated or conflict with each other, the one(s) given last  
1821           shall take precedence. For example, if `-p eme` is specified, file  
1822           modification times shall be preserved.
- 1823    `-s replstr`   Modify file or archive member names named by *pattern* or *file*  
1824                   operands according to the substitution expression *replstr*, using  
1825                   the syntax of the `ed` utility (see 4.20). The concepts of “address”  
1826                   and “line” are meaningless in the context of the `pax` utility and  
1827                   shall not be supplied. The format shall be
- 1828                                 `-s /old/new/[gp]`
- 1829           where (as in `ed`) *old* is a BRE and *new* can contain an ampersand,  
1830           \*n* (where *n* is a digit) backreferences, or subexpression matching.  
1831           The *old* string also shall be permitted to contain `<newline>`  
1832           characters.
- 1833           Any nonnull character can be used as a delimiter (`/` shown here).  
1834           Multiple `-s` expressions can be specified; the expressions shall be  
1835           applied in the order specified, terminating with the first success-  
1836           ful substitution. The optional trailing `g` shall be as defined in the  
1837           `ed` utility. The optional trailing `p` shall cause successful substitu-  
1838           tions to be written to standard error. File or archive member  
1839           names that are replaced with the empty string shall be ignored    B  
1840           when reading and writing archives.

1841	-t	Cause the access times of the archived files to be the same as they	
1842		were before being read by <code>pax</code> .	
1843	-u	Ignore files that are older (having a less recent file modification	
1844		time) than a pre-existing file or archive member with the same	
1845		name. In read mode, an archive member with the same name as	
1846		a file in the file system shall be extracted if the archive member is	
1847		newer than the file. In write mode, an archive file member with	
1848		the same name as a file in the file system shall be superseded if	
1849		the file is newer than the archive member. If <code>-a</code> is also specified,	B
1850		this is accomplished by appending to the archive; otherwise, it is	B
1851		unspecified if this is accomplished by actual replacement in the	
1852		archive or by appending to the archive. In copy mode, the file in	
1853		the destination hierarchy shall be replaced by the file in the	
1854		source hierarchy or by a link to the file in the source hierarchy if	
1855		the file in the source hierarchy is newer.	
1856	-v	In list mode, produce a verbose table of contents (see 4.48.6.1).	
1857		Otherwise, write archive member pathnames to standard error	
1858		(see 4.48.6.2).	
1859	-x <i>format</i>	Specify the output archive format. The <code>pax</code> utility shall support	B
1860		the following formats:	B
1861		<code>cpio</code> The <code>cpio</code> interchange format specified in 4.48.7.3.	
1862		The default <i>blocksize</i> for this format for character	
1863		special archive files shall be 5120 B. Implementa-	
1864		tions shall support all <i>blocksize</i> values less than or	
1865		equal to 32 256 B that are multiples of 512 B.	
1866		<code>pax</code> The interchange format specified in 4.48.7.1, based	
1867		on an extension to the <code>ustar</code> format. The default	
1868		<i>blocksize</i> for this format for character special archive	
1869		files shall be 10 240 B. Implementations shall sup-	
1870		port all <i>blocksize</i> values less than or equal to	
1871		32 256 B that are multiples of 512 B.	
1872		<code>ustar</code> The <code>ustar</code> interchange format specified in 4.48.7.2.	
1873		The default <i>blocksize</i> for this format for character	
1874		special archive files shall be 10 240 B. Implementa-	
1875		tions shall support all <i>blocksize</i> values less than or	
1876		equal to 32 256 B that are multiples of 512 B.	
1877		Implementation-defined formats shall specify a default block size	
1878		as well as any other block sizes supported for character special	
1879		archive files.	
1880		Any attempt to append to an archive file in a format different	
1881		from the existing archive format shall cause <code>pax</code> to exit immedi-	
1882		ately with a nonzero exit status.	
1883		In copy mode, if no <code>-x format</code> is specified, <code>pax</code> shall behave as if	B
1884		<code>-x pax</code> were specified.	B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1885        -X            When traversing the file hierarchy specified by a pathname, `pax`  
 1886                      shall not descend into directories that have a different device ID  
 1887                      [`st_dev`, see POSIX.1 {8} `stat()`].

1888        The options that operate on the names of files or archive members (`-c`, `-i`, `-n`, `-s`,  
 1889        `-u`, and `-v`) shall interact as follows. In read mode, the archive members shall be  
 1890        “selected” based on the user-specified *pattern* operands as modified by the `-c`, `-n`,  
 1891        and `-u` options. Then, any `-s` and `-i` options shall modify, in that order, the  
 1892        names of the selected files. The `-v` option shall write names resulting from these  
 1893        modifications.

1894        In write mode, the files shall be selected based on the user-specified pathnames as  
 1895        modified by the `-n` and `-u` options. Then, any `-s` and `-i` options shall, in that  
 1896        order, modify the names of these selected files. The `-v` option shall write names  
 1897        resulting from these modifications.

1898        If both the `-u` and `-n` options are specified, `pax` shall not consider a file selected  
 1899        unless it is newer than the file to which it is compared.

#### 1900        **4.48.3.1 List-Mode Format Specifications**

1901        In list mode with the `-o listopt=format` option, the *format* argument shall be  
 1902        applied for each selected file. The `pax` utility shall append a <newline> charac-  
 1903        ter to the `listopt` output for each selected file. B

1904        The *format* argument shall be used as the *format* string described in 2.12, with B  
 1905        the exceptions (1) through (5) defined in 4.50.7, plus the following exceptions:

1906            (6) The sequence (*keyword*) can occur before a format conversion specifier.  
 1907            The conversion argument is defined by the value of *keyword*. The imple-  
 1908            mentation shall support the following keywords:

1909            — Any of the Field Name entries in Table 4-100 and Table 4-102. The  
 1910            implementation may support the `cpio` keywords without the leading  
 1911            `c_` in addition to the form required by Table 4-102.

1912            — Any keyword defined for the the extended header in 4.48.7.1.2.

1913            — Any keyword provided as an implementation-defined extension within  
 1914            the extended header defined in 4.48.7.1.2.

1915            For example, the sequence `%(charset)s` is the string value of the name  
 1916            of the character set in the extended header.

1917            The result of the keyword conversion argument shall be the value from  
 1918            the applicable header field or extended header, without any trailing  
 1919            NULs.

1920            All keyword values used as conversion arguments shall be translated  
 1921            from the UTF8 encoding to the character set appropriate for the local file  
 1922            system, user database, etc., as applicable.

1923            (7) An additional conversion character, `T`, shall be used to specify time for-  
 1924            mats. The `T` conversion character can be preceded by the sequence

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



- 1925           (*keyword*=*subformat*), where *subformat* is a date format as defined by  
 1926           4.15.4.1. The default *keyword* shall be `mtime` and the default *subformat*  
 1927           shall be: `%b %e %H:%M %Y`. B
- 1928       (8) An additional conversion character, `M`, shall be used to specify the file B  
 1929       mode string as defined in 4.39.6.1. If (*keyword*) is omitted, the `mode` B  
 1930       keyword shall be used. For example, `%.1M` writes the single character B  
 1931       corresponding to the `<entry type>` field of the `ls -l` command.
- 1932       (9) An additional conversion character, `D`, shall be used to specify the device  
 1933       for block or special files, if applicable, in an implementation-defined for-  
 1934       mat. If not applicable, and (*keyword*) is specified, then this conversion  
 1935       shall be equivalent to `%(keyword)u`. If not applicable, and (*keyword*) is  
 1936       omitted, then this conversion shall be equivalent to `<space>`.
- 1937       (10) An additional conversion character, `F`, shall be used to specify a path-  
 1938       name. The `F` conversion character can be preceded by a sequence of  
 1939       comma separated keywords:  
 1940                (*keyword*[*keyword*] ... )
- 1941       The values for all the keywords that are non-null shall be concatenated  
 1942       together, each separated by a `/`. The default shall be (`path`) if the key-  
 1943       word `path` is defined; otherwise, the default shall be (`prefix,name`).
- 1944       (11) An additional conversion character, `L`, shall be used to specify a symbolic  
 1945       link expansion. If the current file is a symbolic link, then `%L` shall  
 1946       expand to:  
 1947                "`%s -> %s`", *<value of keyword>*, *<contents of link>*
- 1948       Otherwise, the `%L` conversion character shall be the equivalent of `%F`. B

#### 1949 4.48.4 Operands

1950 The following operands shall be supported by the implementation:

- 1951       *directory*   The destination directory pathname for copy mode.
- 1952       *file*        A pathname of a file to be copied or archived.
- 1953       *pattern*     A pattern matching one or more pathnames of archive members.  
 1954       A pattern shall be given in the name-generating notation of the  
 1955       pattern matching notation in 3.13, including the filename expansion  
 1956       rules in 3.13.3. The default, if no *pattern* is specified, is to  
 1957       select all members in the archive.

1958 **4.48.5 External Influences**1959 **4.48.5.1 Standard Input**

1960 In write mode, the standard input shall be used only if no *file* operands are  
 1961 specified. It shall be a text file containing a list of pathnames, one per line,  
 1962 without leading or trailing <blank>s.

1963 In list and read modes, if *-f* is not specified, the standard input shall be an  
 1964 archive file. (See 4.48.5.2.)

1965 Otherwise, the standard input shall not be used.

1966 **4.48.5.2 Input Files**

1967 The input file named by the *archive* option-argument, or standard input when the  
 1968 archive is read from there, shall be a file formatted according to one of the  
 1969 specifications in 4.48.7 or some other implementation-defined format. B

1970 The file `/dev/tty` shall be used to write prompts and read responses.

1971 **4.48.5.3 Environment Variables**

1972 The following environment variables shall affect the execution of `pax`:

1973     **LANG**                 This variable shall determine the locale to use for the  
 1974                                locale categories when both **LC\_ALL** and the correspond-  
 1975                                ing environment variable (beginning with **LC\_**) do not  
 1976                                specify a locale. See 2.6.

1977     **LC\_ALL**                This variable shall determine the locale to be used to over-  
 1978                                ride any values for locale categories specified by the set-  
 1979                                tings of **LANG** or any environment variables beginning  
 1980                                with **LC\_**.

1981     **LC\_COLLATE**         This variable shall determine the locale for the behavior of  
 1982                                ranges, equivalence classes, and multicharacter collating  
 1983                                elements used in the pattern matching expressions for the  
 1984                                *pattern* operand, the BRE for the *-s* option, and the ERE  
 1985                                defined for the *yesexpr* locale keyword in the  
 1986                                **LC\_MESSAGES** category.

1987     **LC\_CTYPE**            This variable shall determine the locale for the interpreta-  
 1988                                tion of sequences of bytes of text data as characters (e.g.,  
 1989                                single- versus multibyte characters in arguments and  
 1990                                input files) and the behavior of character classes within  
 1991                                REs and pattern matching.

1992     **LC\_MESSAGES**        This variable shall determine the processing of affirmative  
 1993                                responses and the language in which messages should be  
 1994                                written.

1995        **LC\_TIME**                This variable shall determine the format and contents of  
1996                                date and time strings when the `-v` option is specified.

1997        **TMPDIR**                    This variable shall be interpreted as a pathname that pro-  
1998                                vides part of the default global extended header record file  
1999                                name, as described for the `-o globexthdr=` keyword in  
2000                                4.48.3.

#### 2001        **4.48.5.4 Asynchronous Events**

2002        Default.

#### 2003        **4.48.6 External Effects**

##### 2004        **4.48.6.1 Standard Output**

2005        In write mode, if `-f` is not specified, the standard output shall be the archive for-  
2006        matted according to one of the specifications in 4.48.7 or some other  
2007        implementation-defined format. (See `-x format` under 4.48.3.)

2008        In list mode, when the `-o listopt=format` option has been specified, the  
2009        selected archive members shall be written to standard output using the format  
2010        described in 4.48.3.1. In list mode without the `-o listopt=format` option, the  
2011        table of contents of the selected archive members shall be written to standard out-  
2012        put using the following format:

2013                "`%s\n`", *<pathname>*

2014        If the `-v` option is specified in list mode, the table of contents of the selected  
2015        archive members shall be written to standard output using the following formats:

2016        For pathnames representing hard links to previous members of the archive:

2017                "`%sΔ==Δ%s\n`", *<ls -l listing>*, *<linkname>*

2018        For all other pathnames:

2019                "`%s\n`", *<ls -l listing>*

2020        where *<ls -l listing>* shall be the format specified by the `ls` utility (see 4.39) with  
2021        the `-l` option. When writing pathnames in this format, it is unspecified what is  
2022        written for fields for which the underlying archive format does not have the  
2023        correct information, although the correct number of *<blank>*-separated fields  
2024        shall be written.

2025        In list mode, standard output shall not be buffered more than one line at a time.

##### 2026        **4.48.6.2 Standard Error**

2027        If `-v` is specified in read, write, or copy modes, `pax` shall write the pathnames it  
2028        processes to standard error using the following format:

2029           "%s\n", <pathname>

2030       These pathnames shall be written as soon as processing is begun on the file or  
2031       archive member and shall be flushed to standard error. The trailing <newline>,  
2032       which shall not be buffered, shall be written when the file has been read or  
2033       written.

2034       If the `-s` option is specified, and the replacement string has a trailing `p`, substitu-  
2035       tions shall be written to standard error in the following format:

2036           "%sΔ>>Δ%s\n", <original pathname>, <new pathname>

2037       In all operating modes of `pax` (see 4.48.2), optional messages of unspecified format  
2038       concerning the input archive format and volume number, and the number of files,  
2039       blocks, volumes, and media parts, as well as other diagnostic messages, may be  
2040       written to standard error.

2041       In all formats, for both standard output and standard error, it is unspecified how  
2042       nonprintable characters in pathnames or linknames are written.

2043       *Editor's Note: The Draft 10 editing instructions mistakenly called for the following* B  
2044       *paragraph to replace all of 4.48.6.2. I believe the correct action is merely to add it* B  
2045       *to the end of the subclause, as I've done here.* B

2046       When `pax` is in read mode or list mode, using the `-x pax` archive format, and a  
2047       file name, link name, owner name, or any other field in an extended header record  
2048       cannot be translated from the `pax` UTF8 codeset format to the codeset and current  
2049       locale of the implementation, `pax` shall write a diagnostic message to standard B  
2050       error, shall process the file as described for the `-o invalid=` option, and then B  
2051       shall process the next file in the archive.

### 2052       **4.48.6.3 Output Files**

2053       In read mode, the extracted output files shall be of the archived file type. In copy B  
2054       mode, the copied output files shall be the type of the file being copied. In either B  
2055       mode, existing files in the destination hierarchy shall be overwritten only when B  
2056       all permission (`-p`), modification time (`-u`), and invalid-value (`-o invalid=`) tests B  
2057       allow it. B

2058       In write mode, the output file named by the `-f` option argument shall be a file for-  
2059       matted according to one of the specifications in 4.48.7 or some other  
2060       implementation-defined format.

## 2061       **4.48.7 Extended Description**

### 2062       **4.48.7.1 pax Interchange Format**

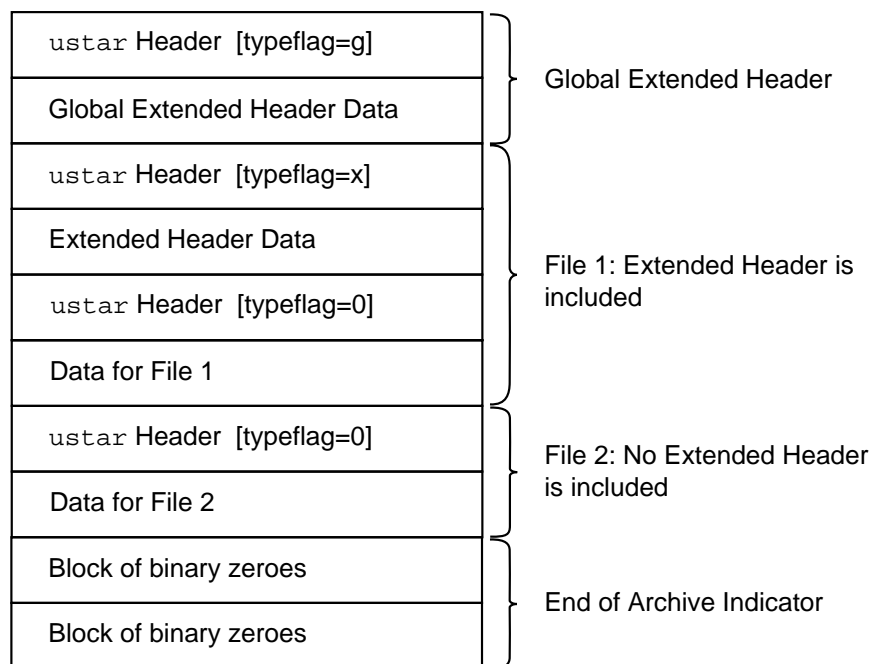
2063       A `pax` archive tape or file produced in the `-x pax` format shall contain a series of  
2064       blocks. The physical layout of the archive shall be identical to the `ustar` format  
2065       described in 4.48.7.2. Each file archived shall be represented by the following  
2066       sequence:

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.

- 2067 — An optional header block with extended header records. This header block  
 2068 is of the form described in 4.48.7.1.1, with a *typeflag* value of *x* or *g*. The  
 2069 extended header records, described in 4.48.7.1.2, are included as the data  
 2070 for this header block.
- 2071 — A header block that describes the file. Any fields in the preceding optional  
 2072 extended header override the associated fields in this header block for this  
 2073 file.
- 2074 — Zero or more blocks that contain the contents of the file. B

2075 At the end of the archive file there shall be two 512 B blocks filled with binary  
 2076 zeroes, interpreted as an end-of-archive indicator.

2077 A schematic of an example archive with global extended header records and two  
 2078 actual files is shown in Figure 4-1. In the example, the second file in the archive  
 2079 has no extended header preceding it, presumably because it has no need for  
 2080 extended attributes.



2081 **Figure 4-1 – pax Format Archive Example**

#### 2082 4.48.7.1.1 Header Block

2083 The header block shall be identical to the *ustar* header block described in  
 2084 4.48.7.2, except that two additional *typeflag* values are defined:

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

2085 'x' Represents extended header records for the following file in the  
 2086 archive (which shall have its own `ustar` header block). The format  
 2087 of these extended header records shall be as described in 4.48.7.1.2.

2088 'g' Represents global extended header records for the following files in  
 2089 the archive. The format of these extended header records shall be  
 2090 as described in 4.48.7.1.2. Each value shall affect all subsequent  
 2091 files that do not override that value in their own extended header  
 2092 record and until another global extended header record is reached  
 2093 that provides another value for the same field. The *typeflag* `g` global  
 2094 headers should not be used with interchange media that could  
 2095 suffer partial data loss in transporting the archive.

2096 For both of these types, the *size* field shall be the size of the extended header  
 2097 records in octets. The other fields in the header block are not meaningful to this  
 2098 version of the `pax` utility. However, if this archive is read by a `pax` utility con-  
 2099 forming to a previous version of this standard, the header block fields are used to  
 2100 create a regular file that contains the extended header records as data. There-  
 2101 fore, header block field values should be selected to provide reasonable file access  
 2102 to this regular file.

2103 A further difference from the `ustar` header block is that data blocks for files of  
 2104 *typeflag* `1` (hard link) may be included, which means that the *size* field may be  
 2105 greater than zero. Archives created by `pax -o linkdata` shall include these data  
 2106 blocks with the hard links.

#### 2107 **4.48.7.1.2 Extended Header**

2108 An extended header contains values that are inappropriate for the `ustar` header  
 2109 block because of limitations in that format: fields requiring a character encoding  
 2110 other than ISO/IEC 646 {1}; fields representing file attributes not described in the  
 2111 `ustar` header; fields whose format or length do not fit the requirements of the  
 2112 `ustar` header. The values in an extended header add attributes to the following  
 2113 file (or files—see the description of the *typeflag* `g` header block) or override values  
 2114 in the following header block(s), as indicated in the following list of keywords.

2115 An extended header shall consist of one or more records, each constructed as  
 2116 follows:

2117 "%d %s=%s\n", *<length>*, *<keyword>*, *<value>*

2118 The extended header records shall be encoded in ISO/IEC 10646 {10} Universal  
 2119 Translation Format 8 (UTF8). The *<length>*, *<blank>*s, equals sign, and *<new-*  
 2120 *line>* shown shall be limited to the portable character set, as encoded in UTF8.  
 2121 The *<keyword>* and *<value>* fields can be any UTF8 characters.

2122 The *<length>* field shall be the decimal length of the extended header record in  
 2123 octets, including the trailing *<newline>*.

2124 The *<keyword>* field shall be one of the entries from the following list or a key-  
 2125 word provided as an implementation extension. Keywords consisting entirely of  
 2126 lowercase letters, digits, and periods are reserved for future standardization. A  
 2127 keyword shall not include an equals sign. [In the following list, the notations

2128		“file(s)” or “block(s)” are used to acknowledge that a keyword affects the following															
2129		single file after a <i>typeflag</i> <i>x</i> extended header, but possibly multiple files after															
2130		<i>typeflag</i> <i>g</i> . Any requirements in the list for <i>pax</i> to include a record when in write															
2131		or copy mode shall apply only when such a record has not already been provided															
2132		through the use of the <i>-o</i> option. When used in copy mode, <i>pax</i> shall behave as if	B														
2133		an archive had been created with applicable extended header records and then	B														
2134		extracted.]	B														
2135	<i>atime</i>	The file access time for the following file(s), equivalent to the															
2136		value of the <i>st_atime</i> member of the <i>stat</i> structure for a file, as															
2137		described in POSIX.1 {8}. The access time shall be restored if the															
2138		process has the appropriate privilege required to do so. The format of the															
2139		<i>&lt;value&gt;</i> shall be as described in 4.48.7.1.4.															
2140	<i>charset</i>	The name of the character set used to encode the data in the fol-															
2141		lowing file(s). The entries in the following table are defined to															
2142		refer to known standards; additional names may be agreed on															
2143		between the originator and recipient.															
2144		<table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; border-bottom: 1px solid black;"><i>&lt;value&gt;</i></th> <th style="text-align: left; border-bottom: 1px solid black;">Formal Standard</th> </tr> </thead> <tbody> <tr> <td>ISO-IRΔ646Δ1990</td> <td>ISO/IEC 646 IRV {1}</td> </tr> <tr> <td>ISO-IRΔ8859Δ1Δ1987</td> <td>ISO 8859-1 {5}</td> </tr> <tr> <td>ISO-IRΔ8859Δ2Δ1987</td> <td>ISO 8859-2 {6}</td> </tr> <tr> <td>ISO-IRΔ10646Δ1993</td> <td>ISO/IEC 10646 {10}</td> </tr> <tr> <td>ISO-IRΔ10646Δ1993ΔUTF8</td> <td>ISO/IEC 10646 {10}, UTF8 encoding</td> </tr> <tr> <td>BINARY</td> <td>None</td> </tr> </tbody> </table>	<i>&lt;value&gt;</i>	Formal Standard	ISO-IRΔ646Δ1990	ISO/IEC 646 IRV {1}	ISO-IRΔ8859Δ1Δ1987	ISO 8859-1 {5}	ISO-IRΔ8859Δ2Δ1987	ISO 8859-2 {6}	ISO-IRΔ10646Δ1993	ISO/IEC 10646 {10}	ISO-IRΔ10646Δ1993ΔUTF8	ISO/IEC 10646 {10}, UTF8 encoding	BINARY	None	
<i>&lt;value&gt;</i>	Formal Standard																
ISO-IRΔ646Δ1990	ISO/IEC 646 IRV {1}																
ISO-IRΔ8859Δ1Δ1987	ISO 8859-1 {5}																
ISO-IRΔ8859Δ2Δ1987	ISO 8859-2 {6}																
ISO-IRΔ10646Δ1993	ISO/IEC 10646 {10}																
ISO-IRΔ10646Δ1993ΔUTF8	ISO/IEC 10646 {10}, UTF8 encoding																
BINARY	None																
2145																	
2146																	
2147																	
2148																	
2149																	
2150																	
2151		The encoding is included in an extended header for information															
2152		only; when <i>pax</i> is used as described in this standard, it shall not															
2153		translate the file data into any other encoding. The BINARY entry															
2154		indicates unencoded binary data.															
2155		When used in write or copy mode, it is implementation defined															
2156		whether <i>pax</i> includes a <i>charset</i> extended header record for a															
2157		file.															
2158	<i>comment</i>	A series of characters used as a comment. All characters in the															
2159		<i>&lt;value&gt;</i> field shall be ignored by <i>pax</i> .															
2160	<i>ctime</i>	The file creation time for the following file(s), equivalent to the															
2161		value of the <i>st_ctime</i> member of the <i>stat</i> structure for a file, as															
2162		described in POSIX.1 {8}. The creation time shall be restored if															
2163		the process has the appropriate privilege required to do so. The															
2164		format of the <i>&lt;value&gt;</i> shall be as described in 4.48.7.1.4.															
2165	<i>gid</i>	The group ID of the group that owns the file, expressed as a	B														
2166		decimal number using digits from ISO/IEC 646 {1}. This record	B														
2167		shall override the <i>gid</i> field in the following header block(s). When	B														
2168		used in write or copy mode, <i>pax</i> shall include a <i>gid</i> extended	B														
2169		header record for each file whose group ID is greater than	B														
2170		99 999 999.	B														

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2171	gname	The group of the file(s), formatted as a group name in the group database. This record shall override the <i>gid</i> and <i>gname</i> fields in the following header block(s), and any <i>gid</i> extended header record. When used in read, copy, or list mode, <i>pax</i> shall translate the name from the UTF8 encoding in the header record to the character set appropriate for the group database on the receiving system. If any of the UTF8 characters cannot be translated, and if the <code>-o invalid=UTF8</code> option is not specified, the results are implementation defined. When used in write or copy mode, <i>pax</i> shall include a <i>gname</i> extended header record for each file whose group name cannot be represented entirely with the letters and digits of the portable character set.	
2172			
2173			
2174			
2175			
2176			
2177			
2178			
2179			
2180			
2181			
2182			
2183	linkpath	The pathname of a link being created to another file, of any type, previously archived. This record shall override the <i>linkname</i> field in the following <i>ustar</i> header block(s).  The following <i>ustar</i> header block shall determine the type of link created. If <i>typeflag</i> of the following header block is 1, it shall be a hard link. If <i>typeflag</i> is 2, it shall be a symbolic link and the <i>linkpath</i> value shall be the contents of the symbolic link.  The <i>pax</i> utility shall translate the name of the link (contents of the symbolic link) from the UTF8 encoding to the character set appropriate for the local file system.  When used in write or copy mode, <i>pax</i> shall include a <i>linkpath</i> extended header record for each link whose pathname cannot be represented entirely with the members of the portable character set other than NUL.	
2184			
2185			
2186			
2187			
2188			
2189			
2190			
2191			
2192			
2193			
2194			
2197	mtime	The file modification time of the following file(s), equivalent to the value of the <i>st_mtime</i> member of the <i>stat</i> structure for a file, as described in POSIX.1 {8}. This record shall override the <i>mtime</i> field in the following header block(s). The modification time shall be restored if the process has the appropriate privilege required to do so. The format of the <i>&lt;value&gt;</i> shall be as described in 4.48.7.1.4.	
2198			
2199			
2200			
2201			
2202			
2203			
2204	path	The pathname of the following file(s). This record shall override the <i>name</i> and <i>prefix</i> fields in the following header block(s). The <i>pax</i> utility shall translate the pathname of the file from the UTF8 encoding to the character set appropriate for the local file system.  When used in write or copy mode, <i>pax</i> shall include a <i>path</i> extended header record for each file whose pathname cannot be represented entirely with the members of the portable character set other than NUL.	
2205			
2206			
2207			
2208			
2209			
2210			
2212	realtime.any	The keywords prefixed by “realtime.” are reserved for future POSIX realtime standardization.	
2213			
2214			

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.



2215	security.	<i>any</i>	
2216		The keywords prefixed by “security.” are reserved for future	
2217		POSIX security standardization.	
2218	size	The size the file in octets, expressed as a decimal number using	B
2219		digits from ISO/IEC 646 {1}. This record shall override the <i>size</i>	B
2220		field in the following header block(s). When used in write or copy	B
2221		mode, <i>pax</i> shall include a <i>size</i> extended header record for each	B
2222		file with a <i>size</i> value greater than 999 999 999 999.	B
2223	uid	The user ID of the file owner, expressed as a decimal number	B
2224		using digits from ISO/IEC 646 {1}. This record shall override the	B
2225		<i>uid</i> field in the following header block(s). When used in write or	B
2226		copy mode, <i>pax</i> shall include a <i>uid</i> extended header record for	B
2227		each file whose owner ID is greater than 99 999 999.	B
2228	uname	The owner of the following file(s), formatted as a user name in the	B
2229		user database. This record shall override the <i>uid</i> and <i>uname</i>	B
2230		fields in the following header block(s), and any <i>uid</i> extended	B
2231		header record. When used in read, copy, or list mode, <i>pax</i> shall	B
2232		translate the name from the UTF8 encoding in the header record	
2233		to the character set appropriate for the user database on the	
2234		receiving system. If any of the UTF8 characters cannot be	B
2235		translated, and if the <i>-o invalid=UTF8</i> option is not specified,	B
2236		the results are implementation defined. When used in write or	B
2237		copy mode, <i>pax</i> shall include a <i>uname</i> extended header record for	
2238		each file whose user name cannot be represented entirely with	
2239		the letters and digits of the portable character set.	
2240	If the <i>&lt;value&gt;</i> field is zero length, it shall delete any header block field, previously		
2241	entered extended header value, or global extended header value of the same		
2242	name.		
2243	If a keyword in an extended header record (or in a <i>-o option-argument</i> ) overrides		
2244	or deletes a corresponding field in the <i>ustar</i> header block, <i>pax</i> shall ignore the		
2245	contents of that header block field.		
2246	Unlike the <i>ustar</i> header block fields, NULs shall not delimit <i>&lt;value&gt;</i> s; all charac-		
2247	ters within the <i>&lt;value&gt;</i> field shall be considered data for the field. None of the		
2248	length limitations of the <i>ustar</i> header block fields in Table 4-100 shall apply to		
2249	the extended header records.		
2250	<b>4.48.7.1.3 Extended Header Keyword Precedence</b>		
2251	This subclause describes the precedence in which the various header records and		
2252	fields and command-line options are selected to apply to a file in the archive.		
2253	When <i>pax</i> is used in read or list modes, it shall determine a file attribute in the		
2254	following sequence:		
2255	(1)	If <i>-o delete=keyword-prefix</i> is used, the affected attributes shall be	
2256		determined from step (7), if applicable, or ignored otherwise.	

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 2257 (2) If `-o keyword:=` is used, the affected attributes shall be ignored.
- 2258 (3) If `-o keyword=value` is used, the affected attribute shall be assigned the  
2259 *value*.
- 2260 (4) If there is a *typeflag* `x` extended header record, the affected attribute  
2261 shall be assigned the *<value>*. When extended header records conflict,  
2262 the last one given in the header shall take precedence.
- 2263 (5) If `-o keyword=value` is used, the affected attribute shall be assigned the  
2264 *value*.
- 2265 (6) If there is a *typeflag* `g` global extended header record, the affected attri-  
2266 bute shall be assigned the *<value>*. When global extended header  
2267 records conflict, the last one given in the global header shall take pre-  
2268 cedence.
- 2269 (7) Otherwise, the attribute shall be determined from the `ustar` header  
2270 block.

#### 2271 4.48.7.1.4 Extended Header File Times

2272 The `pax` utility shall write `atime` and `ctime` records for each file in write or copy  
2273 modes only if the `-o times` option is specified; `pax` shall write a `mtime` record for  
2274 each file in write or copy modes if the file system of the underlying implementa-  
2275 tion supports time granularities smaller than that required by the `ustar` header  
2276 block described in 4.48.7.2. All of these time records shall be formatted as a  
2277 decimal representation of the time in seconds since the Epoch. If a period (.)  
2278 decimal point character is present, the digits to the right of the point shall  
2279 represent the units of a subsecond timing granularity, where the first digit is B  
2280 tenths of a second and each subsequent digit is a tenth of the previous digit. B  
2281 Implementations may ignore any portion of the subsecond digits for which they do B  
2282 not support the necessary timing granularity; they shall not perform any round- B  
2283 ing operation. B

#### 2284 4.48.7.2 `ustar` Interchange Format

2285 A `ustar` archive tape or file shall contain a series of blocks. Each block shall be a  
2286 fixed-size block of 512 octets (see below). Although this format may be thought of  
2287 as being stored on 9-track industry-standard 12,7 mm (0,5 in) magnetic tape,  
2288 other types of transportable media are not excluded. Each file archived shall be  
2289 represented by a header block that describes the file, followed by zero or more  
2290 blocks that give the contents of the file. At the end of the archive file there shall  
2291 be two 512 B blocks filled with binary zeroes, interpreted as an end-of-archive  
2292 indicator.

2293 The blocks may be grouped for physical I/O operations, as described under the  
2294 `-b blocksize` and `-x ustar` options. Each group of blocks may be written with a  
2295 single operation equivalent to the `write()` function in POSIX.1 {8}. On magnetic  
2296 tape, the result of this write shall be a single tape record. The last group of blocks  
2297 always shall be at the full size, so blocks after the two zero blocks may contain

2298 undefined data.

2299 The header block shall be structured as shown in Table 4-100. All lengths and  
2300 offsets are in decimal.

2301 **Table 4-100 – ustar Header Block**

Field Name	Offset (in octets)	Length (in octets)
<i>name</i>	0	100
<i>mode</i>	100	8
<i>uid</i>	108	8
<i>gid</i>	116	8
<i>size</i>	124	12
<i>mtime</i>	136	12
<i>chksum</i>	148	8
<i>typeflag</i>	156	1
<i>linkname</i>	157	100
<i>magic</i>	257	6
<i>version</i>	263	2
<i>uname</i>	265	32
<i>gname</i>	297	32
<i>devmajor</i>	329	8
<i>devminor</i>	337	8
<i>prefix</i>	345	155

2320 All characters in the header block shall be represented in the coded character set  
2321 of ISO/IEC 646 {1}. For maximum portability between implementations, names  
2322 should be selected from characters represented by the portable filename character  
2323 set as octets with the most significant bit zero. If an implementation supports the  
2324 use of characters outside of slash and the portable filename character set in  
2325 names for files, users, and groups, one or more implementation-defined encodings  
2326 of these characters shall be provided for interchange purposes.

2327 Each field within the header block shall be contiguous; that is, there shall be no  
2328 padding used. Each character on the archive medium shall be stored  
2329 contiguously.

2330 The fields *magic*, *uname*, and *gname* shall be character strings each terminated  
2331 by a NUL character. The fields *name*, *linkname*, and *prefix* shall be NUL-  
2332 terminated character strings except when all characters in the array contain  
2333 non-NUL characters including the last character. The *version* field shall be two  
2334 octets containing the characters "00" (zero-zero). The *typeflag* shall contain a  
2335 single character. All other fields shall be leading zero-filled octal numbers using  
2336 digits from ISO/IEC 646 {1} IRV. Each numeric field shall be terminated by one or  
2337 more <space> or NUL characters.

2338 The *name* and the *prefix* fields shall produce the pathname of the file. A new  
2339 pathname shall be formed, if *prefix* is not an empty string (its first character is  
2340 not NUL), by concatenating *prefix* (up to the first NUL character), a slash charac-  
2341 ter, and *name*; otherwise, *name* shall be used alone. In either case, *name* shall be

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2342 terminated at the first NUL character. If *prefix* begins with a NUL character, it  
 2343 shall be ignored. In this manner, pathnames of at most 256 characters can be  
 2344 supported. If a pathname does not fit in the space provided, the `pax` utility shall  
 2345 notify the user of the error, and shall not attempt to store any part of the file—  
 2346 header or data—on the medium.

2347 The *linkname* field, described below, shall not use the *prefix* to produce a path-  
 2348 name. As such, a *linkname* is limited to 100 characters. If the name does not fit  
 2349 in the space provided, the `pax` utility shall notify the user of the error, and shall  
 2350 not attempt to store the link on the medium.

2351 The *mode* field provides 12 b encoded in ISO/IEC 646 {1} octal digit representation.  
 2352 The encoded bits shall represent the bitwise inclusive OR of the values in  
 2353 Table 4-101, expressed in terms of their equivalent POSIX.1 {8} bits.

2354 **Table 4-101 – ustar *mode* Field**

2355	Bit Value	POSIX.1 {8} Bit	Description
2356	04 000	S_ISUID	Set user ID on execution
2357	02 000	S_ISGID	Set group ID on execution
2358	01 000	<reserved>	Reserved for future standardization
2359	00 400	S_IRUSR	Read permission for file owner class
2360	00 200	S_IWUSR	Write permission for file owner class
2361	00 100	S_IXUSR	Execute/search permission for file owner class
2362	00 040	S_IRGRP	Read permission for file group class
2363	00 020	S_IWGRP	Write permission for file group class
2364	00 010	S_IXGRP	Execute/search permission for file group class
2365	00 004	S_IROTH	Read permission for file other class
2366	00 002	S_IWOTH	Write permission for file other class
2367	00 001	S_IXOTH	Execute/search permission for file other class

2368 When appropriate privilege is required to set one of these mode bits, and the user  
 2369 restoring the files from the archive does not have the appropriate privilege, the  
 2370 mode bits for which the user does not have appropriate privilege shall be ignored.  
 2371 Some of the mode bits in the archive format are not mentioned elsewhere in this  
 2372 standard or POSIX.1 {8}. If the implementation does not support those bits, they  
 2373 may be ignored.

2374 The *uid* and *gid* fields shall be the user and group ID of the owner and group of  
 2375 the file, respectively.

2376 The *size* field shall be the size of the file in octets. If the *typeflag* field is set to  
 2377 specify a file to be of type 1 (hard link) or 2 (symbolic link), the *size* field shall be  
 2378 specified as zero. If the *typeflag* field is set to specify a file of type 5 (directory),  
 2379 the *size* field shall be interpreted as described under the definition of that record  
 2380 type. No data blocks shall be stored for types 1, 2, or 5. If the *typeflag* field is set  
 2381 to 3 (character special file), 4 (block special file), or 6 (FIFO), the meaning of the  
 2382 *size* field is unspecified by this standard, and no data blocks shall be stored on the  
 2383 medium. Additionally, for 6, the *size* field shall be ignored when reading. If the  
 2384 *typeflag* field is set to any other value, the number of blocks written following the

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 2385 header shall be  $(size+511)/512$ , ignoring any fraction in the result of the division.
- 2386 The *mtime* field shall be the modification time of the file at the time it was  
2387 archived. It is the ISO/IEC 646 {1} representation of the octal value of the  
2388 modification time obtained from the equivalent of the POSIX.1 {8} *stat()* function.
- 2389 The *chksum* field shall be the ISO/IEC 646 {1} IRV representation of the octal value  
2390 of the simple sum of all octets in the header block. Each octet in the header shall  
2391 be treated as an unsigned value. These values shall be added to an unsigned  
2392 integer, initialized to zero, the precision of which shall be not less than 17 b.  
2393 When calculating the checksum, the *chksum* field shall be treated as if it were all  
2394 `<space>s`.
- 2395 The *typeflag* field shall specify the type of file archived. If a particular implemen-  
2396 tation does not recognize the type, or the user does not have appropriate privilege  
2397 to create that type, the file shall be extracted as if it were a regular file if the file  
2398 type is defined to have a meaning for the size field that could cause data blocks to  
2399 be written on the medium (see the previous description for *size*). If conversion to a  
2400 regular file occurs, the `pax` utility shall produce an error indicating that the  
2401 conversion took place. All of the *typeflag* fields shall be coded in ISO/IEC 646 {1}  
2402 IRV:
- 2403     '0'       Represents a regular file. For backward compatibility, a *typeflag*  
2404               value of binary zero ('`\0`') should be recognized as meaning a regu-  
2405               lar file when extracting files from the archive. Archives written  
2406               with this version of the archive file format shall create regular files  
2407               with a *typeflag* value of ISO/IEC 646 {1} IRV '0'.
- 2408     '1'       Represents a file linked to another file, of any type, previously  
2409               archived. Such files shall be identified by each file having the same  
2410               device and file serial number. The linked-to name shall be specified  
2411               in the *linkname* field with a NUL-character terminator if it is less  
2412               than 100 octets in length.
- 2413     '2'       Represents a symbolic link. The contents of the symbolic link shall  
2414               be stored in the *linkname* field.
- 2415     '3', '4'   Represent character special files and block special files respectively.  
2416               In this case the *devmajor* and *devminor* fields shall contain informa-  
2417               tion defining the device, the format of which is unspecified by this  
2418               standard. Implementations may map the device specifications to  
2419               their own local specification or may ignore the entry.
- 2420     '5'       Specifies a directory or subdirectory. On systems where disk alloca-  
2421               tion is performed on a directory basis, the *size* field shall contain the  
2422               maximum number of octets (which may be rounded to the nearest  
2423               disk block allocation unit) that the directory may hold. A *size* field  
2424               of zero shall indicate no such limiting. Systems that do not support  
2425               limiting in this manner should ignore the *size* field.
- 2426     '6'       Specifies a FIFO special file. Note that the archiving of a FIFO file  
2427               archives the existence of this file and not its contents.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2428 '7' Reserved to represent a file to which an implementation has associ-  
 2429 ated some high performance attribute. Implementations without  
 2430 such extensions should treat this file as a regular file (type '0').

2431 'A'-'Z' The letters A through Z are reserved for custom implementations.  
 2432 All other values are reserved for specification in future revisions of  
 2433 this standard.

2434 The *magic* field is the specification that this archive was output in this archive  
 2435 format. If this field contains "ustar" (the five ISO/IEC 646 {1} IRV characters  
 2436 shown followed by NUL), the *uname* and *gname* fields shall contain the  
 2437 ISO/IEC 646 {1} IRV representation of the owner and group of the file respectively  
 2438 (truncated to fit, if necessary). When the file is restored by a privileged,  
 2439 protection-preserving version of the utility, the password and group files shall be  
 2440 scanned for these names. If found, the user and group IDs contained within these  
 2441 files shall be used rather than the values contained within the *uid* and *gid* fields.

#### 2442 4.48.7.3 *cpio* Interchange Format

2443 The octet-oriented *cpio* archive format shall be a series of entries, each compris-  
 2444 ing a header that describes the file, the name of the file, and then the contents of  
 2445 the file.

2446 An archive may be recorded as a series of fixed-size blocks of octets. This blocking  
 2447 shall be used only to make physical I/O more efficient. The last group of blocks  
 2448 always shall be at the full size.

2449 For the octet-oriented *cpio* archive format, the individual entry information shall  
 2450 be in the order indicated and described by Table 4-102.

##### 2451 4.48.7.3.1 *cpio* Header

2452 For each file in the archive, a header as defined previously shall be written. The  
 2453 information in the header fields shall be written as streams of ISO/IEC 646 {1}  
 2454 characters interpreted as octal numbers. The octal numbers shall be extended to  
 2455 the necessary length by appending ISO/IEC 646 {1} IRV zeros at the most-  
 2456 significant-digit end of the number; the result is written to the stream of octets  
 2457 most-significant-digit first. The fields shall be interpreted as follows:

2458 *c\_magic* Identifies the archive as being a transportable archive by contain-  
 2459 ing the identifying value "070707".

2460 *c\_dev*  
 2461 *c\_ino* Contains values that uniquely identify the file within the archive  
 2462 (i.e., no files shall contain the same pair of *c\_dev* and *c\_ino* values  
 2463 unless they are links to the same file). The values shall be deter-  
 2464 mined in an unspecified manner.

2465

**Table 4-102 – Octet-Oriented `cpio` Archive Entry**

2466

Header		
Field Name	Length (in octets)	Interpreted as
<i>c_magic</i>	6	Octal number
<i>c_dev</i>	6	Octal number
<i>c_ino</i>	6	Octal number
<i>c_mode</i>	6	Octal number
<i>c_uid</i>	6	Octal number
<i>c_gid</i>	6	Octal number
<i>c_nlink</i>	6	Octal number
<i>c_rdev</i>	6	Octal number
<i>c_mtime</i>	11	Octal number
<i>c_namesize</i>	6	Octal number
<i>c_filesize</i>	11	Octal number
File Name		
Field Name	Length	Interpreted as
<i>c_name</i>	<i>c_namesize</i>	Pathname string
File Data		
Field Name	Length	Interpreted as
<i>c_filedata</i>	<i>c_filesize</i>	Data

2479

2480

2481

2482

2483

2484

2485

*c\_mode*

The encoded bits shall represent the bitwise inclusive OR of the values in Table 4-103, expressed in terms of their equivalent POSIX.1 {8} bits, added to one of the values in Table 4-104. Directories, FIFOs, and regular files shall be supported on a system conforming to this standard; additional values defined previously are reserved for compatibility with existing systems. Additional file types may be supported; however, such files should not be written on archives intended for transport to portable systems.

2486

2487

2488

2489

2490

2491

2492

2493

*c\_uid*

Contains the user ID of the owner.

2494

*c\_gid*

Contains the group ID of the group.

2495

*c\_nlink*

Contains the number of links referencing the file at the time the archive was created.

2496

2497

*c\_rdev*

Contains implementation-defined information for character or block special files.

2498

2499

*c\_mtime*

Contains the latest time of modification of the file at the time the archive was created.

2500

2501

*c\_namesize*

Contains the length of the pathname, including the terminating NUL character.

2502

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2503

**Table 4-103 – cpio *c\_mode* File Modes**

Bit Value	POSIX.1 {8} Bit	Description
2505 04 000	S_ISUID	Set user ID on execution
2506 02 000	S_ISGID	Set group ID on execution
2507 01 000	<reserved>	Reserved for future standardization
2508 00 400	S_IRUSR	Read permission for file owner class
2509 00 200	S_IWUSR	Write permission for file owner class
2510 00 100	S_IXUSR	Execute/search permission for file owner class
2511 00 040	S_IRGRP	Read permission for file group class
2512 00 020	S_IWGRP	Write permission for file group class
2513 00 010	S_IXGRP	Execute/search permission for file group class
2514 00 004	S_IROTH	Read permission for file other class
2515 00 002	S_IWOTH	Write permission for file other class
2516 00 001	S_IXOTH	Execute/search permission for file other class

2517

**Table 4-104 – cpio *c\_mode* File Types**

Bit Value	Description
2518 040 000	Directory
2519 010 000	FIFO
2520 0100 000	Regular file
2521 060 000	Block special file
2522 020 000	Character special file
2523 0110 000	Reserved for future standardization
2524 0120 000	Symbolic link
2525 0140 000	Reserved for future standardization
2526	

2527 *c\_filesize* Contains the length of the file in octets. This shall be the length  
 2528 of the data section following the header structure.

#### 2529 4.48.7.3.2 cpio File Name

2530 The *c\_name* field shall contain the pathname of the file. The length of this field in  
 2531 octets shall be the value of *c\_namesize*.

2532 All characters shall be represented in ISO/IEC 646 {1} IRV. For maximum porta-  
 2533 bility between implementations, names should be selected from characters  
 2534 represented by the portable filename character set as octets with the most  
 2535 significant bit zero. If an implementation supports the use of characters outside  
 2536 of slash and the portable filename character set in names for files, users, and  
 2537 groups, one or more implementation-defined encodings of these characters shall  
 2538 be provided for interchange purposes. B

#### 2539 4.48.7.3.3 cpio File Data

2540 Following *c\_name*, there shall be *c\_filesize* octets of data. Interpretation of such  
 2541 data shall occur in a manner dependent on the file. If *c\_filesize* is zero, no data  
 2542 shall be contained in *c\_filedata*. Only regular files have data to be restored. B



**2543 4.48.7.3.4 cpio Special Entries**

2544 FIFO special files, directories, and the trailer shall be recorded with *c\_filesize*  
2545 equal to zero. For other special files, *c\_filesize* is unspecified by this standard.  
2546 The header for the next file entry in the archive shall be written directly after the  
2547 last octet of the file entry preceding it. A header denoting the file name  
2548 “TRAILER!!!” shall indicate the end of the archive; the contents of octets in the  
2549 last block of the archive following such a header are undefined.

**2550 4.48.8 Exit Status**

2551 The `pax` utility shall exit with one of the following values:

- 2552       0     All files were processed successfully.  
2553       >0    An error occurred.

**2554 4.48.9 Consequences of Errors**

2555 If `pax` cannot create a file or a link when reading an archive; cannot find a file  
2556 when writing an archive; or cannot preserve the user ID, group ID, or file mode  
2557 when the `-p` option is specified; a diagnostic message shall be written to standard  
2558 error and a nonzero exit status shall be returned, but processing shall continue.  
2559 In the case where `pax` cannot create a link to a file, `pax` shall not, by default,  
2560 create a second copy of the file.

2561 If the extraction of a file from an archive is prematurely terminated by a signal or  
2562 error, `pax` may have only partially extracted the file or (if the `-n` option was not  
2563 specified) may have extracted a file of the same name as that specified by the  
2564 user, but that is not the file the user wanted. Additionally, the file modes of  
2565 extracted directories may have additional bits from the `S_IRWXU` mask set as well  
2566 as incorrect modification and access times.

2567 **4.51 pwd – Return working directory name**2568 ⇒ **4.51.1 pwd Synopsis.** *Change the Synopsis to:*2569       pwd [ -L | -P ] B2570 ⇒ **4.51.2 pwd Description.** *Change this subclause to:* B2571       The `pwd` utility shall write to standard output an absolute pathname of the B  
2572       current working directory, which does not contain the filenames `dot` or `dot-dot`. B2573 ⇒ **4.51.3 pwd Options.** *Change the entire subclause to:*2574       The `pwd` utility shall conform to the utility argument syntax guidelines B  
2575       described in 2.10.2. B

2576       The following options shall be supported by the implementation:

2577           -L           If the **PWD** environment variable contains an absolute path- B  
2578                       name of the current directory that does not contain the B  
2579                       filenames `dot` or `dot-dot`, `pwd` shall write this pathname to C  
2580                       standard output. Otherwise, the `-L` option shall behave as the C  
2581                       `-P` option. C2582           -P           The absolute pathname written shall not contain filenames B  
2583                       that, in the context of the pathname, refer to files of type sym- B  
2584                       bolic link. B2585       If both `-L` and `-P` are specified, the last one shall apply. If neither `-L` nor `-P` is C  
2586       specified, the `pwd` utility shall behave as if `-L` had been specified. C2587 ⇒ **4.51.5.3 pwd Environment Variables.** *Add the following variable in the C*  
2588       *correct sorted order:* C2589           **PWD**           If the `-P` option is in effect, this variable shall be set to an C  
2590                       absolute pathname of the current working directory that does C  
2591                       not contain any components that specify symbolic links, does C  
2592                       not contain any components that are `dot`, and does not contain C  
2593                       any components that are `dot-dot`. If an application sets or C  
2594                       unsets the value of **PWD**, the behavior of `pwd` is unspecified. C2595 C

2596 **4.53 rm – Remove directory entries**2597 ⇒ **4.53.2 rm Description.** *Replace item (2c) with:*

2598 For each entry contained in *file*, other than dot or dot-dot, the four steps listed  
 2599 here [(1)-(4)] shall be taken with the entry as if it were a *file* operand. The `rm`  
 2600 utility shall not traverse directories by following symbolic links into other  
 2601 parts of the hierarchy, but shall remove the links themselves.

2602 ⇒ **4.53.8 rm Exit Status.** *Change the description of the 0 value to:*

B

2603 0 All of the named directory entries for which `rm` performed actions B  
 2604 equivalent to the POSIX.1 {8} `rmdir()` or `unlink()` functions were B  
 2605 removed. B

2606 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B  
 2607 #75 submitted for IEEE Std 1003.2-1992. B

2608 *Editor's Note: The following rationale will be added to E.4.53, but is kept here with*  
 2609 `rm` *for this draft:*

2610 **rm Rationale.** *(This subclause is not a part of P1003.2b)*

2611 The `rm` utility removes symbolic links themselves, not the files they refer to, as a  
 2612 consequence of the dependence on the POSIX.1 {8} `unlink()` functionality, per the  
 2613 Description. When removing hierarchies with `-r` or `-R`, the prohibition on follow-  
 2614 ing symbolic links has to be made explicit.

2615 **4.55 sed – Stream editor**

2616 **Rationale:** The changes to `sed` are to align with historical practice and are the B  
 2617 result of interpretation requests PASC 1003.2-92 #34 and #35 submitted for IEEE B  
 2618 Std 1003.2-1992. B

2619 ⇒ **4.55.5.2 sed Input Files.** *Replace this subclause with the following:* B

2620 The input files shall be text files. The *script\_files* named by the `-f` option shall B  
 2621 consist of editing commands. B

2622 ⇒ **4.55.7 sed Extended Description.** *Replace the entire Extended Description*  
 2623 *with the following.*

2624 *Editor's Note: There were numerous terminology changes in this clause, which*  
 2625 *would have resulted in many dozens of individual change descriptions. I chose to*  
 2626 *reprint the entire clause with the changes embedded. Lines changed from*  
 2627 *POSIX.2-1992 are diffmarked for Draft 10 only; these are the lines subject to*  
 2628 *P1003.2b balloting. The diffmarks were removed in Draft 11.* B

2629 **4.55.7 Extended Description**

2630 The *script* shall consist of editing commands of the following form: B

2631 `[address[,address]]function`

2632 where *function* represents a single-character command verb from the list in  
 2633 4.55.7.3, followed by any applicable arguments.

2634 Zero or more `<blank>`s shall be accepted before the first address and before *func-*  
 2635 *tion*. Any number of semicolons shall be accepted before the first *address*.

2636 In default operation, `sed` cyclically shall copy a line of input, less its terminating  
 2637 `<newline>`, into a *pattern space* (unless there is something left after a `D` com-  
 2638 mand), apply in sequence all commands whose addresses select that pattern  
 2639 space, and at the end of the script copy the pattern space to standard output  
 2640 (except when `-n` is specified) and delete the pattern space. Whenever the pattern  
 2641 space is written to standard output or a named file, `sed` shall immediately follow  
 2642 it with a `<newline>`.

2643 Some of the editing commands use a *hold space* to save all or part of the *pattern*  
 2644 *space* for subsequent retrieval. The *pattern* and *hold spaces* shall each be able to  
 2645 hold at least 8192 B.

2646 **4.55.7.1 sed Addresses**

2647 An address is either a decimal number that counts input lines cumulatively  
 2648 across files, a `$` character that addresses the last line of input, or a context

- 2649 address (which consists of a BRE, as described in 4.55.7.2, preceded and followed  
2650 by a delimiter, usually a slash).
- 2651 An editing command with no addresses shall select every pattern space. B
- 2652 An editing command with one address shall select each pattern space that B  
2653 matches the address. B
- 2654 An editing command with two addresses shall select the inclusive range from the B  
2655 first pattern space that matches the first address through the next pattern space  
2656 that matches the second. (If the second address is a number less than or equal to  
2657 the line number first selected, only one line shall be selected.) Starting at the  
2658 first line following the selected range, *sed* shall look again for the first address.  
2659 Thereafter, the process shall be repeated. Omitting either or both of the *address*  
2660 components in the [*address*[,*address*]] form produces undefined results.
- 2661 B

#### 2662 4.55.7.2 *sed* REs

2663 The *sed* utility shall support the BREs described in 2.8.3, with the following  
2664 additions:

- 2665 (1) In a context address, the construction `\cBREc`, where *c* is any character  
2666 other than backslash or `<newline>`, shall be identical to `/BRE/`. If the  
2667 character designated by *c* appears following a backslash, then it shall be  
2668 considered to be that literal character, which shall not terminate the  
2669 BRE. For example, in the context address `\xabc\xdefx`, the second `x`  
2670 stands for itself, so that the BRE is `abcxdef`.
- 2671 (2) The escape sequence `\n` shall match a `<newline>` embedded in the pat-  
2672 tern space. A literal `<newline>` character shall not be used in the BRE  
2673 of a context address or in the substitute function.
- 2674 (3) If an RE is empty (i.e., no pattern is specified) *sed* shall behave as if the  
2675 last RE used in the last command applied (either as an address or as part  
2676 of a substitute command) was specified.

#### 2677 4.55.7.3 *sed* Editing Commands

2678 In the following list of editing commands, the maximum number of permissible  
2679 addresses for each function is indicated by [*0addr*], [*1addr*], or [*2addr*],  
2680 representing zero, one, or two addresses.

2681 The argument *text* shall consist of one or more lines. Each embedded `<newline>`  
2682 in the text shall be preceded by a backslash. Other backslashes in text shall be  
2683 removed, and the following character shall be treated literally.

2684 The *r* and *w* command verbs, and the *w* flag to the *s* command, take an optional B  
2685 *rfile* (or *wfile*) parameter, separated from the command verb letter or flag by one B  
2686 or more `<blank>`s; implementations may allow zero separation as an extension.

2687 The argument *rfile* or the argument *wfile* shall terminate the editing command.  
 2688 Each *wfile* shall be created before processing begins. Implementations shall sup-  
 2689 port at least nine *wfile* arguments in the script; the actual number ( $\geq 9$ ) that shall  
 2690 be supported by the implementation is unspecified. The use of the *wfile* param-  
 2691 eter shall cause that file to be initially created, if it does not exist, or shall replace  
 2692 the contents of an existing file.

2693 The *b*, *r*, *s*, *t*, *w*, *y*, and *:* command verbs shall accept additional arguments. The B  
 2694 following synopses indicate which arguments shall be separated from the com-  
 2695 mand verbs by a single `<space>`.

2696 The *a* and *r* commands schedule text for later output. The text specified for the *a* B  
 2697 command, and the contents of the file specified for the *r* command, shall be writ- B  
 2698 ten to standard output just before the next attempt to fetch a line of input when B  
 2699 executing the *N* or *n* commands, or when reaching the end of the script. If written B  
 2700 when reaching the end of the script, and the *-n* option was not specified, the text B  
 2701 shall be written after copying the pattern space to standard output. The contents B  
 2702 of the file specified for the *r* command shall be as of the time the output is writ- B  
 2703 ten, not the time the *r* command is applied. The text shall be output in the order B  
 2704 in which the *a* and *r* commands were applied to the input. B

2705 Command verbs other than `{`, *a*, *b*, *c*, *i*, *r*, *t*, *w*, *:*, and *#* can be followed by a B  
 2706 semicolon, optional `<blank>s`, and another command verb. However, when the *s* B  
 2707 command verb is used with the *w* flag, following it with another command in this B  
 2708 manner produces undefined results.

2709 A function can be preceded by one or more `!` characters, in which case the func- B  
 2710 tion shall be applied if the addresses do not select the pattern space. Zero or more B  
 2711 `<blank>s` shall be accepted before the first `!` character. It is unspecified if B  
 2712 `<blank>` characters can follow a `!` character, and conforming applications shall B  
 2713 not follow a `!` character with `<blank>s`. B

2714 `[2addr] { function` B  
 2715 `function` B  
 2716 `...` B  
 2717 `}` Execute a list of `sed` functions only when the pattern space is B  
 2718 selected. The list of `sed` functions shall be surrounded by braces B  
 2719 and separated by `<newline>s`, as follows. The braces can be pre- B  
 2720 ceded or followed by `<blank>s`. The *functions* can be preceded by B  
 2721 `<blank>s`, but shall not be followed by `<blank>s`. The `<right-` B  
 2722 `brace>` shall be preceded by a `<newline>` and can be preceded B  
 2723 or followed by `<blank>s`. B

2724 `[1addr]a\`  
 2725 `text` Write *text* to standard output as described previously.

2726 `[2addr]b [label]`  
 2727 Branch to the `:` function bearing the *label*. If *label* is not  
 2728 specified, branch to the end of the script. The implementation  
 2729 shall support *labels* recognized as unique up to at least 8 charac-  
 2730 ters; the actual length ( $\geq 8$ ) that shall be supported by the imple-

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2731		mentation is unspecified. It is unspecified whether exceeding a	
2732		label length causes an error or a silent truncation.	
2733	<b>[2addr]c\</b>		
2734	<i>text</i>	Delete the pattern space. With 0 or 1 address or at the end of a	
2735		2-address range, place <i>text</i> on the output and start the next cycle.	B
2736	<b>[2addr]d</b>	Delete the pattern space and start the next cycle.	
2737	<b>[2addr]D</b>	Delete the initial segment of the pattern space through the first	
2738		<newline> and start the next cycle.	
2739	<b>[2addr]g</b>	Replace the contents of the pattern space by the contents of the	
2740		hold space.	
2741	<b>[2addr]G</b>	Append to the pattern space a <newline> followed by the con-	
2742		tents of the hold space.	
2743	<b>[2addr]h</b>	Replace the contents of the hold space with the contents of the	
2744		pattern space.	
2745	<b>[2addr]H</b>	Append to the hold space a <newline> followed by the contents	
2746		of the pattern space.	
2747	<b>[1addr]i\</b>		
2748	<i>text</i>	Write <i>text</i> to standard output.	
2749	<b>[2addr]l</b>	(The letter ell.) Write the pattern space to standard output in a	
2750		visually unambiguous form. The characters listed in Table 2-16	
2751		(see 2.12), except for \n, shall be written as the corresponding	
2752		escape sequence. Nonprintable characters not in Table 2-16 shall	
2753		be written as one three-digit octal number (with a preceding	
2754		backslash) for each byte in the character (most significant byte	
2755		first). If the size of a byte on the system is greater than 9 b, the	
2756		format used for nonprintable characters is implementation	
2757		defined.	
2758		Long lines shall be folded, with the point of folding indicated by	
2759		writing <backslash><newline>; the length at which folding	
2760		occurs is unspecified, but should be appropriate for the output	
2761		device. The end of each line shall be marked with a \$.	
2762	<b>[2addr]n</b>	Write the pattern space to standard output if the default output	
2763		has not been suppressed, and replace the pattern space with the	
2764		next line of input.	
2765		If no next line of input is available, the n command verb shall	B
2766		branch to the end of the script and quit without starting a new	
2767		cycle.	

2768	[2addr]N	Append the next line of input to the pattern space, using an embedded <newline> to separate the appended material from the original material. Note that the current line number changes.	
2769			
2770			
2771		If no next line of input is available, the N command verb shall	B
2772		branch to the end of the script and quit without starting a new	B
2773		cycle or copying the pattern space to standard output.	B
2774	[2addr]P	Write the pattern space to standard output.	
2775	[2addr]P	Write the pattern space, up to the first <newline>, to standard	
2776		output.	
2777	[1addr]Q	Branch to the end of the script and quit without starting a new	
2778		cycle.	
2779	[1addr]r <i>rfile</i>		
2780		Copy the contents of <i>rfile</i> to standard output as described previ-	B
2781		ously. If <i>rfile</i> does not exist or cannot be read, it shall be treated	B
2782		as if it were an empty file, causing no error condition.	
2783	[2addr]s / BRE / replacement / flags		
2784		Substitute the <i>replacement</i> string for instances of the <i>BRE</i> in the	
2785		pattern space. Any character other than backslash or <newline>	
2786		can be used instead of a slash to delimit the BRE and the replace-	
2787		ment. Within the BRE and the replacement, the BRE delimiter	
2788		itself can be used as a literal character if it is preceded by a	
2789		backslash.	
2790		An ampersand (&) appearing in the <i>replacement</i> shall be replaced	
2791		by the string matching the BRE. The special meaning of & in this	
2792		context can be suppressed by preceding it by backslash. The	
2793		characters \n, where <i>n</i> is a digit, shall be replaced by the text	
2794		matched by the corresponding backreference expression (see	
2795		2.8.3.3). For each backslash (\) encountered in scanning <i>replace-</i>	
2796		<i>ment</i> from beginning to end, the backslash shall be discarded and	
2797		the following character shall lose its special meaning (if any). It	
2798		is unspecified what special meaning is given to any character	
2799		other than &, \, or digits.	
2800		A line can be split by substituting a <newline> character into it.	
2801		The application shall escape the <newline> in the <i>replacement</i>	
2802		by preceding it by backslash. A substitution shall be considered	
2803		to have been performed even if the replacement string is identical	
2804		to the string that it replaces. Any backslash used to alter the	B
2805		default meaning of a subsequent character shall be discarded	B
2806		from the BRE or the replacement before evaluating the BRE or	B
2807		using the replacement.	B



2808                   The value of *flags* shall be zero or more of

2809                    *n*       Substitute for the *n*th occurrence only of the *BRE*  
2810                               found within the pattern space.

2811                    *g*       Globally substitute for all nonoverlapping instances  
2812                               of the *BRE* rather than just the first one. If both *g*  
2813                               and *n* are specified, the results are unspecified.

2814                    *p*       Write the pattern space to standard output if a  
2815                               replacement was made.

2816                    *w wfile* Write. Append the pattern space to *wfile* if a  
2817                               replacement was made. A conforming application  
2818                               shall precede the *wfile* argument with one or more  
2819                               <blank>s. If the *w* flag is not the last flag value  
2820                               given in a concatenation of multiple flag values, the  
2821                               results are undefined.

2822        [2addr]t [*label*]  
2823                    Test. Branch to the : command verb bearing the *label* if any sub-  
2824                               stitutions have been made since the most recent reading of an  
2825                               input line or execution of a t. If *label* is not specified, branch to  
2826                               the end of the script.

2827        [2addr]w *wfile*  
2828                    Append [write] the pattern space to *wfile*.

2829        [2addr]x    Exchange the contents of the pattern and hold spaces.

2830        [2addr]y/ *string1*/*string2*/  
2831                    Replace all occurrences of characters in *string1* with the  
2832                               corresponding characters in *string2*. If a backslash followed by an    B  
2833                               *n* appear in *string1* or *string2*, the two characters shall be han-    B  
2834                               dled as a single <newline> character. If the number of charac-    B  
2835                               ters in *string1* and *string2* are not equal, or if any of the charac-    B  
2836                               ters in *string1* appear more than once, the results are undefined.  
2837                    Any character other than backslash or <newline> can be used  
2838                               instead of slash to delimit the strings. If the delimiter is not *n*,  
2839                               within *string1* and *string2*, the delimiter itself can be used as a  
2840                               literal character if it is preceded by a backslash. If a backslash  
2841                               character is immediately followed by a backslash character in  
2842                               *string1* or *string2*, the two backslash characters shall be counted  
2843                               as a single literal backslash character. The meaning of a  
2844                               backslash followed by any character that is not *n*, a backslash, or    B  
2845                               the delimiter character is undefined.                                B

2846        [0addr]: *label*  
2847                    Do nothing. This command bears a *label* to which the *b* and *t*  
2848                               commands branch.

- 2849     **[1addr]=** Write the following to standard output:  
 2850                    "%d\n", <current line number>  
 2851     **[0addr]** Ignore this empty command.  
 2852     **[0addr]#** Ignore the # and the remainder of the line (treat them as a com-  
 2853                   ment), with the single exception that if the first two characters in  
 2854                   the script are #n, the default output shall be suppressed; this  
 2855                   shall be the equivalent of specifying -n on the command line.

2856     *Editor's Note: The following rationale will be added to E.4.51, but is kept here with*  
 2857     *sed for this draft:*

2858     **sed Rationale.** *(This subclause is not a part of P1003.2b)*

2859

B

2860     The b, t, and : commands are documented to ignore leading white space, but no  
 2861     mention is made of trailing white space. Historical implementations of sed  
 2862     assigned different locations to the labels 'x' and 'x '. This is not useful, and  
 2863     leads to subtle programming errors, but it is historical practice, and changing it  
 2864     could theoretically break working scripts. Implementors are encouraged to pro-  
 2865     vide warning messages about labels that are never used or jumps to labels that do  
 2866     not exist.

2867     *Editor's Note: The terminology changes in the normative text will carry over into*  
 2868     *the rationale as well. They are summarized here using POSIX.2-1992 line numbers*  
 2869     *within E.4.55:*

2870     Line 8018: change "commands" to "editing commands."

2871     Line 8021: change "command" to "function."

2872     Line 8029: change "command lines" to "editing commands."

2873     Line 8035: change "command line" to "editing command."

2874     Line 8038: change "command" to "command verb."

2875     Line 8050: change "command" to "function."

2876     Line 8067: change "commands" to "command verbs."

2877     Line 8078: change "command" to "function."

2878     Line 8081: change "command" to "editing command."

2879     Lines 8083–8084: change "commands" to "editing commands."

2880     *Editor's Note: Replace the rationale paragraph (E.4.55, POSIX.2-1992 lines 8083-*  
 2881     *86) with:*

B  
B

2882     Historically, the sed ! and } editing commands did not permit multiple com-  
 2883     mands on a single line using a semicolon as a command delimiter. Implementa-  
 2884     tions are permitted, but not required, to support this extension.

B  
B  
B

2885	<b>4.56 sh – Shell, the standard command language interpreter</b>	B
2886	⇒ <b>4.56.4 sh Operands.</b> <i>Change the <code>command_string</code> description to:</i>	B
2887	<i><code>command_string</code></i>	B
2888	A string that shall be interpreted by the shell as one or more	B
2889	commands, as if the string were the argument to the	B
2890	POSIX.1 {8} <i>system()</i> function. If the <i>command_string</i> operand	B
2891	is an empty string, <code>sh</code> shall exit with a zero exit status.	B
2892	<b>Rationale:</b> This change is part of a general cleanup to remove references to the	B
2893	now-deleted Chapter 7. All of the applicable functions are now in POSIX.1-199x,	B
2894	the version created by the currently balloting P1003.1a.	B
2895	⇒ <b>4.56.5.3 sh Environment Variables.</b> <i>Change the description of <code>ENV</code> to:</i>	B
2896	<b>ENV</b>	B
2897	This variable, when and only when an interactive shell is	B
2898	invoked, shall be subjected to parameter expansion (see	B
2899	3.6.2) by the shell, and the resulting value shall be used	B
2900	as a pathname of a file containing shell commands to	B
2901	execute in the current environment. The file need not be	B
2902	executable. If the expanded value of <code>ENV</code> is not an abso-	B
2903	lute pathname, the results are unspecified. <code>ENV</code> shall be	B
2904	ignored if the real and effective user IDs or real and effec-	B
2905	tive group IDs of the user are different.	C
2906	<b>Rationale:</b> The preceding change is the result of interpretation request PASC	B
2907	1003.2-92 #110 submitted for IEEE Std 1003.2-1992.	B
2908	⇒ <b>4.56.5.3 sh Environment Variables.</b> <i>Add the following variable in proper</i>	B
2909	<i>sorted order:</i>	B
2910	<b>PWD</b>	B
2911	This variable shall represent an absolute pathname of	B
2912	the current working directory. Assignments to this vari-	B
2913	able may be ignored unless the value is an absolute path-	B
	name of the current working directory and there are no	B
	filename components of dot or dot-dot.	B

2914 **4.62 test – Evaluate expression**

2915 ⇒ **4.62.4 test Operands.** *Replace the `-r`, `-w`, and `-x` descriptions with the fol-*  
 2916 *lowing:*

2917       `-r file`       True if *file* exists and is readable. True shall indicate that  
 2918                           permission to read from *file* will be granted, as defined in  
 2919                           2.2.2.66.

2920       `-w file`       True if *file* exists and is writable. True shall indicate that  
 2921                           permission to write to *file* will be granted, as defined in  
 2922                           2.2.2.66.

2923       `-x file`       True if *file* exists and is executable. True shall indicate that  
 2924                           permission to execute *file* will be granted, as defined in  
 2925                           2.2.2.66. If *file* is a directory, true shall indicate that permis-  
 2926                           sion to search *file* will be granted.

2927 **Rationale:** This change is a clarification and is the result of interpretation  
 2928 request PASC 1003.2-92 #23 submitted for IEEE Std 1003.2-1992.

2929 ⇒ **4.62.4 test Operands.** *Add the following primary in the proper sorted order:*

2930       `-h file`       True if *file* exists and is a symbolic link.

2931 ⇒ **4.62.4 test Operands.** *Add the following at the end of the primaries list*  
 2932 *(before the paragraph that begins “A primary can be preceded by ...”*

2933       With the exception of the `-h file` primary, if a *file* argument is a symbolic link,  
 2934       test shall evaluate the expression by resolving the symbolic link and using  
 2935       the file referenced by the link.

2936 **4.64 tr – Translate characters**

2937 **Rationale:** The following changes related to `-C` are the result of interpretation  
 2938 requests PASC 1003.2-92 #24 and #25 submitted for IEEE Std 1003.2-1992.

2939 ⇒ **4.64.1 tr Synopsis.** *Change the Synopsis to:*

2940 `tr [-c|-C] [-s] string1 string2`

2941 `tr -s [-c|-C] string1`

2942 `tr -d [-c|-C] string1`

2943 `tr -ds [-c|-C] string1 string2`

2944 ⇒ **4.64.3 tr Options.** *Change the description of `-c` to:*

2945 `-c` Complement the range of values specified by *string1*. See  
 2946 4.64.7.

2947 `-C` Complement the set of characters specified by *string1*. See  
 2948 4.64.7.

2949 ⇒ **4.64.7 tr Extended Description.** *Change the description of `\octal` to:*

2950 `\octal` Represents octal sequences that can be used to represent  
 2951 specific coded values. An octal sequence shall consist of a  
 2952 backslash followed by the longest sequence of one, two, or  
 2953 three octal-digit characters (01234567). The sequence shall  
 2954 cause the value whose encoding is represented by the one,  
 2955 two, or three digit octal integer to be placed into the array. If  
 2956 the size of a byte on the system is greater than 9 b, the valid  
 2957 escape sequence used to represent a byte is implementation  
 2958 defined.

2959 ⇒ **4.64.7 tr Extended Description.** *Change the description of `\c-c` to:*

2960 `c-c` Represents the range of characters between the range end- C  
 2961 points (as long as neither endpoint is an octal sequence of the C  
 2962 form `\octal`), inclusive, as defined by the current setting of C  
 2963 the LC\_COLLATE locale category. The starting endpoint shall C  
 2964 precede the second endpoint in the current collation order. C  
 2965 The characters in the range shall be placed in the array in C  
 2966 ascending collation sequence.

2967 If either or both of the range endpoints are octal sequences of  
 2968 the form `\octal`, this shall represent the range of specific  
 2969 coded values between the two range endpoints, inclusive.

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

2970 ⇒ **4.64.7 tr Extended Description.** *In the dashed list following “When the -d*  
 2971 *option is not specified”, change the second item to:*

2972 — If the -C option is specified, the complement of the characters specified by  
 2973 *string1*—the set of all characters in the current character set, as defined by  
 2974 the current setting of LC\_CTYPE, except for those actually specified in the  
 2975 *string1* operand—shall be placed in the array in ascending collation  
 2976 sequence, as defined by the current setting of LC\_COLLATE.

2977 — If the -c option is specified, the complement of the values specified by  
 2978 *string1* shall be placed in the array in ascending order by binary value.

2979 ⇒ **4.64.7 tr Extended Description.** *In the dashed list following “When the -d*  
 2980 *option is specified”, change the second item to:*

2981 — When the -C option is specified with -d, all characters except those  
 2982 specified by *string1* shall be deleted. The contents of *string2* shall be  
 2983 ignored, unless the -s option is also specified.

2984 — When the -c option is specified with -d, all values except those specified by  
 2985 *string1* shall be deleted. The contents of *string2* shall be ignored, unless  
 2986 the -s option is also specified.

2987 *Editor’s Note: The following rationale will be added to E.4.64, but is kept here with*  
 2988 *tr for this draft:*

2989 **tr Rationale.** *(This subclause is not a part of P1003.2b)*

2990 A prior version of this standard had a -c option that behaved similarly to the -C  
 2991 option, but did not supply functionality equivalent to the -c option specified in  
 2992 this standard. This meant that historical practice of being able to specify `tr -d`  
 2993 `\200-\377` (which would delete all bytes with the top bit set) would have no  
 2994 effect because, in the C locale, bytes with the values octal 200 to octal 377 are not  
 2995 characters.

2996 The earlier standard also said that octal sequences referred to collating elements  
 2997 and could be placed adjacent to each other to specify multibyte characters. How-  
 2998 ever, it was noted that this caused ambiguities because `tr` would not be able to  
 2999 tell whether adjacent octal sequences were intending to specify multibyte charac-  
 3000 ters or multiple single byte characters. This standard specifies that octal  
 3001 sequences always refer to single byte binary values.

3002

B

3003 **4.72 xargs – Construct argument list(s) and invoke utility**3004 ⇒ **4.72.1 xargs Synopsis.** *Change the synopsis to:*

3005       xargs [-t] [-E eofstr] [-n number [-x]] [-s size] [utility [argument  
3006       ... ]]

3007 **Rationale:** This change is required to match historical practice and is the result  
3008 of interpretation request PASC 1003.2-92 #53 submitted for IEEE Std 1003.2-1992.  
3009 See the added rationale in E.4.72 and the following three changes.

3010 ⇒ **4.72.2 xargs Description.** *Replace the last sentence of the first paragraph of  
3011 the Description (the one beginning with “This sequence shall ...”) with:*

3012       This sequence shall be repeated until one of the following occurs:

- 3013       — An end-of-file condition is detected on standard input
- 3014       — The logical end-of-file string (see the -E eofstr option) is found on standard  
3015       input after double-quote processing, apostrophe processing, and backslash  
3016       escape processing (see next paragraph)
- 3017       — An invocation of a constructed command line returns an exit status of 255

3018 **Rationale:** See 4.72.1 change.

3019 ⇒ **4.72.2 xargs Description.** *In the second paragraph, replace the second-to-  
3020 last sentence (“The utility shall be executed one or more times until the end-  
3021 of-file is reached.”) with:*

3022       The *utility* shall be executed one or more times until the end-of-file is reached  
3023       or the logical end-of-file string is found.

3024 **Rationale:** See 4.72.1 change.

3025 ⇒ **4.72.3 xargs Options.** *Add the following option:*

3026       -E eofstr    Use eofstr as the logical end-of-file string. If -E is not  
3027                    specified, it is unspecified whether the logical end-of-file string  
3028                    is the underscore character (\_) or the end-of-file string capabil-  
3029                    ity is disabled. When eofstr is the null string, the logical end-  
3030                    of-file string capability shall be disabled and underscore char-  
3031                    acters shall be taken literally.

3032 **Rationale:** See 4.72.1 change.

3033 *Editor's Note: The following rationale will be added to E.4.72, but is kept here with*  
 3034 *xargs for this draft:*

3035 **xargs Rationale.** (*This subclause is not a part of P1003.2b*)

3036 The `-e` option was omitted from IEEE Std 1003.2-1992 in the belief that the *eofstr*  
 3037 option-argument was recognized only when it was on a line by itself and before  
 3038 quote and escape processing were performed and that the logical end-of-file pro-  
 3039 cessing was only enabled if a `-e` option was specified. In that case, a simple *sed*  
 3040 script could be used to duplicate the `-e` functionality. Further investigation  
 3041 revealed that

3042 — The logical end-of-file string was checked for *after* quote and escape pro-  
 3043 cessing, making a *sed* script that provided equivalent functionality much  
 3044 more difficult to write

3045 — The default was to perform logical end-of-file processing with an underscore  
 3046 as the logical end-of-file string

3047 To correct this misunderstanding, the `-E eofstr` option was adopted from XPG4  
 3048 {B49} in the first revision of this standard. Users should note that the description  
 3049 of the `-E` option matches historical documentation of the `-e` option (which was not  
 3050 adopted because it did not support the utility syntax guidelines), by saying that if  
 3051 *eofstr* is the null string, logical end-of-file processing is disabled. Historical imple-  
 3052 mentations of *xargs* actually did not disable logical end-of-file processing; they  
 3053 treated a null argument found in the input as a logical end-of-file string. (A null  
 3054 string argument could be generated using single or double quotes ( ' ' or " " ).  
 3055 Since this behavior was not documented historically, it is considered to be a bug.

3056 *Editor's Note: The rationale in E.4.72 will also be modified editorially to remove*  
 3057 *the now incorrect reference to `-e eofstr` being replaced by a *sed* script (IEEE Std*  
 3058 *1003.2-1992 page 970, lines 8986–87).*

3059 ⇒ **4.73 iconv — Convert file codesets.** *Add the following new clause:*

3060 **Rationale:** This addition was adopted from XPG4 {B49} to satisfy the following  
 3061 requirement from ISO/IEC 9945-2: 1993 Annex H.1:

3062 (10) A utility (or feature of another utility, such as `tr`) should be provided  
 3063 that converts between character sets encodings based on two charmap  
 3064 files.



3065 **4.73 iconv – Convert file codesets**3066 **4.73.1 Synopsis**3067 `iconv [-cs] [-f fromcode] [-t tocode] [file ...]`3068 `iconv -l`3069 **4.73.2 Description**

3070 The `iconv` utility shall convert the encoding of characters in *file* from one codeset  
3071 to another and write the results to standard output.

3072 When the options indicate that charmap files are used to specify the codesets (see  
3073 4.73.3), the codeset conversion shall be accomplished by performing a logical join  
3074 on the symbolic character names in the two charmaps. The implementation need B  
3075 not support the use of charmap files for codeset conversion unless the B  
3076 {POSIX2\_LOCALEDEF} symbol is defined on the system; see 2.13.2. B

3077 **4.73.3 Options**

3078 The `iconv` utility shall conform to the utility argument syntax guidelines  
3079 described in 2.10.2.

3080 The following options shall be supported by the implementation:

3081 `-c` Omit any invalid characters from the output. When `-c` is not  
3082 used, the results of encountering invalid characters in the input  
3083 stream (either those that are not valid members of the *fromcode*  
3084 or those that have no corresponding value in *tocode*) shall be  
3085 specified in the system documentation. The presence or absence  
3086 of `-c` shall not affect the exit status of `iconv`.

3087 `-f fromcode`  
3088 Identify the codeset of the input file. If the option-argument con-  
3089 tains a slash character, `iconv` shall attempt to use it as the path-  
3090 name of a charmap file, as defined in 2.4.1. If the pathname does  
3091 not represent a valid, readable charmap file, the results are  
3092 undefined. If the option-argument does not contain a slash, it  
3093 shall be considered the name of one of the codeset descriptions  
3094 provided by the system, in an unspecified format. The valid  
3095 values of the option-argument without a slash are implementa-  
3096 tion defined. If this option is omitted, the codeset of the current  
3097 locale shall be used.

3098 `-l` Write all supported *fromcode* and *tocode* values to standard out-  
3099 put in an unspecified format.

- 3100        -s            Suppress any messages written to standard error concerning  
 3101                    invalid characters. When -s is not used, the results of encountering  
 3102                    invalid characters in the input stream (either those that are  
 3103                    not valid members of the *fromcode* or those that have no  
 3104                    corresponding value in *toctype*) shall be specified in the system  
 3105                    documentation. The presence or absence of -s shall not affect the  
 3106                    exit status of *iconv*.
- 3107        -t *toctype*   Identify the codeset of the output file. The semantics are  
 3108                    equivalent to the -f *fromcode* option.
- 3109    If either -f or -t represents a charmap file, but the other does not (or is omitted),  
 3110    or both -f and -t are omitted, the results are undefined.

#### 3111    **4.73.4 Operands**

3112    The following operands shall be supported by the implementation:

- 3113        *file*            A pathname of an input file. If no *file* operands are specified, or if  
 3114                    a *file* operand is -, the standard input shall be used.

#### 3115    **4.73.5 External Influences**

##### 3116    **4.73.5.1 Standard Input**

3117    The standard input shall be used only if no *file* operands are specified, or if a *file*  
 3118    operand is -. See Input Files.

##### 3119    **4.73.5.2 Input Files**

3120    The input files shall be text files.

##### 3121    **4.73.5.3 Environment Variables**

3122    The following environment variables shall affect the execution of *iconv*:

- 3123        **LANG**            This variable shall determine the locale to use for the  
 3124                    locale categories when both **LC\_ALL** and the correspond-  
 3125                    ing environment variable (beginning with **LC\_**) do not  
 3126                    specify a locale. See 2.6.
- 3127        **LC\_ALL**            This variable shall determine the locale to be used to over-  
 3128                    ride any values for locale categories specified by the set-  
 3129                    tings of **LANG** or any environment variables beginning  
 3130                    with **LC\_**.
- 3131        **LC\_CTYPE**        This variable shall determine the locale for the interpreta-  
 3132                    tion of sequences of bytes of text data as characters (e.g.,  
 3133                    single- versus multibyte characters in arguments and  
 3134                    input files). During translation of the file, this variable

3135 shall be superseded by the use of the *fromcode* and *tocode*  
3136 option-arguments.

3137 **LC\_MESSAGES** This variable shall determine the language in which mes-  
3138 sages should be written.

#### 3139 **4.73.5.4 Asynchronous Events**

3140 Default.

#### 3141 **4.73.6 External Effects**

##### 3142 **4.73.6.1 Standard Output**

3143 When the `-l` option is used, the standard output shall contain all supported *from-*  
3144 *code* and *tocode* values, written in an unspecified format.

3145 When the `-l` option is not used, the standard output shall contain the sequence of  
3146 characters read from the input file(s), translated to the specified codeset. Nothing  
3147 else shall be written to the standard output.

##### 3148 **4.73.6.2 Standard Error**

3149 Used only for diagnostic messages.

##### 3150 **4.73.6.3 Output Files**

3151 None.

#### 3152 **4.73.7 Extended Description**

3153 None.

#### 3154 **4.73.8 Exit Status**

3155 The `iconv` utility shall exit with one of the following values:

3156     0     All input files were output successfully.

3157     >0    An error occurred.

3158 **4.73.9 Consequences of Errors**

3159 Default.

3160 **4.73.10 Rationale.** *(This subclause is not a part of P1003.2b)*3161 **Usage, Examples**

3162 The `iconv` utility can be used portably only when the user provides two charmap  
3163 files as option-arguments. This is because a single charmap provided by the user  
3164 cannot reliably be joined with the names in a system-provided character set  
3165 description. The valid values for *fromcode* and *tocode* are implementation defined  
3166 and do not have to have any relation to the charmap mechanisms. As an aid to  
3167 interactive users, the `-l` option was adopted from the Plan 9 operating system. It  
3168 writes information concerning these implementation-defined values. The format  
3169 is unspecified because there are many possible useful formats that could be  
3170 chosen, such as a matrix of valid combinations of *fromcode* and *tocode*. The `-l`  
3171 option is not intended for shell script usage; portable applications will have to use  
3172 charmaps.

3173 The user must ensure that both charmap files use the same symbolic names for  
3174 characters the two codesets have in common.

3175 **History of Decisions Made**

3176 The `iconv` utility is based on one of the same name in XPG4 {B49}. Because of  
3177 requirements from WG15, the ability to use charmap files for the conversion was  
3178 added.

## Section 5: Revisions to User Portability Utilities Option

### 1     **5.2 at – Execute commands at a later time**

2     ⇒ **5.2.1 at Synopsis.** *Change the first synopsis line to:*

3             at [-m] [-f *file*] [-q *queuename*] -t *time\_arg*

4     ⇒ **5.2.3 at Options.** *Change the description of the -t time option to:*

5             -t *time\_arg*

6                     Submit the job to be run at the time specified by the *time\_arg*  
7                     option-argument, which shall have the format as specified by  
8                     the touch -t *time* argument (see 4.63).

9     **Rationale:** The two preceding changes satisfy the following corrigendum request  
10     from ISO/IEC 9945-2: 1993 Annex H.2:

11         (8) In 5.2, the at utility description is confusing because the same symbol  
12         *time* is used for two different values: the -t *time* option-argument and  
13         one of the *timespec* fields.

14     ⇒ **5.2.3 at Options.** *Add the following sentence to the end of the -q description:*

15             If -q b is specified along with either of the -t *time\_arg* or *timespec* argu-  
16             ments, the results are unspecified.

17     **Rationale:** This change satisfies the following requirement from ISO/IEC 9945-  
18     2: 1993 Annex H.1:

19         (21) The effects of the combined use of the at -q b option and the *timespec*  
20         operand should be specified.

21 ⇒ **5.2.6.2 at Standard Error.** *Change the beginning of the first sentence from*  
 22 *“The following shall be written to standard error . . . ” to:*

23 In the POSIX Locale, the following shall be written to standard error . . .

24 **Rationale:** This change satisfies the following corrigendum request from ISO/IEC  
 25 9945-2: 1993 Annex H.2:

26 (9) In 5.2.6.2, the `at` message

27 `"job %s at %s\n", at_job_id, <date>`

28 is in English, but there is no indication of whether it is dependent on the  
 29 POSIX Locale.

### 30 **5.3 batch – Execute commands at a later time**

31 ⇒ **5.3 batch <title>.** *Change the clause title to be:*

#### 32 **5.3 batch – Schedule commands to be executed in a batch queue**

33 ⇒ **5.3.2 batch Description.** *Change the first sentence from “The batch utility*  
 34 *shall read commands to be executed at a later time ” to:*

35 The `batch` utility shall read commands from standard input and schedule  
 36 them for execution in a batch queue.

37 **Rationale:** The preceding two changes satisfy the following requirement from  
 38 ISO/IEC 9945-2: 1993 Annex H.1:

39 (22) The title and description of the `batch` utility should be re-examined for  
 40 their appropriateness and accuracy. Specific reference to “execution in a  
 41 batch queue” should be included.

42 **5.6 `csplit` – Split files based on context**

43 ⇒ **5.6.4 `csplit` Operands.** *Change the descriptions of the `rexp` operands as fol-*  
 44 *lows:*

45 `/rexp[offset]`

46 A file shall be created using the content of the lines from the C  
 47 current line up to, but not including, the line that results from  
 48 the evaluation of the BRE with *offset*, if any, applied. The BRE  
 49 *rexp* shall follow the rules described in 2.8.3. The application  
 50 shall use the sequence `\/` to specify a slash character within  
 51 the *rexp*. The optional *offset* shall be a positive or negative  
 52 integer value representing a number of lines. A positive B  
 53 integer value can be preceded by `+`. If the selection of lines B  
 54 from an offset expression of this type would create a file with  
 55 zero lines, or one with greater than the number of lines left in  
 56 the input file, the results are unspecified. After the section is  
 57 created, the current line shall be set to the line that results  
 58 from the evaluation of the BRE with any offset applied. If the C  
 59 current line is the first line in the file and an RE operation has C  
 60 not yet been performed, the pattern match of *rexp* shall be C  
 61 applied from the current line to the end of the file. Otherwise, C  
 62 the pattern match of *rexp* shall be applied from the line follow- C  
 63 ing the current line to the end of the file. C

64 `%rexp%[offset]`

65 Equivalent to `/rexp[offset]`, except that no file shall be created  
 66 for the selected section of the input file. The application shall  
 67 use the sequence `\%` to specify a percent-sign character within  
 68 the *rexp*.

69 **Rationale:** These `csplit` changes are required to match historical practice and  
 70 are the result of interpretation request PASC 1003.2-92 #59 submitted for IEEE  
 71 Std 1003.2-1992.

72	<b>5.7 ctags – Create a tags file</b>	B
73 74	⇒ <b>5.7.2 ctags Description.</b> <i>Change the third sentence (“A locator consists ...”) to:</i>	B B
75 76	A locator consists of a name, pathname, and either a search pattern or a line number that can be used in searching for the object definition.	B B
77	⇒ <b>5.7.6.3 ctags Output Files.</b> <i>Change this subclause to:</i>	B
78	When the <code>-x</code> option is not specified, the format of the output file shall be	B
79	<code>"%s\t%s\t/%s/\n", &lt;identifier&gt;, &lt;filename&gt;, &lt;pattern&gt;</code>	B
80 81 82	where <i>&lt;pattern&gt;</i> is a search pattern that could be used by an editor to find the defining instance of <i>&lt;identifier&gt;</i> in <i>&lt;filename&gt;</i> (where “defining instance” is indicated by the declarations listed in 5.7.7).	B B B
83 84 85 86 87 88 89	An optional circumflex (^) can be added as a prefix to <i>&lt;pattern&gt;</i> , and an optional dollar sign can be appended to <i>&lt;pattern&gt;</i> to indicate that the pattern is anchored to the beginning (end) of a line of text (see 2.8.4.6). Any slash or backslash characters in <i>&lt;pattern&gt;</i> shall be preceded by a backslash character. The anchoring circumflex, dollar sign, and escaping backslash characters shall not be considered part of the search pattern. All other characters in the search pattern shall be considered literal characters.	B B B B B B B
90	An alternative format is	B
91	<code>"%s\t%s\t?%s?\n", &lt;identifier&gt;, &lt;filename&gt;, &lt;pattern&gt;</code>	B
92 93 94	which is identical to the first format except that slashes in <i>&lt;pattern&gt;</i> shall not be preceded by escaping backslash characters, and question mark characters in <i>&lt;pattern&gt;</i> shall be preceded by backslash characters.	B B B
95	A second alternative format is	B
96	<code>"%s\t%s\t%d\n", &lt;identifier&gt;, &lt;filename&gt;, &lt;lineno&gt;</code>	B
97 98	where <i>&lt;lineno&gt;</i> is a decimal line number that could be used by an editor to find <i>&lt;identifier&gt;</i> in <i>&lt;filename&gt;</i> .	B B
99 100 101	Neither alternative format shall be produced by <code>ctags</code> when it is used as described by this standard, but the standard utilities that process tags files shall be able to process those formats as well as the first format.	B B B
102 103	In any of these formats, the file shall be sorted by identifier, based on the collation sequence in the POSIX Locale.	B B



104 **Rationale:** The preceding changes are the result of interpretation request PASC B  
 105 1003.2-92 #116 submitted for IEEE Std 1003.2-1992. Note related rationale B  
 106 changes in E.5.7. B

## 107 5.9 `du` – Estimate file usage

108 ⇒ **5.9.1 `du` Synopsis.** *Modify the Synopsis to be:*

109 `du [ -a | -s ] [-kx] [ -H | -L ] [file ... ]`

110 ⇒ **5.9.2 `du` Description.** *Add a new sentence in the first paragraph, following*  
 111 *the sentence beginning with “The `du` utility, by default ... ”*

112 By default, when a symbolic link is encountered on the command line or in the B  
 113 file hierarchy, `du` shall count the size of the symbolic link (rather than the file B  
 114 referenced by the link), and shall not follow the link to another portion of the B  
 115 file hierarchy. B

116 ⇒ **5.9.3 `du` Options.** *Add the following options in the proper sorted order:*

117 `-H` If a symbolic link is specified on the command line, `du` shall B  
 118 count the size of the symbolic link and the size of the file or file B  
 119 hierarchy referenced by the link. B

120 `-L` If a symbolic link is specified on the command line or encoun- B  
 121 tered during the traversal of a file hierarchy, `du` shall count B  
 122 the size of the symbolic link and the size of the file or file B  
 123 hierarchy referenced by the link. B

124 ⇒ **5.9.3 `du` Options.** *Add the following paragraph to the end of the subclause:* B

125 Specifying more than one of the mutually exclusive options `-H` and `-L` shall not B  
 126 be considered an error. The last option specified shall determine the behavior B  
 127 of the utility. B

128 ⇒ **5.10 ex – Text editor.** *Replace the entire ex clause with the following.* B

129 *Editor's Note: All of this clause has been changed in Draft 11 from the POSIX.2- B*  
 130 *1992 version. To avoid clutter, it is not further diffmarked. The rationale in B*  
 131 *Annex E is also completely replaced.* B

132 **Rationale:** The changes to the `ex` and `vi` clauses are the result of interpretation C  
 133 requests PASC 1003.2-92 #31, 38, 49, 50, 51, 52, 55, 56, 57, 61, 62, 63, 64, 65, and B  
 134 78, submitted for IEEE Std 1003.2-1992. B

## 135 **5.10 ex – Text editor**

### 136 **5.10.1 Synopsis**

137 `ex [-rR] [-s | -v] [-c command] [-t tagstring] [-w size] [file ...]`

138 *Obsolescent Version:*

139 `ex [-rR] [- | -v] [+command] [-t tagstring] [-w size] [file ...]`

### 140 **5.10.2 Description**

141 The `ex` utility is a line-oriented text editor. There are two other modes of the  
 142 editor—open and visual—in which screen-oriented editing is available. This is  
 143 described more fully by the `ex open` and `visual` commands and in 5.35. The  
 144 user can switch back and forth between `ex` and the screen-oriented editor modes.  
 145 All three modes can be further partitioned into two other modes: command mode  
 146 and text input mode. In command mode, the user is entering commands for the  
 147 editor to execute; in text input mode, the user is entering text into the edit buffer.

148 This clause uses the term “edit buffer” to describe the current working text. No  
 149 specific implementation is implied by this term. All editing changes are per-  
 150 formed on the edit buffer, and no changes to it shall affect any file until an editor  
 151 command writes a file.

152 Certain terminals do not have all the capabilities necessary to support the com-  
 153 plete `ex` definition, such as the full-screen editing commands (open and visual).  
 154 When these commands cannot be supported on such terminals, this condition  
 155 shall neither produce an error message such as “not an editor command” nor  
 156 report a syntax error. The implementation may either accept the commands and  
 157 produce results on the screen that are the result of an unsuccessful attempt to  
 158 meet the requirements of this standard or report an error describing the  
 159 terminal-related deficiency.

160 **5.10.3 Options**

161 The `ex` utility shall conform to the utility argument syntax guidelines described  
 162 in 2.10.2, except for the obsolescent `+command` and `-` “options,” and that the C  
 163 order of presentation of the `+command` and `-c` options is significant. The follow- C  
 164 ing options shall be supported by the implementation:

- 165 `-c command`  
 166 `+command` (Obsolescent.)  
 167 Specify an initial command to be executed in the first edit buffer  
 168 loaded from an existing file (see 5.10.7.1). Implementations may C  
 169 support more than a single `+command` or `-c` option. In such C  
 170 implementations, the specified commands shall be executed in the C  
 171 order specified on the command line. C
- 172 `-r` Recover the named files (see 5.10.7.1). Recovery information for a  
 173 file shall be saved during an editor or system crash (e.g., when  
 174 the editor is terminated by a signal which the editor can catch), or  
 175 after the use of an `ex preserve` command.
- 176 A “crash” in this context is an unexpected failure of the system or  
 177 utility that requires restarting the failed system or utility. A sys-  
 178 tem crash implies that any utilities running at the time also  
 179 crash. In the case of an editor or system crash, the number of  
 180 changes to the edit buffer (since the most recent `preserve` com-  
 181 mand) that will be recovered is unspecified.
- 182 If no *file* operands are given and the `-t` option is not specified, all  
 183 other options, the **EXINIT** variable, and any `.exrc` files shall be  
 184 ignored; a list of all recoverable files available to the invoking  
 185 user shall be written, and the editor shall exit normally without  
 186 further action.
- 187 `-R` Set the readonly edit option.
- 188 `-s`  
 189 `-` (Obsolescent.)  
 190 Prepare `ex` for batch use by taking the following actions:
- 191 — Suppress writing prompts and informational (but not diagnos- C  
 192 tic) messages.
  - 193 — Ignore the value of **TERM** and any implementation default ter-  
 194 minal type and assume the terminal is a type incapable of sup-  
 195 porting open or visual modes; see 5.10.7.5.19, 5.10.7.5.37, and  
 196 the description of `vi` in 5.35.
  - 197 — Suppress the use of the **EXINIT** environment variable (see  
 198 5.10.5.3) and the reading of any `.exrc` file (see 5.10.7.1).
  - 199 — Suppress autoindentation, ignoring the value of the `autoin-` C  
 200 `dent` edit option. C

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

- 201        -t *tagstring*  
 202            Edit the file containing the specified tagstring; see 5.10.7.1 and  
 203            ctags in 5.7. The tags feature represented by -t *tagstring* and  
 204            the tag command (see 5.10.7.5.32) is optional. It shall be pro-  
 205            vided on any system that also provides a conforming implementa-  
 206            tion of ctags; otherwise, the use of -t produces undefined  
 207            results. On any system, it shall be an error to specify more than  
 208            a single -t option. C C
- 209        -v            Begin in visual mode (see 5.35).
- 210        -w *size*        Set the value of the window edit option to *size*.

## 211    5.10.4 Operands

212    The following operand shall be supported by the implementation:

- 213        *file*            A pathname of a file to be edited.

## 214    5.10.5 External Influences

### 215    5.10.5.1 Standard Input

216    The standard input consists of a series of commands and input text, as described  
 217    in 5.10.7. The implementation may limit each line of standard input to a length  
 218    of {LINE\_MAX}. C C

219    If the standard input is not a terminal device, it shall be as if the -s option had  
 220    been specified.

221    If a read from the standard input returns an error, or if the editor detects an end-  
 222    of-file condition from the standard input, it shall be equivalent to a SIGHUP asyn-  
 223    chronous event.

### 224    5.10.5.2 Input Files

225    Input files shall be text files or files that would be text files except for an incom-  
 226    plete last line that is not longer than {LINE\_MAX} - 1 B in length and contains no  
 227    NUL characters. By default, any incomplete last line shall be treated as if it had a  
 228    trailing <newline> character. Other forms of files may optionally be edited by  
 229    implementations. The .exrc files (see 5.10.7.1) and source (see 5.10.7.5.30) files  
 230    shall be text files consisting of ex commands.

231    By default, the editor shall read lines from the files to be edited without interpret-  
 232    ing any of those lines as any form of editor command.

233 **5.10.5.3 Environment Variables**234 The following environment variables shall affect the execution of `ex`:

235	<b>COLUMNS</b>	This variable shall override the system-selected horizontal screen size. See 2.6 for valid values and results when it is	
236		unset or null.	
237			
238	<b>EXINIT</b>	This variable shall be interpreted to contain <code>ex</code> com-	
239		mands, executed during startup. See 5.10.7.1 for more	
240		details.	
241	<b>HOME</b>	This variable shall be interpreted as a pathname of a	
242		directory that shall be searched for an editor startup file	
243		named <code>.exrc</code> ; see 5.10.7.1 for details.	
244	<b>LANG</b>	This variable shall determine the locale to use for the	
245		locale categories when both <code>LC_ALL</code> and the correspond-	
246		ing environment variable (beginning with <code>LC_</code> ) do not	
247		specify a locale. See 2.6.	
248	<b>LC_ALL</b>	This variable shall determine the locale to be used to over-	
249		ride any values for locale categories specified by the set-	
250		tings of <code>LANG</code> or any environment variables beginning	
251		with <code>LC_</code> .	
252	<b>LC_COLLATE</b>	This variable shall determine the locale for character col-	
253		lation information in REs.	
254	<b>LC_CTYPE</b>	This variable shall determine the interpretation of	
255		sequences of bytes of text data as characters (e.g., single-	
256		versus multibyte characters in arguments and input files),	
257		the behavior of character classes within REs, the	
258		classification of characters as upper- or lowercase letters,	
259		the case conversion of letters, and the detection of word	
260		boundaries.	
261	<b>LC_MESSAGES</b>	This variable shall determine the processing of affirmative	
262		responses and the language in which messages should be	
263		displayed or written.	
264	<b>LINES</b>	This variable shall override the system-selected vertical	C
265		screen size, and shall set the value of the <code>window edit</code>	C
266		option. See 2.6 for valid values and results when it is	C
267		unset or null.	
268	<b>PATH</b>	This variable shall determine the search path for the shell	
269		command specified in the <code>ex</code> editor commands <code>!</code> , <code>shell</code> ,	
270		<code>read</code> , <code>write</code> , and the open and visual mode command <code>!</code> ;	
271		see the description of command search and execution in	
272		3.9.1.1.	

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

273	<b>SHELL</b>	This variable shall be used as the default value of the	
274		<code>shell</code> edit option. See 5.10.7.8.18.	
275	<b>TERM</b>	This variable shall be interpreted as the name of the ter-	
276		terminal type. If this variable is unset or null, an	
277		unspecified default terminal type shall be used.	

#### 278 5.10.5.4 Asynchronous Events

279 The following symbol is used in this and following subclauses to specify command  
280 and asynchronous event actions:

281	<i>complete write</i>		
282		A complete write is a write of the entire contents of the edit buffer to	
283		a file of a type other than a terminal device, or, the saving of the edit	
284		buffer caused by the user executing the <code>ex preserve</code> command.	C
285		Writing the contents of the edit buffer to a temporary file that will be	C
286		removed when the editor exits shall not be considered a complete	C
287		write.	C

288 The following actions shall be taken upon receipt of signals:

289	<b>SIGINT</b>	If the standard input is not a terminal device, <code>ex</code> shall not write	C
290		the file or return to command or text input mode, and shall exit	C
291		with a nonzero exit status.	C
292		Otherwise, if executing an open or visual text input mode com-	C
293		mand, <code>ex</code> in receipt of SIGINT shall behave identically to its	C
294		receipt of the <ESC> character.	C
295		Otherwise:	C
296		(1) If executing an <code>ex</code> text input mode command, all input lines	C
297		that have been completely entered shall be resolved into the	C
298		edit buffer, and any partially entered line shall be	C
299		discarded.	C
300		(2) If there is a currently executing command, it shall be	C
301		aborted and a message displayed. Unless otherwise	C
302		specified by the <code>ex</code> or <code>vi</code> command descriptions, it is	C
303		unspecified if any lines modified by the executing command	C
304		appear modified, or as they were before being modified by	C
305		the executing command, in the buffer.	C
306		If the currently executing command was a motion command,	C
307		its associated command shall be discarded.	C
308		(3) If in open or visual command mode, the terminal shall be	C
309		alerted.	C
310		(4) The editor shall then return to command mode.	C
311	<b>SIGCONT</b>	The screen shall be refreshed if in open or visual mode.	

312       SIGHUP  
313       SIGTERM    If the edit buffer has been modified since the last complete write,  
314                    ex shall attempt to save the edit buffer so that it can be recovered  
315                    later using the `-r` option or the `ex recover` command. The edi-  
316                    tor shall not write the file or return to command or text input  
317                    mode, and shall terminate with a nonzero exit status.

318   The action taken for all other signals is unspecified.

## 319   **5.10.6 External Effects**

### 320   **5.10.6.1 Standard Output**

321   The standard output shall be used only for writing prompts to the user, for infor-  
322   mational messages, and for writing lines from the edit buffer.

### 323   **5.10.6.2 Standard Error**

324   Used only for diagnostic messages.

### 325   **5.10.6.3 Output Files**

326   The output from `ex` shall be text files.

## 327   **5.10.7 Extended Description**

328   Only the `ex` mode of the editor is described in this subclause. See 5.35 for addi-  
329   tional editing capabilities available in `ex`.

330   When an error occurs, `ex` shall write a message. If the terminal supports a stan-  
331   dout mode (such as inverse video), the message shall be written in standout mode.  
332   If the terminal does not support a standout mode, and the edit option `error-`  
333   `bells` is set, an alert action shall precede the error message.

334   By default, `ex` shall start in command mode, which shall be indicated by a “:”  
335   prompt (see 5.10.7.8.12). Text input mode can be entered by the `append`, `insert`,  
336   or `change` commands; it can be exited (and command mode re-entered) by typing  
337   a period (.) alone at the beginning of a line.

338 **5.10.7.1 *ex* and *vi* Initialization**

339 The following symbols are used in this and following clauses to specify locations  
340 in the edit buffer.

341 *alternate and current pathnames* C

342 Two pathnames, named *current* and *alternate*, are maintained by the edi- C  
343 tor. Any *ex* commands that take file names as arguments shall set them as C  
344 follows: C

345 (1) If a file argument is specified to the *ex edit*, *ex*, or *recover* com- C  
346 mands, or if an *ex tag* command replaces the contents of the edit C  
347 buffer. C

348 (a) If the command replaces the contents of the edit buffer, the C  
349 current pathname shall be set to the *file* argument or the file C  
350 indicated by the tag, and the alternate pathname shall be set to C  
351 the previous value of the current pathname. C

352 (b) Otherwise, the alternate pathname shall be set to the *file* argu- C  
353 ment. C

354 (2) If a *file* argument is specified to the *ex next* command: C

355 (a) If the command replaces the contents of the edit buffer, the C  
356 current pathname shall be set to the first *file* argument, and the C  
357 alternate pathname shall be set to the previous value of the C  
358 current pathname. C

359 (3) If a *file* argument is specified to the *ex file* command, the current C  
360 pathname shall be set to the *file* argument, and the alternate path- C  
361 name shall be set to the previous value of the current pathname. C

362 (4) If a *file* argument is specified to the *ex read* and *write* commands C  
363 (i.e., when reading or writing a file, and not to the program named by C  
364 the *shell edit* option), or a *file* argument is specified to the *ex exit* C  
365 command: C

366 (a) If the current pathname has no value, the current pathname C  
367 shall be set to the *file* argument. C

368 (b) Otherwise, the alternate pathname shall be set to the *file* argu- C  
369 ment. C

370 If the alternate pathname is set to the previous value of the current C  
371 pathname when the current pathname had no previous value, then C  
372 the alternate pathname shall have no value as a result. C

373 *current line*

374 The line of the edit buffer referenced by the cursor. Each command C  
375 description specifies the current line after the command has been C  
376 executed, as the *Current line* value. When the edit buffer contains C  
377 no lines, the current line shall be zero; see 5.10.7.2. C



378 *current column*  
 379 The current screen column occupied by the cursor. (The columns C  
 380 shall be numbered beginning at 1.) Each command description C  
 381 specifies the current column after the command has been executed,  
 382 as the *Current column* value. This column is an “ideal” column that  
 383 is remembered over the lifetime of the editor. The actual screen  
 384 column upon which the cursor rests may be different from the  
 385 current column; see the cursor positioning discussion in vi (5.35.7.2).

386 *set to nonblank*  
 387 A description for a current column value, meaning that the current  
 388 column shall be set to the last screen column on which is displayed  
 389 any part of the first nonblank character of the line. If the line has no  
 390 nonblank characters, the current column shall be set to the last  
 391 screen column on which is displayed any part of the last character in  
 392 the line. If the line is empty, the current column shall be set to  
 393 column position 1.

394 The length of lines in the edit buffer may be limited to {LINE\_MAX} bytes.  
 395 In open and visual mode, the length of lines in the edit buffer may be lim-  
 396 ited to the number of characters that will fit in the display. If either limit  
 397 is exceeded during editing, an error message shall be written. If either  
 398 limit is exceeded by a line read in from a file, an error message shall be  
 399 written and the edit session may be terminated.

400 C  
 401 If the editor stops running due to any reason other than a user command,  
 402 and the edit buffer has been modified since the last complete write, it shall  
 403 be equivalent to a SIGHUP asynchronous event. If the system crashes, it  
 404 shall be equivalent to a SIGHUP asynchronous event.

405 During initialization (before the first file is copied into the edit buffer or  
 406 any user commands from the terminal are processed)

407 (1) If the environment variable **EXINIT** is set, the editor shall execute the  
 408 `ex` commands contained in that variable.

409 (2) If the **EXINIT** variable is not set, and all of the following are true:

410 (a) The **HOME** environment variable is not null and not empty.

411 (b) The file `.exrc` in the directory referred to by the **HOME** environ-  
 412 ment variable

413 [1] exists

414 [2] is owned by the same user ID as the real user ID of the pro-  
 415 cess or the process has appropriate privileges

416 [3] is not writeable by anyone other than the owner

417 the editor shall execute the `ex` commands contained in that file.

- 418 (3) If and only if all of the following are true:
- 419 (a) The current directory is not referred to by the **HOME** environ-  
420 ment variable.
- 421 (b) A command in the **EXINIT** environment variable or a command  
422 in the `.exrc` file in the directory referred to by the **HOME**  
423 environment variable sets the editor option `exrc`.
- 424 (c) The `.exrc` file in the current directory
- 425 [1] exists
- 426 [2] is owned by the same user ID as the real user ID of the pro-  
427 cess, or by one of a set of implementation defined user IDs
- 428 [3] is not writeable by anyone other than the owner
- 429 the editor shall attempt to execute the `ex` commands contained in  
430 that file.

431 Lines in any `.exrc` file that contain no characters or only `<blank>` charac-  
432 ters shall be ignored. If any `.exrc` file exists, but is not read for ownership C  
433 or permission reasons, it shall be an error. C

434 After the **EXINIT** variable and any `.exrc` files are processed, the first file  
435 specified by the user shall be edited, as follows:

- 436 (1) If the user specified the `-t` option, the effect shall be as if the `ex` tag  
437 command was entered with the specified argument, with the excep-  
438 tion that if tag processing does not result in a file to edit, the effect  
439 shall be as described in step (3) below.
- 440 (2) Otherwise, if the user specified any command-line *file* arguments, the  
441 effect shall be as if the `ex edit` command was entered with the first  
442 of those arguments as its *file* argument.
- 443 (3) Otherwise, the effect shall be as if the `ex edit` command was entered  
444 with a nonexistent file name as its file argument. It is unspecified if  
445 this action shall set the current pathname. In an implementation C  
446 where this action does not set the current pathname, any editor com- C  
447 mand using the current pathname shall fail until an editor command  
448 sets the current pathname.

449 If the `-r` option was specified, the first time a file in the initial argument  
450 list or a file specified by the `-t` option is edited, if recovery information has  
451 previously been saved about it, that information shall be recovered and the  
452 editor shall behave as if the contents of the edit buffer have already been  
453 modified. If there are multiple instances of the file to be recovered, the one  
454 most recently saved shall be recovered, and an informational message that  
455 there are previous versions of the file that can be recovered shall be writ-  
456 ten. If no recovery information about a file is available, an informational  
457 message to this effect shall be written, and the edit shall proceed as usual. C

458 If the `-c` option was specified, the first time a file that already exists  
 459 (including a file that might not exist but for which recovery information is  
 460 available, when the `-r` option is specified) replaces or initializes the con- C  
 461 tents of the edit buffer, the current line shall be set to the last line of the  
 462 edit buffer, the current column shall be set to nonblank, and the `ex` com- C  
 463 mands specified with the `-c` option shall be executed. In this case, the C  
 464 current line and current column shall not be set as described for the com- C  
 465 mand associated with the replacement or initialization of the edit buffer C  
 466 contents. However, if the `-t` option or a `tag` command is associated with C  
 467 this action, the `-c` option commands shall be executed and then the move- C  
 468 ment to the tag shall be performed. C

469 The current argument list shall initially be set to the file names specified  
 470 by the user on the command line. If no file names are specified by the user,  
 471 the current argument list shall be empty. If the `-t` option was specified, it  
 472 is unspecified if any file name resulting from tag processing shall be  
 473 prepended to the current argument list. In the case where the file name is  
 474 added as a prefix to the current argument list, the current argument list  
 475 reference shall be set to that file name. In the case where the file name is  
 476 not added as a prefix to the current argument list, the current argument  
 477 list reference shall logically be located before the first of the file names  
 478 specified on the command line (e.g., a subsequent `ex next` command shall  
 479 edit the first file name from the command line). If the `-t` option was not  
 480 specified, the current argument list reference shall be to the first of the file  
 481 names on the command line.

482 C

### 483 5.10.7.2 Addressing

484 Addressing in `ex` relates to the current line and the current column; the address  
 485 of a line is its 1-based line number, the address of a column is its 1-based count  
 486 from the beginning of the line. Generally, the current line is the last line affected  
 487 by a command. The current line number is the address of the current line. In  
 488 each command description, the effect of the command on the current line number  
 489 and the current column is described.

490 Addresses are constructed as follows:

- 491 (1) The character `.` (period) shall address the current line.
- 492 (2) The character `$` shall address the last line of the edit buffer.
- 493 (3) The positive decimal number  $n$  shall address the  $n$ -th line of the edit  
 494 buffer.
- 495 (4) `'x` shall address the line marked with the mark name character  $x$ , which  
 496 shall be a lowercase letter from the portable character set or one of the  
 497 characters ``` or `'`. It shall be an error if the line that was marked is not  
 498 currently present in the edit buffer or the mark has not been set. Lines  
 499 can be marked with the `ex mark` or `k` commands, or the `vi m` command.

- 500 (5) An RE enclosed by slashes (/) shall address the first line found by search-  
 501 ing forwards from the line following the current line toward the end of  
 502 the edit buffer and stopping at the first line containing a string matching  
 503 the RE. [As stated in 5.10.7.6, an address consisting of a null RE delim-  
 504 ited by slashes (//) shall address the next line containing the last RE  
 505 encountered.] In addition, the second slash can be omitted at the end of  
 506 a command line. If the `wrapsan` edit option is set, the search shall  
 507 wrap around to the beginning of the edit buffer and continue up to and  
 508 including the current line, so that the entire edit buffer is searched.  
 509 Within the RE, the sequence `\ /` shall represent a literal slash instead of  
 510 the RE delimiter.
- 511 (6) An RE enclosed in question marks (?) shall address the first line found by  
 512 searching backwards from the line preceding the current line toward the  
 513 beginning of the edit buffer and stopping at the first line containing a  
 514 string matching the RE. The second question mark can be omitted at the  
 515 end of a command line. If the `wrapsan` edit option is set, the search  
 516 shall wrap around from the beginning of the edit buffer to the end of the  
 517 edit buffer and continue up to and including the current line, so that the  
 518 entire edit buffer is searched. Within the RE, the sequence `\?` shall  
 519 represent a literal question mark instead of the RE delimiter.
- 520 (7) A `+` or `-` immediately followed by a decimal number shall address the  
 521 current line plus or minus the number. A `+` or `-` not followed by a  
 522 decimal number shall address the current line plus or minus 1.

523 Addresses can be followed by zero or more address offsets, optionally `<blank>`  
 524 separated. Address offsets are constructed as follows:

- 525 (1) A `+` or `-` immediately followed by a decimal number shall add (subtract)  
 526 the indicated number of lines to (from) the address. A `+` or `-` not followed  
 527 by a decimal number shall add (subtract) 1 to (from) the address.
- 528 (2) A decimal number shall add the indicated number of lines to the address.

529 It shall not be an error for an intermediate address value to be less than zero or  
 530 greater than the last line in the edit buffer. It shall be an error for the final  
 531 address value to be less than zero or greater than the last line in the edit buffer.

532 Commands take zero, one, or two addresses; see the descriptions of `1addr` and  
 533 `2addr` in 5.10.7.5. If more than the required number of addresses are provided to  
 534 a command that requires zero addresses, it shall be an error. Otherwise, if more  
 535 than the required number of addresses are provided to a command, the addresses  
 536 specified first shall be evaluated and then discarded until the maximum number  
 537 of valid addresses remain.

538 Addresses shall be separated from each other by a comma (,) or a semicolon (;).  
 539 If no address is specified before or after a comma or semicolon separator, it shall  
 540 be as if the address of the current line was specified before or after the separator.  
 541 In the case of a semicolon separator, the current line (.) shall be set to the first  
 542 address, and only then will the next address be calculated. This feature can be

543 used to determine the starting line for forwards and backwards searches [see  
544 rules (5) and (6)].

545 A percent sign (%) shall be equivalent to entering the two addresses 1, \$.

546 Any delimiting <blank> characters between addresses, address separators, or C  
547 address offsets shall be discarded. C

### 548 **5.10.7.3 ex Command-Line Parsing**

549 The following symbol is used in this and following subclauses to describe parsing  
550 behavior:

551 *escape*

552 If a character is referred to as “backslash escaped” or “<control-V>  
553 escaped,” it shall mean that the character acquired or lost a special  
554 meaning by virtue of being preceded, respectively, by a backslash or  
555 <control-V> character. Unless otherwise specified, the escaping char-  
556 acter shall be discarded at that time and shall not be further considered  
557 for any purpose.

558 Command-line parsing shall be done in the following steps. For each step, char-  
559 acters already evaluated shall be ignored; i.e., the phrase “leading character”  
560 refers to the next character that has not yet been evaluated.

- 561 (1) Leading colon characters shall be skipped.
- 562 (2) Leading <blank> characters shall be skipped.
- 563 (3) If the leading character is a double-quote character, the characters up to  
564 and including the next non-backslash-escaped <newline> character C  
565 shall be discarded, and any subsequent characters shall be parsed as a  
566 separate command.
- 567 (4) Leading characters that can be interpreted as addresses shall be  
568 evaluated; see 5.10.7.2.
- 569 (5) Leading <blank> characters shall be skipped.
- 570 (6) If the next character is a vertical-line character or a <newline>  
571 character:
  - 572 (a) If the next character is a <newline> character:
    - 573 [1] If *ex* is in open or visual mode, the current line shall be set to  
574 the last address specified, if any.
    - 575 [2] Otherwise, if the last command was terminated by a vertical-  
576 line character, no action shall be taken; e.g., the command  
577 “|<newline>” shall execute two implied commands, not three.
    - 578 [3] Otherwise, step (6b) shall apply. C
  - 579 (b) Otherwise, the implied command shall be the *print* command. C  
580 The last #, p, and l flags specified to any *ex* command shall be C  
581 remembered and shall apply to this implied command. Executing C

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

582 the `ex` number, `print`, or `list` command shall set the remembered C  
 583 flags to `#`, `nothing`, and `l`, respectively, plus any other flags specified C  
 584 for that execution of the `number`, `print`, or `list` command. C

585 If `ex` is not currently performing a `global` or `v` command, and no  
 586 address or count is specified, the current line shall be incremented  
 587 by 1 before the command is executed. If incrementing the current  
 588 line would result in an address past the last line in the edit buffer,  
 589 the command shall fail, and the increment shall not happen.

590 (c) The `<newline>` or vertical-line character shall be discarded and  
 591 any subsequent characters shall be parsed as a separate command.

592 (7) The command name shall be comprised of the next character, (if the char- C  
 593 acter is not alphabetic) or the next character and any subsequent C  
 594 alphabetic characters (if the character is alphabetic), with the following C  
 595 exceptions: C

596 (a) Commands that consist of any prefix of the characters in the com-  
 597 mand name `delete`, followed immediately by any of the characters  
 598 `l`, `p`, `+`, `-`, or `#` shall be interpreted as a `delete` command, followed  
 599 by a `<blank>` character, followed by the characters that were not  
 600 part of the prefix of the `delete` command. The maximum number  
 601 of characters shall be matched to the command name `delete`; e.g.,  
 602 “`del`” shall not be treated as “`de`” followed by the flag `l`.

603 (b) Commands that consist of the character `k`, followed by a character  
 604 that can be used as the name of a mark, shall be equivalent to the  
 605 `mark` command followed by a `<blank>` character, followed by the  
 606 character that followed the `k`.

607 (c) Commands that consist of the character `s`, followed by character(s)  
 608 that could be interpreted as valid options to the `s` command, shall  
 609 be the equivalent of the `s` command, without any pattern or replace-  
 610 ment values, followed by a `<blank>` character, followed by the char-  
 611 acters after the `s`.

612 C

613 (8) The command name shall be matched against the possible command  
 614 names, and a command name that contains a prefix matching the charac-  
 615 ters specified by the user shall be the executed command. In the case of  
 616 commands where the characters specified by the user could be ambigu-  
 617 ous, the executed command shall be as follows:

618	a	append	n	next	t	t
619	c	change	p	print	u	undo
620	ch	change	pr	print	un	undo
621	e	edit	r	read	v	v

622	m	move	re	read	w	write
623	ma	mark	s	s		

624 Implementation extensions with names causing similar ambiguities shall  
 625 not be checked for a match until all possible matches for commands  
 626 specified by this standard have been checked.

627 (9) If the command is a `!` command, or if the command is a `read` command  
 628 followed by zero or more `<blank>` characters and a `!`, or if the command  
 629 is a `write` command followed by one or more `<blank>` characters and a  
 630 `!`, the rest of the command shall include all characters up to a non-  
 631 backslash-escaped `<newline>`. The `<newline>` shall be discarded and  
 632 any subsequent characters shall be parsed as a separate `ex` command.

633 (10) Otherwise, if the command is an `edit`, `ex` or `next` command, or a  
 634 `visual` command while in open or visual mode, the next part of the com-  
 635 mand shall be parsed as follows:

636 (a) Any `!` character immediately following the command shall be  
 637 skipped and be part of the command.

638 (b) Any leading `<blank>` characters shall be skipped and be part of the  
 639 command.

640 (c) If the next character is a `+`, characters up to the first non-  
 641 backslash-escaped `<newline>` or non-backslash-escaped `<blank>`  
 642 shall be skipped and be part of the command. C

643 (d) The rest of the command shall be determined by the steps specified C  
 644 in paragraph 12. C

645 (11) Otherwise, if the command is a `global`, `open`, `s`, or `v` command, the next C  
 646 part of the command shall be parsed as follows: C

647 (a) Any leading `<blank>` characters shall be skipped and be part of the C  
 648 command. C

649 (b) If the next character is not an alphanumeric, double-quote, `<new-` C  
 650 `line>`, backslash, or vertical-line character: C

651 [1] The next character shall be used as a command delimiter. C

652 [2] If the command is a `global`, `open`, or `v` command, characters C  
 653 up to the first non-backslash-escaped `<newline>` character, or C  
 654 first non-backslash-escaped delimiter character, shall be C  
 655 skipped and be part of the command. C

656 [3] If the command is an `s` command, characters up to the first C  
 657 non-backslash-escaped `<newline>` character, or second non- C  
 658 backslash-escaped delimiter character, shall be skipped and be C  
 659 part of the command. C

660 (c) If the command is a `global` or `v` command, characters up to the C  
 661 first non-backslash-escaped `<newline>` character shall be skipped C  
 662 and be part of the command. C

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 663 (d) Otherwise, the rest of the command shall be determined by the C  
 664 steps specified in paragraph 12. C
- 665 (12) Otherwise:
- 666 (a) If the command was a `map`, `unmap`, `abbreviate`, or `unabbreviate`  
 667 command, characters up to the first non-`<control-V>`-escaped  
 668 `<newline>`, vertical-line, or double-quote character shall be  
 669 skipped and be part of the command.
- 670 (b) Otherwise, characters up to the first non-backslash-escaped `<new-`  
 671 `line>`, vertical-line, or double-quote character shall be skipped and  
 672 be part of the command.
- 673 (c) If the command was an `append`, `change`, or `insert` command, and  
 674 the step (12b) ended at a vertical-line character, any subsequent  
 675 characters, up to the next non-backslash-escaped `<newline>` char-  
 676 acter shall be used as input text to the command.
- 677 (d) If the command was ended by a double-quote character, all subse-  
 678 quent characters, up to the next non-backslash-escaped `<newline>`  
 679 character shall be discarded.
- 680 (e) The terminating `<newline>` or vertical-line character shall be dis-  
 681 carded and any subsequent characters shall be parsed as a separate  
 682 `ex` command.

683 Command arguments shall be parsed as described by the synopsis and description  
 684 of each individual `ex` command. This parsing shall not be `<blank>`-sensitive, C  
 685 except for the `!` argument, which must follow the command name without inter- C  
 686 vening `<blank>` characters, and where it would otherwise be ambiguous. For C  
 687 example, *count* and *flag* arguments need not be `<blank>`-separated because  
 688 “`d22p`” is not ambiguous, but *file* arguments to the `ex next` command must be  
 689 separated by one or more `<blank>` characters. Any `<blank>` character in com-  
 690 mand arguments for the `abbreviate`, `unabbreviate`, `map`, and `unmap` com-  
 691 mands can be `<control-V>`-escaped, in which case the `<blank>` character shall  
 692 not be used as an argument delimiter. Any `<blank>` character in the command  
 693 argument for any other command can be backslash-escaped, in which case that  
 694 `<blank>` character shall not be used as an argument delimiter.

695 Within command arguments for the `abbreviate`, `unabbreviate`, `map`, and  
 696 `unmap` commands, any character can be `<control-V>`-escaped. All such escaped  
 697 characters shall be treated literally and shall have no special meaning. Within  
 698 command arguments for all other `ex` commands that are not REs or replacement  
 699 strings, any character that would otherwise have a special meaning can be  
 700 backslash escaped. Escaped characters shall be treated literally, without special  
 701 meaning as shell expansion characters or `!`, `%`, and `#` expansion characters. See  
 702 5.10.7.6 and 5.10.7.7 for descriptions of command arguments that are REs or  
 703 replacement strings.

704 Non-backslash-escaped `%` characters appearing in *file* arguments to any `ex` com-  
 705 mand shall be replaced by the current pathname; unescaped `#` characters shall be  
 706 replaced by the alternate pathname. It shall be an error if `%` or `#` characters



707 appear unescaped in an argument and their corresponding values are not set.

708 Non-backslash-escaped ! characters in the arguments to either the `ex !` com-  
 709 mand or the open and visual mode ! command, or in the arguments to the `ex`  
 710 `read` command, where the first non-`<blank>` character after the command name  
 711 is a ! character, or in the arguments to the `ex write` command where the com-  
 712 mand name is followed by one or more `<blank>` characters and the first non-  
 713 `<blank>` character after the command name is a ! character, shall be replaced  
 714 with the arguments to the last of those three commands as they appeared after all  
 715 unescaped %, #, and ! characters were replaced. It shall be an error if ! charac-  
 716 ters appear unescaped in one of these commands and there has been no previous  
 717 execution of one of these commands.

718 If an error occurs during the parsing or execution of an `ex` command:

- 719 — An informational message to this effect shall be written. Execution of the C  
 720 `ex` command shall stop, and the cursor (e.g., the current line and column) C  
 721 shall not be further modified. C
- 722 — If the `ex` command resulted from a map expansion, all characters from that  
 723 map expansion shall be discarded, except as otherwise specified by the `map`  
 724 command (see 5.10.7.5.14).
- 725 — Otherwise, if the `ex` command resulted from the processing of an **EXINIT**  
 726 environment variable, a `.exrc` file, a `:source` command, a `-c` option, or a  
 727 `+command` specified to an `ex edit`, `ex next`, or `visual` command, no  
 728 further commands from the source of the commands shall be executed.
- 729 — Otherwise, if the `ex` command resulted from the execution of a `buffer` or a  
 730 `global` or `v` command, no further commands caused by the execution of the  
 731 `buffer` or the `global` or `v` command shall be executed.
- 732 — Otherwise, if the `ex` command was not terminated by a `<newline>` charac- C  
 733 ter, all characters up to and including the next non-backslash-escaped C  
 734 `<newline>` shall be discarded. C

#### 735 5.10.7.4 `ex` Input Editing

736 The following symbols are used in this and following clauses to specify command  
 737 actions.

738 *word* In the POSIX Locale, a word consists of a maximal sequence of letters,  
 739 digits, and underscores, delimited at both ends by characters other than  
 740 letters, digits, or underscores, or by the beginning or end of a line or the  
 741 edit buffer.

742 When accepting input characters from the user, in either `ex` command mode or `ex`  
 743 text input mode, `ex` shall enable canonical mode input processing, as defined in  
 744 POSIX.1 {8}.

745 If in `ex` text input mode:

746 (1) If the `number edit` option is set, `ex` shall prompt for input using the line  
747 number that would be assigned to the line if it is entered, in the format  
748 specified for the `ex number` command.

749 (2) If the `autoindent edit` option is set, `ex` shall prompt for input using  
750 `autoindent` characters, as described by the `autoindent edit` option.  
751 `Autoindent` characters shall follow the line number, if any.

752 If in `ex` command mode:

753 (1) If the `prompt edit` option is set, input shall be prompted for using a single  
754 `:` character; otherwise, there shall be no prompt.

755 The input characters in the following subclauses shall have the following effects  
756 on the input line.

#### 757 **5.10.7.4.1 *eof***

758 *Synopsis:* `eof`

759 See the description of the `stty eof` character in 4.59.

760 If in `ex` command mode:

761 If the `eof` character is the first character entered on the line, the line shall  
762 be evaluated as if it contained two characters: a `<control-D>` and a `<new-`  
763 `line>` character.

764 Otherwise, the `eof` character shall have no special meaning.

765 If in `ex` text input mode:

766 If the cursor follows an `autoindent` character, the `autoindent` characters C  
767 in the line shall be modified so that a part of the next text input character C  
768 will be displayed on the first column in the line after the previous C  
769 `shiftwidth` edit option column boundary, and the user shall be prompted C  
770 again for input for the same line. C

771 Otherwise, if the cursor follows a `0`, which follows an `autoindent` character C  
772 and the `0` was the previous text input character, the `0` and all `autoin-` C  
773 `dent` characters in the line shall be discarded, and the user shall be C  
774 prompted again for input for the same line. C

775 Otherwise, if the cursor follows a `^`, which follows an `autoindent` character C  
776 and the `^` was the previous text input character, the `^` and all `autoin-` C  
777 `dent` characters in the line shall be discarded, and the user shall be C  
778 prompted again for input for the same line. In addition, the `autoindent` C  
779 level for the next input line shall be derived from the same line from which C  
780 the `autoindent` level for the current input line was derived. C

781 Otherwise, if there are no `autoindent` or text input characters in the line, C  
782 the `eof` character shall be discarded. C

783 Otherwise, the `eof` character shall have no special meaning. C

784 **5.10.7.4.2 <newline>**785 *Synopsis:* <newline>786 *Synopsis:* <control-J>

787

C

788 If in *ex* command mode:789 Cause the command line to be parsed; <control-J> shall be mapped to C  
790 the <newline> character for this purpose. C791 If in *ex* text input mode:792 Terminate the current line. If there are no characters other than autoin-  
793 dent characters on the line, all characters on the line shall be discarded.  
794 Prompt for text input on a new line after the current line. If the *autoin-*  
795 *dent edit* option is set, an appropriate number of autoindent characters  
796 shall be added as a prefix to the line as described by the *ex autoindent*  
797 *edit* option.798 **5.10.7.4.3 <backslash>**

C

799 *Synopsis:* <backslash>

C

800 Allow the entry of a subsequent <newline> or <control-J> as a literal charac- C  
801 ter, removing any special meaning that it may have to the editor during text C  
802 input mode. The backslash character shall be retained and evaluated when the C  
803 command line is parsed, or retained and included when the input text becomes C  
804 part of the edit buffer. C805 **5.10.7.4.4 <control-V>**806 *Synopsis:* <control-V>807 Allow the entry of any subsequent character as a literal character, removing any C  
808 special meaning that it may have to the editor during text input mode. The C  
809 <control-V> character shall be discarded before the command line is parsed or  
810 the input text becomes part of the edit buffer.811 If the “literal next” functionality is performed by the underlying system, it is  
812 implementation defined if a character other than <control-V> performs this  
813 function.814 **5.10.7.4.5 <control-W>**815 *Synopsis:* <control-W>816 Discard the <control-W>, and the word previous to it in the input line, including  
817 any <blank> characters following the word and preceding the <control-W>.818 If the “word erase” functionality is performed by the underlying system, it is  
819 implementation-defined if a character other than <control-W> performs this  
820 function.

821 **5.10.7.5 `ex` Command Descriptions**

822 The following symbols are used in this subclause to represent command modifiers.  
 823 Some of these modifiers can be omitted, in which case the specified defaults shall  
 824 be used.

825 *1addr* A single address, given in any of the forms described in 5.10.7.2; the  
 826 default shall be the current line (`.`), unless otherwise specified.

827 If the line address is zero, it shall be an error, unless otherwise  
 828 specified in the following command descriptions.

829 If the edit buffer is empty, and the address is specified with a com-  
 830 mand other than `=`, `append`, `insert`, `open`, `put`, `read`, or `visual`, C  
 831 or the address is not zero, it shall be an error.

832 *2addr* Two addresses specifying an inclusive range of lines. If no addresses  
 833 are specified, the default for *2addr* shall be the current line only  
 834 (`.,.`), unless otherwise specified in the following command descrip-  
 835 tions. If one address is specified, *2addr* shall specify that line only,  
 836 unless otherwise specified in the following command descriptions.

837 It shall be an error if the first address is greater than the second  
 838 address.

839 If the edit buffer is empty, and the two addresses are specified with a  
 840 command other than the `!`, `write`, `wq`, or `xit` commands, or either  
 841 address is not zero, it shall be an error.

842 *count* A positive decimal number. If *count* is specified, it shall be  
 843 equivalent to specifying an additional address to the command,  
 844 unless otherwise specified by the following command descriptions.  
 845 The additional address shall be equal to the last address specified to  
 846 the command (either explicitly or by default) plus *count* - 1.

847 If this would result in an address greater than the last line of the  
 848 edit buffer, it shall be corrected to equal the last line of the edit  
 849 buffer.

850 *flags* One or more of the characters `+`, `-`, `#`, `p`, or `l` (ell). The *flag* charac-  
 851 ters can be <blank>-separated, and in any order or combination.

852 The characters `#`, `p`, and `l` shall cause line(s) to be written in the for- C  
 853 mat specified by the `print` command with the specified flags. C

854 The line(s) to be written are as follows:

855 (1) All edit buffer lines written during the execution of the `ex` `&`, `~`,  
 856 `list`, `number`, `open`, `print`, `s`, `visual`, and `z` commands shall  
 857 be written as specified by any flags.

858 (2) After the completion of an `ex` command with a flag as an argu-  
 859 ment, the current line shall be written as specified by the  
 860 flag(s), unless the current line was the last line written by the  
 861 command.

862 The characters + and – cause the value of the current line after the  
 863 execution of the `ex` command to be adjusted by the offset address as  
 864 described in section 5.10.7.2. This adjustment shall occur before the  
 865 current line is written as described in (2) above.

866 The default for *flags* shall be none.

867 *buffer* One of a number of named areas for holding text. The named buffers  
 868 are specified by the alphanumeric characters of the POSIX Locale.  
 869 There shall also be one “unnamed” buffer. When no buffer is  
 870 specified for editor commands that use a buffer, the unnamed buffer  
 871 shall be used. Commands that store text into buffers shall store the  
 872 text as it was before the command took effect, and shall store text  
 873 occurring earlier in the file before text occurring later in the file,  
 874 regardless of how the text region was specified. Commands that  
 875 store text into buffers shall store the text into the unnamed buffer as  
 876 well as any specified buffer.

877 In `ex` commands, buffer names are specified as the name by itself. C  
 878 In open or visual mode commands the name is preceded by a double  
 879 quote (") character.

880 If the specified buffer name is an uppercase character, and the buffer  
 881 contents are to be modified, the buffer shall be appended to rather  
 882 than being overwritten. If the buffer is not being modified, specify-  
 883 ing the buffer name in lowercase and uppercase shall have identical  
 884 results.

885 There shall also be buffers named by the numbers 1 through 9. In  
 886 open and visual mode, if a region of text including characters from  
 887 more than a single line is being modified by the `vi c` or `d` commands, C  
 888 the motion character associated with the `c` or `d` commands specifies C  
 889 that the buffer text shall be in line mode, or the commands `%`, ```, `/`, `?`, C  
 890 `(`, `)`, `N`, `n`, `{`, or `}` are used to define a region of text for the `c` or `d`  
 891 commands, the contents of buffers 1 through 8 shall be moved into  
 892 the buffer named by the next numerically greater value, the contents  
 893 of buffer 9 shall be discarded, and the region of text shall be copied  
 894 into buffer 1. This shall be in addition to copying the text into a  
 895 user-specified buffer or unnamed buffer, or both. Numeric buffers C  
 896 can be specified as a source buffer for open and visual mode com-  
 897 mands; however, specifying a numeric buffer as the write target of C  
 898 an open or visual mode command shall have unspecified results.

899 The text of each buffer shall have the characteristic of being in either  
 900 line or character mode. Appending text to a nonempty buffer shall  
 901 set the mode to match the characteristic of the text being appended.  
 902 Appending text to a buffer shall cause the creation of at least one  
 903 additional line in the buffer. All text stored into buffers by `ex` com-  
 904 mands shall be in line mode. The `ex` commands that use buffers as  
 905 the source of text specify individually how buffers of different modes  
 906 are handled. Each open or visual mode command that uses buffers

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

907 for any purpose specifies individually the mode of the text stored into  
 908 the buffer and how buffers of different modes are handled.

909 *file* Command text used to derive a pathname. The default shall be the  
 910 current pathname, as defined previously, in which case, if no current  
 911 pathname has yet been established it shall be an error, except where  
 912 specifically noted in the individual command descriptions that follow.  
 913 If the command text contains any of the characters ~, {, [, \*, ?, \$, \,  
 914 ', ", and \, it shall be subjected to the process of “shell expansions,”  
 915 as described below; if more than a single pathname results and the  
 916 command expects only one, it shall be an error.

917 The process of shell expansions in the editor shall be done as follows.  
 918 The *ex* utility shall pass two arguments to the program named by  
 919 the *shell* edit option; the first shall be *-c*, and the second shall be  
 920 the string “echo ” and the command text as a single argument. The  
 921 standard output and standard error of that command shall replace  
 922 the command text.

923 *!* A character that can be appended to the command name to modify  
 924 its operation, as detailed in the individual command descriptions. C  
 925 With the exception of the *ex read*, *write*, and *!* commands, the *!* C  
 926 character shall only act as a modifier if there are no <blank> charac- C  
 927 ters between it and the command name. C

928 *remembered search direction* C  
 929 The *vi* commands *N* and *n* begin searching in a forwards or back- C  
 930 wards direction in the edit buffer based on a remembered search C  
 931 direction, which is initially unset, and is set by the *ex global*, *v*, *s*, C  
 932 and *tag* commands, and the *vi /* and *?* commands. C

933 **5.10.7.5.1 abbreviate**

934 *Synopsis:* *ab*[*breviate*] [*lhs rhs*]

935 If *lhs* and *rhs* are not specified, write the current list of abbreviations and do  
 936 nothing more.

937 Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except  
 938 that printable characters and <blank>s shall not be restricted. Additional res-  
 939 trictions shall be implementation defined.

940 In both *lhs* and *rhs*, any character may be escaped with a <control-V>, in which  
 941 case the character shall not be used to delimit *lhs* from *rhs*, and the escaping  
 942 <control-V> shall be discarded.

943 In open and visual text input mode, if a nonword or <ESC> character that is not  
 944 escaped by a <control-V> character is entered after a word character, a check  
 945 shall be made for a set of characters matching *lhs*, in the text input entered dur- C  
 946 ing this command. If it is found, the effect shall be as if *rhs* was entered instead C  
 947 of *lhs*.

948 The set of characters that are checked is defined as follows:

- 949 (1) If there are no characters inserted before the word and nonword or <ESC>  
 950 characters that triggered the check, the set of characters shall consist of  
 951 the word character.
- 952 (2) If the character inserted before the word and nonword or <ESC> charac-  
 953 ters that triggered the check is a word character, the set of characters  
 954 shall consist of the characters inserted immediately before the triggering  
 955 character(s) that are word characters, plus the triggering word character. C
- 956 (3) If the character inserted before the word and nonword or <ESC> charac-  
 957 ters that triggered the check is not a word character, the set of characters  
 958 shall consist of the characters that were inserted before the triggering  
 959 character(s) that are neither <blank>s nor word characters, plus the C  
 960 triggering word character. C

961 It is unspecified if the *lhs* argument entered for the `ex` abbreviate and unabbrevi-  
 962 ate commands is replaced in this fashion. Regardless of whether or not the  
 963 replacement occurs, the effect of the command shall be as if the replacement had  
 964 not occurred.

965 *Current line*: Unchanged.

966 *Current column*: Unchanged.

#### 967 **5.10.7.5.2** `append`

968 *Synopsis*: [*laddr*] a[ppend][!]

969 Enter `ex` text input mode; the input text shall be placed after the specified line. If  
 970 the line is zero, the text shall be placed at the beginning of the edit buffer.

971 This command shall be affected by the `number` and `autoindent` edit options; fol-  
 972 lowing the command name with `!` shall cause the `autoindent` edit option setting  
 973 to be toggled for the duration of this command only.

974 *Current line*: Set to the last input line; if no lines were input, set to the specified  
 975 line, or to the first line of the edit buffer if a line of zero was specified, or zero if  
 976 the edit buffer is empty.

977 *Current column*: Set to nonblank.

#### 978 **5.10.7.5.3** `args`

979 *Synopsis*: ar[gs]

980 Write the current argument list, with the current argument-list entry, if any,  
 981 between [ and ] characters.

982 *Current line*: Unchanged.

983 *Current column*: Unchanged.

984 **5.10.7.5.4 change**985 *Synopsis:* [2addr] c[hange][!] [count]

986 Enter `ex` text input mode; the input text shall replace the specified lines. The  
987 specified lines shall be copied into the unnamed buffer, which shall become a line  
988 mode buffer.

989 This command shall be affected by the `number` and `autoindent` edit options; fol-  
990 lowing the command name with `!` shall cause the `autoindent` edit option setting  
991 to be toggled for the duration of this command only.

992 *Current line:* Set to the last input line; if no lines were input, set to the line before  
993 the first address, or to the first line of the edit buffer if there are no lines preced-  
994 ing the first address, or to zero if the edit buffer is empty.

995 *Current column:* Set to nonblank.

996 **5.10.7.5.5 chdir**997 *Synopsis:* chd[ir][!] [file]998 *Synopsis:* cd[!] [file]

999 Change the current working directory to *file*.

1000 If no *file* argument is specified, and the **HOME** environment variable is set to a  
1001 nonnull and nonempty value, *file* shall default to the value named in the **HOME**  
1002 environment variable. If the **HOME** environment variable is empty or is  
1003 undefined, the default value of *file* is implementation defined.

1004 If no `!` is appended to the command name, and the edit buffer has been modified  
1005 since the last complete write, and the current pathname does not begin with a `/`,  
1006 it shall be an error.

1007 *Current line:* Unchanged.

1008 *Current column:* Unchanged.

1009 **5.10.7.5.6 copy**1010 *Synopsis:* [2addr] co[py] 1addr [flags]1011 *Synopsis:* [2addr] t 1addr [flags]

1012 Copy the specified lines after the specified destination line; line zero specifies that  
1013 the lines shall be placed at the beginning of the edit buffer.

1014 *Current line:* Set to the last line copied.

1015 *Current column:* Set to nonblank.



1016 **5.10.7.5.7 delete**1017 *Synopsis:* [*2addr*] d[*delete*] [*buffer*] [*count*] [*flags*]1018 Delete the specified lines into a buffer (defaulting to the unnamed buffer), which  
1019 shall become a line-mode buffer.

1020 Flags can immediately follow the command name; see 5.10.7.3.

1021 *Current line:* Set to the line following the deleted lines, or to the last line in the  
1022 edit buffer if that line is past the end of the edit buffer, or to zero if the edit buffer  
1023 is empty.1024 *Current column:* Set to nonblank.1025 **5.10.7.5.8 edit**1026 *Synopsis:* e[*dit*][!] [*+command*] [*file*]1027 *Synopsis:* ex[!] [*+command*] [*file*]1028 If no ! is appended to the command name, and the edit buffer has been modified  
1029 since the last complete write, it shall be an error.1030 If *file* is specified, replace the current contents of the edit buffer with the current  
1031 contents of *file*, and set the current pathname to *file*. If *file* is not specified, replace  
1032 the current contents of the edit buffer with the current contents of the file named  
1033 by the current pathname. If for any reason the current contents of the file cannot  
1034 be accessed, the edit buffer shall be empty.1035 The *+command* option shall be <blank>-delimited; <blank> characters within C  
1036 *+command* can be escaped by preceding them with a backslash character. The  
1037 *+command* shall be interpreted as an *ex* command immediately after the contents  
1038 of the edit buffer have been replaced and the current line and column have been  
1039 set.

1040 If the edit buffer is empty:

1041 *Current line:*

1042 Set to 0.

1043 *Current column:*

1044 Set to 1.

1045 Otherwise, if executed while in *ex* command mode or if the *+command* argument  
1046 is specified:1047 *Current line:*

1048 Set to the last line of the edit buffer.

1049 *Current column:*

1050 Set to nonblank.

1051 Otherwise, if *file* is omitted or results in the current pathname:

1052        *Current line:*  
 1053           Set to the first line of the edit buffer.

1054        *Current column:*  
 1055           Set to nonblank.

1056   Otherwise, if *file* is the same as the last file edited, the line and column shall be  
 1057   set as follows; if the file was previously edited, the line and column may be set as  
 1058   follows:

1059        *Current line:*  
 1060           Set to the last value held when that file was last edited. If this value is not  
 1061           a valid line in the new edit buffer, set to the first line of the edit buffer.

1062        *Current column:*  
 1063           If the current line was set to the last value held when the file was last  
 1064           edited, set to the last value held when the file was last edited. Otherwise,  
 1065           or if the last value is not a valid column in the new edit buffer, set to non-  
 1066           blank.

1067   Otherwise:

1068        *Current line:*  
 1069           Set to the first line of the edit buffer.

1070        *Current column:*  
 1071           Set to nonblank.

1072   **5.10.7.5.9 file**

1073   *Synopsis:*   f[ile] [*file*]

1074   If a *file* argument is specified, the alternate pathname shall be set to the current  
 1075   pathname, and the current pathname shall be set to *file*.

1076   Write an informational message. If the file has a current pathname, it shall be   C  
 1077   included in this message; otherwise, the message shall indicate that there is no   C  
 1078   current pathname. If the edit buffer contains lines, the current line number and   C  
 1079   the number of lines in the edit buffer shall be included in this message; otherwise,   C  
 1080   the message shall indicate that the edit buffer is empty. If the edit buffer has   C  
 1081   been modified since the last complete write, this fact shall be included in this   C  
 1082   message. If the `readonly` edit option is set, this fact shall be included in this   C  
 1083   message. The message may contain other unspecified information.

1084   *Current line:* Unchanged.

1085   *Current column:* Unchanged.

1086 **5.10.7.5.10 global**

1087 *Synopsis:* [*2addr*] g[lobal][!] /[[*pattern*]/ [*commands*]] C

1088 *Synopsis:* [*2addr*] v /[[*pattern*]/ [*commands*]] C

1089 The optional ! character after the global command shall be the same as execut-  
1090 ing the v command.

1091 If *pattern* is empty (e.g., //) or not specified, the last RE used in the editor com- C  
1092 mand shall be used as the pattern. The pattern can be delimited by slashes  
1093 (shown in the Synopsis line), as well as any nonalphanumeric or non-*<blank>*  
1094 character other than backslash, vertical line, double quote, or *<newline>*.

1095 If no lines are specified, the lines shall default to the entire file.

1096 The global and v commands are logically two-pass operations. First, mark the  
1097 lines within the specified lines that match (global) or do not match (v or glo- C  
1098 bal!) the specified pattern. Second, execute the ex command(s) given by *com-* C  
1099 *mands*, with the current line (.) set to each marked line. If an error occurs dur-  
1100 ing this process, or the contents of the edit buffer are replaced (e.g., by the ex  
1101 :edit command) an error message shall be written and no more commands  
1102 resulting from the execution of this command shall be processed.

1103 Multiple ex commands can be specified by entering multiple commands on a sin-  
1104 gle line using a vertical line to delimit them, or one per line, by escaping each  
1105 *<newline>* with a backslash.

1106 If no commands are specified:

1107 (1) If in ex command mode, it shall be as if the print command were  
1108 specified.

1109 (2) Otherwise, no command shall be executed.

1110 For the append, change, and insert commands, the input text shall be included  
1111 as part of the command, and the terminating period can be omitted if the com-  
1112 mand ends the list of commands. The open and visual commands can be  
1113 specified as one of the commands, in which case each marked line shall cause the  
1114 editor to enter open or visual mode. If open or visual mode is exited using the vi  
1115 Q command, the current line shall be set to the next marked line, and open or  
1116 visual mode reentered, until the list of marked lines is exhausted.

1117 The global, v, and undo commands cannot be used in *commands*. Marked lines  
1118 may be deleted by commands executed for lines occurring earlier in the file than  
1119 the marked lines. In this case, no commands shall be executed for the deleted  
1120 lines.

1121 If the remembered search direction is not set, the global and v commands shall C  
1122 set it to forward. C

1123 The autoprint and autoindent edit options shall be inhibited for the duration C  
1124 of the g or v command.

1125 *Current line:*  
 1126 If no commands executed, set to the last marked line. Otherwise, as  
 1127 specified for the executed `ex` commands.

1128 *Current column:*  
 1129 If no commands are executed, set to nonblank; otherwise, as specified for  
 1130 the individual `ex` commands.

#### 1131 **5.10.7.5.11 insert**

1132 *Synopsis:* [*1addr*] i[nsert][!]

1133 Enter `ex` text input mode; the input text shall be placed before the specified line.  
 1134 If the line is zero or 1, the text shall be placed at the beginning of the edit buffer.

1135 This command shall be affected by the `number` and `autoindent` edit options; fol-  
 1136 lowing the command name with `!` shall cause the `autoindent` edit option setting  
 1137 to be toggled for the duration of this command only.

1138 *Current line:* Set to the last input line; if no lines were input, set to the line before  
 1139 the specified line, or to the first line of the edit buffer if there are no lines preced-  
 1140 ing the specified line, or zero if the edit buffer is empty.

1141 *Current column:* Set to nonblank.

#### 1142 **5.10.7.5.12 join**

1143 *Synopsis:* [*2addr*] j[oin][!] [*count*] [*flags*]

1144 If *count* is specified:

1145 If no address was specified, the `join` command shall behave as if *2addr*  
 1146 were the current line and the current line plus *count* (`., . + count`).

1147 If one address was specified, the `join` command shall behave as if *2addr*  
 1148 were the specified address and the specified address plus *count* (`addr,addr`  
 1149 `+ count`).

1150 If two addresses were specified, the `join` command shall behave as if an  
 1151 additional address, equal to the last address plus *count* - 1  
 1152 (`addr1,addr2,addr2 + count - 1`), was specified.

1153 If this would result in a second address greater than the last line of the edit  
 1154 buffer, it shall be corrected to be equal to the last line of the edit buffer.

1155 If no *count* is specified:

1156 If no address was specified, the `join` command shall behave as if *2addr*  
 1157 were the current line and the next line (`., . +1`).

1158 If one address was specified, the `join` command shall behave as if *2addr*  
 1159 were the specified address and the next line (`addr,addr + 1`).

1160 Join the text from the specified lines into a single line, which shall replace the  
 1161 specified lines.

1162 If a `!` character is appended to the command name, the join shall be without  
1163 modification of any line, independent of the current locale.

1164 Otherwise, in the POSIX Locale, set the current line to the first of the specified  
1165 lines, and then, for each subsequent line, proceed as follows:

- 1166 (1) Discard leading spaces from the line to be joined.
- 1167 (2) If the line to be joined is now empty, delete it, and skip steps (3) through  
1168 (5).
- 1169 (3) If the current line ends in a `<blank>` character, or the first character of  
1170 the line to be joined is a `)` character, join the lines without further  
1171 modification.
- 1172 (4) If the last character of the current line is a `.`, join the lines with two  
1173 `<space>` characters between them.
- 1174 (5) Otherwise, join the lines with a single `<space>` character between them.

1175 *Current line*: Set to the first line specified.

1176 *Current column*: Set to nonblank.

#### 1177 **5.10.7.5.13 list**

1178 *Synopsis*: `[2addr] l[ist] [count] [flags]`

1179 This command shall be equivalent to the `ex` command:

1180 `[2addr] p[rint] [count] l[flags]`

1181 See 5.10.7.5.21.

#### 1182 **5.10.7.5.14 map**

1183 *Synopsis*: `map[!] [lhs rhs]`

1184 If *lhs* and *rhs* are not specified:

- 1185 (1) If `!` is specified, write the current list of text input mode maps.
- 1186 (2) Otherwise, write the current list of command mode maps.
- 1187 (3) Do nothing more.

1188 Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except  
1189 that printable characters and `<blank>`s shall not be restricted. Additional res-  
1190 trictions shall be implementation defined.

1191 In both *lhs* and *rhs*, any character can be escaped with a `<control-V>`, in which  
1192 case the character shall not be used to delimit *lhs* from *rhs*, and the escaping  
1193 `<control-V>` shall be discarded.

1194 If the character `!` is appended to the `map` command name, the mapping shall be  
1195 effective during open or visual text input mode rather than open or visual com-  
1196 mand mode. This allows *lhs* to have two different map definitions at the same  
1197 time: one for command mode and one for text input mode.

1198 For command mode mappings:

1199 When the *lhs* is entered as any part of a `vi` command in open or visual  
1200 mode (but not as part of the arguments to the command), the action shall  
1201 be as if the corresponding *rhs* had been entered.

1202 If any character in the command, other than the first, is escaped using a  
1203 `<control-V>` character, that character shall not be part of a match to an  
1204 *lhs*.

1205

C

1206 It is unspecified if implementations shall support command maps where  
1207 the *lhs* is more than a single character in length, where the first character  
1208 of the *lhs* is printable.

1209 If *lhs* contains more than one character and the first character is `#`, addi-  
1210 tional, unspecified character(s), representing the function key named by the  
1211 characters in *lhs* following the `#`, may be mapped to *rhs*. It is unspecified  
1212 how function keys are named or what function keys are supported.

1213 For text input mode mappings:

1214 When the *lhs* is entered as any part of text entered in open or visual text  
1215 input modes, the action shall be as if the corresponding *rhs* had been  
1216 entered.

1217 If any character in the input text is escaped using a `<control-V>` charac-  
1218 ter, that character shall not be part of a match to an *lhs*.

1219 It is unspecified if the *lhs* argument entered for the `map` or `unmap` com- C  
1220 mands is replaced in this fashion. Regardless of whether or not the  
1221 replacement occurs, the effect of the command shall be as if the replace-  
1222 ment had not occurred.

1223 If only part of the *lhs* is entered, it is unspecified how long the editor will wait for  
1224 additional, possibly matching characters before treating the already entered char-  
1225 acters as not matching the *lhs*.

1226 The *rhs* characters shall themselves be subject to remapping, unless otherwise  
1227 specified by the `remap edit` option, except that if the characters in *lhs* occur as  
1228 prefix characters in *rhs*, those characters shall not be remapped.

1229 On block-mode terminals, the mapping need not occur immediately (for example,  
1230 it may occur after the terminal transmits a group of characters to the system), but  
1231 it shall achieve the same results as if it occurred immediately.

1232 *Current line*: Unchanged.

1233 *Current column*: Unchanged.

1234 **5.10.7.5.15 mark**1235 *Synopsis:* [*laddr*] ma[*rk*] *character*1236 *Synopsis:* [*laddr*] k *character*

1237 Implementations shall support *character* values of a single lowercase letter of the  
 1238 POSIX Locale and the characters ` and `; support of other characters is imple-  
 1239 mentation defined.

1240 If executing the `vi m` command, set the specified mark to the current line and 1- C  
 1241 based numbered character referenced by the current column, if any; otherwise, C  
 1242 column position 1. C

1243 Otherwise, set the specified mark to the specified line and 1-based numbered first C  
 1244 non-<blank> character in the line, if any; otherwise, the last character in the  
 1245 line, if any; otherwise, column position 1. C

1246 The mark shall remain associated with the line until the mark is reset or the line C  
 1247 is deleted. If a deleted line is restored by a subsequent `undo` command, any C  
 1248 marks previously associated with the line, which have not been reset, shall be C  
 1249 restored as well. Any use of a mark not associated with a current line in the edit C  
 1250 buffer shall be an error. C

1251 The marks ` and ` shall be set as described previously, immediately before the  
 1252 following events occur in the editor:

1253 (1) The use of \$ as an `ex` address1254 (2) The use of a positive decimal number as an `ex` address1255 (3) The use of a search command as an `ex` address1256 (4) The use of a mark reference as an `ex` address

1257 (5) The use of the following open and visual mode commands:

1258 &lt;control-]&gt; % ( ) [ ] { }

1259 (6) The use of the following open and visual mode commands:

1260 ' G H L M z

1261 if the current line will change as a result of the command

1262 (7) The use of the open and visual mode commands:

1263 / ? N ` n

1264 if the current line or column will change as a result of the command

1265 (8) The use of the `ex` mode commands:

1266 z undo global v

1267 For rules (1), (2), (3), and (4), the ` and ` marks shall not be set if the `ex` com-  
 1268 mand is parsed as specified by rule (6a) in 5.10.7.3.

1269 For rules (5), (6), and (7), the ` and ` marks shall not be set if the commands are  
 1270 used as motion commands in open and visual mode.

- 1271 For rules (1), (2), (3), (4), (5), (6), (7), and (8), the ` and ` marks shall not be set if C  
 1272 the command fails. C
- 1273 The ` and ` marks shall be set as described previously, each time the contents of C  
 1274 the edit buffer are replaced (including the editing of the initial buffer), if in open C  
 1275 or visual mode, or if in `ex` mode and the edit buffer is not empty, before any com- C  
 1276 mands or movements (including commands or movements specified by the `-c` or C  
 1277 `-t` options or the `+command` argument) are executed on the edit buffer. If in open C  
 1278 or visual mode, the marks shall be set as if executing the `vi m` command; other- C  
 1279 wise, as if executing the `ex mark` command. C
- 1280 When changing from `ex` mode to open or visual mode, if the ` and ` marks are C  
 1281 not already set, the ` and ` marks shall be set as described previously. C
- 1282 *Current line*: Unchanged.
- 1283 *Current column*: Unchanged.
- 1284 **5.10.7.5.16 move**
- 1285 *Synopsis*: [`2addr`] `m[ove]` [`1addr`] [`flags`]
- 1286 Move the specified lines after the specified destination line. A destination of line C  
 1287 zero specifies that the lines shall be placed at the beginning of the edit buffer. It C  
 1288 shall be an error if the destination line is within the range of lines to be moved.
- 1289 *Current line*: Set to the last of the moved lines.
- 1290 *Current column*: Set to nonblank.
- 1291 **5.10.7.5.17 next**
- 1292 *Synopsis*: `n[ext][!]` [`+command`] [`file ...`]
- 1293 If no `!` is appended to the command name, and the edit buffer has been modified C  
 1294 since the last complete write, it shall be an error, unless the file is successfully C  
 1295 written as specified by the `autowrite` option.
- 1296 If one or more files is specified:
- 1297 (1) Set the argument list to the specified file names.
- 1298 (2) Set the current argument list reference to be the first entry in the argu- C  
 1299 ment list.
- 1300 (3) Set the current pathname to the first file name specified.
- 1301 Otherwise:
- 1302 (1) It shall be an error if there are no more file names in the argument list C  
 1303 after the file name currently referenced.
- 1304 (2) Set the current pathname and the current argument list reference to the C  
 1305 file name after the file name currently referenced in the argument list.
- 1306 Replace the contents of the edit buffer with the contents of the file named by the C  
 1307 current pathname. If for any reason the contents of the file cannot be accessed,



- 1308 the edit buffer shall be empty.
- 1309 This command shall be affected by the `autowrite` and `writeany` edit options.
- 1310 The *+command* option shall be <blank>-delimited; <blank> characters can be  
 1311 escaped by preceding them with a backslash character. The *+command* shall be  
 1312 interpreted as an `ex` command immediately after the contents of the edit buffer  
 1313 have been replaced and the current line and column have been set.
- 1314 *Current line*: Set as described for the `edit` command.
- 1315 *Current column*: Set as described for the `edit` command.
- 1316 **5.10.7.5.18 number**
- 1317 *Synopsis*: [*2addr*] nu[mber] [*count*] [*flags*]  
 1318 *Synopsis*: [*2addr*] # [*count*] [*flags*]
- 1319 These commands shall be equivalent to the `ex` command: C
- 1320        [*2addr*] p[rint] [*count*] #[*flags*] C
- 1321 See 5.10.7.5.21. C
- 1322 **5.10.7.5.19 open**
- 1323 *Synopsis*: [*1addr*] o[pen] [/ *pattern* [/]] [*flags*] C
- 1324 This command need not be supported on block-mode terminals or terminals with  
 1325 insufficient capabilities. If standard input, standard output, or standard error are  
 1326 not terminal devices, the results are unspecified.
- 1327 Enter open mode.
- 1328 The trailing delimiter can be omitted from *pattern* at the end of the command C  
 1329 line. If *pattern* is empty (e.g., `/`) or not specified, the last RE used in the editor C  
 1330 shall be used as the pattern. The pattern can be delimited by slashes (shown in C  
 1331 the *Synopsis* line), as well as any alphanumeric, or non-<blank> character other C  
 1332 than backslash, vertical line, double quote, or <newline>. C
- 1333 If a match is found for the optional RE in the line, the cursor shall be placed at the  
 1334 start of the matching pattern. If the pattern is not found, it shall be an error.
- 1335 *Current line*: Set to the specified line.
- 1336 *Current column*: Set to nonblank.
- 1337 **5.10.7.5.20 preserve**
- 1338 *Synopsis*: pre[serve]
- 1339 Save the edit buffer in a form that can later be recovered by using the `-r` option or  
 1340 by using the `ex recover` command. After the file has been preserved, a mail  
 1341 message shall be sent to the user. This message shall be readable by invoking the  
 1342 `mailx` utility (see 4.40). The message shall contain the name of the file, the time  
 1343 of preservation, and an `ex` command that could be used to recover the file.

- 1344 Additional, unspecified, information may be included in the mail message.
- 1345 *Current line*: Unchanged.
- 1346 *Current column*: Unchanged.
- 1347 **5.10.7.5.21 print**
- 1348 *Synopsis*: [*2addr*] p[rint] [*count*] [*flags*]
- 1349 Write the addressed lines. The behavior is unspecified if the number of columns C  
 1350 on the display is less than the number of columns required to write any single C  
 1351 character in the line(s) being written. C
- 1352 Nonprintable characters, except for <tab>, shall be written as implementation- C  
 1353 defined multicharacter sequences. C
- 1354 If the # flag is specified or the number edit option is set, each line shall be pre- C  
 1355 ceded by its line number in the following format: C
- 1356 "%6dΔΔ", <line number> C
- 1357 If the l flag is specified or the list edit option is set: C
- 1358 (1) The characters listed in Table 2-16 (see 2.12) shall be written as the C  
 1359 corresponding escape sequence. C
- 1360 (2) Nonprintable characters not in Table 2-16 shall be written as one three- C  
 1361 digit octal number (with a preceding <backslash>) for each byte in the C  
 1362 character (most significant byte first). If the size of a byte on the system C  
 1363 is greater than 9 b, the format used for nonprintable characters is imple- C  
 1364 mentation defined. C
- 1365 (3) The end of each line shall be marked with a \$, and literal \$ characters C  
 1366 within the line shall be written with a preceding backslash. C
- 1367 Long lines shall be folded. The length at which folding occurs is unspecified, but C  
 1368 folding should be as appropriate for the output terminal, considering the number C  
 1369 of columns of the terminal. C
- 1370 If a line is folded, and the l flag is specified or the list edit option is set: C
- 1371 (1) The point of folding shall be indicated by writing <backslash> <new- C  
 1372 line>. C
- 1373 (2) A multicolumn character at the folding position shall be neither C  
 1374 separated nor discarded. C
- 1375 If a line is folded, and the l flag is not specified and the list edit option is not C  
 1376 set, it is unspecified if a multicolumn character at the folding position is C  
 1377 separated; it shall not be discarded. C
- 1378 *Current line*: Set to the last line written.
- 1379 *Current column*: Unchanged if the current line is unchanged; otherwise, set to C  
 1380 nonblank.

1381 **5.10.7.5.22 put**1382 *Synopsis:* [*laddr*] pu[t] [*buffer*]

1383 Append text from the specified buffer (by default, the unnamed buffer) to the  
 1384 specified line; line zero specifies that the text shall be placed at the beginning of  
 1385 the edit buffer. Each portion of a line in the buffer shall become a new line in the  
 1386 edit buffer, regardless of the mode of the buffer.

1387 *Current line:* Set to the last line entered into the edit buffer.1388 *Current column:* Set to nonblank.1389 **5.10.7.5.23 quit**1390 *Synopsis:* q[uit][!]

1391 If no ! is appended to the command name

1392 (1) If the edit buffer has been modified since the last complete write, it shall  
 1393 be an error.

1394 (2) If there are file names in the argument list after the file name currently  
 1395 referenced, and the last command was not a quit, wq, xit, or ZZ (see  
 1396 5.35.7.2.85) command, it shall be an error.

1397 Otherwise, terminate the editing session.

1398 **5.10.7.5.24 read**1399 *Synopsis:* [*laddr*] r[ead][!] [*file*]

1400 If ! is not the first non-`<blank>` character to follow the command name, a copy of  
 1401 the specified file shall be appended into the edit buffer after the specified line; line  
 1402 zero specifies that the copy shall be placed at the beginning of the edit buffer. The  
 1403 number of lines and bytes read shall be written. If no *file* is named, the current  
 1404 pathname shall be the default. If there is no current pathname, then *file* shall  
 1405 become the current pathname. If there is no current pathname or *file* operand, it  
 1406 shall be an error. Specifying a *file* that is not of type regular shall have  
 1407 unspecified results.

1408 Otherwise, if *file* is preceded by !, the rest of the line after the ! shall have %, #,  
 1409 and ! characters expanded as described in 5.10.7.3.

1410 The `ex` utility shall then pass two arguments to the program named by the `shell`  
 1411 edit option; the first shall be “-c” and the second shall be the expanded argu- C  
 1412 ments to the `read` command as a single argument. The standard input of the pro- C  
 1413 gram shall be set to the standard input of the `ex` program when it was invoked.  
 1414 The standard error and standard output of the program shall be appended into C  
 1415 the edit buffer after the specified line.

1416 Each line in the copied file or program output (as delimited by `<newline>` charac-  
 1417 ters or the end of the file or output if it is not immediately preceded by a `<new-`  
 1418 `line>` character), shall be a separate line in the edit buffer. Any occurrences of  
 1419 `<carriage-return>` and `<newline>` character pairs in the output shall be

1420 treated as single `<newline>` characters.

1421 The special meaning of the `!` following the `read` command can be overridden by  
1422 escaping it with a backslash character.

1423 *Current line:*

1424 If no lines are added to the edit buffer, unchanged.

1425 Otherwise, if in open or visual mode, set to the first line entered into the  
1426 edit buffer.

1427 Otherwise, set to the last line entered into the edit buffer.

1428 *Current column:*

1429 Set to nonblank.

#### 1430 **5.10.7.5.25 recover**

1431 *Synopsis:* `rec[over][!] [file]`

1432 If no `!` is appended to the command name, and the edit buffer has been modified  
1433 since the last complete write, it shall be an error.

1434 If no *file* operand is specified, then the current pathname shall be used. If there is C  
1435 no current pathname or *file* operand, it shall be an error. C

1436 If no recovery information has previously been saved about *file*, the `recover` com-  
1437 mand shall behave identically to the `edit` command, and an informational mes-  
1438 sage to this effect shall be written.

1439 Otherwise, set the current pathname to *file*, and replace the current contents of  
1440 the edit buffer with the recovered contents of *file*. If there are multiple instances  
1441 of the file to be recovered, the one most recently saved shall be recovered, and an  
1442 informational message that there are previous versions of the file that can be  
1443 recovered shall be written. The editor shall behave as if the contents of the edit  
1444 buffer have already been modified.

1445 *Current line:* Set as described for the `edit` command.

1446 *Current column:* Set as described for the `edit` command.

#### 1447 **5.10.7.5.26 rewind**

1448 *Synopsis:* `rew[ind][!]`

1449 If no `!` is appended to the command name, and the edit buffer has been modified  
1450 since the last complete write, it shall be an error, unless the file is successfully  
1451 written as specified by the `autowrite` option.

1452 If the argument list is empty, it shall be an error.

1453 The current argument list reference and the current pathname shall be set to the  
1454 first file name in the argument list.

1455 Replace the contents of the edit buffer with the contents of the file named by the  
 1456 current pathname. If for any reason the contents of the file cannot be accessed,  
 1457 the edit buffer shall be empty.

1458 This command shall be affected by the `autowrite` and `writeany` edit options.

1459 *Current line*: Set as described for the `edit` command.

1460 *Current column*: Set as described for the `edit` command.

#### 1461 **5.10.7.5.27 s**

1462 *Synopsis*: [*2addr*] `s` [/*[pattern]*[/*[repl]*[/]]] [*options*] [*count*] [*flags*] C

1463 *Synopsis*: [*2addr*] `&` [*options*] [*count*] [*flags*]

1464 *Synopsis*: [*2addr*] `~` [*options*] [*count*] [*flags*]

1465 Replace the first instance of *pattern* with the string *repl* on each specified line.  
 1466 (See 5.10.7.6 and 5.10.7.7.) Any nonalphabetic, nonblank delimiter other than \,  
 1467 |, double quote or <newline> can be used instead of /. Backslash characters can  
 1468 be used to escape delimiters, backslash characters, and other special characters.

1469 The trailing delimiter can be omitted from *pattern* or from *repl* at the end of the  
 1470 command line. If both *pattern* and *repl* are not specified or are empty (e.g., //), C  
 1471 the last `s` command shall be repeated. If only *pattern* is not specified or is empty, C  
 1472 the last RE used in the editor shall be used as the pattern. If only *repl* is not C  
 1473 specified or is empty, the pattern shall be replaced by nothing. If the entire C  
 1474 replacement pattern is %, the last replacement pattern to an `s` command shall be  
 1475 used.

1476 Entering a <carriage-return> in *repl* (which requires an escaping backslash in  
 1477 `ex` mode and an escaping <control-V> in `open` or `vi` mode) shall split the line at  
 1478 that point, creating a new line in the edit buffer. The <carriage-return> shall  
 1479 be discarded.

1480 If *options* includes the letter `g` (`global`), all nonoverlapping instances of the *pat-*  
 1481 *tern* in the line shall be replaced.

1482 If *options* includes the letter `c` (`confirm`), then before each substitution the line  
 1483 shall be written; the written line shall reflect all previous substitutions. On the C  
 1484 following line, <space> characters shall be written beneath the characters from C  
 1485 the line that are before the *pattern* to be replaced, and ^ characters written  
 1486 beneath the characters included in the *pattern* to be replaced. The `ex` utility shall  
 1487 then wait for a response from the user. An affirmative response shall cause the  
 1488 substitution to be done, while any other input shall not make the substitution.  
 1489 An affirmative response shall consist of a line with the affirmative response (as  
 1490 defined by the current locale) at the beginning of the line. This line shall be sub-  
 1491 ject to editing in the same way as the `ex` command line.

1492 If interrupted (see 5.10.5.4), any modifications confirmed by the user shall be C  
 1493 preserved in the edit buffer after the interrupt. C

1494 If the remembered search direction is not set, the `s` command shall set it to for- C  
 1495 ward (see 5.35.7.2.63 and 5.35.7.2.64). C

1496 In the second synopsis, the `&` command shall repeat the previous substitution, as  
 1497 if the `&` command were replaced by `s/pattern/repl/`, where *pattern* and *repl* are  
 1498 as specified in the previous `s`, `&`, or `~` command.

1499 In the third synopsis, the `~` command shall repeat the previous substitution, as if  
 1500 the `~` were replaced by `s/pattern/repl/`, where *pattern* shall be the last RE  
 1501 specified to the editor, and *repl* shall be from the previous substitution (including  
 1502 `&` and `~`) command.

1503 These commands shall be affected by the `LC_MESSAGES` environment variable.

1504 *Current line*: Set to the last line in which a substitution occurred, or, unchanged if  
 1505 no substitution occurred.

1506 *Current column*: Set to nonblank.

#### 1507 **5.10.7.5.28 set**

1508 *Synopsis*: `se[t] [option[=value]] . . . ] [[no]option . . . ] [option? . . . ] [all]`

1509 When no arguments are specified, write the value of the `term` edit option and  
 1510 those options whose values have been changed from the default settings; when  
 1511 the argument `all` is specified, write all of the option values.

1512 Giving an option name followed by the character `?` shall cause the current value  
 1513 of that option to be written. The `?` can be separated from the option name by zero  
 1514 or more `<blank>s`. The `?` shall be necessary only for Boolean valued options.  
 1515 Boolean options can be given values by the form `set option` to turn them on or  
 1516 `set nooption` to turn them off; string and numeric options can be assigned by the  
 1517 form `set option=value`. Any `<blank>s` in strings can be included as is by preced-  
 1518 ing each `<blank>` with an escaping backslash. More than one option can be set  
 1519 or listed by a single `set` command by specifying multiple arguments, each  
 1520 separated from the next by one or more `<blank>s`.

1521 See 5.10.7.8 for details about specific options.

1522 *Current line*: Unchanged.

1523 *Current column*: Unchanged.

#### 1524 **5.10.7.5.29 shell**

1525 *Synopsis*: `sh[ell]`

1526 Invoke the program named by the `shell` edit option with the single argument `-i`.  
 1527 Editing shall be resumed when the program exits.

1528 *Current line*: Unchanged.

1529 *Current column*: Unchanged.

1530 **5.10.7.5.30 source**1531 *Synopsis:* so[urce] *file*1532 Read and execute *ex* commands from *file*. Lines in the file that contain no charac-  
1533 ters or only <blank> characters shall be ignored.1534 *Current line:* As specified for the individual *ex* commands.1535 *Current column:* As specified for the individual *ex* commands.1536 **5.10.7.5.31 suspend**1537 *Synopsis:* su[suspend][!]1538 *Synopsis:* st[op][!]1539 Allow control to return to the invoking process; *ex* shall suspend itself as if it had  
1540 received the SIGTSTP signal. If the system does not support job control as  
1541 described in POSIX.1 {8}, it shall be an error. If job control is not enabled for any  
1542 reason, the results of the command are unspecified.1543 These commands shall be affected by the *autowrite* and *writeln* edit options.1544 The current *susp* character (see *stty* in 4.59) shall have the same effect as the  
1545 *suspend* command.1546 *Current line:* Unchanged.1547 *Current column:* Unchanged.1548 **5.10.7.5.32 tag**1549 *Synopsis:* ta[g][!] *tagstring*1550 The results are unspecified if the format of a tags file is not as specified by the  
1551 *ctags* utility (5.7) description.1552 The *tag* command shall search for *tagstring* in the tag file(s) referred to by the  
1553 *tag* edit option, in the order they are specified, until a reference to *tagstring* is  
1554 found. Files shall be searched from beginning to end. If no reference is found, it C  
1555 shall be an error and an error message to this effect shall be written. If no refer- C  
1556 ence is found and a file referred to by the *tag* edit option does not exist, is not  
1557 readable, or has an unspecified problem, an error message shall be written. This  
1558 error message shall only be displayed the first time a tag is not found and a file in  
1559 the *tag* edit option has a problem.1560 Otherwise, if the tags file contained a pattern, the pattern shall be treated as an C  
1561 RE used in the editor; e.g., for the purposes of the *s* command. C1562 If the *tagstring* is in a file with a different name than the current pathname, set  
1563 the current pathname to the name of that file, and replace the contents of the edit  
1564 buffer with the contents of that file. In this case, if no ! is appended to the com- C  
1565 mand name, and the edit buffer has been modified since the last complete write, it  
1566 shall be an error, unless the file is successfully written as specified by the  
1567 *autowrite* option.

1568 This command shall be affected by the `autowrite`, `tag`, `taglength`, and `wri-`  
1569 `teany` edit options.

1570 *Current line:*

1571 If the tags file contained a line number, set to that line number. If the line  
1572 number is larger than the last line in the edit buffer, an error message  
1573 shall be written and the current line shall be set as specified for the `edit`  
1574 command.

1575 If the tags file contained a pattern, set to the first occurrence of the pattern  
1576 in the file. If no matching pattern is found, an error message shall be writ-  
1577 ten and the current line shall be set as specified for the `edit` command.

1578 *Current column:*

1579 If the tags file contained a line-number reference and that line-number was  
1580 not larger than the last line in the edit buffer, or if the tags file contained a  
1581 pattern and that pattern was found, set to `nonblank`.

1582 Otherwise, set as specified for the `edit` command.

### 1583 **5.10.7.5.33 unabbreviate**

1584 *Synopsis:* `una[bbreviate] lhs`

1585 If `lhs` is not an entry in the current list of abbreviations (see 5.10.7.5.1), it shall be  
1586 an error. Otherwise, delete `lhs` from the list of abbreviations.

1587 *Current line:* Unchanged.

1588 *Current column:* Unchanged.

### 1589 **5.10.7.5.34 undo**

1590 *Synopsis:* `u[ndo]`

1591 Reverse the changes made by the last command that modified the contents of the  
1592 edit buffer, including `undo`. For this purpose, the `global`, `v`, `open`, and `visual`  
1593 commands, and commands resulting from buffer executions and mapped charac-  
1594 ter expansions, are considered single commands.

1595 If no action that can be undone preceded the `undo` command, it shall be an error.

1596 If the `undo` command restores lines that were marked, the mark shall also be  
1597 restored unless it was reset subsequent to the deletion of the lines.

1598 *Current line:*

1599 (1) If lines are added or changed in the file, set to the first line added or  
1600 changed.

1601 (2) Set to the line before the first line deleted, if it exists.

1602 (3) Set to 1 if the edit buffer is not empty.



1603 (4) Set to zero.

1604 *Current column*: Set to nonblank.

1605 **5.10.7.5.35 unmap**

1606 *Synopsis*: unm[ap][!] *lhs*

1607 If ! is appended to the command name, and if *lhs* is not an entry in the list of text  
1608 input mode map definitions, it shall be an error. Otherwise, delete *lhs* from the  
1609 list of text input mode map definitions.

1610 If no ! is appended to the command name, and if *lhs* is not an entry in the list of  
1611 command mode map definitions, it shall be an error. Otherwise, delete *lhs* from  
1612 the list of command mode map definitions.

1613 *Current line*: Unchanged.

1614 *Current column*: Unchanged.

1615 **5.10.7.5.36 version**

1616 *Synopsis*: ve[rsion]

1617 Write a message containing version information for the editor. The format of the  
1618 message is unspecified.

1619 *Current line*: Unchanged.

1620 *Current column*: Unchanged.

1621 **5.10.7.5.37 visual**

1622 *Synopsis*: [*laddr*] vi[sual] [*type*] [*count*] [*flags*]

1623 If *ex* is currently in open or visual mode, the *Synopsis* and behavior of the  
1624 *visual* command shall be the same as the *edit* command, as specified by  
1625 5.10.7.5.8.

1626 Otherwise, this command need not be supported on block-mode terminals or ter- C  
1627 minals with insufficient capabilities. If standard input, standard output, or stan-  
1628 dard error are not terminal devices, the results are unspecified.

1629 If *count* is specified, the value of the *window edit* option shall be set to *count* (as  
1630 described in 5.10.7.8.29). If the ^ type character was also specified, the *window* C  
1631 *edit* option shall be set before being used by the ^ type character. C

1632 Enter visual mode. If *type* is not specified, it shall be as if a *type* of + was  
1633 specified. The *type* shall cause the following effects:

1634 + Place the beginning of the specified line at the top of the display.

1635 - Place the end of the specified line at the bottom of the display.

1636 . Place the beginning of the specified line in the middle of the display.

1637       <sup>^</sup> If the specified line is less than or equal to the value of the `window edit`  
 1638       option, set the line to 1; otherwise, decrement the line by the value of the  
 1639       `window edit` option minus 1. Place the beginning of this line as close to the  
 1640       bottom of the displayed lines as possible, while still displaying the value of  
 1641       the `window edit` option number of lines.

1642   *Current line*: Set to the specified line.

1643   *Current column*: Set to nonblank.

#### 1644   **5.10.7.5.38 write**

1645   *Synopsis*: [*2addr*] `w[rite][!]` [*>>*] [*file*]

1646   *Synopsis*: [*2addr*] `w[rite]` [*!*] [*file*]

1647   *Synopsis*: [*2addr*] `wq[!]` [*>>*] [*file*]

1648   If no lines are specified, the lines shall default to the entire file.

1649   The command `wq` shall be equivalent to a `write` command followed by a `quit`  
 1650   command; `wq!` shall be equivalent to `write!` followed by `quit`. In both cases, if  
 1651   the `write` fails, the `quit` shall not be attempted.

1652   If the command name is not followed by one or more `<blank>`s, or *file* is not pre- C  
 1653   ceded by a `!` character, the `write` shall be to a file. C

1654       (1) If the `>>` argument is specified, and the file already exists, the lines shall C  
 1655       be appended to the file instead of replacing its contents. If the `>>` argu- C  
 1656       ment is specified, and the file does not already exist, it is unspecified if C  
 1657       the `write` shall proceed as if the `>>` argument had not been specified or if C  
 1658       the `write` shall fail. C

1659       (2) If the `readonly` edit option is set (see 5.10.7.8.13), the `write` shall fail.

1660       (3) If *file* is specified, and is not the current pathname, and the file exists,  
 1661       the `write` shall fail.

1662       (4) If *file* is not specified, the current pathname shall be used. If there is no  
 1663       current pathname, the `write` command shall fail.

1664       (5) If the current pathname is used, and the current pathname has been  
 1665       changed by the `file` or `read` commands, and the file exists, the `write`  
 1666       shall fail. If the `write` is successful, subsequent `writes` shall not fail for  
 1667       this reason (unless the current pathname is changed again). C

1668       (6) If the whole edit buffer is not being written, and the file to be written  
 1669       exists, the `write` shall fail.

1670   For rules (1), (2), (4), and (5), the `write` can be forced by appending the character `!`  
 1671   to the command name.

1672   For rules (2), (4), and (5), the `write` can be forced by setting the `writeln` edit  
 1673   option.

1674   Additional, implementation-defined tests may cause the `write` to fail.

- 1675 If the edit buffer is empty, a file without any contents shall be written.
- 1676 An informational message shall be written noting the number of lines and bytes C  
1677 written.
- 1678 Otherwise, if the command is followed by one or more <blank>s, and *file* is pre-  
1679 ceded by !, the rest of the line after the ! shall have %, #, and ! characters  
1680 expanded as described in 5.10.7.3.
- 1681 The `ex` utility shall then pass two arguments to the program named by the `shell`  
1682 `edit` option; the first shall be “-c” and the second shall be the expanded argu- C  
1683 ments to the `write` command as a single argument. The specified lines shall be C  
1684 written to the standard input of the command. The standard error and standard C  
1685 output of the program, if any, shall be written as described for the `print` com- C  
1686 mand. If the last character in that output is not a <newline> character, a <new-  
1687 line> shall be written at the end of the output.
- 1688 The special meaning of the ! following the `write` command can be overridden by  
1689 escaping it with a backslash character.
- 1690 *Current line:* Unchanged.
- 1691 *Current column:* Unchanged.
- 1692 **5.10.7.5.39 `xit`**
- 1693 *Synopsis:* [*2addr*] `x[it][!]` [*file*]
- 1694 If the edit buffer has not been modified since the last complete write, `xit` shall be  
1695 equivalent to the `quit` command, or if a ! is appended to the command name, to  
1696 `quit!`.
- 1697 Otherwise, `xit` shall be equivalent to the `wq` command, or if a ! is appended to  
1698 the command name, to `wq!`.
- 1699 *Current line:* Unchanged.
- 1700 *Current column:* Unchanged.
- 1701 **5.10.7.5.40 `yank`**
- 1702 *Synopsis:* [*2addr*] `ya[nk]` [*buffer*] [*count*]
- 1703 Copy the specified lines to the specified buffer (by default, the unnamed buffer),  
1704 which shall become a line-mode buffer.
- 1705 *Current line:* Unchanged.
- 1706 *Current column:* Unchanged.

1707 **5.10.7.5.41 z**

1708 *Synopsis:* [*laddr*] z[!][*type ...*] [*count*] [*flags*] C

1709 If no line is specified, the current line shall be the default; if *type* is omitted as  
1710 well, the current line value shall first be incremented by 1. If incrementing the  
1711 current line would cause it to be greater than the last line in the edit buffer, it  
1712 shall be an error.

1713 If there are <blank> characters between the *type* argument and the preceding z C  
1714 command name or optional ! character, it shall be an error. C

1715 If *count* is specified, the value of the `window` edit option shall be set to *count* (as  
1716 described in 5.10.7.8.29). If *count* is omitted, it shall default to 2 times the value  
1717 of the `scroll` edit option, or if ! was specified, the number of lines in the display  
1718 minus 1.

1719 If *type* is omitted, then *count* lines starting with the specified line shall be written.  
1720 Otherwise, *count* lines starting with the line specified by the *type* argument shall  
1721 be written.

1722 The *type* argument shall change the line(s) to be written. The possible values of  
1723 *type* are as follows:

1724 – The specified line shall be decremented by the following value:

1725 
$$(((\text{number of “-” characters}) \times \text{count}) - 1)$$

1726 If the calculation would result in a number less than 1, it shall be an error.  
1727 Write lines from the edit buffer, starting at the new value of line, until C  
1728 *count* lines or the last line in the edit buffer has been written. C

1729 + The specified line shall be incremented by the following value:

1730 
$$(((\text{number of “+” characters}) - 1) \times \text{count}) + 1$$

1731 If the calculation would result in a number greater than the last line in the  
1732 edit buffer, it shall be an error. Write lines from the edit buffer, starting at C  
1733 the new value of line, until *count* lines or the last line in the edit buffer has C  
1734 been written.

1735 =

1736 . If more than a single . or = is specified, it shall be an error. The following  
1737 steps shall be taken:

1738 (1) If *count* is zero, nothing shall be written.

1739 (2) Write as many of the *N* lines before the current line in the edit buffer C  
1740 as exist. If *count* or ! was specified, *N* shall be C

1741 
$$(\text{count} - 1) / 2$$
 C

1742 Otherwise, *N* shall be C

1743 
$$(\text{count} - 3) / 2$$
 C

- 1744 If  $N$  is a number less than 3, no lines shall be written. C
- 1745 (3) If = was specified as the *type* character, write a line consisting of the  
1746 smaller of: the number of columns in the display divided by two, or 40  
1747 “-” characters.
- 1748 (4) Write the current line.
- 1749 (5) Repeat step 3.
- 1750 (6) Write as many of the  $N$  lines after the current line in the edit buffer C  
1751 as exist.  $N$  shall be defined as in step (2). If  $N$  is a number less than C  
1752 3, no lines shall be written. C
- 1753 lines after the current line in the edit buffer as exist. If *count* is less  
1754 than 3, no lines shall be written.
- 1755 ^ The specified line shall be decremented by the following value:
- 1756 
$$(((\text{number of “^” characters}) + 1) \times \text{count}) - 1$$
- 1757 If the calculation would result in a number less than 1, it shall be an error.  
1758 Write lines from the edit buffer, starting at the new value of line, until C  
1759 *count* lines or the last line in the edit buffer has been written. C
- 1760 *Current line*: Set to the last line written, unless the type is =, in which case, set to  
1761 the specified line.
- 1762 *Current column*: Set to nonblank.
- 1763 **5.10.7.5.42 !**
- 1764 *Synopsis*: [*2addr*]! *command*
- 1765 The contents of the line after the ! shall have %, #, and ! characters expanded as  
1766 described in 5.10.7.3. If the expansion causes the text of the line to change, it C  
1767 shall be redisplayed, preceded by a single ! character. C
- 1768 The *ex* utility shall execute the program named by the *shell* edit option. It shall  
1769 pass two arguments to the program; the first shall be “-c”, and the second shall C  
1770 be the expanded arguments to the ! command as a single argument. C
- 1771 If no lines are specified, the standard input, standard output, and standard error  
1772 of the program shall be set to the standard input, standard output, and standard  
1773 error of the *ex* program when it was invoked. In addition, a warning message  
1774 shall be written if the edit buffer has been modified since the last complete write,  
1775 and the *warn* edit option is set.
- 1776 If lines are specified, they shall be passed to the program as standard input, and  
1777 the standard output and standard error of the program shall replace those lines  
1778 in the edit buffer. Each line in the program output (as delimited by <newline>  
1779 characters or the end of the output if it is not immediately preceded by a <new-  
1780 line> character), shall be a separate line in the edit buffer. Any occurrences of  
1781 <carriage-return> and <newline> character pairs in the output shall be  
1782 treated as single <newline> characters. The specified lines shall be copied into  
1783 the unnamed buffer before they are replaced, and the unnamed buffer shall

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1784 become a line-mode buffer.

1785 If in `ex` mode, a single `!` character shall be written when the program completes. C

1786 This command shall be affected by the `shell` and `warn edit` options. If no lines  
 1787 are specified, this command shall be affected by the `autowrite` and `writeany`  
 1788 edit options. If lines are specified, this command shall be affected by the `auto-` C  
 1789 `print` edit option. C

1790 *Current line:*

1791 (1) If no lines are specified, unchanged.

1792 (2) Otherwise, set to the last line read in, if any lines are read in.

1793 (3) Otherwise, set to the line before the first line of the lines specified, if  
 1794 that line exists.

1795 (4) Otherwise, set to the first line of the edit buffer if the edit buffer is not  
 1796 empty.

1797 (5) Otherwise, set to zero.

1798 *Current column:*

1799 If no lines are specified, unchanged.

1800 Otherwise, set to nonblank.

1801 **5.10.7.5.43 <**

1802 *Synopsis:* [*2addr*] <[<... ] [*count*] [*flags*]

1803 Shift the specified lines toward the start of the line; the number of column posi-  
 1804 tions to be shifted shall be the number of command characters times the value of  
 1805 the `shiftwidth` edit option. Only leading `<blank>`s shall be deleted or changed  
 1806 into other `<blank>` characters in shifting; other characters shall not be affected.

1807 Lines to be shifted shall be copied into the unnamed buffer, which shall become a  
 1808 line-mode buffer.

1809 This command shall be affected by the `autoprint` edit option. C

1810 *Current line:* Set to the last line in the lines specified.

1811 *Current column:* Set to nonblank.

1812 **5.10.7.5.44 >**

1813 *Synopsis:* [*2addr*] >[>... ] [*count*] [*flags*]

1814 Shift the specified lines away from the start of the line; the number of column  
 1815 positions to be shifted shall be the number of command characters times the value  
 1816 of the `shiftwidth` edit option. The shift shall be accomplished by adding  
 1817 `<blank>`s as a prefix to the line or changing leading `<blank>` characters into  
 1818 other `<blank>` characters. Empty lines shall not be changed.

1819 Lines to be shifted shall be copied into the unnamed buffer, which shall become a  
1820 line-mode buffer.

1821 This command shall be affected by the `autoprint` edit option. C

1822 *Current line*: Set to the last line in the lines specified.

1823 *Current column*: Set to nonblank.

1824 **5.10.7.5.45** <control-D>

1825 *Synopsis*: <control-D>

1826 Write the next  $n$  lines, where  $n$  is the minimum of the values of the `scroll` edit  
1827 option and the number of lines after the current line in the edit buffer. If the  
1828 current line is the last line of the edit buffer it shall be an error.

1829 *Current line*: Set to the last line written.

1830 *Current column*: Set to nonblank.

1831 **5.10.7.5.46** =

1832 *Synopsis*: [*1addr*] = [*flags*]

1833 If line is not specified, it shall default to the last line in the edit buffer. Write the  
1834 line number of the specified line.

1835 *Current line*: Unchanged.

1836 *Current column*: Unchanged.

1837 **5.10.7.5.47** @

1838 *Synopsis*: [*2addr*] @ [*buffer*]

1839 *Synopsis*: [*2addr*] \* [*buffer*]

1840 If no *buffer* is specified or is specified as @ or \*, the last *buffer* executed shall be  
1841 used. If no previous *buffer* has been executed, it shall be an error.

1842 For each line specified by the addresses, set the current line (.) to the specified  
1843 line, and execute the contents of the named *buffer* (as they were at the time the @  
1844 command was executed) as `ex` commands. For each line of a line-mode buffer,  
1845 and all but the last line of a character-mode buffer, the `ex` command parser shall  
1846 behave as if the line was terminated by a <newline> character.

1847 If an error occurs during this process, or a line specified by the addresses does not  
1848 exist when the current line would be set to it, or more than a single line was C  
1849 specified by the addresses, and the contents of the edit buffer are replaced (e.g., by C  
1850 the `ex :edit` command) an error message shall be written, and no more com-  
1851 mands resulting from the execution of this command shall be processed.

1852 *Current line*: As specified for the individual `ex` commands.

1853 *Current column*: As specified for the individual `ex` commands.

### 1854 **5.10.7.6 REs**

1855 The `ex` utility shall support REs that are a superset of the BREs described in  
1856 2.8.3. A null RE (`//` or `??`) shall be equivalent to the last RE encountered.

1857 REs can be used in addresses to specify lines and, in some commands (for exam-  
1858 ple, the `s` command), to specify portions of a line to be substituted.

1859 The following constructs can be used to enhance the BREs:

1860 `\<` Match the beginning of a word. (See the definition of word at the begin-  
1861 ning of 5.10.7.4.)

1862 `\>` Match the end of a word.

1863 `~` Match the replacement part of the last `s` command. The tilde (`~`) char-  
1864 acter can be escaped with a backslash in an RE to become a normal  
1865 character with no special meaning. The backslash shall be discarded.

1866 When the `magic` edit option is not set, the only characters with special meanings  
1867 shall be `^` at the beginning of a pattern, `$` at the end of a pattern, and backslash.  
1868 The characters `.`, `*`, `[`, and `~` shall be treated as ordinary characters unless pre-  
1869 ceded by a backslash; when preceded by a backslash they shall regain their spe-  
1870 cial meaning, or in the case of backslash, be handled as a single backslash.  
1871 Backslashes used to escape other characters shall be discarded.

### 1872 **5.10.7.7 Replacement Strings**

1873 The character `&` (`\&` if the `magic` edit option is not set) in the replacement string  
1874 shall stand for the text matched by the pattern to be replaced. The character `~`  
1875 (`\~` if the `magic` edit option is not set) shall be replaced by the replacement part  
1876 of the previous `s` command. The sequence `\n`, where `n` is an integer, shall be  
1877 replaced by the text matched by the pattern enclosed in the `n`th set of parentheses  
1878 `\(` and `\)`.

1879 The strings `\l`, `\u`, `\L`, and `\U` can be used to modify the case of elements in the  
1880 replacement string. The string `\l` (`\u`) shall cause the character that follows to  
1881 be converted to lowercase (uppercase). The string `\L` (`\U`) shall cause all charac-  
1882 ters subsequent to it to be converted to lowercase (uppercase) until the string `\e`  
1883 or `\E`, or the end of the replacement string, is encountered.

1884 Otherwise, any character following a backslash shall be treated as that literal  
1885 character, and the escaping backslash shall be discarded.

### 1886 **5.10.7.8 Edit Options**

1887 The `ex` utility has a number of options that modify its behavior. These options  
1888 have default settings, which can be changed using the `set` command.

1889 Options are Boolean unless otherwise specified.



1890 **5.10.7.8.1 autoindent, ai**1891 [Default: *unset*]

1892 If `autoindent` is set, each line in text input mode shall be indented (first using  
 1893 as many `<tab>`s as possible, as determined by the `tabstop` edit option, and then  
 1894 using `<space>`s) to align with another line, as follows:

- 1895 (1) If in open or visual mode and the text input is part of a line-oriented com-  
 1896 mand (see 5.35.7), align to the first column.
- 1897 (2) Otherwise, if in open or visual mode, indentation for each line shall be C  
 1898 set as follows: C
- 1899 (a) If a line was previously inserted as part of this command, it shall be C  
 1900 set to the indentation of the last inserted line by default, or as oth- C  
 1901 erwise specified for the `<control-D>` character in 5.35.7.3.2. C
- 1902 (b) Otherwise, it shall be set to the indentation of the previous current C  
 1903 line, if any; otherwise, to the first column. C
- 1904 (3) For the `ex a`, `i`, and `c` commands, indentation for each line shall be set as C  
 1905 follows: C
- 1906 (a) If a line was previously inserted as part of this command, it shall be C  
 1907 set to the indentation of the last inserted line by default, or as oth- C  
 1908 erwise specified for the `eof` character in 5.10.7.4.1. C
- 1909 (b) Otherwise, if the command is the `ex a` command, it shall be set to C  
 1910 the line appended after, if any; otherwise to the first column. C
- 1911 (c) Otherwise, if the command is the `ex i` command, it shall be set to C  
 1912 the line inserted before, if any; otherwise to the first column. C
- 1913 (d) Otherwise, if the command is the `ex c` command, it shall be set to C  
 1914 the indentation of the line replaced. C

1915 **5.10.7.8.2 autoprint, ap**1916 [Default: *set*]

1917 If `autoprint` is set, the current line shall be written after each `ex` command that  
 1918 modifies the contents of the current edit buffer, and after each `tag` command for  
 1919 which the tag search pattern was found or tag line number was valid, unless:

- 1920 (1) The command was executed while in open or visual mode.
- 1921 (2) The command was executed as part of a `global` or `v` command or `@`  
 1922 buffer execution.
- 1923 (3) The command was the form of the `read` command that reads a file into  
 1924 the edit buffer.
- 1925 (4) The command was the `append`, `change`, or `insert` command.
- 1926 (5) C

- 1927       (6) The command was not terminated by a <newline> character. C
- 1928       (7) The current line shall be written by a flag specified to the command; e.g., C  
 1929       “delete #” shall write the current line as specified for the flag modifier C  
 1930       to the delete command, and not as specified by the autoprint edit C  
 1931       option. C
- 1932   **5.10.7.8.3 autowrite, aw**
- 1933   [Default: *unset*]
- 1934   If *autowrite* is set, and the edit buffer has been modified since it was last com- C  
 1935   pletely written to any file, the contents of the edit buffer shall be written as if the C  
 1936   *ex write* command had been specified without arguments, before each command C  
 1937   affected by the *autowrite* edit option is executed. Appending the character ! to  
 1938   the command name of any of the *ex* commands except ! shall prevent the write.  
 1939   If the write fails, it shall be an error and the command shall not be executed.
- 1940   **5.10.7.8.4 errorbells, eb**
- 1941   [Default: *unset*]
- 1942   If the editor is in *ex* mode, and the terminal does not support a standout mode  
 1943   (such as inverse video), and *errorbells* is set, error messages shall be preceded  
 1944   by alerting the terminal.
- 1945   **5.10.7.8.5 exrc, ex**
- 1946   [Default: *unset*]
- 1947   If *exrc* is set, *ex* shall access any *.exrc* file in the current directory, as described  
 1948   in 5.10.7.1. If *exrc* is not set, *ex* shall ignore any *.exrc* file in the current direc-  
 1949   tory during initialization, unless the current directory is named by the **HOME**  
 1950   environment variable.
- 1951   **5.10.7.8.6 ignorecase, ic**
- 1952   [Default: *unset*]
- 1953   If *ignorecase* is set, characters that have uppercase and lowercase representa-  
 1954   tions shall have those representations considered as equivalent for use in REs.
- 1955   The *ignorecase* edit option shall affect all remembered REs; e.g., unsetting the  
 1956   *ignorecase* edit option shall cause a subsequent *vi n* command to search for the  
 1957   last BRE in a case-sensitive fashion.
- 1958   **5.10.7.8.7 list**
- 1959   [Default: *unset*]
- 1960   If *list* is set, edit buffer lines written while in *ex* command mode shall be writ-  
 1961   ten as specified for the *print* command with the *l* flag specified. C

1962 In open or visual mode, each edit buffer line shall be displayed as specified for the  
 1963 `ex print` command with the `l` flag specified. In open or visual text input mode, C  
 1964 when the cursor does not rest on any character in the line, it shall rest on the `$` C  
 1965 marking the end of the line. C

#### 1966 **5.10.7.8.8** `magic`

1967 [Default: *set*]

1968 If `magic` is set, modify the interpretation of characters in REs and substitution  
 1969 replacement strings as described in 5.10.7.6 and 5.10.7.7.

#### 1970 **5.10.7.8.9** `mesg`

1971 [Default: *set*]

1972 If `mesg` is set, the permission for others to use the `write` or `talk` commands to  
 1973 write to the terminal shall be set while in open or visual mode. The shell-level  
 1974 command `mesg n` (see 5.17) shall take precedence over any setting of the `mesg`  
 1975 edit option; i.e., if `mesg y` was issued before the editor started (or in a shell  
 1976 escape, such as `!:mesg y`), the `mesg` edit option in the editor shall suppress  
 1977 incoming messages, but the `mesg edit` option shall not enable incoming mes-  
 1978 sages if `mesg n` was issued.

#### 1979 **5.10.7.8.10** `number`, `nu`

1980 [Default: *unset*]

1981 If `number` is set, edit buffer lines written while in `ex` command mode shall be  
 1982 written with line numbers, in the format specified by the `print` command with C  
 1983 the `#` flag specified. In `ex` text input mode, each line shall be preceded by the line  
 1984 number it will have in the file.

1985 In open or visual mode, each edit buffer line shall be displayed with a preceding  
 1986 line number, in the format specified by the `ex print` command with the `#` flag C  
 1987 specified. This line number shall not be considered part of the line for the pur-  
 1988 poses of evaluating the current column; i.e., column position 1 shall be the first  
 1989 column position after the format specified by the `print` command. C

#### 1990 **5.10.7.8.11** `paragraphs`, `para`

1991 [Default in the POSIX Locale: `IPLPPPQPP LIpplpipbp`]

1992 The `paragraphs` edit option shall define additional paragraph boundaries for  
 1993 open and visual mode commands. The `paragraphs` edit option can be set to a  
 1994 character string consisting of zero or more character pairs; it shall be an error to  
 1995 set it to an odd number of characters.

- 1996 **5.10.7.8.12 prompt**
- 1997 [Default: *set*]
- 1998 If `prompt` is set, `ex` command mode input shall be prompted for with a colon (:)  
1999 character; when unset, no prompt shall be written.
- 2000 **5.10.7.8.13 readonly, ro**
- 2001 [Default: *see text*]
- 2002 If the `readonly` edit option is set, read-only mode shall be enabled (see C  
2003 5.10.7.5.38). The `readonly` edit option shall be initialized to set if either of the C  
2004 following conditions are true: C
- 2005 — The command-line option `-R` was specified. C
  - 2006 — Performing actions equivalent to the POSIX.1 {8} `access()` function, called C  
2007 with the following arguments indicates that the file lacks write permission: C
- 2008 (1) The current pathname is used as the *path* argument. C
  - 2009 (2) The constant `W_OK` is used as the *amode* argument. C
- 2010 The `readonly` edit option may be initialized to set for other, implementation- C  
2011 defined reasons. The `readonly` edit option shall not be initialized to unset based C  
2012 on any special privileges of the user or process. C
- 2013 The `readonly` edit option shall be reinitialized each time that the contents of the C  
2014 edit buffer are replaced (e.g., by an `edit` or `next` command) unless the user has C  
2015 explicitly set it, in which case it shall remain set until the user explicitly unsets C  
2016 it. Once unset, it shall again be reinitialized each time that the contents of the C  
2017 edit buffer are replaced. C
- 2018 **5.10.7.8.14 remap**
- 2019 [Default: *set*]
- 2020 If `remap` is set, map translation shall allow for maps defined in terms of other  
2021 maps; translation shall continue until a final product is obtained. If unset, only a  
2022 one-step translation shall be done.
- 2023 **5.10.7.8.15 report**
- 2024 [Default: 5]
- 2025 The value of the `report` edit option specifies what number of lines being added, C  
2026 copied, deleted or modified in the edit buffer will cause an informational message C  
2027 to be written to the user. The following conditions shall cause an informational C  
2028 message. The message shall contain the number of lines added, copied, deleted, C  
2029 or modified, but is otherwise unspecified. C
- 2030 — An `ex` or `vi` editor command, other than `open`, `undo`, or `visual`, that C  
2031 modifies at least the value of the `report` edit option number of lines, and C  
2032 which is not part of an `ex global` or `v` command, or `ex` or `vi` buffer C

- 2033 execution, shall cause an informational message to be written. C
- 2034 — An `ex yank` or `vi y` or `Y` command, that copies at least the value of the C  
 2035 report edit option plus 1 number of lines, and which is not part of an `ex` C  
 2036 `global` or `v` command, or `ex` or `vi` buffer execution, shall cause an infor- C  
 2037 mational message to be written. C
- 2038 — An `ex global`, `v, open`, `undo`, or `visual` command or `ex` or `vi` buffer exe- C  
 2039 cution, that adds or deletes a total of at least the value of the report edit C  
 2040 option number of lines, and which is not part of an `ex global` or `v` com- C  
 2041 mand, or `ex` or `vi` buffer execution, shall cause an informational message C  
 2042 to be written. (For example, if 3 lines were added and 8 lines deleted dur- C  
 2043 ing an `ex visual` command, 5 would be the number compared against the C  
 2044 report edit option after the command completed. C
- 2045 **5.10.7.8.16 scroll, scr**
- 2046 [Default: (number of lines in the display – 1) / 2]
- 2047 The value of the `scroll` edit option shall affect the number of lines scrolled by C  
 2048 the `ex <control-D>` and `z` commands. For the `vi <control-D>` and C  
 2049 `<control-U>` commands, it shall be the initial number of lines to scroll when no  
 2050 previous `<control-D>` or `<control-U>` command has been executed.
- 2051 **5.10.7.8.17 sections, sect**
- 2052 [Default in the POSIX Locale: `NHSHH HUnhsh`]
- 2053 The `sections` edit option shall define additional section boundaries for open and  
 2054 visual mode commands. The `sections` edit option can be set to a character  
 2055 string consisting of zero or more character pairs; it shall be an error to set it to an  
 2056 odd number of characters.
- 2057 **5.10.7.8.18 shell, sh**
- 2058 [Default: from the environment variable `SHELL`]
- 2059 The value of this edit option shall be a string. The default shall be taken from the  
 2060 `SHELL` environment variable. If the `SHELL` environment variable is null or  
 2061 empty, the `sh` (see 4.56) utility shall be the default.
- 2062 **5.10.7.8.19 shiftwidth, sw**
- 2063 [Default: 8]
- 2064 The value of this edit option shall give the width in columns of an indentation  
 2065 level used during autoindentation and by the `ex` and `vi < and >` commands. C

- 2066 **5.10.7.8.20 showmatch, sm**
- 2067 [Default: *unset*]
- 2068 The functionality described for the `showmatch` edit option need not be supported  
2069 on block-mode terminals or terminals with insufficient capabilities.
- 2070 If the `showmatch` option is set, in open and visual text input modes, when a `)` or  
2071 `}` is typed, if the matching `(` or `{` is currently visible on the display, the matching  
2072 `(` or `{` shall be flagged by moving the cursor to its location for an unspecified  
2073 amount of time.
- 2074 **5.10.7.8.21 showmode, smd** C
- 2075 [Default: *unset*]
- 2076 If `showmode` is set in open or visual mode, the current mode of the editor shall be C  
2077 displayed on the last line of the display. Command mode and text input mode C  
2078 shall be differentiated; other unspecified modes and implementation-defined C  
2079 information may be displayed.
- 2080 **5.10.7.8.22 slowopen**
- 2081 [Default: *unset*]
- 2082 If `slowopen` is set during open and visual text input modes, the editor shall not  
2083 update portions of the display other than those screen columns that display the  
2084 characters entered by the user (see 5.35.7.3).
- 2085 **5.10.7.8.23 tabstop, ts**
- 2086 [Default: 8]
- 2087 The value of this edit option shall specify the column boundary used by a `<tab>` C  
2088 character in the display (see 5.10.7.8.2 and 5.35.7.2). C
- 2089 **5.10.7.8.24 taglength, tl**
- 2090 [Default: zero]
- 2091 The value of this edit option shall specify the maximum number of characters that  
2092 are considered significant in the user-specified tag name and in the tag name  
2093 from the tags file. If the value is zero, all characters in both tag names shall be  
2094 significant.
- 2095 **5.10.7.8.25 tag, tags**
- 2096 [Default: *unspecified*]
- 2097 The value of this edit option shall be a string of `<blank>`-delimited pathnames of  
2098 files used by the `tag` command. The default value is unspecified.

- 2099 **5.10.7.8.26 term**
- 2100 [Default: from the environment variable **TERM**]
- 2101 The value of this edit option shall be a string. The default shall be taken from the  
2102 **TERM** environment variable. If the **TERM** environment variable is empty or null,  
2103 the default is unspecified. The editor shall use the value of this edit option to  
2104 determine the type of the display device.
- 2105 The results are unspecified if the user changes the value of the `term` edit option  
2106 after editor initialization.
- 2107 **5.10.7.8.27 terse**
- 2108 [Default: *unset*]
- 2109 If `terse` is set, error messages may be less verbose. However, except for this  
2110 caveat, error messages are unspecified.
- 2111 **5.10.7.8.28 warn**
- 2112 [Default: *set*]
- 2113 If `warn` is set, and the contents of the edit buffer have been modified since they  
2114 were last completely written, the editor shall write a warning message before cer-  
2115 tain ! commands (see 5.10.7.5.42).
- 2116 **5.10.7.8.29 window, wi**
- 2117 [Default: *see text*]
- 2118 A value used in open and visual mode, by the <control-B> and <control-F>  
2119 commands, and, in visual mode, to specify the number of lines displayed when the  
2120 screen is repainted.
- 2121 If the `-w` command-line option is not specified, the default value shall be set to the  
2122 value of the **LINES** environment variable. If the **LINES** environment variable is  
2123 empty or null, the default shall be the number of lines in the display minus 1.
- 2124 Setting the `window` edit option to zero or to a value greater than the number of  
2125 lines in the display minus 1 (either explicitly or based on the `-w` option or the  
2126 **LINES** environment variable) shall cause the `window` edit option to be set to the  
2127 number of lines in the display minus 1.
- 2128 The baud rate of the terminal line may change the default in an implementation-  
2129 defined manner.
- 2130 **5.10.7.8.30 wrapmargin, wm**
- 2131 [Default: zero]
- 2132 If the value of this edit option is zero, it shall have no effect.
- 2133 If not in the POSIX Locale, the effect of this edit option is implementation-defined.

2134 Otherwise, it shall specify a number of columns from the ending margin of the  
2135 terminal.

2136 During open and visual text input modes, for each character for which any part of  
2137 the character is displayed in a column that is less than `wrapmargin` columns  
2138 from the ending margin of the screen, the editor shall behave as follows:

2139 (1) If the character triggering this event is a `<blank>`, it, and all immedi-  
2140 ately preceding `<blank>` characters on the current line entered during  
2141 the execution of the current text input command shall be discarded, and  
2142 the editor shall behave as if the user had entered a single `<newline>`  
2143 character instead. In addition, if the next user-entered character is a  
2144 `<space>`, it shall be discarded as well.

2145 (2) Otherwise, if there are one or more `<blank>` characters on the current  
2146 line immediately preceding the last group of inserted non-`<blank>` char- C  
2147 acters which was entered during the execution of the current text input  
2148 command, the `<blank>` characters shall be replaced as if the user had  
2149 entered a single `<newline>` character instead.

2150 If the `autoindent` edit option is set, and the events described in (1) or (2) are  
2151 performed, any `<blank>` characters at or after the cursor in the current line shall  
2152 be discarded.

2153 The ending margin shall be determined by the system or overridden by the user,  
2154 as described for **COLUMNS** in 5.10.5.3 and 2.6.

#### 2155 **5.10.7.8.31** `wrapsan`, `ws`

2156 [Default: *set*]

2157 If `wrapsan` is set, searches (the `ex /` and `?` addresses, or open and visual mode C  
2158 `/`, `?`, `N`, and `n` commands) shall wrap around the beginning or end of the edit  
2159 buffer; when unset, searches shall stop at the beginning or end of the edit buffer.

#### 2160 **5.10.7.8.32** `writeany`, `wa`

2161 [Default: *unset*]

2162 If `writeany` is set, some of the checks performed when executing the `ex write`  
2163 commands shall be inhibited, as described in 5.10.7.5.38.

### 2164 **5.10.8 Exit Status**

2165 The `ex` utility shall exit with one of the following values:

2166 0 Successful completion.

2167 >0 An error occurred.



2168 **5.10.9 Consequences of Errors**

2169 When any error is encountered and the standard input is not a terminal device C  
 2170 file, `ex` shall not write the file or return to command or text input mode, and shall C  
 2171 terminate with a nonzero exit status. C

2172 Otherwise, when an unrecoverable error is encountered it shall be equivalent to a C  
 2173 `SIGHUP` asynchronous event. C

2174 Otherwise, when an error is encountered, the editor shall behave as specified in C  
 2175 5.10.7.3. C

2176 **5.11 expand – Convert tabs to spaces**

2177 ⇒ **5.11.5.3 expand Environment Variables.** *In the description of `LC_CTYPE`,*  
 2178 *change the phrase "... width in column positions each character would occupy*  
 2179 *on a constant-width-font output device" to:*

2180 ... width in column positions each character would occupy on an output  
 2181 device.

2182 **Rationale:** This change partially satisfies the following corrigendum request  
 2183 from ISO/IEC 9945-2: 1993 Annex H.2:

2184 (15) In 5.11.5.3 and 5.32.5.3, in the last sentence of the `LC_CTYPE` paragraph  
 2185 for `expand` and `unexpand`, the phrase "on a constant-width-font output  
 2186 device" may be redundant because of definitions elsewhere in the  
 2187 standard.

2188 **5.14 file – Determine file type**

2189 **Rationale:** The changes in this clause, except for those related to symbolic links,  
2190 satisfy the following requirement from ISO/IEC 9945-2: 1993 Annex H.1:

2191 (12) The `file` utility should allow user-specified algorithms for file type  
2192 recognition, similar to those used in the historical `/etc/magic` file.

2193 ⇒ **5.14.1 file Synopsis.** *Modify the Synopsis to be:*

2194 `file [-dhi] [-M file] [-m file] file...`

2195 ⇒ **5.14.2 file Description.** *Add a new paragraph at the end of the subclause:*

2196 If `file` is a symbolic link, by default the link shall be resolved and `file` shall  
2197 test the type of file referenced by the symbolic link.

2198 ⇒ **5.14.3 file Options.** *Replace the entire Options subclause with:*

2199 The `file` utility shall conform to the utility argument syntax guidelines  
2200 described in 2.10.2.

2201 The following options shall be supported by the implementation:

2202 `-d` Apply any default system tests to the file.

2203 `-h` When a symbolic link is encountered, identify the file as a  
2204 symbolic link. If `-h` is not specified and `file` is a symbolic link  
2205 that refers to a nonexistent file, `file` shall identify the file as  
2206 a symbolic link, as if `-h` had been specified.

2207 `-i` If a file is a regular file, do not attempt to classify the type of  
2208 the file further, but identify the file as specified in 5.14.6.1,  
2209 using a `<type>` string that contains the string `regular file`.

2210 `-M file` Specify the name of a file containing tests that shall be applied  
2211 to a file in order to classify it (see 5.14.7). No default system  
2212 tests shall be applied.

2213 `-m file` Specify the name of a file containing tests that shall be applied  
2214 to a file in order to classify it (see 5.14.7).

2215 If multiple instances of the `-m`, `-d`, or `-M` options are specified, the concatena-  
2216 tion of the tests specified, in the order specified, shall be the set of tests that  
2217 are applied. If a `-M` option is specified, no tests other than those specified  
2218 using the `-d`, `-M`, and `-m` options shall be applied to the file. If neither the `-d`  
2219 nor `-M` options are specified, any default system tests shall be applied after any  
2220 tests specified using the `-m` option.

2221 ⇒ **5.14.6.1 file Standard Output.** *Insert a new paragraph between the second*  
 2222 *and third (the one beginning “If the file named ...”) paragraphs:*

2223 If *file* is identified as a symbolic link (see `-h`), the following alternative output  
 2224 format shall be used:

2225 `"%s: %s %s\n", <file>, <type>, <contents of link>`

2226 ⇒ **5.14.6.1 file Standard Output.** *Change the third paragraph (the one*  
 2227 *beginning with “If the file named ...”) to:*

2228 If the file named by the *file* operand does not exist or cannot be read, the string  
 2229 cannot open shall be included as part of the *<type>* field, but this shall not  
 2230 be considered an error that affects the exit status. If the type of the file named  
 2231 by the *file* operand cannot be determined, the string `data` shall be included as  
 2232 part of the *<type>* field, but this shall not be considered an error that affects  
 2233 the exit status.

C

2234

B

2235 ⇒ **5.14.6.1 file Standard Output.** *Add the following entry to the table named*  
 2236 *file Output Strings:*

2237           If *file* is a    <type> shall contain the string

2238 symbolic link    symbolic link to

2239 ⇒ **5.14.7 file Extended Description.** *Change the entire subclause to:*

2240 A file specified as an option-argument to the `-m` or `-M` options shall contain one  
 2241 test per line, which shall be applied to the file. If the test succeeds, the *mes-*  
 2242 *sage* field of the line shall be printed and no further tests shall be applied, with  
 2243 the exception that tests on immediately following lines beginning with a single  
 2244 `>` character shall be applied.

2245 Each line shall be composed of the following four `<blank>`-separated fields:

2246     *offset* An unsigned number (optionally preceded by a single `>` character)  
 2247           specifying the offset, in bytes, of the value in the file that is to be  
 2248           compared against the value field of the line. If the file is shorter  
 2249           than the specified offset, the test shall fail.

2250 If the offset begins with the character `>`, the test contained in the  
 2251 line shall not be applied to the file unless the test on the last line for  
 2252 which the offset did not begin with a `>` was successful. By default,  
 2253 the offset shall be interpreted as an unsigned decimal number.  
 2254 With a leading `0x` or `0X`, the offset shall be interpreted as a hexade-  
 2255 cimal number; otherwise, with a leading `0`, the offset shall be inter-  
 2256 preted as an octal number.

2257           *type*   The type of the value in the file to be tested. The type shall consist  
 2258                   of the type specification characters *c*, *d*, *f*, *s*, and *u*, specifying char-  
 2259                   acter, signed decimal, floating point, string, and unsigned decimal,  
 2260                   respectively.

2261                   The type string shall be interpreted as the bytes from the file start-  
 2262                   ing at the specified offset and including the same number of bytes  
 2263                   specified by the *value* field. If insufficient bytes remain in the file  
 2264                   past the offset to match the value field, the test shall fail.

2265                   The type specification characters *d*, *f*, and *u* can be followed by an  
 2266                   optional unsigned decimal integer that specifies the number of bytes  
 2267                   represented by the type. The type specification character *f* can be  
 2268                   followed by an optional *F*, *D*, or *L*, indicating that the value is of  
 2269                   type *float*, *double*, or *long double*, respectively. The type  
 2270                   specification characters *d* and *u* can be followed by an optional *C*, *S*,  
 2271                   *I*, or *L*, indicating that the value is of type *char*, *short*, *int*, or *long*,  
 2272                   respectively.

2273                   The default number of bytes represented by the type specifiers *d*, *f*,  
 2274                   and *u* shall correspond to their respective C-language types as fol-  
 2275                   lows. If the system claims conformance to the C-Language Develop-  
 2276                   ment Utilities Option, those specifiers shall correspond to the  
 2277                   default sizes used in the *c89* utility. Otherwise, the default sizes  
 2278                   shall be implementation defined.

2279                   For the type specifier characters *d* and *u*, the default number of  
 2280                   bytes shall correspond to the size of the basic integral data type of  
 2281                   the implementation. For these specifier characters, the implemen-  
 2282                   tation shall support values of the optional number of bytes to be  
 2283                   converted corresponding to the number of bytes in the C-language  
 2284                   types *char*, *short*, *int*, or *long*. These numbers can also be specified  
 2285                   by an application as the characters *C*, *S*, *I*, and *L*, respectively. The  
 2286                   byte order used when interpreting numeric values is implementa-  
 2287                   tion defined, but shall correspond to the order in which a constant  
 2288                   of the corresponding type is stored in memory on the system.

2289                   For the type specifier *f*, the default number of bytes shall  
 2290                   correspond to the number of bytes in the basic double precision  
 2291                   floating-point data type of the underlying implementation. The  
 2292                   implementation shall support values of the optional number of  
 2293                   bytes to be converted corresponding to the number of bytes in the  
 2294                   C-language types *float*, *double*, and *long double*. These numbers can  
 2295                   also be specified by an application as the characters *F*, *D*, and *L*,  
 2296                   respectively.

2297                   All type specifiers, except for *s*, can be followed by a mask specifier  
 2298                   of the form *&number*. The mask value shall be ANDed with the  
 2299                   value before the comparison with the value from the file is made.  
 2300                   By default, the mask shall be interpreted as an unsigned decimal  
 2301                   number. With a leading *0x* or *0X*, the mask shall be interpreted as

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 2302 a unsigned hexadecimal number; otherwise, with a leading 0, the  
2303 mask shall be interpreted as an unsigned octal number.
- 2304 The strings `byte`, `short`, `long`, and `string` shall also be sup-  
2305 ported as type fields, being interpreted as `dC`, `dS`, `dL`, and `s`,  
2306 respectively.
- 2307 *value* The value to be compared with the value from the file.
- 2308 Any value that contains a character that is not a digit, other than a  
2309 leading sign (+ or -) or a leading `0x` or `0X`, shall be interpreted as a  
2310 string. The test shall succeed only when a string value exactly  
2311 matches the bytes from the file.
- 2312 If the value is a string, it can contain the following sequences:
- 2313 *\character*  
2314 The backslash-escape sequences in Table 2-16 (see 2.12). The  
2315 results of using any other character, other than an octal digit,  
2316 following the backslash are unspecified.
- 2317 *\octal*  
2318 Octal sequences that can be used to represent characters  
2319 with specific coded values. An octal sequence shall consist of  
2320 a backslash followed by the longest sequence of one, two, or  
2321 three octal-digit characters (01234567). If the size of a byte  
2322 on the system is greater than 9 b, the valid escape sequence  
2323 used to represent a byte is implementation defined.
- 2324 By default, any value that is not a string shall be interpreted as a  
2325 signed decimal number. Any such value, with a leading `0x` or `0X`,  
2326 shall be interpreted as an unsigned hexadecimal number; other-  
2327 wise, with a leading zero, the value shall be interpreted as an  
2328 unsigned octal number.
- 2329 If the value is not a string, it can be preceded by a character indi-  
2330 cating the comparison to be performed. Permissible characters and  
2331 the comparisons they specify are as follows:
- 2332 = The test shall succeed if the value from the file equals the  
2333 value field.
- 2334 < The test shall succeed if the value from the file is less than  
2335 the value field.
- 2336 > The test shall succeed if the value from the file is greater  
2337 than the value field.
- 2338 & The test shall succeed if all of the bits in the value field are  
2339 set in the value from the file.
- 2340 ^ The test shall succeed if at least one of the bits in the value  
2341 field is not set in the value from the file.

2342 x The test shall succeed if there is any value in the file.

2343 *message*

2344 The message to be printed if the test succeeds. The message shall  
 2345 be interpreted using the notation for the `printf` formatting  
 2346 specification; see 4.50.7. If the *value* field was a string, the the  
 2347 value from the file shall be the argument for the `printf` formatting  
 2348 specification; otherwise, the value from the file shall be the argu-  
 2349 ment.

B  
B

2350 *Editor's Note: The rationale in E.5.14 (IEEE Std 1003.2-1992 pages 987-88, lines*  
 2351 *9703-49) will be replaced by the following:*

2352 **file Rationale.** (*This subclause is not a part of P1003.2b*)

2353 Historical systems have used a “magic file” named `/etc/magic` to help identify  
 2354 file types. Because it is generally useful for users and scripts to be able to identify  
 2355 special file types, the `-m` flag and a portable format for user-created magic files  
 2356 has been specified. No requirement is made that an implementation of `file` use  
 2357 this method of identifying files, only that users be permitted to add their own  
 2358 classifying tests.

2359 In addition, three options have been added to historical practice. The `-d` flag has  
 2360 been added to permit users to cause their tests to follow any default system tests.  
 2361 The `-i` flag has been added to permit users to test portably for regular files in  
 2362 shell scripts. The `-M` flag has been added to permit users to ignore any default  
 2363 system tests.

2364 The historical `-c` option was omitted as not particularly useful to users or port-  
 2365 able shell scripts. In addition, a reasonable implementation of the `file` utility  
 2366 would report any errors found each time the magic file is read.

2367 The historical format of the magic file was the same as that specified by the  
 2368 rationale in the previous version of this standard for the *offset*, *value*, and *mes-*  
 2369 *sage* fields; however, it used less precise *type* fields than the format specified by  
 2370 the current normative text. The new *type* field values are a superset of the histor-  
 2371 ical ones.

2372 The following is an example magic file:

2373	0	short	070707	cpio archive
2374	0	short	0143561	byte-swapped cpio archive
2375	0	string	070707	ASCII cpio archive
2376	0	long	0177555	very old archive
2377	0	short	0177545	old archive
2378	0	short	017437	old packed data
2379	0	string	\037\036	packed data
2380	0	string	\377\037	compacted data
2381	0	string	\037\235	compressed data
2382	>2	byte&0x80	>0	block compressed
2383	>2	byte&0x1f	x	%d bits
2384	0	string	\032\001	Compiled Terminfo Entry
2385	0	short	0433	Curses screen image

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2386	0	short	0434	Curses screen image
2387	0	string	<ar>	System V Release 1 archive
2388	0	string	!<arch>\n___.SYMDEF	archive random library
2389	0	string	!<arch>	archive
2390	0	string	ARF_BEGARF	PHIGS clear text archive
2391	0	long	0x137A2950	scalable OpenFont binary
2392	0	long	0x137A2951	encrypted scalable OpenFont binary

2393 ⇒ **5.18 more – Display files on a page-by-page basis.** *Replace the entire* B  
 2394 *more clause with the following.* B

2395 *Editor’s Note: All of this clause has been changed in Draft 11 from the POSIX.2-* B  
 2396 *1992 version. The rationale in Annex E is also completely replaced. Only the por-* B  
 2397 *tions changed from Draft 10 and the 1992 standard are diffmarked.* B

2398 **Rationale:** The changes to this clause are the result of interpretation requests B  
 2399 PASC 1003.2-92 #37 and 109, submitted for IEEE Std 1003.2-1992, and the follow- B  
 2400 ing requirement from ISO/IEC 9945-2: 1993 Annex H.1: B

2401 (27) The `more` utility should be able to handle underlined and emboldened B  
 2402 displays of characters that are wider than a single column position. B

## 2403 **5.18 more – Display files on a page-by-page basis**

### 2404 **5.18.1 Synopsis**

2405 `more [-ceisu] [-n number] [-t tagstring] [-p command] [file ...]`

2406 *Obsolescent Version:*

2407 `more [-ceisu] [-n number] [+command] [-t tagstring] [file ...]`

### 2408 **5.18.2 Description**

2409 The `more` utility shall read files and either write them to the terminal on a page-  
 2410 by-page basis or filter them to standard output. If standard output is not a termi-  
 2411 nal device, all input files shall be copied to standard output in their entirety,  
 2412 without modification, except as specified for the `-s` option. If standard output is a B  
 2413 terminal device, the files shall be written a number of lines (one “screenful”) at a  
 2414 time under the control of user commands; see 5.18.7.

2415 Certain block-mode terminals do not have all the capabilities necessary to support  
 2416 the complete `more` definition; they are incapable of accepting commands that are  
 2417 not terminated with a `<newline>`. Implementations that support such terminals  
 2418 shall provide an operating mode to `more` in which all commands can be ter-  
 2419 minated with a `<newline>` on those terminals. This mode shall

- 2420 — Be documented in the system documentation
- 2421 — At invocation, inform the user of the terminal deficiency that requires the
- 2422 <newline> usage and provide instructions on how this warning can be
- 2423 suppressed in future invocations
- 2424 — Not be required for implementations supporting only fully capable
- 2425 terminals
- 2426 — Not affect commands already requiring <newline>s
- 2427 — Not affect users on the capable terminals from using `more` as described in
- 2428 this standard

### 2429 5.18.3 Options

- 2430 The `more` utility shall conform to the utility argument syntax guidelines  
 2431 described in 2.10.2, except that *+command* of the obsolescent version uses a non-  
 2432 standard syntax, and that the order of presentation of the `-p` and `-t` options is  
 2433 significant. B
- 2434 The following options shall be supported by the implementation:
- 2435 `-c` If a screen is to be written that has no lines in common with the  
 2436 current screen, or `more` is writing its first screen, do not scroll the  
 2437 screen, but instead redraw each line of the screen in turn, from  
 2438 the top of the screen to the bottom. In addition, if `more` is writing  
 2439 its first screen, clear the screen. This option may be silently B  
 2440 ignored on devices with insufficient terminal capabilities. B
  - 2441 `-e` By default, `more` shall exit immediately after writing the last line B  
 2442 of the last file in the argument list. If the `-e` option is specified: B
    - 2443 (1) If there is only a single file in the argument list and that file B  
 2444 was completely displayed on a single screen, `more` shall exit B  
 2445 immediately after writing the last line of that file. B
    - 2446 (2) Otherwise, `more` shall exit only after reaching end-of-file on B  
 2447 the last file in the argument list twice without an interven- B  
 2448 ing operation; see 5.18.7. B
  - 2449 `-i` Perform pattern matching in searches without regard to case.  
 2450 See 2.8.2.
  - 2451 `-n number` Specify the number of lines per screenful. The *number* argument  
 2452 is a positive decimal integer. The `-n` option shall override any  
 2453 values obtained from any other source. B
  - 2454 `-p command`  
 2455 *+command* (Obsolescent.)  
 2456 Each time a screen from a new file is displayed or redisplayed B  
 2457 (including as a result of `more` commands; e.g., `:p`), execute the B  
 2458 `more` command(s) in the *command* arguments in the order B  
 2459 specified, as if entered by the user after the first screen has been B

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.



2460		displayed. No intermediate results shall be displayed (i.e., if the	B
2461		command is a movement to a screen different than the normal	B
2462		first screen, only the screen resulting from the command shall be	B
2463		displayed.) If any of the commands fail for any reason, an infor-	B
2464		mational message to this effect shall be written, and no further	B
2465		commands specified using the <code>-p</code> or <code>+command</code> options shall be	B
2466		executed for this file.	B
2467	<code>-s</code>	Behave as if consecutive empty lines were a single empty line.	B
2468	<code>-t tagstring</code>		
2469		Write the screenful of the file containing the tag named by the	
2470		<code>tagstring</code> argument. See the <code>ctags</code> utility in 5.7. The tags	
2471		feature represented by <code>-t tagstring</code> and the <code>:t</code> command (see	
2472		5.18.7.23) is optional. It shall be provided on any system that	
2473		also provides a conforming implementation of <code>ctags</code> ; otherwise,	
2474		the use of <code>-t</code> produces undefined results.	
2475		The file name resulting from the <code>-t</code> option shall be logically	B
2476		added as a prefix to the list of command-line files, as if specified	B
2477		by the user. If the tag named by the <code>tagstring</code> argument is not	B
2478		found, it shall be an error, and <code>more</code> shall take no further action.	B
2479		If the tag specifies a line number, the first line of the display shall	B
2480		contain the beginning of that line. If the tag specifies a pattern,	B
2481		the first line of the display shall contain the beginning of the	B
2482		matching text from the first line of the file that contains that pat-	B
2483		tern. If the line does not exist in the file or matching text is not	B
2484		found, an informational message to this effect shall be displayed,	B
2485		and <code>more</code> shall display the default screen as if <code>-t</code> had not been	B
2486		specified.	B
2487		If both the <code>-t tagstring</code> and <code>-p command</code> (or the obsolescent	B
2488		<code>+command</code> ) options are given, the <code>-t tagstring</code> shall be processed	B
2489		first; i.e., the file and starting line for the display shall be as	B
2490		specified by <code>-t</code> , and then the <code>-p</code> or <code>+command</code> <code>more</code> commands	B
2491		shall be executed. If the line (matching text) specified by the <code>-t</code>	B
2492		command does not exist (is not found), no <code>-p</code> or <code>+command</code> <code>more</code>	B
2493		commands shall be executed for this file at any time.	B
2494	<code>-u</code>	Treat <code>&lt;backspace&gt;</code> as a printable control character, displayed as	
2495		an implementation-defined character sequence (see 5.18.7),	
2496		suppressing backspacing and the special handling that produces	
2497		underlined or standout-mode text on some terminal types. Also,	
2498		do not ignore a <code>&lt;carriage-return&gt;</code> character at the end of a	
2499		line.	
2500			B

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.

2501 **5.18.4 Operands**

2502 The following operand shall be supported by the implementation:

2503     *file*           A pathname of an input file. If no *file* operands are specified, the  
 2504                    standard input shall be used. If a *file* is `-`, the standard input  
 2505                    shall be read at that point in the sequence.

2506 **5.18.5 External Influences**2507 **5.18.5.1 Standard Input**

2508 The standard input shall be used only if no *file* operands are specified, or if a *file*  
 2509 operand is `-`.

2510 **5.18.5.2 Input Files**

2511 The input files being examined shall be files of any type. If standard output is a     B  
 2512 terminal, standard error shall be used to read commands from the user. If stan-  
 2513 dard output is a terminal, standard error is not readable, and command input is  
 2514 needed, `more` may attempt to obtain user commands from the controlling termi-  
 2515 nal (e.g., `/dev/tty`); otherwise, `more` shall terminate with an error indicating  
 2516 that it was unable to read user commands. If standard output is not a terminal,  
 2517 no error shall result if standard error cannot be opened for reading.

2518 **5.18.5.3 Environment Variables**2519 The following environment variables shall affect the execution of `more`:

2520     **COLUMNS**           This variable shall override the system-selected horizontal  
 2521                    screen size. See 2.6 for valid values and results when it is  
 2522                    unset or null.

2523     **EDITOR**             This variable shall be used by the `v` command to select an  
 2524                    editor; see 5.18.7.

2525     **LANG**               This variable shall determine the locale to use for the  
 2526                    locale categories when both `LC_ALL` and the correspond-  
 2527                    ing environment variable (beginning with `LC_`) do not  
 2528                    specify a locale. See 2.6.

2529     **LC\_ALL**             This variable shall determine the locale to be used to over-  
 2530                    ride any values for locale categories specified by the set-  
 2531                    tings of `LANG` or any environment variables beginning  
 2532                    with `LC_`.

2533     **LC\_COLLATE**        This variable shall determine the locale for character col-  
 2534                    lation information in BREs.

2535       **LC\_CTYPE**       This variable shall determine the interpretation of  
 2536       sequences of bytes of text data as characters (e.g., single-  
 2537       versus multibyte characters in arguments and input files),  
 2538       and the behavior of character classes within BREs.

2539       **LC\_MESSAGES**   This variable shall determine the language in which mes-  
 2540       sages should be written.

2541       **LINES**           This variable shall override the system-selected vertical  
 2542       screen size, used as the number of lines in a screenful.  
 2543       See 2.6 for valid values and results when it is unset or  
 2544       null. The `-n` option shall take precedence over the **LINES**  
 2545       variable for determining the number of lines in a  
 2546       screenful.

2547       **MORE**            This variable shall be interpreted as a string containing  
 2548       options described in 5.18.3, preceded with hyphens and  
 2549       <blank>-separated as on the command line. Any  
 2550       command-line options shall be processed after those in the  
 2551       **MORE** variable, as if the command line were

2552                            `more $MORE options operands`

2553                            The **MORE** variable shall take precedence over the **TERM**  
 2554                            and **LINES** variables for determining the number of lines  
 2555                            in a screenful.

2556       **TERM**            This variable shall be interpreted as the name of the ter-  
 2557       minal type. If this variable is unset or null, an  
 2558       unspecified default terminal type shall be used.

#### 2559   **5.18.5.4 Asynchronous Events**

2560   Default.

#### 2561   **5.18.6 External Effects**

##### 2562   **5.18.6.1 Standard Output**

2563   The standard output shall be used to write the contents of the input files.

##### 2564   **5.18.6.2 Standard Error**

2565   Used for diagnostic messages and user commands (see 5.18.5.2) and, if standard  
 2566   output is a terminal device, to write a prompting string. The prompting string  
 2567   shall appear on the screen line below the last line of the file displayed in the   B  
 2568   current screenful. The prompt shall contain the name of the file currently being   B  
 2569   examined and shall contain an end-of-file indication and the name of the next file,   B  
 2570   if any, when prompting at the end-of-file. If an error or informational message is   B  
 2571   displayed, it is unspecified if it is contained in the prompt. If it is not contained in   B

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

2572 the prompt, it shall be displayed and then the user shall be prompted for a con- B  
 2573 tinuation character, at which point another message or the user prompt may be B  
 2574 displayed. The prompt is otherwise unspecified. It is unspecified if informational B  
 2575 messages are written for other user commands.

### 2576 5.18.6.3 Output Files

2577 None.

### 2578 5.18.7 Extended Description

2579 The following subclause describes the behavior of `more` when the standard output B  
 2580 is a terminal device. If the standard output is not a terminal device, no options B  
 2581 other than `-s` shall have any effect, and all input files shall be copied to standard B  
 2582 output otherwise unmodified, at which time `more` shall exit without further B  
 2583 action. B

2584 The number of lines available per “screen” shall be determined by the `-n` option, if B  
 2585 present, or by examining values in the environment (see 5.18.5.3). If neither B  
 2586 method yields a number, an unspecified number of lines shall be used.

2587 The maximum number of lines written shall be one less than this number because B  
 2588 the screen line after the last line written shall be used to write a user prompt and B  
 2589 user input. If the number of lines in the screen is less than two, the results are B  
 2590 undefined. It is unspecified if user input is permitted to be longer than the B  
 2591 remainder of a single line where the prompt has been written. B

2592 The number of columns available per line shall be determined by examining B  
 2593 values in the environment (see 5.18.5.3), with a default value as described in 2.6. C  
 2594 Lines that are longer than the display shall be folded; the length at which folding C  
 2595 occurs is unspecified, but should be appropriate for the output device. Folding C  
 2596 may occur between glyphs of single characters that take up multiple display C  
 2597 columns. C

2598 When standard output is a terminal and `-u` is not specified, `more` shall treat B  
 2599 `<backspace>s` and `<carriage-return>s` specially:

2600 — A character, followed first by a sequence of  $n$  `<backspace>s` (where  $n$  is B  
 2601 the same as the number of column positions that the character occupies), B  
 2602 then by  $n$  underscores (`_`), shall cause that character to be written as under- B  
 2603 lined text, if the terminal type supports that. The  $n$  underscores, followed B  
 2604 first by  $n$  `<backspace>s`, then any character with  $n$  column positions, also B  
 2605 shall cause that character to be written as underlined text, if the terminal B  
 2606 type supports that.

2607 — A sequence of  $n$  `<backspace>s` (where  $n$  is the same as the number of B  
 2608 column positions that the previous character occupies) that appears B  
 2609 between two identical printable characters shall cause the first of those two B  
 2610 characters to be written as emboldened text (i.e., visually brighter, stan- B  
 2611 d-out mode, or inverse-video mode), if the terminal type supports that, and B  
 2612 the second to be discarded. Immediately subsequent occurrences of

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2613 <backspace>s/character pairs for that same character also shall be dis-  
 2614 carded. (For example, the sequence a\ba\ba\ba is interpreted as a single  
 2615 emboldened a.)

2616 — The `more` utility shall logically discard all other <backspace> characters C  
 2617 from the line as well as the character which precedes them, if any. B

2618 — A <carriage-return> at the end of a line shall be ignored, rather than  
 2619 being written as a nonprintable character, as described in the next para- B  
 2620 graph. B

2621 It is implementation defined how other nonprintable characters are written.  
 2622 Implementations should use the same format that they use for the `exprint` com-  
 2623 mand; see 5.10.7.5.21. It is unspecified if a multicolumn character shall be B  
 2624 separated if it crosses a logical line boundary; it shall not be discarded. The B  
 2625 behavior is unspecified if the number of columns on the display is less than the B  
 2626 number of columns any single character in the line being displayed would occupy. B

2627 When each new file is displayed (or redisplayed), `more` shall write the first screen B  
 2628 of the file. Once the initial screen has been written, `more` shall prompt for a user B  
 2629 command. If the execution of the user command results in a screen that has lines B  
 2630 in common with the current screen, and the device has sufficient terminal capa- B  
 2631 bilities, `more` shall scroll the screen; otherwise, it is unspecified if the screen is B  
 2632 scrolled or redrawn. B

2633 For all files but the last (including standard input if no file was specified, and for B  
 2634 the last file as well, if the `-e` option was not specified), when `more` has written the C  
 2635 last line in the file, `more` shall prompt for a user command. This prompt shall B  
 2636 contain the name of the next file as well as an indication that `more` has reached B  
 2637 end-of-file. If the user command is `f`, <control-F>, <space>, `j`, <newline>, `d`, B  
 2638 <control-D>, or `s`, `more` shall display the next file. Otherwise, if displaying the B  
 2639 last file, `more` shall exit. Otherwise, `more` shall execute the user command B  
 2640 specified. B

2641 Several of the commands described in this clause display a previous screen from B  
 2642 the input stream. In the case that text is being taken from a nonrewindable B  
 2643 stream, such as a pipe, it is implementation defined how much backwards motion  
 2644 is supported. If a command cannot be executed because of a limitation on back- B  
 2645 wards motion, an error message to this effect shall be displayed, the current B  
 2646 screen shall not change, and the user shall be prompted for another command. B

2647 If a command cannot be performed because there are insufficient lines to display, B  
 2648 `more` shall alert the terminal. If a command cannot be performed because there B  
 2649 are insufficient lines to display or a `/` command fails: if the input is the standard B  
 2650 input, the last screen in the file may be displayed; otherwise, the current file and B  
 2651 screen shall not change, and the user shall be prompted for another command. B

2652 The interactive commands in the following subclauses shall be supported. Some  
 2653 commands can be preceded by a decimal integer, called *count* in the following  
 2654 descriptions. If not specified with the command, *count* shall default to 1.

2655 In the following descriptions, *pattern* is a BRE, as described in 2.8.3. The term  
 2656 “examine” is historical usage meaning “open the file for viewing”; for example,

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2657	<code>more foo</code> would be expressed as “examining” file <code>foo</code> . In the following descriptions, unless otherwise specified, <i>line</i> is a logical line in the <code>more</code> display, not a line from the file being examined.	B
2658		B
2659		B
2660	In the following descriptions, the “current position” refers to two things:	C
2661	— The position of the current line on the screen	C
2662	— The line number (in the file) of the current line on the screen	C
2663	Usually, the line on the screen corresponding to the current position is the third line on the screen. If this is not possible (there are fewer than three lines to display, or this is the first page of the file, or it is the last page of the file), then	C
2664	the current position is either the first or last line on the screen as described later.	C
2665		C
2666		C
2667	<b>5.18.7.1 Help</b>	
2668	<i>Synopsis:</i> <code>h</code>	
2669	Write a summary of these commands and other implementation-defined commands. The behavior shall be as if the <code>more</code> utility were executed with the <code>-e</code>	B
2670	option on a file that contained the summary information. The user shall be	B
2671	prompted as described in 5.18.7 when end-of-file is reached. If the user command	B
2672	is one of those specified to continue to the next file, <code>more</code> shall return to the file	B
2673	and screen state from which the <code>h</code> command was executed.	B
2674		B
2675	<b>5.18.7.2 Scroll forwards one screenful</b>	B
2676	<i>Synopsis:</i> <code>[count]f</code>	
2677	<i>Synopsis:</i> <code>[count]&lt;control-F&gt;</code>	
2678	Scroll forwards <i>count</i> lines, with a default of one screenful. If <i>count</i> is more than	B
2679	the screen size, only the final screenful shall be written.	B
2680	<b>5.18.7.3 Scroll backwards one screenful</b>	B
2681	<i>Synopsis:</i> <code>[count]b</code>	
2682	<i>Synopsis:</i> <code>[count]&lt;control-B&gt;</code>	
2683	Scroll backwards <i>count</i> lines, with a default of one screenful. If <i>count</i> is more	B
2684	than the screen size, only the final screenful shall be written.	B
2685	<b>5.18.7.4 Scroll forwards one line</b>	
2686	<i>Synopsis:</i> <code>[count]&lt;space&gt;</code>	
2687	<i>Synopsis:</i> <code>[count]j</code>	
2688	<i>Synopsis:</i> <code>[count]&lt;newline&gt;</code>	
2689	Scroll forwards <i>count</i> lines. The default <i>count</i> for <code>&lt;space&gt;</code> shall be one screenful;	
2690	for <code>j</code> and <code>&lt;newline&gt;</code> , one line. The entire <i>count</i> lines shall be written, even if	
2691	<i>count</i> is more than the screen size.	B

2692 **5.18.7.5 Scroll backwards one line**2693 *Synopsis:* [*count*]k

2694 Scroll backwards *count* lines. The entire *count* lines shall be written, even if B  
 2695 *count* is more than the screen size.

2696 **5.18.7.6 Scroll forwards one-half screenful**2697 *Synopsis:* [*count*]d2698 *Synopsis:* [*count*]<control-D>

2699 Scroll forwards *count* lines, with a default of one half of the screen size. If *count*  
 2700 is specified, it shall become the new default for subsequent d <control-D>, u, B  
 2701 and <control-U> commands. The entire *count* lines shall be written, even if B  
 2702 *count* is more than the screen size. B

2703 **5.18.7.7 Skip forwards one line**2704 *Synopsis:* [*count*]s

2705 Display the screenful beginning with the line *count* lines after the last line on the B  
 2706 current screen. If *count* would cause the current position to be such that less than  
 2707 one screenful would be written, the last screenful in the file shall be written.

2708 **5.18.7.8 Scroll backwards one-half screenful**2709 *Synopsis:* [*count*]u2710 *Synopsis:* [*count*]<control-U>

2711 Scroll backwards *count* lines, with a default of one half of the screen size. If *count*  
 2712 is specified, it shall become the new default for subsequent d <control-D>, u, B  
 2713 and <control-U> commands. The entire *count* lines shall be written, even if B  
 2714 *count* is more than the screen size. B

2715 **5.18.7.9 Go to beginning of file**2716 *Synopsis:* [*count*]g

2717 Display the screenful beginning with the line *count*. B

2718 **5.18.7.10 Go to end-of-file**2719 *Synopsis:* [*count*]G

2720 If *count* is specified, display the screenful beginning with the line *count*. Other- B  
 2721 wise, display the last screenful of the file. B

2722 **5.18.7.11 Refresh the screen**2723 *Synopsis:* r2724 *Synopsis:* <control-L>

2725 Refresh the screen.

2726 **5.18.7.12 Discard and refresh**2727 *Synopsis:* R

2728 Refresh the screen, discarding any buffered input. If the current file is nonseek-  
 2729 able, buffered input shall not be discarded, and the R command is equivalent to  
 2730 the r command.

2731 **5.18.7.13 Mark position**2732 *Synopsis:* m*letter*

2733 Mark the current position with the letter named by *letter*, where *letter* represents  
 2734 the name of one of the lowercase letters of the portable character set. When a  
 2735 new file is examined, all marks may be lost.

2736 **5.18.7.14 Return to mark**2737 *Synopsis:* ' *letter*

2738 Return to the position that was previously marked with the letter named by *letter*,  
 2739 making that line the current position.

2740 **5.18.7.15 Return to previous position**2741 *Synopsis:* ' '

2742 Return to the position from which the last large movement command was exe-  
 2743 cuted (where a “large movement” is defined as any movement of more than a  
 2744 screenful of lines). If no such movements have been made, return to the begin-  
 2745 ning of the file.

2746 **5.18.7.16 Search forwards for pattern**2747 *Synopsis:* [*count*]/[!]*pattern*<newline>

2748 Display the screenful beginning with the *count*-th line containing the pattern. B  
 2749 The search shall start after the first line currently displayed. The null BRE B  
 2750 (/<newline>) shall repeat the search using the previous BRE, with a default B  
 2751 *count*. If the character ! is included, matching lines shall be those that do not B  
 2752 contain the pattern. If no match is found for the pattern, a message to that effect B  
 2753 shall be displayed. B



2754 **5.18.7.17 Search backwards for pattern**2755 *Synopsis:* `[count]?[!]pattern<newline>`

2756 Display the screenful beginning with the *count*-th previous line containing the B  
 2757 pattern. The search shall start on the last line before the first line currently B  
 2758 displayed. The null BRE (?<newline>) shall repeat the search using the previous B  
 2759 BRE, with a default *count*. If the character ! is included, matching lines shall be B  
 2760 those that do not contain the pattern. If no match is found for the pattern, a mes- B  
 2761 sages to that effect shall be displayed. B

2762 **5.18.7.18 Repeat search**2763 *Synopsis:* `[count]n`

2764 Repeat the previous search for *count*-th line containing the last *pattern* (or not B  
 2765 containing the last *pattern*, if the previous search was /! or ?!). B

2766 **5.18.7.19 Repeat search in reverse**2767 *Synopsis:* `[count]N`

2768 Repeat the search in the opposite direction of the previous search for the *count*-th B  
 2769 line containing the last *pattern* (or not containing the last *pattern*, if the previous B  
 2770 search was /! or ?!).

2771 **5.18.7.20 Examine new file**2772 *Synopsis:* `:e [filename]<newline>`

2773 Examine a new file. If the *filename* argument is not specified, the “current” file B  
 2774 (see the :n and :p commands below) shall be re-examined. The *filename* shall be B  
 2775 subjected to the process of shell word expansions (see 3.6); if more than a single B  
 2776 pathname results, the effects are unspecified. If *filename* is a number sign (#), B  
 2777 the previously examined file shall be re-examined. If *filename* is not accessible for B  
 2778 any reason (including that it is a nonseekable file), an error message to this effect B  
 2779 shall be displayed and the current file and screen shall not change. B

2780 **5.18.7.21 Examine next file**2781 *Synopsis:* `[count]:n`

2782 Examine the next file. If a number *count* is specified, the *count*-th next file shall B  
 2783 be examined. If *filename* refers to a nonseekable file, the results are unspecified.

2784 **5.18.7.22 Examine previous file**2785 *Synopsis:* `[count]:p`

2786 Examine the previous file. If a number *count* is specified, the *count*-th previous B  
 2787 file shall be examined. If *filename* refers to a nonseekable file, the results are

2788 unspecified.

### 2789 **5.18.7.23 Go to tag**

2790 *Synopsis:* :t *tagstring*<newline>

2791 If the file containing the tag named by the *tagstring* argument is not the current B  
 2792 file, examine the file, as if the :e command was executed with that file as the B  
 2793 argument. Otherwise, or in addition, display the screenful beginning with the B  
 2794 tag, as described for the -t option (see 5.18.3). If the ctags utility is not sup- B  
 2795 ported by the system, the use of :t produces undefined results.

### 2796 **5.18.7.24 Invoke editor**

2797 *Synopsis:* v

2798 Invoke an editor to edit the current file being examined. If standard input is  
 2799 being examined, the results are unspecified. The name of the editor shall be B  
 2800 taken from the environment variable **EDITOR** or shall default to vi. If the last B  
 2801 pathname component in **EDITOR** is either “ex” or “vi,” the editor shall be B  
 2802 invoked with a -c *linenumber* command-line argument, where *linenumber* is the B  
 2803 line number of the physical line containing the logical line currently displayed as B  
 2804 the first line of the screen. It is implementation defined whether line-setting  
 2805 options are passed to editors other than vi and ex.

2806 When the editor exits, more shall resume with the same file and screen as when B  
 2807 the editor was invoked. B

### 2808 **5.18.7.25 Display position**

2809 *Synopsis:* =

2810 *Synopsis:* <control-G>

2811 Write a message for which the information references the first byte of the line B  
 2812 after the last line of the file on the screen. This message shall include the name of B  
 2813 the file currently being examined, its number relative to the total number of files B  
 2814 there are to examine, the physical line number, the byte number and the total B  
 2815 bytes in the file, and what percentage of the file precedes the current position. If B  
 2816 more is reading from standard input, or the file is shorter than a single screen, B  
 2817 the line number, the byte number, the total bytes, and the percentage need not be B  
 2818 written. B

### 2819 **5.18.7.26 Quit**

2820 *Synopsis:* q

2821 *Synopsis:* :q

2822 *Synopsis:* ZZ

2823 Exit more.

2824 **5.18.8 Exit Status**2825 The `more` utility shall exit with one of the following values:

2826       0     Successful completion.

2827       &gt;0    An error occurred.

2828 **5.18.9 Consequences of Errors**

2829 If an error is encountered accessing a file when using the `:n` command, `more`  
 2830 shall attempt to examine the next file in the argument list, but the final exit  
 2831 status shall be affected. If an error is encountered accessing a file via the `:p` com-  
 2832 mand, `more` shall attempt to examine the previous file in the argument list, but  
 2833 the final exit status shall be affected. If an error is encountered accessing a file  
 2834 via the `:e` command, `more` shall remain in the current file, and the final exit  
 2835 status shall not be affected.

2836 **5.22 patch – Apply changes to files**2837 ⇒ **5.22.3 patch Options.** *Change the `-D` description to:* B2838       `-D define`   Mark changes with one of the following C preprocessor con- B  
2839                    structs: B2840                    `#ifdef define` B

2841                    ... B

2842                    `#endif` B2843                    `#ifndef define` B

2844                    ... B

2845                    `#endif` B2846                    optionally combined with the C preprocessor construct `#else`. B2847 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B

2848 #69 submitted for IEEE Std 1003.2-1992. B

2849 ⇒ **5.22.7.2 patch Filename Determination.** *Replace the entire subclause*  
2850 *with:*2851 If no *file* operand is specified, `patch` shall perform the following steps to deter-  
2852 mine the filename to use:

- 2853 (1) If the type of diff is context, the `patch` utility shall delete pathname com-  
 2854 ponents (as specified by the `-p` option) from the filename on the line  
 2855 beginning with `***`, then test for the existence of this file relative to the  
 2856 current directory (or the directory specified with the `-d` option). If the file  
 2857 exists, the `patch` utility shall use this filename.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 2858 (2) If the type of diff is context, the `patch` utility shall delete the pathname  
 2859 components (as specified by the `-p` option) from the filename on the line  
 2860 beginning with `---`, then test for the existence of this file relative to the  
 2861 current directory (or the directory specified with the `-d` option). If the file  
 2862 exists, the `patch` utility shall use this filename.
- 2863 (3) If the header information contains a line beginning with the string  
 2864 `Index:`, the `patch` utility shall delete pathname components (as  
 2865 specified by the `-p` option) from this line, then test for the existence of  
 2866 this file relative to the current directory (or the directory specified with  
 2867 the `-d` option). If the file exists, the `patch` utility shall use this filename.
- 2868 (4) The `patch` utility shall write a prompt to standard output and request a  
 2869 filename interactively from the controlling terminal (e.g., `/dev/tty`).

2870 **Rationale:** The change substituting `/dev/tty` for standard input corrects an  
 2871 error that deviated from historical practice and is the result of interpretation  
 2872 request PASC 1003.2-92 #19 submitted for IEEE Std 1003.2-1992.

2873 The other wording changes are required to match historical practice and are the  
 2874 result of interpretation request PASC 1003.2-92 #15 submitted for IEEE Std  
 2875 1003.2-1992.

## 2876 **5.24 renice – Set system scheduling priorities of running** B 2877 **processes** B

2878 ⇒ **5.24.1 renice Synopsis.** *Change the first Synopsis line (the non-Obsolescent* B  
 2879 *one) to:* B

2880 `renice -n increment [ -g | -p | -u ] ID...` B

2881 ⇒ **5.24.2 renice Description.** *Delete the second paragraph, which currently* B  
 2882 *contains:* B

2883 The system scheduling priority shall be bounded in an implementation-defined B  
 2884 manner. If the requested *increment* (or *nice\_value* in the obsolescent versions) B  
 2885 would raise or lower the system scheduling priority of the executed utility B  
 2886 beyond implementation-defined limits, then the limit whose value was B  
 2887 exceeded shall be used. B

2888	⇒ <b>5.24.3 renice Options.</b> <i>Change the full description of <code>-n</code> to:</i>	B
2889	<code>-n</code> <i>increment</i>	B
2890	The <code>-n</code> option for the <code>renice</code> utility shall behave as described	B
2891	for the <code>-n</code> option for the <code>nice</code> utility (see 5.24).	B
2892	⇒ <b>5.24.4 renice Operands.</b> <i>Change the description of the <code>nice_value</code> operand:</i>	B
2893	<code>nice_value</code> (Obsolescent.) The value specified shall be taken as the actual	B
2894	system scheduling priority, rather than as an increment to the	B
2895	existing system scheduling priority. The system scheduling	B
2896	priority shall be bounded in an implementation-defined	B
2897	manner. If the requested <code>nice_value</code> would raise or lower the	B
2898	system scheduling priority of the executed utility beyond	B
2899	implementation-defined limits, then the limit whose value was	B
2900	exceeded shall be used. Specifying a scheduling priority	B
2901	higher than that of the existing process may require appropri-	B
2902	ate privileges.	B
2903	<b>Rationale:</b> The preceding changes are the result of interpretation requests PASC	B
2904	1003.2-92 #83 and #84 submitted for IEEE Std 1003.2-1992.	B

## 2905 **5.32 unexpand – Convert spaces to tabs**

2906 ⇒ **5.32.5.3 unexpand Environment Variables.** *In the description of*  
 2907 **LC\_CTYPE**, *change the phrase “... width in column positions each character*  
 2908 *would occupy on a constant-width-font output device” to:*

2909 ... width in column positions each character would occupy on an output  
 2910 device.

2911 **Rationale:** This change partially satisfies the following corrigendum request  
 2912 from ISO/IEC 9945-2: 1993 Annex H.2:

2913 (15) In 5.11.5.3 and 5.32.5.3, in the last sentence of the `LC_CTYPE` paragraph  
 2914 for `expand` and `unexpand`, the phrase “on a constant-width-font output  
 2915 device” may be redundant because of definitions elsewhere in the  
 2916 standard.

2917 **5.33 uuencode – Decode a binary file**

2918 **Rationale:** This change partially satisfies the following corrigendum request  
 2919 from ISO/IEC 9945-2: 1993 Annex H.2:

2920 (14) The `uuencode` utility should support the BASE64-encoding specified in  
 2921 the MIME-RFC currently under consideration for Internet use. The  
 2922 `uuencode` utility should allow the user to override the output file name  
 2923 that is embedded in the file. Both utilities should be moved from Section  
 2924 5 to Section 4.

2925 ⇒ **5.33.1 uuencode Synopsis.** *Change the Synopsis to be:*

2926 `uuencode [-o outfile] [file]`

2927 ⇒ **5.33.2 uuencode Description.** *Replace the first paragraph with:*

2928 The `uuencode` utility shall read a file, or standard input if no file is specified,  
 2929 that includes data created by the `uuencode` utility (see 5.34). The `uuencode`  
 2930 utility shall scan the input file, searching for data compatible with one of the  
 2931 formats specified in 5.34.6.1 and attempt to create or overwrite the file  
 2932 described by the data (or overridden by the `-o` option). The pathname shall be  
 2933 contained in the data or specified by the `-o` option. The file access permission  
 2934 bits and contents for the file to be produced shall be contained in that data.  
 2935 The mode bits of the created file (other than standard output) shall be set from  
 2936 the file access permission bits contained in the data; i.e., other attributes of the  
 2937 mode, including the file mode creation mask (see `umask` in 4.67), shall not  
 2938 affect the file being produced.

2939 ⇒ **5.33.3 uuencode Options.** *Change this subclause to:*

2940 The `uuencode` utility shall conform to the utility argument syntax guidelines  
 2941 described in 2.10.2.

2942 The following option shall be supported by the implementation:

2943 `-o outfile` A pathname of a file that shall be used instead of any path-  
 2944 name contained in the input data. Specifying an *outfile*  
 2945 option-argument of `/dev/stdout` shall indicate standard  
 2946 output.

2947 ⇒ **5.33.6.1 uuencode Standard Output.** *Change this subclause to:*

2948       If the file data header encoded by uuencode is – or the –o /dev/stdout  
2949       option overrides the file data, the standard output shall be in the same format  
2950       as the file originally encoded by uuencode. Otherwise, the standard output  
2951       shall not be used.

2952       *Editor’s Note: The following rationale will be added to E.5.33, but is kept here with*  
2953       *uuencode for this draft:*

2954       **uuencode Rationale.** *(This subclause is not a part of P1003.2b)*

2955       The –o option is not historical practice, but was added at the request of WG15 so  
2956       that the user could override the target pathname without having to edit the input  
2957       data itself.

2958       In an early draft, the [–o outfile] option-argument allowed the use of – to mean  
2959       standard output. The symbol – has only been used previously in this standard as  
2960       a standard input indicator. The developers of the standard did not wish to over-  
2961       load the meaning of – in this manner. The /dev/stdout concept exists on most  
2962       modern systems. The /dev/stdout syntax does not refer to a new special file. It  
2963       is just a magic cookie to specify standard output.

## 2964       **5.34 uuencode – Encode a binary file**

2965       **Rationale:** This change partially satisfies the following corrigendum request  
2966       from ISO/IEC 9945-2: 1993 Annex H.2:

2967       (14) The uuencode utility should support the BASE64-encoding specified in  
2968       the MIME-RFC currently under consideration for Internet use. The  
2969       uuencode utility should allow the user to override the output file name  
2970       that is embedded in the file. Both utilities should be moved from Section  
2971       5 to Section 4.

2972 ⇒ **5.34.1 uuencode Synopsis.** *Change the Synopsis to be:*

2973       uuencode [–m] [file] decode\_pathname

2974 ⇒ **5.34.2 uuencode Description.** *Change the phrase “the algorithm” to “one of*  
2975       *the algorithms”.*

2976 ⇒ **5.34.3 uuencode Options.** *Change this subclause to:*

2977 The uuencode utility shall conform to the utility argument syntax guidelines  
2978 described in 2.10.2.

2979 The following option shall be supported by the implementation:

2980	-m	Encode the output using the MIME Base64 algorithm	
2981		described in 5.34.6.1.1. If -m is not specified, the historical	C
2982		algorithm described in 5.34.6.1.2 shall be used.	C

2983 ⇒ **5.34.4 uuencode Operands.** *Change the description of decode\_pathname to:*

2984 *decode\_pathname*

2985 The pathname of the file into which the uudecode utility (see  
2986 5.33) shall place the decoded file. Specifying a  
2987 *decode\_pathname* operand of /dev/stdout shall indicate that  
2988 uudecode is to use standard output. If there are characters  
2989 in *decode\_pathname* that are not in the portable filename  
2990 character set (see Section 2.2.2.131 in POSIX.1 {8}), the results  
2991 are unspecified.

2992 ⇒ **5.34.6.1 uuencode Standard Output.** *Replace this subclause with the*  
2993 *following:*

#### 2994 **5.34.6.1.1 uuencode Base64 Algorithm**

2995 The standard output shall be a text file (encoded in the character set of the  
2996 current locale) that begins with the line:

2997 "begin-base64 $\Delta$ %s $\Delta$ %s\n", <mode>, *decode\_pathname*

2998 and ends with the line:

2999 "====\n"

3000 In both cases, the lines shall have no preceding or trailing <blank>s.

3001 The encoding process represents 24 b groups of input bits as output strings of four  
3002 encoded characters. Preceding from left to right, a 24 b input group shall be  
3003 formed by concatenating three 8 b input groups. These 24 b then shall be treated  
3004 as four concatenated 6 b groups, each of which shall be translated into a single  
3005 digit in the base64 alphabet. When encoding a bit stream via the base64 encod-  
3006 ing, the bit stream shall be presumed to be ordered with the most-significant-bit  
3007 first. That is, the first bit in the stream shall be the high-order bit in the first  
3008 byte, and the eighth bit shall be the low-order bit in the first byte, and so on.

3009 Each 6 b group is used as an index into an array of 64 printable characters, as  
3010 shown in Table 5-100.



3011

**Table 5-100 – uuencode Base64 Values**

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

3030 The character referenced by the index shall be placed in the output string.

3031 The output stream (encoded bytes) shall be represented in lines of no more than  
 3032 76 characters each. All line breaks or other characters not found in the table  
 3033 shall be ignored by decoding software (see `uudecode` in 5.33).

3034 Special processing shall be performed if fewer than 24 b are available at the end  
 3035 of a message or encapsulated part of a message. A full encoding quantum shall be  
 3036 always completed at the end of a message. When fewer than 24 input bits are  
 3037 available in an input group, zero bits shall be added (on the right) to form an  
 3038 integral number of 6 b groups. Output character positions that are not required  
 3039 to represent actual input data shall be set to the character =. Since all base64  
 3040 input is an integral number of octets, only the following cases can arise:

- 3041 (1) The final quantum of encoding input is an integral multiple of 24 b; here,  
 3042 the final unit of encoded output shall be an integral multiple of 4 charac-  
 3043 ters with no = padding.
- 3044 (2) The final quantum of encoding input is exactly 8 b; here, the final unit of  
 3045 encoded output shall be two characters followed by two = padding  
 3046 characters.
- 3047 (3) The final quantum of encoding input is exactly 16 b; here, the final unit  
 3048 of encoded output shall be three characters followed by one = padding  
 3049 character.
- 3050 (4) The terminating ==== evaluates to nothing and denotes the end of the  
 3051 encoded data.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

3052 **5.34.6.1.2 uuencode Historical Algorithm**3053 *<current contents of 5.34.6.1>*3054 *Editor's Note: The following rationale will be added to E.5.34, but is kept here with*  
3055 *uuencode for this draft:*3056 **uuencode Rationale.** *(This subclause is not a part of P1003.2b)*3057 A new algorithm was added at the request of the international community to  
3058 parallel work in the Internet MIME RFC 2045 {B90}. As with the historical uuen- C  
3059 code format, the Base64 Content-Transfer-Encoding is designed to represent  
3060 arbitrary sequences of octets in a form that is not humanly readable. A 65-  
3061 character subset of ISO/IEC 646 {1} is used, enabling 6 b to be represented per  
3062 printable character. (The extra 65th character, =, is used to signify a special pro-  
3063 cessing function.)3064 This subset has the important property that it is represented identically in all  
3065 versions of ISO/IEC 646 {1}, including US ASCII, and all characters in the subset  
3066 are also represented identically in all versions of EBCDIC. The historical uuen-  
3067 code algorithm does not share this property, which is the reason that a second  
3068 algorithm was added to POSIX.2.3069 The string ==== was used for the termination instead of the end used in the origi-  
3070 nal format because the latter is a string that could be valid encoded input.3071 In an early draft, the -m option was named -b (for Base64), but it was renamed to  
3072 reflect its relationship to the Internet MIME RFC. A -u was also present to invoke C  
3073 the default algorithm, but since this was not historical practice, it was omitted as  
3074 being unnecessary.

3075 See the uudecode rationale for the derivation of the /dev/stdout symbol.

3076 ⇒ **5.35 vi – Screen-oriented (visual) display editor.** *Replace the entire vi B*  
3077 *clause with the following.* B3078 *Editor's Note: All of this clause has been changed in Draft 11 from the POSIX.2- B*  
3079 *1992 version. It is not further diffmarked. The rationale in Annex E is also com- B*  
3080 *pletely replaced.* B

## 3081 **5.35 vi – Screen-oriented (visual) display editor**

3082 This utility shall be provided on systems that both support the User Portability  
3083 Utilities Option and define the {POSIX2\_CHAR\_TERM} symbol. On other systems,  
3084 it is optional.

### 3085 **5.35.1 Synopsis**

3086 `vi [-rR] [-c command] [-t tagstring] [-w size] [file ...]`

3087 *Obsolescent Version:*

3088 `vi [-rR] [+command] [-t tagstring] [-w size] [file ...]`

### 3089 **5.35.2 Description**

3090 The `vi` (visual) utility is a screen-oriented text editor. Only the open and visual C  
3091 modes of the editor are described in this clause. See the line editor `ex` (5.10) for C  
3092 additional editing capabilities used in `vi`. The user can switch back and forth  
3093 between `vi` and `ex`, and execute `ex` commands from within `vi`.

3094 This clause uses the term “edit buffer” to describe the current working text. No  
3095 specific implementation is implied by this term. All editing changes are per-  
3096 formed on the edit buffer, and no changes to it shall affect any file until an editor  
3097 command writes the file.

3098 When using `vi`, the terminal screen acts as a “window” into the edit buffer.  
3099 Changes made to the edit buffer shall be reflected in the screen display, and the  
3100 position of the cursor on the screen shall indicate the current position within the  
3101 edit buffer.

3102 Certain terminals do not have all the capabilities necessary to support the com-  
3103 plete `vi` definition. When these commands cannot be supported on such termi-  
3104 nals, this condition shall neither produce an error message such as “not an editor  
3105 command” nor report a syntax error. The implementation may either accept the  
3106 commands and produce results on the screen that are the result of an unsuccess-  
3107 ful attempt to meet the requirements of this standard or report an error describ-  
3108 ing the terminal-related deficiency.

### 3109 **5.35.3 Options**

3110 The `vi` utility shall conform to the utility argument syntax guidelines described  
3111 in 2.10.2, except for the obsolescent *+command* “option.”

3112 The following options shall be supported by the implementation:

- 3113        -c *command*  
 3114        +*command* (Obsolescent.)  
 3115                See the `ex` command description of the `-c` and `+command` options  
 3116                (5.10.3).
- 3117        -r                See the `ex` command description of the `-r` option (5.10.3).
- 3118        -R                See the `ex` command description of the `-R` option (5.10.3).
- 3119        -t *tagstring*  
 3120                See the `ex` command description of the `-t` option (5.10.3).
- 3121        -w *size*        See the `ex` command description of the `-w` option (5.10.3).

### 3122    **5.35.4 Operands**

- 3123    See the description of the operands supported by the `ex` command (5.10.4) for a  
 3124    description of the operands supported by the `vi` command.

### 3125    **5.35.5 External Influences**

#### 3126    **5.35.5.1 Standard Input**

- 3127    If standard input is not a terminal device, undefined results occur. The standard  
 3128    input consists of a series of commands and input text, as described in 5.35.7.
- 3129    If a read from the standard input returns an error, or if the editor detects an end-  
 3130    of-file condition from the standard input, it shall be equivalent to a SIGHUP asyn-  
 3131    chronous event.

#### 3132    **5.35.5.2 Input Files**

- 3133    See the description of the input files supported by the `ex` command (5.10.5.2) for a  
 3134    description of the input files supported by the `vi` command.

#### 3135    **5.35.5.3 Environment Variables**

- 3136    See the description of the environment variables that affect the execution of the  
 3137    `ex` command (5.10.5.3) for a description of the environment variables that shall  
 3138    affect the `vi` command.

#### 3139    **5.35.5.4 Asynchronous Events**

- 3140    See the description of the asynchronous events that affect the execution of the `ex`  
 3141    command (5.10.5.4) for a description of the asynchronous events that shall affect  
 3142    the `vi` command.

3143 **5.35.6 External Effects**3144 **5.35.6.1 Standard Output**

3145 If standard output is not a terminal device, undefined results occur.

3146 Standard output may be used for writing prompts to the user, for informational  
3147 and error messages, and for writing lines from the edit buffer.3148 **5.35.6.2 Standard Error**

3149 If standard error is not a terminal device, undefined results occur.

3150 Used only for diagnostic messages.

3151 **5.35.6.3 Output Files**3152 See the description of the output files supported by the `ex` command (5.10.6.3) for  
3153 a description of the output files supported by the `vi` command.3154 **5.35.7 Extended Description**3155 If the terminal does not have the capabilities necessary to support an unspecified  
3156 portion of the `vi` definition, implementations shall start initially in `ex` mode (see  
3157 5.10) or open mode. Otherwise, after initialization, `vi` shall be in command mode;  
3158 text input mode can be entered by one of several commands used to insert or  
3159 change text. In text input mode, `<ESC>` can be used to return to command mode  
3160 (see 5.35.7.3.9); other uses of `<ESC>` are described in 5.35.7.2.14.3161 **5.35.7.1 `ex` and `vi` Initialization**3162 See the description of `ex` and `vi` Initialization for the `ex` command (5.10.7.1) for a  
3163 description of `ex` and `vi` Initialization for the `vi` utility.3164 **5.35.7.2 `vi` Command Descriptions**3165 The following symbols are used in this clause to represent arguments to  
3166 commands.3167 *buffer* See the description of *buffer* in 5.10.7.5.3168 In open and visual mode, when a command synopsis shows both  
3169 [*buffer*] and [*count*] preceding the command name, they can be C  
3170 specified in either order.3171 *count* A positive integer used as an optional argument to most commands,  
3172 normally to give a repeat count or as a size. This argument is  
3173 optional and shall default to 1 unless otherwise specified.

3174 The Synopsis lines for the vi commands <control-G>, <control-  
 3175 L>, <control-R>, <control-]>, %, &, ^, D, m, M, Q, u, U, and ZZ do  
 3176 not have *count* as an optional argument. Regardless, it shall not be  
 3177 an error to specify a count to these commands, and any specified  
 3178 count shall be ignored.

3179 *motion* An optional trailing argument used by the !, <, >, c, d, and y com-  
 3180 mands, which is used to indicate the region of text that shall be  
 3181 affected by the command. The motion can be either one of the com-  
 3182 mand characters repeated or one of several other vi commands  
 3183 (listed in the following table). Each of the applicable commands  
 3184 specifies the region of text matched by repeating the command; each  
 3185 command that can be used as a motion command specifies the region  
 3186 of text it affects.

3187 Commands that take *motion* arguments operate on either lines or C  
 3188 characters, depending on the circumstances. When operating on C  
 3189 lines, all lines that fall partially or wholly within the text region C  
 3190 specified for the command shall be affected. When operating on C  
 3191 characters, only the exact characters in the specified text region shall C  
 3192 be affected. Each motion command specifies this individually. C

3193 When command that may be motion commands are not used as C  
 3194 motion commands, they shall set the current position to the current C  
 3195 line and column as specified.

3196 The following shall be valid cursor motion commands:

3197	<control-H>	;	<i>\ character</i>
3198	<newline>	?	b
3199	<carriage-return>	B	e
3200	<control-N>	E	f
3201	<control-P>	F	h
3202	<space>	G	j
3203	\$	H	k
3204	%	L	l
3205	' <i>character</i>	M	n
3206	(	N	t
3207	)	T	w
3208	+	W	{
3209	,	[[	
3210	-	]]	}
3211	/	^	0
3212	—		

3213 Any *count* that is specified to a command that has an associated C  
 3214 motion command shall be applied to the motion command. If a *count* C  
 3215 is applied to both the command and its associated motion command, C  
 3216 the effect shall be multiplicative. C

3217 The following symbol is used in this clause to specify locations in the edit buffer:

- 3218 *current character*  
 3219 The character that is currently displayed by the cursor. C
- 3220 The following symbols are used in this clause to specify command actions:
- 3221 *bigword*  
 3222 In the POSIX Locale, `vi` shall recognize four kinds of bigwords:
- 3223 (1) A maximal sequence of nonblank characters preceded and followed by  
 3224 <blank> characters or the beginning or end of a line or the edit buffer
  - 3225 (2) One or more sequential empty or <blank>-filled lines
  - 3226 (3) The first character in the edit buffer
  - 3227 (4) The last character in the edit buffer
- 3228 *word*  
 3229 In the POSIX Locale, `vi` shall recognize five kinds of words:
- 3230 (1) A maximal sequence of letters, digits and underscores, delimited at  
 3231 both ends by: characters other than letters, digits, or underscores; the  
 3232 beginning or end of a line; the beginning or end of the edit buffer
  - 3233 (2) A maximal sequence of characters other than letters, digits, under-  
 3234 scores, or <blank>s, delimited at both ends by: a letter, digit, under-  
 3235 score, or <blank>s; the beginning or end of a line; the beginning or  
 3236 end of the edit buffer
  - 3237 (3) One or more sequential empty or <blank>-filled lines
  - 3238 (4) The first character in the edit buffer
  - 3239 (5) The last character in the edit buffer
- 3240 *section boundary*
- 3241 (1) A line whose first character is a <form-feed>
  - 3242 (2) A line whose first character is an open curly brace ({})
  - 3243 (3) A line whose first character is a period and whose second and third  
 3244 characters match a two-character pair in the `sections` edit option  
 3245 (see 5.10.7.8.17)
  - 3246 (4) A line whose first character is a period and whose only other charac-  
 3247 ter matches the first character of a two-character pair in the `sec-`  
 3248 `tions` edit option, where the second character of the two-character  
 3249 pair is a <space>
  - 3250 (5) The first line of the edit buffer
  - 3251 (6) The last line of the edit buffer if the last line of the edit buffer is  
 3252 empty or if it is a `]]` or `}` command; otherwise, the last character of  
 3253 the last line of the edit buffer

- 3254 *paragraph boundary*
- 3255 (1) A section boundary
- 3256 (2) A line whose first character is a period and whose second and third  
3257 characters match a two-character pair in the `paragraphs` edit option  
3258 (see 5.10.7.8.11)
- 3259 (3) A line whose first character is a period and whose only other charac-  
3260 ter matches the first character of a two-character pair in the `para-`  
3261 `graphs` edit option, where the second character of the two-character  
3262 pair is a `<space>`
- 3263 (4) One or more sequential empty or `<blank>`-filled lines C
- 3264 *remembered search direction* C
- 3265 See the description of *remembered search direction* in 5.10.7.5.
- 3266 *sentence boundary*
- 3267 (1) A paragraph boundary
- 3268 (2) The first nonblank character that occurs after a paragraph boundary
- 3269 (3) The first nonblank character that occurs after a period (.), exclama-  
3270 tion point (!) or question mark (?), followed by two `<space>`s or the  
3271 end of a line; any number of closing parenthesis ()), closing brackets  
3272 (]), double quote ("), or single quote (') characters can appear  
3273 between the punctuation mark and the two `<space>`s or end-of-line
- 3274 Any lines displayed on the screen that logically represent lines after the last line  
3275 in the edit buffer shall consist of a single tilde (~) character.
- 3276 The last line of the screen shall be used to report errors or display informational  
3277 messages. It shall also be used to display the input for "line-oriented commands"  
3278 (/, ?, :, and !). When a line-oriented command is executed, the editor shall enter  
3279 text input mode on the last line on the screen, using the respective command  
3280 characters as prompt characters. (In the case of the ! command, the associated  
3281 motion shall be entered by the user before the editor enters text input mode.) The  
3282 line entered by the user shall be terminated by a `<newline>`, a non-`<control-`  
3283 `V>`-escaped `<carriage-return>`, or unescaped `<ESC>`. It is unspecified if more C  
3284 characters than require a display width minus one column number of screen C  
3285 columns can be entered. C
- 3286 If any command is executed that overwrites a portion of the screen other than the  
3287 last line of the screen, (e.g., the `ex suspend`, or ! commands), other than the `ex`  
3288 `shell` command, the user shall be prompted for a character before the screen is  
3289 refreshed and the edit session continued.
- 3290 Lines that are longer than the display shall be folded; the length at which folding  
3291 occurs is unspecified, but should be appropriate for the output device. Folding  
3292 may occur between glyphs of single characters that take up multiple display C  
3293 columns. By default, no `vi` command shall operate on screen lines; instead, all C  
3294 commands shall operate on physical lines, which may occupy one or more logical C  
3295 (i.e., display) lines. Unless otherwise specified, *line* refers to a physical line. C



- 3296 Tab characters shall take up the number of columns on the screen set by the  
 3297 `tabstop` edit option (see 5.10.7.8.23), unless there are less than that number of  
 3298 columns before the display margin that will cause the displayed line to be folded; C  
 3299 in this case, they shall only take up the number of columns up to that boundary.
- 3300 The cursor shall be placed on the current line and relative to the current column  
 3301 as specified by each command described in the following subclauses.
- 3302 In open mode, if the current line is not already displayed, then it shall be  
 3303 displayed.
- 3304 In visual mode, if the current line is not displayed, then the lines that are  
 3305 displayed shall be expanded, scrolled or redrawn to cause an unspecified portion  
 3306 of the current line to be displayed. If the screen is redrawn, no more than the C  
 3307 number of logical lines specified by the value of the `window` edit option shall be C  
 3308 displayed (unless the current line cannot be completely displayed in the number C  
 3309 of logical lines specified by the `window` edit option) and the current line shall be C  
 3310 positioned as close to the center of the displayed lines as possible (within the con-  
 3311 straints imposed by the distance of the line from the beginning or end of the edit  
 3312 buffer). If the current line is before the first line in the display and the screen is  
 3313 scrolled, an unspecified portion of the current line shall be placed on the first line  
 3314 of the display. If the current line is after the last line in the display and the  
 3315 screen is scrolled, an unspecified portion of the current line shall be placed on the  
 3316 last line of the display.
- 3317 In visual mode, if a line from the edit buffer (other than the current line) does not  
 3318 entirely fit into the lines at the bottom of the display that are available for its  
 3319 presentation, the editor may choose not to display any portion of the line. The  
 3320 lines of the display that do not contain text from the edit buffer for this reason  
 3321 shall each consist of a single @ character.
- 3322 In visual mode, the editor may choose for unspecified reasons to not update lines  
 3323 in the display to correspond to the underlying edit buffer text. The lines of the  
 3324 display that do not correctly correspond to text from the edit buffer for this reason  
 3325 shall consist of a single @ character, and the `<control-R>` command shall cause  
 3326 the editor to update the screen to correctly represent the edit buffer.
- 3327 Open and visual mode commands that set the current column set it to a column  
 3328 position in the display, and not a character position in the line. In this case, how-  
 3329 ever, the column position in the display shall be calculated for a infinite width  
 3330 display; e.g., the column related to a character that is part of a line that has been  
 3331 folded onto additional screen lines will be offset from the screen column where the  
 3332 physical line begins, not from the beginning of a particular screen line.
- 3333 The physical cursor column in the display is based on the value of the current  
 3334 column, as follows, with each rule applied in turn:
- 3335 (1) If the current column is after the last screen column used by the  
 3336 displayed line, the physical cursor column shall be set to the last screen C  
 3337 column occupied by the last character in the current line; otherwise, the  
 3338 physical cursor column shall be set to the current column.
  - 3339 (2) If the character of which some portion is displayed in the screen column  
 3340 specified by the physical cursor column requires more than a single

- 3341 screen column:
- 3342 (a) If in text input mode, the physical cursor column shall be adjusted  
3343 to the first screen column in which any portion of that character is  
3344 displayed.
- 3345 (b) Otherwise, the physical cursor column shall be adjusted to the last C  
3346 screen column in which any portion of that character is displayed.
- 3347 The current column shall not be changed by these adjustments to the physical  
3348 cursor column.
- 3349 If an error occurs during the parsing or execution of a `vi` command: C
- 3350 — The terminal shall be alerted. Execution of the `vi` command shall stop, C  
3351 and the cursor (e.g., the current line and column) shall not be further C  
3352 modified. C
  - 3353 — Unless otherwise specified by the following command subclauses, it is C  
3354 unspecified if an informational message shall be displayed. C
  - 3355 — Any partially entered `vi` command shall be discarded. C
  - 3356 — If the `vi` command resulted from a map expansion, all characters from that C  
3357 map expansion shall be discarded, except as otherwise specified by the `map` C  
3358 command (see 5.10.7.5.14). C
  - 3359 — If the `vi` command resulted from the execution of a buffer, no further com- C  
3360 mands caused by the execution of the buffer shall be executed. C
- 3361 The following subclauses describe command characters entered during command  
3362 mode.
- 3363 **5.35.7.2.1 <control-B>**
- 3364 *Synopsis:* [*count*] <control-B>
- 3365 If in open mode, the <control-B> command shall behave identically to the `z` C  
3366 command (see 5.35.7.2.84). Otherwise, if the current line is the first line of the C  
3367 edit buffer, it shall be an error.
- 3368 If the `window edit` option is less than 3, display a screen where the last line of the  
3369 display shall be some portion of: C
- 3370  $(\textit{current first line}) - 1$
- 3371 otherwise, display a screen where the first line of the display shall be some por- C  
3372 tion of: C
- 3373  $(\textit{current first line}) - \textit{count} \times ((\textit{window edit option}) - 2)$  C
- 3374 If this calculation would result in a line that is before the first line of the edit  
3375 buffer, the first line of the display shall display some portion of the first line of the  
3376 edit buffer.
- 3377 *Current line:* If no lines from the previous display remain on the screen, set to the  
3378 last line of the display; otherwise, set to (*line - the number of new lines displayed*  
3379 *on this screen*).

- 3380 *Current column*: Set to nonblank.
- 3381 **5.35.7.2.2** <control-D>
- 3382 *Synopsis*: [*count*] <control-D>
- 3383 If the current line is the last line of the edit buffer, it shall be an error.
- 3384 If no *count* is specified, *count* shall default to the *count* associated with the previ-  
 3385 ous <control-D> or <control-U> command. If there was no previous  
 3386 <control-D> or <control-U> command, *count* shall default to the value of the  
 3387 scroll edit option.
- 3388 If in open mode: Write lines starting with the line after the current line, until  
 3389 *count* lines or the last line of the file have been written.
- 3390 *Current line*: If the current line + *count* is past the last line of the edit buffer, set  
 3391 to the last line of the edit buffer; otherwise, set to the current line + *count*.
- 3392 *Current column*: Set to nonblank.
- 3393 **5.35.7.2.3** <control-E>
- 3394 *Synopsis*: [*count*] <control-E>
- 3395 C
- 3396 Display the line *count* lines after the last line currently displayed.
- 3397 If the last line of the edit buffer is displayed, it shall be an error. If there is no  
 3398 line *count* lines after the last line currently displayed, the last line of the display  
 3399 shall display some portion of the last line of the edit buffer.
- 3400 *Current line*: Unchanged if the previous current character is displayed; otherwise, C  
 3401 set to the first line displayed.
- 3402 *Current column*: Unchanged.
- 3403 **5.35.7.2.4** <control-F>
- 3404 *Synopsis*: [*count*] <control-F>
- 3405 If in open mode, the <control-F> command shall behave identically to the z C  
 3406 command (see 5.35.7.2.84). C
- 3407 Otherwise, if the current line is the last line of the edit buffer, it shall be an error. C
- 3408 If the window edit option is less than 3, display a screen where the first line of the  
 3409 display shall be some portion of: C
- 3410  $(\textit{current last line}) + 1$
- 3411 otherwise, display a screen where the first line of the display shall be some por- C  
 3412 tion of: C
- 3413  $(\textit{current first line}) + \textit{count} \times ((\textit{window edit option}) - 2)$
- 3414 If this calculation would result in a line that is after the last line of the edit  
 3415 buffer, the last line of the display shall display some portion of the last line of the

3416 edit buffer.

3417 *Current line*: If no lines from the previous display remain on the screen, set to the  
3418 first line of the display; otherwise, set to (line + the number of new lines displayed  
3419 on this screen).

3420 *Current column*: Set to nonblank.

3421 **5.35.7.2.5** <control-G>

3422 *Synopsis*: <control-G>

3423 This command shall be equivalent to the `ex file` command (see 5.10.7.5.9).

3424 **5.35.7.2.6** <control-H>

3425 *Synopsis*: [*count*] <control-H>

3426 *Synopsis*: [*count*] h

3427 *Synopsis*: the current *erase* character (see `stty` in 4.59)

3428 If there are no characters before the current character on the current line, it shall  
3429 be an error. If there are less than *count* previous characters on the current line,  
3430 *count* shall be adjusted to the number of previous characters on the line.

3431 If used as a motion command:

3432 (1) The text region shall be from the character before the starting cursor up  
3433 to and including the *count*-th character before the starting cursor.

3434 (2) Any text copied to a buffer shall be in character mode.

3435 If not used as a motion command:

3436 *Current line*: Unchanged.

3437 *Current column*: Set to (*column* – the number of columns occupied by the *count*  
3438 characters ending in the previous current column).

3439 **5.35.7.2.7** <newline>

3440 *Synopsis*: [*count*] <newline>

3441 *Synopsis*: [*count*] <control-J>

3442 *Synopsis*: [*count*] <control-M>

3443 *Synopsis*: [*count*] <control-N>

3444 *Synopsis*: [*count*] j

3445 *Synopsis*: [*count*] <carriage-return>

3446 *Synopsis*: [*count*] +

3447 If there are less than *count* lines after the current line in the edit buffer, it shall  
3448 be an error.

3449 If used as a motion command:

3450 (1) The text region shall include the starting line and the next *count* – 1  
3451 lines.

C  
C

3452 (2) Any text copied to a buffer shall be in line mode.

3453 If not used as a motion command:

3454 *Current line*: Set to current line + *count*.

3455 *Current column*: Set to nonblank for the <carriage-return>, <control-M>, C  
 3456 and + commands; otherwise, unchanged. C

3457 **5.35.7.2.8** <control-L>

3458 *Synopsis*: <control-L>

3459 If in open mode, clear the screen and redisplay the current line.

3460 Otherwise, clear and redisplay the screen.

3461 *Current line*: Unchanged.

3462 *Current column*: Unchanged.

3463 **5.35.7.2.9** <control-P>

3464 *Synopsis*: [*count*] <control-P>

3465 *Synopsis*: [*count*] k

3466 *Synopsis*: [*count*] -

3467 If there are less than *count* lines before the current line in the edit buffer, it shall  
 3468 be an error.

3469 If used as a motion command:

3470 (1) The text region shall include the starting line and the previous *count*  
 3471 lines.

3472 (2) Any text copied to a buffer shall be in line mode.

3473 If not used as a motion command:

3474 *Current line*: Set to current line - *count*.

3475 *Current column*: Set to nonblank for the - command; otherwise, unchanged.

3476 **5.35.7.2.10** <control-R>

3477 *Synopsis*: <control-R>

3478 If any lines have been deleted from the logical screen in visual mode, and flagged  
 3479 as deleted on the terminal using the @ convention (see 5.35.7), they shall be  
 3480 redisplayed to match the contents of the edit buffer.

3481 It is unspecified if lines flagged with @ because they do not fit on the terminal  
 3482 display shall be affected.

3483 *Current line*: Unchanged.

3484 *Current column*: Unchanged.

3485 **5.35.7.2.11** <control-U>3486 *Synopsis:* [*count*] <control-U>

3487 If the current line is the first line of the edit buffer, it shall be an error.

3488 If no *count* is specified, *count* shall default to the *count* associated with the previ-  
 3489 ous <control-D> or <control-U> command. If there was no previous  
 3490 <control-D> or <control-U> command, *count* shall default to the value of the  
 3491 scroll edit option.

3492 *Current line:* If *count* is greater than the current line, set to 1; otherwise, set to  
 3493 the current line – *count*.

3494 *Current column:* Set to nonblank.3495 **5.35.7.2.12** <control-Y>3496 *Synopsis:* [*count*] <control-Y>

3497

C

3498 Display the line *count* lines before the first line currently displayed.

3499 If the current line is the first line of the edit buffer, it shall be an error. If this  
 3500 calculation would result in a line that is before the first line of the edit buffer, the  
 3501 first line of the display shall display some portion of the first line of the edit  
 3502 buffer.

3503 *Current line:* Unchanged if the previous current character is displayed; otherwise, C  
 3504 set to the first line displayed.

3505 *Current column:* Unchanged.3506 **5.35.7.2.13** <control-^>3507 *Synopsis:* <control-^>

3508 This command shall be equivalent to the `ex edit` command (see 5.10.7.5.8), with  
 3509 the alternate pathname as its argument.

3510 **5.35.7.2.14** <ESC>3511 *Synopsis:* <ESC>

3512 If a partial `vi` command (as defined by at least one, non-*count* character) has been  
 3513 entered, discard the *count* and the command character(s). C

3514 Otherwise, if no command characters have been entered, and the <ESC> was the C  
 3515 result of a map expansion, the terminal shall be alerted and the <ESC> character C  
 3516 shall be discarded, but it shall not be an error. C

3517 Otherwise, it shall be an error. C

3518 *Current line:* Unchanged.3519 *Current column:* Unchanged.

3520 **5.35.7.2.15** <control-]>3521 *Synopsis:* <control-]>

3522 If the current character is not a word or &lt;blank&gt; character, it shall be an error.

3523 This command shall be equivalent to the `ex tag` command (see 5.10.7.5.32), with  
3524 the argument to that command defined as follows:

3525 If the current character is a &lt;blank&gt;

3526 (1) Skip all &lt;blank&gt; characters after the cursor up to the end of the line.

3527 (2) If the end of the line is reached, it shall be an error.

3528 Then, the argument to the `ex tag` command shall be the current character and all  
3529 subsequent characters, up to the first nonword character or the end of the line.3530 **5.35.7.2.16** <space>3531 *Synopsis:* [*count*] <space>3532 *Synopsis:* [*count*] 1 (ell)3533 If there are less than *count* characters after the cursor on the current line, *count*  
3534 shall be adjusted to the number of characters after the cursor on the line.

3535 If used as a motion command:

3536 (1) If the current or *count*-th character after the cursor is the last character  
3537 in the line, the text region shall be comprised of the current character up  
3538 to and including the last character in the line. Otherwise, the text region  
3539 shall be from the current character up to, but not including, the *count*-th  
3540 character after the cursor.

3541 (2) Any text copied to a buffer shall be in character mode.

3542 If not used as a motion command:

3543 If there are no characters after the current character on the current line, it shall  
3544 be an error.3545 *Current line:* Unchanged.3546 *Current column:* Set to the last column that displays any portion of the *count*-th  
3547 character after the current character.3548 **5.35.7.2.17** !3549 *Synopsis:* [*count*] ! *motion shell-command(s)* <newline>3550 If the *motion* command is the ! command repeated:3551 (1) If the edit buffer is empty and no *count* was supplied, the command shall  
3552 be the equivalent of the `ex :read !` command, with the text input, and  
3553 no text shall be copied to any buffer.

3554 (2) Otherwise:

3555 (a) If there are less than *count* – 1 lines after the current line in the  
3556 edit buffer, it shall be an error.

3557 (b) The text region shall be from the current line up to and including  
3558 the next *count* – 1 lines.

3559 Otherwise, the text region shall be the lines in which any character of the text  
3560 region specified by the motion command appear.

3561 Any text copied to a buffer shall be in line mode.

3562 This command shall be equivalent to the `ex !` command (see 5.10.7.5.42) for the  
3563 specified lines.

### 3564 **5.35.7.2.18** §

3565 *Synopsis:* [*count*] §

3566 It shall be an error if there are less than (*count* – 1) lines after the current line in  
3567 the edit buffer.

3568 If used as a motion command:

3569 (1) If *count* is 1:

3570 (a) It shall be an error if the line is empty.

3571 (b) Otherwise, the text region shall consist of all characters from the  
3572 starting cursor to the last character in the line, inclusive, and any  
3573 text copied to a buffer shall be in character mode. C

3574 (2) Otherwise, if the starting cursor position is at or before the first non-  
3575 blank in the line, the text region shall consist of the current and the next  
3576 *count* – 1 lines, and any text saved to a buffer shall be in line mode.

3577 (3) Otherwise, the text region shall consist of all characters from the starting  
3578 cursor to the last character in the line that is *count* – 1 lines forward  
3579 from the current line, and any text copied to a buffer shall be in character  
3580 mode.

3581 If not used as a motion command:

3582 *Current line:* Set to current line + *count* – 1.

3583 *Current column:* The current column is set to the last screen column of the last  
3584 character in the line, or column position 1 if the line is empty.

3585 The current column shall be adjusted to be on the last screen column of the last  
3586 character of the current line as subsequent commands change the current line,  
3587 until a command changes the current column.



3588 **5.35.7.2.19** %

3589 *Synopsis:* %

3590 If the character at the current position is not a parenthesis, bracket, or curly  
3591 brace, search forwards in the line to the first one of those characters. If no such  
3592 character is found, it shall be an error.

3593 The matching character shall be the parenthesis, bracket, or curly brace matching  
3594 the parenthesis, bracket, or curly brace, respectively, that was at the current posi-  
3595 tion or that was found on the current line.

3596 Matching shall be determined as follows, for a open parenthesis:

- 3597 (1) Set a counter to 1.
- 3598 (2) Search forwards until a parenthesis is found or the end of the edit buffer  
3599 is reached.
- 3600 (3) If the end of the edit buffer is reached, it shall be an error.
- 3601 (4) If a open parenthesis is found, increment the counter by 1.
- 3602 (5) If a close parenthesis is found, decrement the counter by 1.
- 3603 (6) If the counter is zero, the current character is the matching character.

3604 Matching for a close parenthesis shall be equivalent, except that the search shall  
3605 be backwards, from the starting character to the beginning of the buffer, a close  
3606 parenthesis shall increment the counter by 1, and a open parenthesis shall decre-  
3607 ment the counter by 1.

3608 Matching for brackets and curly braces shall be equivalent, except that searching  
3609 shall be done for open and close brackets or open and close curly braces.

3610 It is implementation-defined if other characters are searched for and matched as  
3611 well.

3612 If used as a motion command:

- 3613 (1) If the matching cursor was after the starting cursor in the edit buffer,  
3614 and the starting cursor position was at or before the first nonblank in the  
3615 starting line, and the matching cursor position was at or after the last  
3616 nonblank in the matching line, the text region shall consist of the current  
3617 line to the matching line, inclusive, and any text copied to a buffer shall  
3618 be in line mode.
- 3619 (2) If the matching cursor was before the starting cursor in the edit buffer,  
3620 and the starting cursor position was at or after the last nonblank in the  
3621 starting line, and the matching cursor position was at or before the first  
3622 nonblank in the matching line, the text region shall consist of the current  
3623 line to the matching line, inclusive, and any text copied to a buffer shall  
3624 be in line mode.
- 3625 (3) Otherwise, the text region shall consist of the starting character to the  
3626 matching character, inclusive, and any text copied to a buffer shall be in  
3627 character mode.

3628 If not used as a motion command:

3629 *Current line*: Set to the line where the matching character is located.

3630 *Current column*: Set to the last column where any portion of the matching character is displayed.  
3631

3632 **5.35.7.2.20** &

3633 *Synopsis*: &

3634 This command shall be equivalent to the `ex &` command with the current line as C  
3635 its addresses, and without *options*, *count*, or *flags* (see 5.10.7.5.27). C

3636 **5.35.7.2.21** ' ,

3637 *Synopsis*: ' , character

3638 It shall be an error if the marked line is no longer in the edit buffer.

3639 If used as a motion command:

3640 (1) If the starting cursor is after the marked cursor, then the locations of the  
3641 starting cursor and the marked cursor in the edit buffer shall be logically  
3642 swapped.

3643 (2) The text region shall consist of the starting line up to and including the  
3644 marked line, and any text copied to a buffer shall be in line mode.

3645 If not used as a motion command:

3646 *Current line*: Set to the line referenced by the mark.

3647 *Current column*: Set to nonblank.

3648 **5.35.7.2.22** \

3649 *Synopsis*: \ character

3650 It shall be an error if the marked line is no longer in the edit buffer. If the  
3651 marked line no longer contains a character in the saved numbered character posi-  
3652 tion, it shall be as if the marked position is the first nonblank.

3653 If used as a motion command:

3654 (1) It shall be an error if the marked cursor references the same character in  
3655 the edit buffer as the starting cursor.

3656 (2) If the starting cursor is after the marked cursor, then the locations of the  
3657 starting cursor and the marked cursor in the edit buffer shall be logically  
3658 swapped.

3659 (3) If the starting line is empty or the starting cursor is at or before the first  
3660 nonblank character of the starting line, and the marked cursor line is  
3661 empty or the marked cursor references the first character of the marked  
3662 cursor line, the text region shall consist of all lines containing characters  
3663 from the starting cursor to the line before the marked cursor line,  
3664 inclusive, and any text copied to a buffer shall be in line mode.

3665 (4) Otherwise, if the marked cursor line is empty or the marked cursor refer-  
 3666 ences a character at or before the first nonblank of the marked cursor  
 3667 line, the region of text shall be from the starting cursor to the last charac-  
 3668 ter of the line before the marked cursor line, inclusive, and any text  
 3669 copied to a buffer shall be in character mode.

3670 (5) Otherwise, the region of text shall be from the starting cursor (inclusive),  
 3671 to the marked cursor (exclusive), and any text copied to a buffer shall be  
 3672 in character mode.

3673 If not used as a motion command:

3674 *Current line*: Set to the line referenced by the mark.

3675 *Current column*: Set to the last column in which any portion of the character  
 3676 referenced by the mark is displayed.

3677 **5.35.7.2.23** [ [

3678 *Synopsis*: [*count*] [ [

3679 Move the cursor backward through the edit buffer to the first character of the pre- C  
 3680 vious section boundary, *count* times. C

3681 If used as a motion command:

3682 (1) If the starting cursor was at the first character of the starting line or the  
 3683 starting line was empty, and the first character of the boundary was the  
 3684 first character of the boundary line, the text region shall consist of the  
 3685 current line up to and including the line where the *count*-th next bound-  
 3686 ary starts, and any text copied to a buffer shall be in line mode.

3687 (2) If the boundary was the last line of the edit buffer or the last character of  
 3688 the last line of the edit buffer, the text region shall consist of the last  
 3689 character in the edit buffer up to and including the starting character,  
 3690 and any text saved to a buffer shall be in character mode.

3691 (3) Otherwise, the text region shall consist of the starting character up to  
 3692 but not including the first character in the *count*-th next boundary, and  
 3693 any text copied to a buffer shall be in character mode.

3694 If not used as a motion command:

3695 *Current line*: Set to the line where the *count*-th next boundary in the edit buffer  
 3696 starts.

3697 *Current column*: Set to the last column in which any portion of the first character  
 3698 of the *count*-th next boundary is displayed, or column position 1 if the line is  
 3699 empty.

3700 **5.35.7.2.24** ] ]3701 *Synopsis:* [ *count* ] ]3702 Move the cursor forward through the edit buffer to the first character of the next C  
3703 section boundary, *count* times. C

3704 If used as a motion command:

3705 (1) If the starting cursor was at the first character of the starting line or the  
3706 starting line was empty, and the first character of the boundary was the  
3707 first character of the boundary line, the text region shall consist of the  
3708 current line up to and including the line where the *count*-th previous  
3709 boundary starts, and any text copied to a buffer shall be in line mode.3710 (2) If the boundary was the first line of the edit buffer, the text region shall  
3711 consist of the first character in the edit buffer up to but not including the  
3712 starting character, and any text copied to a buffer shall be in character  
3713 mode.3714 (3) Otherwise, the text region shall consist of the first character in the  
3715 *count*-th previous section boundary up to but not including the starting  
3716 character, and any text copied to a buffer shall be in character mode.

3717 If not used as a motion command:

3718 *Current line:* Set to the line where the *count*-th previous boundary in the edit  
3719 buffer starts.3720 *Current column:* Set to the last column in which any portion of the first character  
3721 of the *count*-th previous boundary is displayed, or column position 1 if the line is  
3722 empty.3723 **5.35.7.2.25** ^3724 *Synopsis:* ^

3725 If used as a motion command:

3726 (1) If the line has no nonblank characters, or if the cursor is at the first non-  
3727 blank character of the line, it shall be an error.3728 (2) If the cursor is before the first nonblank character of the line, the text  
3729 region shall be comprised of the current character, up to, but not includ-  
3730 ing, the first nonblank character of the line.3731 (3) If the cursor is after the first nonblank character of the line, the text  
3732 region shall be from the character before the starting cursor up to and  
3733 including the first nonblank character of the line.

3734 (4) Any text copied to a buffer shall be in character mode.

3735 If not used as a motion command:

3736 *Current line:* Unchanged.3737 *Current column:* Set to nonblank.

3738 **5.35.7.2.26** \_3739 *Synopsis:* [*count*] \_

3740 If there are less than *count* – 1 lines after the current line in the edit buffer, it  
3741 shall be an error.

3742 If used as a motion command:

- 3743 (1) If *count* is less than 2, the text region shall be the current line. C
- 3744 (2) Otherwise, the text region shall include the starting line and the next  
3745 *count* – 1 lines.
- 3746 (3) Any text copied to a buffer shall be in line mode.

3747 If not used as a motion command:

3748 *Current line:* Set to current line + *count* – 1.3749 *Current column:* Set to nonblank.3750 **5.35.7.2.27** (3751 *Synopsis:* [*count*] (

3752 This command shall be equivalent to the [[] command, with the exception that  
3753 sentence boundaries shall be used instead of section boundaries.

3754 **5.35.7.2.28** )3755 *Synopsis:* [*count*] )

3756 This command shall be equivalent to the []] command, with the exception that  
3757 sentence boundaries shall be used instead of section boundaries.

3758 **5.35.7.2.29** {3759 *Synopsis:* [*count*] {

3760 This command shall be equivalent to the [[] command, with the exception that  
3761 paragraph boundaries shall be used instead of section boundaries.

3762 **5.35.7.2.30** }3763 *Synopsis:* [*count*] }

3764 This command shall be equivalent to the []] command, with the exception that  
3765 paragraph boundaries shall be used instead of section boundaries.

3766 **5.35.7.2.31** |3767 *Synopsis:* [*count*] |

3768 For the purposes of this command, lines that are too long for the current display  
3769 and that have been folded shall be treated as having a single, 1-based, number of  
3770 columns.

3771 If there are less than *count* columns in which characters from the current line are  
 3772 displayed on the screen, *count* shall be adjusted to be the last column in which  
 3773 any portion of the line is displayed on the screen.

3774 If used as a motion character:

3775 (1) If the line is empty, or the cursor character is the same as the character  
 3776 on the *count*-th column of the line, it shall be an error.

3777 (2) If the cursor is before the *count*-th column of the line, the text region  
 3778 shall be comprised of the current character, up to but not including the  
 3779 character on the *count*-th column of the line.

3780 (3) If the cursor is after the *count*-th column of the line, the text region shall  
 3781 be from the character before the starting cursor up to and including the  
 3782 character on the *count*-th column of the line.

3783 (4) Any text copied to a buffer shall be in character mode.

3784 If not used as a motion character:

3785 *Current line*: Unchanged.

3786 *Current column*: Set to the last column in which any portion of the character that  
 3787 is displayed in the *count* column of the line is displayed.

3788 **5.35.7.2.32** ,

3789 *Synopsis*: [*count*] ,

3790 If the last F, f, T, or t command was F, f, T, or t, this command shall be C  
 3791 equivalent to an f, F, t, or T command, respectively, with the specified *count* and C  
 3792 the same search character. C

3793 If there was no previous F, f, T, or t command, it shall be an error. C

3794 **5.35.7.2.33** .

3795 *Synopsis*: [*count*] .

3796 Repeat the last !, <, >, A, C, D, I, J, O, P, R, S, X, Y, a, c, d, i, o, p, r, s, x, y, or ~  
 3797 command. It shall be an error if none of these commands have been executed.  
 3798 Commands (other than commands that enter text input mode) executed as a  
 3799 result of map expansions, shall not change the value of the last repeatable  
 3800 command.

3801 Repeated commands with associated motion commands shall repeat the motion  
 3802 command as well; however, any specified *count* shall replace the *count*(s) that  
 3803 were originally specified to the repeated command or its associated motion com- C  
 3804 mand. C

3805 If the motion component of the repeated command is f, F, t, or T, the repeated  
 3806 command shall not set the remembered search character for the ; and , com-  
 3807 mands.

3808 If the repeated command is p or P, and the buffer associated with that command  
 3809 was a numeric buffer named with a number less than 9, the buffer associated

3810 with the repeated command shall be set to be the buffer named by the name of the  
3811 previous buffer logically incremented by 1. C

3812 If the repeated character is a text input command, the input text associated with  
3813 that command is repeated literally:

3814 — Input characters are neither macro or abbreviation expanded.

3815 — Input characters are not interpreted in any special way with the exception  
3816 that <newline> and <carriage-return> behave as described in  
3817 5.35.7.3.4, and <control-T> behaves as described in 5.35.7.3.5.

3818 *Current line*: Set as described for the repeated command.

3819 *Current column*: Set as described for the repeated command.

### 3820 **5.35.7.2.34** /

3821 *Synopsis*: /

3822 If the input line contains no characters, it shall be equivalent to a line containing  
3823 only the last RE encountered. The enhanced REs supported by vi are described in  
3824 ex; see 5.10.7.6.

3825 Otherwise, the line shall be interpreted as one or more REs, optionally followed by  
3826 an address offset or a vi z command.

3827 If the RE is not the last RE on the line, or if a line offset or z command is specified,  
3828 the RE shall be terminated by an unescaped / character, which shall not be used  
3829 as part of the RE. If the RE is not the first RE on the line, it shall be preceded by  
3830 zero or more <blank> characters, a semicolon, zero or more <blank> characters,  
3831 and a leading / character, which shall not be interpreted as part of the RE. It  
3832 shall be an error to precede any RE with any characters other than these.

3833 Each search shall begin from the character after the first character of the last  
3834 match (or, if it is the first search, after the cursor). If the wrapscan edit option is  
3835 set, the search shall continue to the character before the starting cursor charac- C  
3836 ter; otherwise, to the end of the edit buffer. It shall be an error if any search fails C  
3837 to find a match, and an informational message to this effect shall be displayed. C

3838 An optional address offset (see 5.10.7.2) can be specified after the last RE by  
3839 including a trailing / character after the RE and specifying the address offset.  
3840 This offset will be from the line containing the match for the last RE specified. It  
3841 shall be an error if the line offset would indicate a line address less than 1 or  
3842 greater than the last line in the edit buffer. An address offset of zero shall be sup-  
3843 ported. It shall be an error to follow the address offset with any other characters  
3844 than <blank>s.

3845 If not used as a motion command, an optional z command (see 5.35.7.2.84) can be  
3846 specified after the last RE by including a trailing / character after the RE, zero or  
3847 more <blank> characters, a z, zero or more <blank> characters, an optional new  
3848 window edit option value, zero or more <blank> characters, and a location char-  
3849 acter. The effect shall be as if the z command was executed after the /  
3850 command(s). It shall be an error to follow the z command with any other charac-  
3851 ters than <blank>s.

- 3852 The remembered search direction shall be set to forward.
- 3853 C
- 3854 If used as a motion command:
- 3855 (1) It shall be an error if the last match references the same character in the C  
 3856 edit buffer as the starting cursor. C
- 3857 (2) If any address offset is specified, the last match shall be adjusted by the C  
 3858 specified offset as described previously. C
- 3859 (3) If the starting cursor is after the last match, then the locations of the  
 3860 starting cursor and the last match in the edit buffer shall be logically  
 3861 swapped.
- 3862 (4) If any address offset is specified, the text region shall consist of all lines  
 3863 containing characters from the starting cursor to the last match line,  
 3864 inclusive, and any text copied to a buffer shall be in line mode.
- 3865 (5) Otherwise, if the starting line is empty or the starting cursor is at or  
 3866 before the first nonblank character of the starting line, and the last  
 3867 match line is empty or the last match starts at the first character of the  
 3868 last match line, the text region shall consist of all lines containing char-  
 3869 acters from the starting cursor to the line before the last match line,  
 3870 inclusive, and any text copied to a buffer shall be in line mode.
- 3871 (6) Otherwise, if the last match line is empty or the last match begins at a  
 3872 character at or before the first nonblank of the last match line, the region  
 3873 of text shall be from the current cursor to the last character of the line  
 3874 before the last match line, inclusive, and any text copied to a buffer shall  
 3875 be in character mode.
- 3876 (7) Otherwise, the region of text shall be from the current cursor (inclusive),  
 3877 to the first character of the last match (exclusive), and any text copied to  
 3878 a buffer shall be in character mode.
- 3879 If not used as a motion command:
- 3880 *Current line*: If a match is found, set to the last matched line plus the address  
 3881 offset, if any; otherwise, unchanged.
- 3882 *Current column*: Set to the last column on which any portion of the first character  
 3883 in the last matched string is displayed, if a match is found, otherwise, unchanged.
- 3884 **5.35.7.2.35 0**
- 3885 *Synopsis*: 0 (zero)
- 3886 The character 0 shall not be interpreted as a command if it is immediately pre- C  
 3887 ceded by a digit. C
- 3888 If used as a motion command:
- 3889 (1) If the cursor character is the first character in the line, it shall be an  
 3890 error.



3891 (2) The text region shall be from the character before the cursor character up  
3892 to and including the first character in the line.

3893 (3) Any text copied to a buffer shall be in character mode.

3894 If not used as a motion command:

3895 *Current line*: Unchanged.

3896 *Current column*: The last column in which any portion of the first character in the  
3897 line is displayed, or if the line is empty, unchanged.

3898 **5.35.7.2.36 :**

3899 *Synopsis*: :

3900 Execute one or more `ex` command(s).

3901

C

3902 If any portion of the screen other than the last line of the screen was overwritten  
3903 by any `ex` command (except `shell`), `vi` shall display a message indicating that it  
3904 is waiting for an input from the user, and shall then read a character. This action  
3905 may also be taken for other, unspecified reasons.

3906 If the next character entered is a `:`, another `ex` command shall be accepted and  
3907 executed. Any other character shall cause the screen to be refreshed and `vi` shall  
3908 return to command mode.

3909 *Current line*: As specified for the `ex` command(s).

3910 *Current column*: As specified for the `ex` command(s).

3911 **5.35.7.2.37 ;**

3912 *Synopsis*: [*count*] ;

3913 This command shall be equivalent to the last `F`, `f`, `T`, or `t` command, with the  
3914 specified *count*, and with the same search character used for the last `F`, `f`, `T`, or `t`  
3915 command.

3916 If there was no previous `F`, `f`, `T`, or `t` command, it shall be an error.

C

3917 **5.35.7.2.38 <**

3918 *Synopsis*: [*count*] < *motion*

3919 If the *motion* command is the `<` command repeated:

3920 (1) If there are less than *count* - 1 lines after the current line in the edit  
3921 buffer, it shall be an error.

3922 (2) The text region shall be from the current line, up to and including the  
3923 next *count* - 1 lines.

3924 Shift any line in the text region specified by the *count* and *motion* command one  
3925 *shiftwidth* (see 5.10.7.8.19) toward the start of the line, as described by the `ex`  
3926 `<` command (see 5.10.7.5.42). The unshifted lines shall be copied to the unnamed

3927 buffer in line mode.

3928 *Current line*: If the motion was from the current cursor position toward the end of  
3929 the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that  
3930 is part of the text region specified by the motion command.

3931 *Current column*: Set to nonblank.

3932 **5.35.7.2.39** >

3933 *Synopsis*: [*count*] > *motion*

3934 If the *motion* command is the > command repeated:

3935 (1) If there are less than *count* – 1 lines after the current line in the edit  
3936 buffer, it shall be an error.

3937 (2) The text region shall be from the current line, up to and including the  
3938 next *count* – 1 lines.

3939 Shift any line with characters in the text region specified by the *count* and motion  
3940 command one *shiftwidth* (see 5.10.7.8.19) away from the start of the line, as  
3941 described by the *ex* > command (see 5.10.7.5.43). The unshifted lines shall be  
3942 copied into the unnamed buffer in line mode.

3943 *Current line*: If the motion was from the current cursor position toward the end of  
3944 the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that  
3945 is part of the text region specified by the motion command.

3946 *Current column*: Set to nonblank.

3947 **5.35.7.2.40** ?

3948 *Synopsis*: ?

3949 The ? command shall be equivalent to the / command (see 5.35.7.2.34) with the  
3950 following exceptions:

3951 (1) The input prompt shall be a ?.

3952 (2) Each search shall begin from the character before the first character of  
3953 the last match (or, if it is the first search, the character before the cursor  
3954 character).

3955 (3) The search direction shall be from the cursor toward the beginning of the  
3956 edit buffer, and the *wraps* edit option shall affect whether the search  
3957 wraps to the end of the edit buffer and continues.

3958

C

3959 (4) The remembered search direction shall be set to backward.

3960 **5.35.7.2.41 @**3961 *Synopsis:* [*count*] @*buffer*3962 If the *buffer* is specified as @, the last buffer executed shall be used. If no previous  
3963 buffer has been executed, it shall be an error.3964 Behave as if the contents of the named buffer were entered as standard input.  
3965 After each line of a line-mode buffer, and all but the last line of a character mode  
3966 buffer, behave as if a <newline> character were entered as standard input.3967 If an error occurs during this process, an error message shall be written, and no C  
3968 more characters resulting from the execution of this command shall be processed.3969 If a *count* is specified, behave as if that count were entered as user input before  
3970 the characters from the @ buffer were entered.3971 *Current line:* As specified for the individual commands. C3972 *Current column:* As specified for the individual commands. C3973 **5.35.7.2.42 ~**3974 *Synopsis:* [*count*] ~3975 Reverse the case of the current character and the next *count* – 1 characters, such  
3976 that lowercase characters that have uppercase counterparts shall be changed to  
3977 uppercase characters, and uppercase characters that have lowercase counterparts  
3978 shall be changed to lowercase characters, as prescribed by the current locale. No  
3979 other characters shall be affected by this command.3980 If there are less than *count* – 1 characters after the cursor in the edit buffer, *count*  
3981 shall be adjusted to the number of characters after the cursor in the edit buffer C  
3982 minus 1. C3983 For the purposes of this command, the next character after the last character on  
3984 the line shall be the next character in the edit buffer.3985 *Current line:* Set to the line including the (*count* – 1)-th character after the cursor.3986 *Current column:* Set to the last column in which any portion of the (*count*–1)-th  
3987 character after the cursor is displayed.3988 **5.35.7.2.43 a**3989 *Synopsis:* [*count*] a3990 Enter text input mode after the current cursor position. No characters already in C  
3991 the edit buffer shall be affected by this command. A *count* shall cause the input C  
3992 text to be appended *count* – 1 more times to the end of the input.3993 *Current line/column:* As specified for the text input commands; see 5.35.7.3.

3994 **5.35.7.2.44 a**3995 *Synopsis:* [*count*] A3996 This command shall be equivalent to the `vi` commands `§[count]a` (see  
3997 5.35.7.2.43).3998 **5.35.7.2.45 b**3999 *Synopsis:* [*count*] b4000 With the exception that words are used as the delimiter instead of bigwords, this  
4001 command shall be equivalent to the `B` command; see 5.35.7.2.46.4002 **5.35.7.2.46 B**4003 *Synopsis:* [*count*] B4004 If the edit buffer is empty or the cursor is on the first character of the edit buffer,  
4005 it shall be an error. If less than *count* bigwords begin between the cursor and the  
4006 start of the edit buffer, *count* shall be adjusted to the number of bigword begin-  
4007 nings between the cursor and the start of the edit buffer.

4008 If used as a motion command:

4009 (1) The text region shall be from the first character of the *count*-th previous  
4010 bigword beginning up to but not including the cursor character.

4011 (2) Any text copied to a buffer shall be in character mode.

4012 If not used as a motion command:

4013 *Current line:* Set to the line containing the *current column*.4014 *Current column:* Set to the last column upon which any part of the first character  
4015 of the *count*-th previous bigword is displayed.4016 **5.35.7.2.47 c**4017 *Synopsis:* [*buffer*] [*count*] *c motion*4018 If the *motion* command is the `c` command repeated:

4019 (1) The buffer text shall be in line mode.

4020 (2) If there are less than *count* - 1 lines after the current line in the edit  
4021 buffer, it shall be an error.4022 (3) The text region shall be from the current line up to and including the  
4023 next *count* - 1 lines.4024 Otherwise, the buffer text mode and text region shall be as specified by the `c`  
4025 motion command.4026 The replaced text shall be copied into buffer, if specified, and into the unnamed  
4027 buffer. If the text to be replaced contains characters from more than a single line,  
4028 or the buffer text is in line mode, the replaced text shall be copied into the  
4029 numeric buffers as well.

- 4030 If the buffer text is in line mode:
- 4031 (1) Any lines that contain characters in the region shall be deleted, and the  
4032 editor shall enter text input mode at the beginning of a new line which  
4033 shall replace the first line deleted.
- 4034 (2) If the `autoindent` edit option is set, autoindent characters equal to the  
4035 autoindent characters on the first line deleted shall be inserted as if  
4036 entered by the user.
- 4037 Otherwise, if characters from more than one line are in the region of text:
- 4038 (1) The text shall be deleted.
- 4039 (2) Any text remaining in the last line in the text region shall be appended  
4040 to the first line in the region, and the last line in the region shall be  
4041 deleted.
- 4042 (3) The editor shall enter text input mode after the last character not deleted  
4043 from the first line in the text region, if any; otherwise, on the first column  
4044 of the first line in the region.
- 4045 Otherwise:
- 4046 (1) If the glyph for `$` is smaller than the region, the end of the region shall be C  
4047 marked with a `$`. C
- 4048 (2) The editor shall enter text input mode, overwriting the region of text. C
- 4049 *Current line/column*: As specified for the text input commands; see 5.35.7.3.
- 4050 **5.35.7.2.48 c**
- 4051 *Synopsis*: `[buffer] [count] c` C
- 4052 This command shall be equivalent to the `vi` command `[buffer] [count] c$` (see  
4053 5.35.7.2.47).
- 4054 **5.35.7.2.49 d**
- 4055 *Synopsis*: `[buffer] [count] d motion`
- 4056 If the *motion* command is the `d` command repeated:
- 4057 (1) The buffer text shall be in line mode.
- 4058 (2) If there are less than *count* – 1 lines after the current line in the edit  
4059 buffer, it shall be an error.
- 4060 (3) The text region shall be from the current line up to and including the  
4061 next *count* – 1 lines.
- 4062 Otherwise, the buffer text mode and text region shall be as specified by the C  
4063 *motion* command.
- 4064 If in open mode, and the current line is deleted, and the line remains on the  
4065 display, an `@` character shall be displayed as the first glyph of that line. C

4066 Delete the region of text into buffer, if specified, and into the unnamed buffer. If  
 4067 the text to be deleted contains characters from more than a single line, or the  
 4068 buffer text is in line mode, the deleted text shall be copied into the numeric  
 4069 buffers, as well.

4070 *Current line*: Set to the first text region line that appears in the edit buffer, unless  
 4071 that line has been deleted, in which case it shall be set to the last line in the edit  
 4072 buffer, or line 1 if the edit buffer is empty.

4073 *Current column*:

- 4074 (1) If the line is empty, set to column position 1.
- 4075 (2) Otherwise, if the buffer text is in line mode or the motion was from the  
 4076 cursor toward the end of the edit buffer:
  - 4077 (a) If a character from the current line is displayed in the current  
 4078 column, set to the last column that displays any portion of that  
 4079 character.
  - 4080 (b) Otherwise, set to the last column in which any portion of any char-  
 4081 acter in the line is displayed.
- 4082 (3) Otherwise, if a character is displayed in the column that began the text  
 4083 region, set to the last column that displays any portion of that character.
- 4084 (4) Otherwise, set to the last column in which any portion of any character  
 4085 in the line is displayed.

#### 4086 **5.35.7.2.50 D**

4087 *Synopsis*: [*buffer*] D

4088 This command shall be equivalent to the `vi` command [*buffer*] `d$` (see  
 4089 5.35.7.2.49). C

#### 4090 **5.35.7.2.51 e**

4091 *Synopsis*: [*count*] e

4092 With the exception that words are used instead of bigwords as the delimiter, this  
 4093 command shall be equivalent to the `E` command; see 5.35.7.2.52.

#### 4094 **5.35.7.2.52 E**

4095 *Synopsis*: [*count*] E

4096 If the edit buffer is empty it shall be an error. If less than *count* bigwords end  
 4097 between the cursor and the end of the edit buffer, *count* shall be adjusted to the  
 4098 number of bigword endings between the cursor and the end of the edit buffer.

4099 If used as a motion command:

- 4100 (1) The text region shall be from the last character of the *count*-th next big-  
 4101 word up to and including the cursor character.

4102 (2) Any text copied to a buffer shall be in character mode.

4103 If not used as a motion command:

4104 *Current line*: Set to the line containing the *current column*.

4105 *Current column*: Set to the last column upon which any part of the last character  
4106 of the *count*-th next bigword is displayed.

4107 **5.35.7.2.53 f**

4108 *Synopsis*: [*count*] f *character*

4109 It shall be an error if *count* occurrences of the character do not occur after the cur-  
4110 sor in the line.

4111 If used as a motion command:

4112 (1) The text range shall be from the cursor character up to and including the  
4113 *count*-th occurrence of the specified character after the cursor.

4114 (2) Any text copied to a buffer shall be in character mode.

4115 If not used as a motion command:

4116 *Current line*: Unchanged.

4117 *Current column*: Set to the last column in which any portion of the *count*-th  
4118 occurrence of the specified character after the cursor appears in the line.

4119 **5.35.7.2.54 F**

4120 *Synopsis*: [*count*] F *character*

4121 It shall be an error if *count* occurrences of the character do not occur before the  
4122 cursor in the line.

4123 If used as a motion command:

4124 (1) The text region shall be from the *count*-th occurrence of the specified  
4125 character before the cursor, up to, but not including the cursor character.

4126 (2) Any text copied to a buffer shall be in character mode.

4127 If not used as a motion command:

4128 *Current line*: Unchanged.

4129 *Current column*: Set to the last column in which any portion of the *count*-th  
4130 occurrence of the specified character before the cursor appears in the line.

4131 **5.35.7.2.55 G**

4132 *Synopsis*: [*count*] G

4133 If *count* is not specified, it shall default to the last line of the edit buffer.

4134 If *count* is greater than the last line of the edit buffer, it shall be an error.

4135 If used as a motion command:

4136 (1) The text region shall be from the cursor line up to and including the  
4137 specified line.

4138 (2) Any text copied to a buffer shall be in line mode.

4139 If not used as a motion command:

4140 *Current line*: Set to *count*.

4141 *Current column*: Set to nonblank.

#### 4142 **5.35.7.2.56** H

4143 *Synopsis*: [*count*] H

4144 If the beginning of the line count greater than the first line of which any portion C  
4145 appears on the display does not exist, it shall be an error. C

4146 If used as a motion command:

4147 (1) If in open mode, the text region shall be the current line.

4148 (2) Otherwise, the text region shall be from the starting line up to and  
4149 including (the first line of the display + *count* - 1).

4150 (3) Any text copied to a buffer shall be in line mode.

4151 If not used as a motion command:

4152 If in open mode, this command shall set the current column to nonblank and do  
4153 nothing else.

4154 Otherwise, it shall set the current line and current column as follows:

4155 *Current line*: Set to (the first line of the display + *count* - 1).

4156 *Current column*: Set to nonblank.

#### 4157 **5.35.7.2.57** i

4158 *Synopsis*: [*count*] i

4159 Enter text input mode before the current cursor position. No characters already C  
4160 in the edit buffer shall be affected by this command. A *count* shall cause the C  
4161 input text to be appended *count* - 1 more times to the end of the input.

4162 *Current line/column*: As specified for the text input commands; see 5.35.7.3.

#### 4163 **5.35.7.2.58** I

4164 *Synopsis*: [*count*] I

4165 This command shall be equivalent to the `vi` commands `^[count]i` (see  
4166 5.35.7.2.57).



4167 **5.35.7.2.59** J4168 *Synopsis:* [*count*] J

4169 If the current line is the last line in the edit buffer, it shall be an error.

4170 This command shall be equivalent to the `ex join` command (see 5.10.7.5.12) with  
 4171 no addresses, and an `ex` command *count* value of 1 if *count* was not specified or if  
 4172 a *count* of 1 was specified, and an `ex` command *count* value of *count* – 1 for any  
 4173 other value of *count*, except that the current line and column shall be set as fol-  
 4174 lows:

4175 *Current line:* Unchanged.

4176 *Current column:* The last column in which any portion of the character following  
 4177 the last character in the initial line is displayed, or the last character in the line if  
 4178 no characters were appended. C

4179 **5.35.7.2.60** L4180 *Synopsis:* [*count*] L

4181 If the beginning of the line count less than the last line of which any portion C  
 4182 appears on the display does not exist, it shall be an error. C

4183 If used as a motion command:

- 4184 (1) If in open mode, the text region shall be the current line.  
 4185 (2) Otherwise, the text region shall include all lines from the starting cursor  
 4186 line to (the last line of the display – (*count* – 1)).  
 4187 (3) Any text copied to a buffer shall be in line mode.

4188 If not used as a motion command:

4189 If in open mode, this command shall set the current column to nonblank and do  
 4190 nothing else.

4191 Otherwise, it shall set the current line and current column as follows:

4192 *Current line:* Set to (the last line of the display – (*count* – 1)).4193 *Current column:* Set to nonblank.4194 **5.35.7.2.61** m4195 *Synopsis:* *m* character

4196 This command shall be equivalent to the `ex mark` command (see 5.10.7.5.15) with  
 4197 the specified character as an argument.

4198 **5.35.7.2.62 M**4199 *Synopsis:* M

4200 The middle line of the display shall be calculated as follows:

4201  $(\textit{the top line of the display}) + (((\textit{number of lines displayed}) + 1) / 2) - 1$ 

4202 If used as a motion command:

- 4203 (1) If in open mode, the text region shall be the current line.
- 4204 (2) Otherwise, the text region shall include all lines from the starting cursor  
4205 line up to and including the middle line of the display.
- 4206 (3) Any text copied to a buffer shall be in line mode.

4207 If not used as a motion command:

4208 If in open mode, this command shall set the current column to nonblank and do  
4209 nothing else.

4210 Otherwise, it shall set the current line and current column as follows:

4211 *Current line:* Set to the middle line of the display.4212 *Current column:* Set to nonblank.4213 **5.35.7.2.63 n**4214 *Synopsis:* n

4215 If the remembered search direction was forward, the n command shall be  
4216 equivalent to the vi / command with no characters entered by the user (see  
4217 5.35.7.2.34). Otherwise, it shall be equivalent to the vi ? command with no char-  
4218 acters entered by the user (see 5.35.7.2.40).

4219 If the n command is used as a motion command for the ! command, the editor  
4220 shall not enter text input mode on the last line on the screen, and shall behave as  
4221 if the user entered a single ! character as the text input.

4222 **5.35.7.2.64 N**4223 *Synopsis:* N

4224 If the remembered search direction was forward, the N command shall be  
4225 equivalent to the vi ? command with no characters entered by the user (see  
4226 5.35.7.2.40.) Otherwise, it shall be equivalent to the vi / command with no char-  
4227 acters entered by the user (see 5.35.7.2.34).

4228 If the N command is used as a motion command for the ! command, the editor  
4229 shall not enter text input mode on the last line on the screen, and shall behave as  
4230 if the user entered a single ! character as the text input.

4231 **5.35.7.2.65** ○4232 *Synopsis:* [*count*] ○

4233 Enter text input mode in a new line appended after the current line. A *count*  
 4234 shall cause the input text to be appended *count* – 1 more times to the end of the C  
 4235 already added text, each time starting on a new, appended line. C

4236 *Current line/column:* As specified for the text input commands; see 5.35.7.3.4237 **5.35.7.2.66** ○4238 *Synopsis:* [*count*] ○

4239 Enter text input mode in a new line inserted before the current line. A *count*  
 4240 shall cause the input text to be appended *count* – 1 more times to the end of the C  
 4241 already added text, each time starting on a new, appended line. C

4242 *Current line/column:* As specified for the text input commands; see 5.35.7.3.4243 **5.35.7.2.67** p4244 *Synopsis:* [*buffer*] [*count*] p4245 If no *buffer* is specified, the unnamed buffer shall be used. C

4246 If the buffer text is in line mode, the text shall be appended below the current  
 4247 line, and each line of the buffer shall become a new line in the edit buffer. A  
 4248 *count* shall cause the buffer text to be appended *count* – 1 more times to the end of  
 4249 the already added text, each time starting on a new, appended line.

4250 If the buffer text is in character mode, the text shall be appended into the current  
 4251 line after the cursor, and each line of the buffer other than the first and last shall  
 4252 become a new line in the edit buffer. A *count* shall cause the buffer text to be  
 4253 appended *count* – 1 more times to the end of the already added text, each time  
 4254 starting after the last added character.

4255 *Current line:* If the buffer text is in line mode, set the line to line + 1; otherwise,  
 4256 unchanged.

4257 *Current column:*

4258 If the buffer text is in line mode:

4259 (1) If there is a nonblank character in the first line of the buffer, set to the  
 4260 last column on which any portion of the first nonblank character in the  
 4261 line is displayed.

4262 (2) If there is no nonblank character in the first line of the buffer, set to the  
 4263 last column on which any portion of the last character in the first line of  
 4264 the buffer is displayed.

4265 If the buffer text is in character mode:

4266 (1) If the text in the buffer is from more than a single line, then set to the  
 4267 last column on which any portion of the first character from the buffer is  
 4268 displayed.

4269 (2) Otherwise, if the buffer is the unnamed buffer, set to the last column on  
4270 which any portion of the last character from the buffer is displayed.

4271 (3) Otherwise, set to the first column on which any portion of the first char-  
4272 acter from the buffer is displayed.

4273 **5.35.7.2.68 P**

4274 *Synopsis:* [*buffer*] [*count*] P

4275 If no *buffer* is specified, the unnamed buffer shall be used. C

4276 If the buffer text is in line mode, the text shall be inserted above the current line,  
4277 and each line of the buffer shall become a new line in the edit buffer. A *count*  
4278 shall cause the buffer text to be appended *count* – 1 more times to the end of the  
4279 already added text, each time starting on a new, appended line.

4280 If the buffer text is in character mode, the text shall be inserted into the current  
4281 line before the cursor, and each line of the buffer other than the first and last  
4282 shall become a new line in the edit buffer. A *count* shall cause the buffer text to  
4283 be appended *count* – 1 more times to the end of the already added text, each time  
4284 starting after the last added character.

4285 *Current line:* Unchanged.

4286 *Current column:*

4287 If the buffer text is in line mode:

4288 (1) If there is a nonblank character in the first line of the buffer, set to the  
4289 last column on which any portion of that character is displayed.

4290 (2) If there is no nonblank character in the first line of the buffer, set to the  
4291 last column on which any portion of the last character in the first line of  
4292 the buffer is displayed.

4293 If the buffer text is in character mode:

4294 (1) If the buffer is the unnamed buffer, set to the last column on which any  
4295 portion of the last character from the buffer is displayed.

4296 (2) Otherwise, set to the first column on which any portion of the first char-  
4297 acter from the buffer is displayed.

4298 **5.35.7.2.69 Q**

4299 *Synopsis:* Q

4300 Leave visual or open mode and enter `ex` command mode.

4301 *Current line:* Unchanged.

4302 *Current column:* Unchanged.

4303 **5.35.7.2.70 r**4304 *Synopsis:* [*count*] r *character*

4305 Replace the *count* characters at and after the cursor with the specified character.  
 4306 If there are less than *count* characters at and after the cursor on the line, it shall  
 4307 be an error.

4308 If character is <control-V>, any next character other than <newline> shall be  
 4309 stripped of any special meaning and used as a literal character.

4310 If character is <ESC>, no replacement shall be made and the current line and  
 4311 current column shall be unchanged.

4312 If character is <carriage-return> or <newline>, *count* new lines shall be  
 4313 appended to the current line. All but the last of these lines shall be empty. *Count* C  
 4314 characters at and after the cursor shall be discarded, and any remaining charac- C  
 4315 ters after the cursor in the current line shall be moved to the last of the new lines.  
 4316 If the *autoindent* edit option is set, they shall be preceded by the same number  
 4317 of *autoindent* characters found on the line from which the command was exe-  
 4318 cuted.

4319 *Current line:* Unchanged unless the replacement character is a <carriage-  
 4320 return> or <newline>, in which case it shall be set to *line* + *count*.

4321 *Current column:* Set to the last column position on which a portion of the last  
 4322 replaced character is displayed, or if the replacement character caused new lines  
 4323 to be created, set to nonblank C

4324 **5.35.7.2.71 R**4325 *Synopsis:* [*count*] R

4326 Enter text input mode at the current cursor position. A *count* shall cause the  
 4327 input text to be appended *count* – 1 more times to the end of the input.

4328 *Current line/column:* As specified for the text input commands; see 5.35.7.3.

4329 **5.35.7.2.72 s**4330 *Synopsis:* [*buffer*] [*count*] s

4331 This command shall be equivalent to the vi command [*buffer*] [*count*] c1 (see  
 4332 5.35.7.2.47).

4333 **5.35.7.2.73 s**4334 *Synopsis:* [*buffer*] [*count*] s

4335 This command shall be equivalent to the vi command [*buffer*] [*count*] c\_ (see  
 4336 5.35.7.2.47).

4337 **5.35.7.2.74** **t**4338 *Synopsis:* [*count*] **t** *character*4339 It shall be an error if *count* occurrences of the character do not occur after the cur-  
4340 sor in the line.

4341 If used as a motion command:

4342 (1) The text region shall be from the cursor up to but not including the  
4343 *count*-th occurrence of the specified character after the cursor.

4344 (2) Any text copied to a buffer shall be in character mode.

4345 If not used as a motion command:

4346 *Current line:* Unchanged.4347 *Current column:* Set to the last column in which any portion of the character  
4348 before the *count*-th occurrence of the specified character after the cursor appears  
4349 in the line.4350 **5.35.7.2.75** **T**4351 *Synopsis:* [*count*] **T** *character*4352 It shall be an error if *count* occurrences of the character do not occur before the  
4353 cursor in the line.

4354 If used as a motion command:

4355 (1) If the character before the cursor is the specified character, it shall be an  
4356 error.4357 (2) The text region shall be from the character before the cursor up to but  
4358 not including the *count*-th occurrence of the specified character before the  
4359 cursor.

4360 (3) Any text copied to a buffer shall be in character mode.

4361 If not used as a motion command:

4362 *Current line:* Unchanged.4363 *Current column:* Set to the last column in which any portion of the character after  
4364 the *count*-th occurrence of the specified character before the cursor appears in the  
4365 line.4366 **5.35.7.2.76** **u**4367 *Synopsis:* **u**4368 This command shall be equivalent to the `ex undo` command (see 5.10.7.5.34),  
4369 except that the current line and current column shall be set as follows:

- 4370 *Current line:*
- 4371 Set to the first line added or changed if any; otherwise, move to the line C  
 4372 preceding any deleted text if one exists; otherwise, move to line 1. C
- 4373 *Current column:*
- 4374 If undoing an `ex` command, set to the first nonblank.
- 4375 Otherwise, if undoing a text input command:
- 4376 (1) If the command was an `C`, `c`, `O`, `o`, `R`, `S`, or `s` command, the current  
 4377 column shall be set to the value it held when the text input command  
 4378 was entered.
- 4379 (2) Otherwise, set to the last column in which any portion of the first  
 4380 character after the deleted text is displayed, or, if no characters follow  
 4381 the text deleted from this line, set to the last column in which any  
 4382 portion of the last character in the line is displayed, or 1 if the line is  
 4383 empty.
- 4384 Otherwise, if a single line was modified (i.e., not added or deleted) by the `u`  
 4385 command:
- 4386 (1) If text was added or changed, set to the last column in which any por-  
 4387 tion of the first character added or changed is displayed.
- 4388 (2) If text was deleted, set to the last column in which any portion of the  
 4389 first character after the deleted text is displayed, or, if no characters  
 4390 follow the deleted text, set to the last column in which any portion of  
 4391 the last character in the line is displayed, or 1 if the line is empty.
- 4392 Otherwise, set to nonblank.
- 4393 **5.35.7.2.77 `U`**
- 4394 *Synopsis:* `U`
- 4395 Restore the current line to its state immediately before the most recent time that  
 4396 it became the current line.
- 4397 *Current line:* Unchanged.
- 4398 *Current column:* Set to the first column in the line in which any portion of the  
 4399 first character in the line is displayed.
- 4400 **5.35.7.2.78 `w`**
- 4401 *Synopsis:* [*count*] `w`
- 4402 With the exception that words are used as the delimiter instead of bigwords, this  
 4403 command shall be equivalent to the `W` command; see 5.35.7.2.79.

4404 **5.35.7.2.79 w**4405 *Synopsis:* [*count*] w

4406 If the edit buffer is empty, it shall be an error. If there are less than *count* big-  
 4407 words between the cursor and the end of the edit buffer, *count* shall be adjusted to  
 4408 move the cursor to the last bigword in the edit buffer.

4409 If used as a motion command:

4410 (1) If the associated command is *c*, *count* is 1, and the cursor is on a  
 4411 <blank> character, the region of text shall be the current character and  
 4412 no further action shall be taken.

4413 (2) If there are less than *count* bigwords between the cursor and the end of  
 4414 the edit buffer, then the command shall succeed, and the region of text  
 4415 shall include the last character of the edit buffer.

4416 (3) If there are <blank> characters or an end-of-line that precede the *count*-  
 4417 th bigword, and the associated command is *c*, the region of text shall be  
 4418 up to and including the last character before the preceding <blank>  
 4419 characters or end-of-line.

4420 (4) If there are <blank> characters or an end-of-line that precede the big-  
 4421 word, and the associated command is *d* or *y*, the region of text shall be up  
 4422 to and including the last <blank> character before the start of the big-  
 4423 word or end-of-line.

4424 (5) Any text copied to a buffer shall be in character mode.

4425 If not used as a motion command:

4426 If the cursor is on the last character of the edit buffer, it shall be an error.

4427 *Current line:* Set to the line containing the *current column*.

4428 *Current column:* Set to the last column in which any part of the first character of  
 4429 the *count*-th next bigword is displayed.

4430 **5.35.7.2.80 x**4431 *Synopsis:* [*buffer*] [*count*] x

4432 Delete the *count* characters at and after the current character into buffer, if  
 4433 specified, and into the unnamed buffer.

4434 If the line is empty, it shall be an error. If there are less than *count* characters at  
 4435 and after the cursor on the current line, *count* shall be adjusted to the number of  
 4436 characters at and after the cursor.

4437 *Current line:* Unchanged.

4438 *Current column:* If the line is empty, set to column position 1. Otherwise, if there  
 4439 were *count* or less characters at and after the cursor on the current line, set to the  
 4440 last column that displays any part of the last character of the line. Otherwise,  
 4441 unchanged.



4442 **5.35.7.2.81** **x**4443 *Synopsis:* [*buffer*] [*count*] **x**4444 Delete the *count* characters before the current character into *buffer*, if specified,  
4445 and into the unnamed buffer.4446 If there are no characters before the current character on the current line, it shall  
4447 be an error. If there are less than *count* previous characters on the current line,  
4448 *count* shall be adjusted to the number of previous characters on the line.4449 *Current line:* Unchanged.4450 *Current column:* Set to (*current column* – *the width of the deleted characters*).4451 **5.35.7.2.82** **y**4452 *Synopsis:* [*buffer*] [*count*] **y** *motion*

4453 Copy the region of text into buffer, if specified, and into the unnamed buffer.

4454 If the *motion* command is the **y** command repeated:

- 4455 (1) The buffer shall be in line mode.
- 4456 (2) If there are less than *count* – 1 lines after the current line in the edit  
4457 buffer, it shall be an error.
- 4458 (3) The text region shall be from the current line up to and including the  
4459 next *count* – 1 lines.

4460 Otherwise, the buffer text mode and text region shall be as specified by the **c**  
4461 *motion* command.4462 *Current line:* If the motion was from the current cursor position toward the end of  
4463 the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that  
4464 is part of the text region specified by the *motion* command.4465 *Current column:*

- 4466 (1) If the motion was from the current cursor position toward the end of the  
4467 edit buffer, unchanged.
- 4468 (2) Otherwise, if the current line is empty, set to column position 1.
- 4469 (3) Otherwise, set to the last column that displays any part of the first char-  
4470 acter in the file that is part of the text region specified by the *motion*  
4471 command.

4472 **5.35.7.2.83** **Y**4473 *Synopsis:* [*buffer*] [*count*] **Y**4474 This command shall be equivalent to the **vi** command [*buffer*] [*count*] **Y\_** (see  
4475 5.35.7.2.82).

- 4476 **5.35.7.2.84 z**
- 4477 If in open mode, the `z` command shall have a Synopsis of: C
- 4478 *Synopsis:* `[count] z` C
- 4479 If *count* is not specified, it shall default to the `window edit` option `- 1`. The `z` com- C  
 4480 mand shall be equivalent to the `ex z` command, with a type character of “=” and a C  
 4481 *count* of *count*-2, except that the current line and current column shall be set as C  
 4482 follows, and the `window edit` option shall not be affected. If the calculation for the C  
 4483 *count* argument would result in a negative number, the *count* argument to the `ex` C  
 4484 `z` command shall be zero. A blank line shall be written after the last line is writ- C  
 4485 ten. C
- 4486 *Current line:* Unchanged. C
- 4487 *Current column:* Unchanged. C
- 4488 If not in open mode, the `z` command shall have a Synopsis of: C
- 4489 *Synopsis:* `[line] z [count] character` C
- 4490 If *line* is not specified, it shall default to the current line. If *line* is specified, but is C  
 4491 greater than the number of lines in the edit buffer, it shall default to the number C  
 4492 of lines in the edit buffer.
- 4493 If *count* is specified, the value of the `window edit` option shall be set to *count* (as C  
 4494 described in 5.10.7.8.29), and the screen shall be redrawn.
- 4495 Line shall be placed as specified by the following characters:
- 4496 `<newline>`
- 4497 `<carriage-return>`
- 4498 Place the beginning of the line on the first line of the display.
- 4499 . Place the beginning of the line in the center of the display. The middle line C  
 4500 of the display shall be calculated as described for the `M` command (see C  
 4501 5.35.7.2.62). C
- 4502 - Place an unspecified portion of the line on the last line of the display.
- 4503 + If *line* was specified, equivalent to the `<newline>` case. If *line* was not C  
 4504 specified, display a screen where the first line of the display shall be C  
 4505 (current last line) + 1. If there are no lines after the last line in the display, C  
 4506 it shall be an error.
- 4507 ^ If *line* was specified, display a screen where the last line of the display shall C  
 4508 contain an unspecified portion of the first line of a display that had an C  
 4509 unspecified portion of the specified line on the last line of the display. If C  
 4510 this calculation results in a line before the beginning of the edit buffer, C  
 4511 display the first screen of the edit buffer.
- 4512 Otherwise, display a screen where the last line of the display shall contain C  
 4513 an unspecified portion of (current first line - 1). If this calculation results C  
 4514 in a line before the beginning of the edit buffer, it shall be an error.

4515 *Current line:*

4516 If *line* and the ^ character were specified:

4517 (1) If the first screen was displayed as a result of the command attempting to  
4518 display lines before the beginning of the edit buffer:

4519 If the first screen was already displayed, unchanged; otherwise, set to  
4520 (current first line – 1).

4521 (2) Otherwise, set to the last line of the display.

4522 If *line* and the + character were specified, set to the first line of the display.

4523 Otherwise, if *line* was specified, set to *line*.

4524 Otherwise, unchanged. *Current column:* Set to nonblank.

4525 **5.35.7.2.85 zz**

4526 *Synopsis:* zz

4527 This command shall be equivalent to the `ex xit` command with no addresses,  
4528 trailing `!`, or file name (see 5.10.7.5.39).

### 4529 **5.35.7.3 Input Mode Commands**

4530 In text input mode, the current line shall consist of zero or more of the following  
4531 categories:

4532 (1) Characters preceding the text input entry point:

4533 Characters in this category shall not be modified during text input  
4534 mode.

4535 (2) Autoindent characters:

4536 Autoindent characters shall be automatically inserted into each line  
4537 that is created in text input mode, either as a result of entering a  
4538 `<newline>` or `<carriage-return>` while in text input mode, or as  
4539 an effect of the command itself; e.g., `o` or `o` (see 5.10.7.8.1), as if  
4540 entered by the user.

4541 It shall be possible to erase autoindent characters with the  
4542 `<control-D>` command (see 5.35.7.3.2); it is unspecified if they can  
4543 be erased by `<control-H>`, `<control-U>`, and `<control-W>` char-  
4544 acters (see 5.35.7.3.3, 5.35.7.3.6, and 5.35.7.3.8). Erasing any autoin-  
4545 dent character turns the glyph into erase-columns and deletes the  
4546 character from the edit buffer, but does not change its representation  
4547 on the screen.

4548 (3) Text input characters:

4549 Text input characters are the characters entered by the user. Erasing  
4550 (see 5.35.7.3.3, 5.35.7.3.6, and 5.35.7.3.8) any text input character  
4551 turns the glyph into erase-columns and deletes the character from the  
4552 edit buffer, but does not change its representation on the screen.

4553 Each text input character entered by the user (that does not have a  
4554 special meaning) shall be treated as follows:

4555 (a) The text input character shall be appended to the last character  
4556 in the edit buffer from the first, second, or third categories.

4557 (b) If there are no erase-columns on the screen, the text input com-  
4558 mand was the R command, and characters in the fifth category  
4559 from the original line follow the cursor, the next such character  
4560 shall be deleted from the edit buffer. If the `slowopen` edit  
4561 option is not set, the corresponding glyph on the screen shall  
4562 become erase-columns.

4563 (c) If there are erase-columns on the screen, as many columns as  
4564 they occupy, or as are necessary, shall be overwritten to display  
4565 the text input character. (If only part of a multicolumn glyph is  
4566 overwritten, the remainder shall be left on the screen, and con-  
4567 tinue to be treated as erase-columns; it is unspecified if the  
4568 remainder of the glyph is modified in any way.)

4569 (d) If additional screen columns are needed to display the text input  
4570 character:

4571 [1] If the `slowopen` edit option is set, the text input characters  
4572 shall be displayed on subsequent screen columns, overwrit-  
4573 ing any characters displayed in those columns.

4574 [2] Otherwise, any characters currently displayed on or after  
4575 the column on the screen where the text input character is  
4576 to be displayed shall be pushed ahead the number of screen  
4577 columns necessary to display the rest of the text input  
4578 character.

4579 (4) Erase-columns:

4580 Erase-columns are not logically part of the edit buffer, appearing only  
4581 on the screen, and may be overwritten on the screen by subsequent  
4582 text input characters. When text input mode ends, all erase-columns  
4583 shall no longer appear on the screen.

4584 Erase-columns are initially the region of text specified by the `c`  
4585 command (see 5.35.7.2.47); however, erasing autoindent or text input  
4586 characters causes the glyphs of the erased characters to be treated as  
4587 erase-columns.

4588 (5) Characters following the text region for the `c` command, or the text input  
4589 entry point for all other commands:

4590 Characters in this category shall not be modified during text input  
4591 mode, except as specified in category (3b) for the R text input com- C  
4592 mand, or as `<blank>` characters deleted when a `<newline>` or  
4593 `<carriage-return>` is entered (see 5.35.7.3.4).

4594 It is unspecified if it is an error to attempt to erase past the beginning of a line  
4595 that was created by the entry of a `<newline>` or `<carriage-return>` character

4596 during text input mode. If it is not an error, the editor shall behave as if the eras-  
 4597 ing character was entered immediately after the last text input character entered  
 4598 on the previous line, and all of the characters on the current line shall be treated  
 4599 as erase-columns.

4600 When text input mode is entered, or after a text input mode character is entered  
 4601 (except as specified for the special characters below), the cursor shall be posi-  
 4602 tioned as follows:

- 4603 (1) On the first column that displays any part of the first erase-column, if  
 4604 one exists.
- 4605 (2) Otherwise, if the `slowopen` edit option is set, on the first screen column  
 4606 after the last character in the first, second, or third categories, if one  
 4607 exists.
- 4608 (3) Otherwise, the first column that displays any part of the first character  
 4609 in the fifth category, if one exists.
- 4610 (4) Otherwise, the screen column after the last character in the first, second,  
 4611 or third categories, if one exists.
- 4612 (5) Otherwise, on column position 1.

4613 The characters that are updated on the screen during text input mode are  
 4614 unspecified, other than that the last text input character shall always be updated,  
 4615 and, if the `slowopen` edit option is not set, the current cursor character shall  
 4616 always be updated.

4617 The following specifications are for command characters entered during text input  
 4618 mode.

#### 4619 **5.35.7.3.1** NUL

4620 *Synopsis:* NUL

4621 If the first character of the text input is a NUL, the most recently input text shall  
 4622 be input as if entered by the user, and then text input mode shall be exited. The  
 4623 text shall be input literally; i.e., characters are neither macro or abbreviation  
 4624 expanded, nor are any characters interpreted in any special manner. It is C  
 4625 unspecified if implementations shall support more than 256 bytes of remembered C  
 4626 input text. C

#### 4627 **5.35.7.3.2** <control-D>

4628 *Synopsis:* <control-D>

4629 The <control-D> character shall have no special meaning when in text input  
 4630 mode for a line-oriented command (see 5.35.7.2).

4631 This command need not be supported on block-mode terminals.

4632

C

4633 If the cursor does not follow an autoindent character, or an autoindent character  
 4634 and a 0 or ^ character:

4635 (1) If the cursor is in column position 1, the `<control-D>` character shall be  
4636 discarded and no further action taken.

4637 (2) Otherwise, the `<control-D>` character shall have no special meaning.

4638 If the last input character was a 0, the cursor shall be moved to column position 1.

4639 Otherwise, if the last input character was a `^`, the cursor shall be moved to  
4640 column position 1. In addition, the `autoindent` level for the next input line shall  
4641 be derived from the same line from which the `autoindent` level for the current  
4642 input line was derived.

4643 Otherwise, the cursor shall be moved back to the column after the previous  
4644 `shiftwidth` (see 5.10.7.8.19) boundary.

4645 All of the glyphs on columns between the starting cursor position and (inclusively)  
4646 the ending cursor position shall become erase-columns as described in 5.35.7.3.

4647 *Current line:* Unchanged.

4648 *Current column:* Set to 1 if the `<control-D>` was preceded by a `^` or 0; otherwise,  
4649 set to  $(\text{column} - 1) - ((\text{column} - 2) \% \text{shiftwidth})$ .

### 4650 **5.35.7.3.3** `<control-H>`

4651 *Synopsis:* `<control-H>`

4652 If in text input mode for a line-oriented command, and there are no characters to  
4653 erase, text input mode shall be terminated, no further action shall be done for this  
4654 command, and the current line and column shall be unchanged.

4655 If there are characters other than autoindent characters that have been input on  
4656 the current line before the cursor, the cursor shall move back one character.

4657 Otherwise, if there are autoindent characters on the current line before the cur-  
4658 sor, it is implementation-defined if the `<control-H>` command is an error or if  
4659 the cursor moves back one autoindent character.

4660 Otherwise, if the cursor is in column position 1 and there are previous lines that  
4661 have been input, it is implementation-defined if the `<control-H>` command is an  
4662 error or if it is equivalent to entering `<control-H>` after the last input character  
4663 on the previous input line.

4664 Otherwise, it shall be an error.

4665 All of the glyphs on columns between the starting cursor position and (inclusively)  
4666 the ending cursor position shall become erase-columns as described in 5.35.7.3.

4667 The current *erase* character (see `stty` in 4.59) shall cause an equivalent action to  
4668 the `<control-H>` command, unless the previously inserted character was a  
4669 backslash, in which case it shall be as if the literal current *erase* character had  
4670 been inserted instead of the backslash.

4671 *Current line:* Unchanged, unless previously input lines are erased, in which case  
4672 it shall be set to `line - 1`.

4673 *Current column:* Set to the first column that displays any portion of the character  
4674 backed up over.

4675 **5.35.7.3.4 <newline>**4676 *Synopsis:* <newline>4677 *Synopsis:* <carriage-return>4678 *Synopsis:* <control-J> C4679 *Synopsis:* <control-M> C

4680 If input was part of a line-oriented command, text input mode shall be terminated  
4681 and the command shall continue execution with the input provided.

4682 Otherwise, terminate the current line. If there are no characters other than  
4683 autoindent characters on the line, all characters on the line shall be discarded.  
4684 Otherwise, it is unspecified if the autoindent characters in the line are modified  
4685 by entering these characters. C

4686 Continue text input mode on a new line appended after the current line. If the  
4687 `slowopen` edit option is set, the lines on the screen below the current line shall  
4688 not be pushed down, but the first of them shall be cleared and shall appear to be  
4689 overwritten. Otherwise, the lines of the screen below the current line shall be  
4690 pushed down.

4691 If the `autoindent` edit option is set, an appropriate number of autoindent  
4692 characters shall be added as a prefix to the line as described by the `ex autoin-`  
4693 `dent` edit option (see 5.10.7.8.1). C

4694 All columns after the cursor that are `erase-columns` (as described in 5.35.7.3)  
4695 shall be discarded.

4696 If the `autoindent` edit option is set, all <blank> characters immediately follow- C  
4697 ing the cursor shall be discarded. C

4698 All remaining characters after the cursor shall be transferred to the new line,  
4699 positioned after any autoindent characters.

4700 *Current line:* Set to current line + 1.

4701 *Current column:* Set to the first column that displays any portion of the first char-  
4702 acter after the autoindent characters on the new line, if any, or the first column  
4703 position after the last autoindent character, if any, or column position 1.

4704 **5.35.7.3.5 <control-T>**4705 *Synopsis:* <control-T>

4706 The <control-T> character shall have no special meaning when in text input C  
4707 mode for a line-oriented command (see 5.35.7.2). C

4708 This command need not be supported on block-mode terminals.

4709

4710 Behave as if the user entered the minimum number of <blank> characters neces-  
4711 sary to move the cursor forward to the column position after the next  
4712 shiftwidth (see 5.10.7.8.19) boundary.

4713 *Current line:* Unchanged. C

4714 *Current column*: Set to `column + shiftwidth - ((column - 1) % shiftwidth)`.

4715 **5.35.7.3.6** `<control-U>`

4716 *Synopsis*: `<control-U>`

4717 If there are characters other than autoindent characters that have been input on  
4718 the current line before the cursor, the cursor shall move to the first character  
4719 input after the autoindent characters.

4720 Otherwise, if there are autoindent characters on the current line before the cur-  
4721 sor, it is implementation-defined if the `<control-U>` command is an error or if  
4722 the cursor moves to the first column position on the line.

4723 Otherwise, if the cursor is in column position 1 and there are previous lines that  
4724 have been input, it is implementation-defined if the `<control-U>` command is an  
4725 error or if it is equivalent to entering `<control-U>` after the last input character  
4726 on the previous input line.

4727 Otherwise, it shall be an error.

4728 All of the glyphs on columns between the starting cursor position and (inclusively)  
4729 the ending cursor position shall become erase-columns as described in 5.35.7.3.

4730 The current *kill* character (see `stty` in 4.59) shall cause an equivalent action to  
4731 the `<control-U>` command, unless the previously inserted character was a  
4732 backslash, in which case it shall be as if the literal current *kill* character had been  
4733 inserted instead of the backslash.

4734 *Current line*: Unchanged, unless previously input lines are erased, in which case  
4735 it shall be set to `line - 1`.

4736 *Current column*: Set to the first column that displays any portion of the last char-  
4737 acter backed up over.

4738 **5.35.7.3.7** `<control-V>`

4739 *Synopsis*: `<control-V>`

4740 *Synopsis*: `<control-Q>`

4741 Allow the entry of any subsequent character, other than `<control-J>` or `<new-` C  
4742 `line>`, as a literal character, removing any special meaning that it may have to C  
4743 the editor in text input mode. If a `<control-V>` or `<control-Q>` is entered C  
4744 before a `<control-J>` or `<newline>` character, the `<control-V>` or C  
4745 `<control-Q>` character shall be discarded, and the `<control-J>` or `<newline>` C  
4746 shall behave as described in 5.35.7.3.4. C

4747 For purposes of the display only, the editor shall behave as if a `^` character was  
4748 entered, and the cursor shall be positioned as if overwriting the `^` character.  
4749 When a subsequent character is entered, the editor shall behave as if that charac-  
4750 ter was entered instead of the original `<control-V>` or `<control-Q>` character.

4751 *Current line*: Unchanged.

4752 *Current column*: Unchanged.



4753 **5.35.7.3.8 <control-W>**4754 *Synopsis:* <control-W>

4755 If there are characters other than autoindent characters that have been input on  
 4756 the current line before the cursor, the cursor shall move back over the last word  
 4757 preceding the cursor (including any <blank> characters between the end of the  
 4758 last word and the current cursor); the cursor shall not move to before the first C  
 4759 character after the end of any autoindent characters.

4760 Otherwise, if there are autoindent characters on the current line before the cur-  
 4761 sor, it is implementation-defined if the <control-W> command is an error or if  
 4762 the cursor moves to the first column position on the line.

4763 Otherwise, if the cursor is in column position 1 and there are previous lines that  
 4764 have been input, it is implementation-defined if the <control-W> command is an  
 4765 error or if it is equivalent to entering <control-W> after the last input character  
 4766 on the previous input line.

4767 Otherwise, it shall be an error.

4768 All of the glyphs on columns between the starting cursor position and (inclusively)  
 4769 the ending cursor position shall become erase-columns as described in 5.35.7.3.

4770 *Current line:* Unchanged, unless previously input lines are erased, in which case  
 4771 it shall be set to line – 1.

4772 *Current column:* Set to the first column that displays any portion of the last char-  
 4773 acter backed up over.

4774 **5.35.7.3.9 <ESC>**4775 *Synopsis:* <ESC>

4776 C

4777 If input was part of a line-oriented command:

4778 (1) If <interrupt> was entered, text input mode shall be terminated and  
 4779 the editor shall return to command mode. The terminal shall be alerted.

4780 (2) If <ESC> was entered, text input mode shall be terminated and the com-  
 4781 mand shall continue execution with the input provided.

4782 Otherwise, terminate text input mode and return to command mode. C

4783 Any autoindent characters entered on newly created lines that have no other  
 4784 characters shall be deleted.

4785 Any leading autoindent and <blank> characters on newly created lines shall be  
 4786 rewritten to be the minimum number of <blank> characters possible.

4787 The screen shall be redisplayed as necessary to match the contents of the edit  
 4788 buffer.

4789 *Current line:* Unchanged.

4790 *Current column:*

- 4791 (1) If there are text input characters on the current line, the column shall be  
4792 set to the last column where any portion of the last text input character  
4793 is displayed.
- 4794 (2) Otherwise, if a character is displayed in the current column, unchanged.
- 4795 (3) Otherwise, set to column position 1.

4796 **5.35.8 Exit Status**

4797 The `vi` utility shall exit with one of the following values:

- 4798 0 Successful completion.
- 4799 >0 An error occurred.

4800 **5.35.9 Consequences of Errors**

- 4801 When any error is encountered and the standard input is not a terminal device C  
4802 file, `vi` shall not write the file or return to command or text input mode, and shall C  
4803 terminate with a nonzero exit status. C
- 4804 Otherwise, when an unrecoverable error is encountered it shall be equivalent to a C  
4805 SIGHUP asynchronous event. C
- 4806 Otherwise, when an error is encountered, the editor shall behave as specified in C  
4807 5.35.7.2. C

4808 **5.37 write – Write to another user**

4809 ⇒ **5.37.5.3 write Environment Variables.** *Change the description of*  
4810 **LC\_CTYPE** *to:*

4811 **LC\_CTYPE** This variable shall determine the interpretation of  
4812 sequences of bytes of text data as characters (e.g.,  
4813 single- versus multibyte characters in arguments and  
4814 input files). If the locale of the recipient does not use  
4815 an LC\_CTYPE equivalent to that of the sender, the  
4816 results are undefined.

4817 **Rationale:** This change is the result of interpretation request PASC 1003.2-92  
4818 #26 submitted for IEEE Std 1003.2-1992.



## Section 6: Revisions to Software Development Utilities Option

1	<b>6.1 ar – Create and maintain library archives</b>	B
2	⇒ <b>6.1.6.1 ar Standard Output.</b> <i>Change the two paragraphs:</i> If the <code>-r</code> option is	B
3	used with the <code>-v</code> option, and <i>file</i> is already in the archive, the standard output	B
4	format shall be	B
5	<code>"r - %s\n", &lt;file&gt;</code>	B
6	where <i>file</i> is the operand specified on the command line.	B
7	If <i>file</i> is being added to the archive with the <code>-r</code> option, the standard output for-	B
8	mat shall be	B
9	<code>"a - %s\n", &lt;file&gt;</code>	B
10	where <i>file</i> is the operand specified on the command line.	B
11	<i>to:</i>	B
12	If the <code>-r</code> option is used with the <code>-v</code> option:	B
13	(1) If <i>file</i> is already in the archive, the standard output format shall be	B
14	<code>"r - %s\n", &lt;file&gt;</code>	B
15	where <i>&lt;file&gt;</i> is the operand specified on the command line.	B
16	If <i>file</i> is not already in the archive, the standard output format shall be	B
17	<code>"a - %s\n", &lt;file&gt;</code>	B
18	where <i>&lt;file&gt;</i> is the operand specified on the command line.	B
19	<b>Rationale:</b> This change is the result of interpretation request PASC 1003.2-	B
20	92 #92 submitted for IEEE Std 1003.2-1992.	B

## 21 **6.2 make – Maintain, update, and regenerate groups of programs** B

22 **Rationale:** The changes to `make` are the result of interpretation requests PASC B  
 23 1003.2-92 #94, #100, and #113 submitted for IEEE Std 1003.2-1992. The large B  
 24 majority of these changes change the term “command line” to be specific in each B  
 25 case about whether it is a “`make` utility command line” or a “`makefile` command B  
 26 line.” To avoid clutter, it is not further diffmarked. B

27 ⇒ **6.2.3 make Options.** *Change the text from option `-q` to the end of the sub-*  
 28 *clause to:*

29 `-q` Return a zero exit value if the target file is up-to-date; other-  
 30 wise, return an exit value of 1. Targets shall not be updated if  
 31 this option is specified. However, a `makefile` command line  
 32 (associated with the targets) with a plus-sign (+) prefix shall  
 33 be executed.

34 `-r` Clear the suffix list and do not use the built-in rules.

35 `-S` Terminate `make` if an error occurs while executing the com-  
 36 mands to bring a target up-to-date. This shall be the default  
 37 and the opposite of `-k`.

38 `-s` Do not write `makefile` command lines or touch messages (see  
 39 `-t`) to standard output before executing. This mode shall be  
 40 the same as if the special target `.SILENT` were specified  
 41 without prerequisites. See 6.2.7.2.

42 `-t` Update the modification time of each target as though a `touch`  
 43 *target* had been executed. See `touch` in 4.63. Targets that  
 44 have prerequisites but no commands (see 6.2.7.3), or that are  
 45 already up-to-date, shall not be touched in this manner. Write  
 46 messages to standard output for each target file, indicating the  
 47 name of the file and that it was touched. Normally, the  
 48 `makefile` command lines associated with each target are not  
 49 executed. However, a `makefile` command line with a plus-sign  
 50 (+) prefix shall be executed.

51 Any options specified in the **MAKEFLAGS** environment variable shall be  
 52 evaluated before any options specified on the `make` utility command line. If the  
 53 `-k` and `-S` options are both specified on the `make` utility command line or by the  
 54 **MAKEFLAGS** environment variable, the last option specified shall take pre-  
 55 cedence. If the `-f` or `-p` options appear in the **MAKEFLAGS** environment vari-  
 56 able, the result is undefined.

57 ⇒ **6.2.4 make Operands.** *Change the final paragraph to:*

58 If the *target\_name* and *macro=name* operands are intermixed on the `make` util-  
59 ity command line, the results are unspecified.

60 ⇒ **6.2.5.3 make Environment Variables.** *Change the text from variable*  
61 **MAKEFLAGS** *to the end of the subclause to:*

62           **MAKEFLAGS**           This variable shall be interpreted as a character string  
63 representing a series of option characters to be used as  
64 the default options. The implementation shall accept  
65 both of the following formats (but need not accept them  
66 when intermixed):

67                           (1) The characters are option letters without the  
68 leading hyphens or <blank> separation used on a  
69 `make` utility command line.

70                           (2) The characters are formatted in a manner similar  
71 to a portion of the `make` utility command line:  
72 options are preceded by hyphens and <blank>-  
73 separated as described in 2.10.2. The  
74 *macro=name* macro definition operands can also  
75 be included. The difference between the contents  
76 of **MAKEFLAGS** and the `make` utility command  
77 line is that the contents of the variable shall not  
78 be subjected to the word expansions (see 3.6)  
79 associated with parsing the command-line values.

80 The value of the **SHELL** environment variable shall not be used as a macro  
81 and shall not be modified by defining the **SHELL** macro in a makefile or on the  
82 `make` utility command line. All other environment variables, including those  
83 with null values, shall be used as macros, as defined in 6.2.7.4.

84 ⇒ **6.2.6.1 make Standard Output.** *Add a new sentence to the end of the para-* C  
85 *graph:* C

86 If the `-t` option is present and a file is *touched*, `make` shall write to standard C  
87 output a message of unspecified format indicating that the file was touched, C  
88 including the filename of the file. C

89 ⇒ **6.2.6.3 make Output Files.** *Change this subclause to:* C

90 Files can be created when the `-t` option is present. Additional files can also be C  
91 created by the utilities invoked by `make`. C

92 ⇒ **6.2.7.1 Makefile Syntax.** *Change the first paragraph to:*

93 A makefile can contain rules, macro definitions (see 6.2.7.4), and comments.  
 94 There are two kinds of rules: inference rules (6.2.7.5) and target rules (6.2.7.3).  
 95 The `make` utility shall contain a set of built-in inference rules. If the `-r` option  
 96 is present, the built-in rules shall not be used and the suffix list shall be  
 97 cleared. Additional rules of both types can be specified in a makefile. If a rule  
 98 is defined more than once, the value of the rule shall be that of the last one  
 99 specified. Macros can also be defined more than once, and the value of the  
 100 macro is specified by 6.2.7.4. Comments start with a number sign (#) and con-  
 101 tinue until an unescaped `<newline>` is reached.

102 ⇒ **6.2.7.1 Makefile Syntax.** *Change the fourth paragraph (the one beginning*  
 103 *“The rules in makefiles ...”) to:*

104 The rules in makefiles shall consist of the following types of lines: target rules,  
 105 including special targets (see 6.2.7.3); inference rules (see 6.2.7.5); macro  
 106 definitions (see 6.2.7.4); empty lines; and comments.

107 ⇒ **6.2.7.1 Makefile Syntax.** *Change the fifth paragraph (the one beginning*  
 108 *“When an escaped ...”) to:*

109 When an escaped `<newline>` (one preceded by a backslash) is found anywhere  
 110 in the makefile except in a command line, it shall be replaced, along with any  
 111 leading white space on the following line, with a single `<space>`. When an  
 112 escaped `<newline>` is found in a command line in a makefile, the command  
 113 line shall contain the backslash, the `<newline>`, and the next line, except that  
 114 the first character of the next line shall not be included if it is a `<tab>`.

115 ⇒ **6.2.7.2 Makefile Execution.** *Replace this subclause with:*

116 Makefile command lines shall be processed one at a time by writing the  
 117 makefile command line to the standard output (unless one of the conditions  
 118 listed under “@” suppresses the writing) and executing the command(s) in the  
 119 line. A `<tab>` character may precede the command to standard output. Com-  
 120 mand execution shall be as if the makefile command line were the argument to  
 121 the `system()` function in POSIX.1 {8}. The environment for the command being  
 122 executed shall contain all of the variables in the environment of `make`.

123 By default, when `make` receives a nonzero status from the execution of a com-  
 124 mand, it terminates with an error message to standard error.

125 Makefile command lines can have one or more of the following prefixes: a  
 126 hyphen (-), an at sign (@), or a plus sign (+). These modify the way in which  
 127 `make` processes the command. When a command is written to standard out-  
 128 put, the prefix shall not be included in the output.

129 - If the command prefix contains a hyphen, or if the `-i` option is present,  
 130 or if the special target `.IGNORE` has either the current target as a prere-  
 131 quisite or has no prerequisites, any error found while executing the com-  
 132 mand shall be ignored.



133       @ If the command prefix contains an at sign and the `make` utility  
 134       command-line `-n` option is not specified, or the `-s` option is present, or  
 135       the special target `.SILENT` has either the current target as a prere-  
 136       quisite or has no prerequisites, the command shall not be written to  
 137       standard output before it is executed.

138       + If the command prefix contains a plus sign, this indicates a makefile  
 139       command line that shall be executed even if `-n`, `-q`, or `-t` is specified on  
 140       the `make` utility command line.

141   ⇒ **6.2.7.3 Target Rules.** *In the second paragraph (the one beginning with “Tar-*  
 142   *get entries ...”), change “command lines” to “makefile command lines.”*

143   ⇒ **6.2.7.3 Target Rules.** *Replace the list entry for `.SUFFIXES` with the following:* C

144       `.SUFFIXES` Prerequisites of `.SUFFIXES` shall be appended to the list of C  
 145       known suffixes and are used in conjunction with the inference C  
 146       rules (see 6.2.7.5). If `.SUFFIXES` does not have any prere- C  
 147       quisites, the list of known suffixes shall be cleared. C

148       The special targets `.IGNORE`, `.POSIX`, `.PRECIOUS`, `.SILENT`, and `.SUFFIXES` C  
 149       shall be specified without commands. C

150   ⇒ **6.2.7.4 Macros.** *Delete the following paragraph:*

151       Subsequent appearances of  $\$(string1)$  or  $\${string1}$  shall be replaced by  
 152        $string2$ . The parentheses or braces are optional if  $string1$  is a single character.  
 153       The macro  $$$$  shall be replaced by the single character  $\$$ .

154   ⇒ **6.2.7.4 Macros.** *Change the fifth paragraph (the one beginning “Macros can*  
 155   *appear anywhere ...”) to:*

156       Macros can appear anywhere in the makefile.  $\$(string1)$  or  $\${string1}$  shall  
 157       be replaced by  $string2$ , as follows:

158       (1) Macros in target lines shall be evaluated when the target line is read. C

159       (2) Macros in makefile command lines shall be evaluated when the command C  
 160       is executed.

161       (3) Macros in the string before the equals sign in a macro definition shall be  
 162       evaluated when the macro assignment is made.

163       (4) Macros after the equals sign in a macro definition shall not be evaluated  
 164       until the defined macro is used in a rule or command, or before the  
 165       equals sign in a macro definition.

166       The parentheses or braces are optional if  $string1$  is a single character. The macro  
 167        $$$$  shall be replaced by the single character  $\$$ .

168 ⇒ **6.2.7.4 Macros.** *Change the sixth through eleventh paragraphs (“Macro*  
169 *assignments ... <numbered list> ... shall be reversed.”) to:*

170 Macro definitions shall be taken from the following sources, in the following  
171 logical order, before the makefile(s) are read.

172 (1) Macros specified on the `make` utility command line, in the order specified  
173 on the command line. It is unspecified whether the internal macros  
174 defined in 6.2.7.7 are accepted from this source.

175 (2) Macros defined by the **MAKEFLAGS** environment variable, in the order  
176 specified in the environment variable. It is unspecified whether the  
177 internal macros defined in 6.2.7.7 are accepted from this source.

178 (3) The contents of the environment, excluding the **MAKEFLAGS** and  
179 **SHELL** variables and including the variables with null values.

180 (4) Macros defined in the inference rules built into `make`.

181 Macro definitions from these sources shall not override macro definitions from a  
182 lower-numbered source. Macro definitions from a single source (e.g., the `make`  
183 utility command line, the **MAKEFLAGS** environment variable or the other  
184 environment variables) shall override previous macro definitions from the same  
185 source.

186 Macros defined in the makefile(s) shall override macro definitions that occur  
187 before them in the makefile(s) and macro definitions from source (4). If the `-e`  
188 option is not specified, macros defined in the makefile(s) shall override macro  
189 definitions from source (3). Macros defined in the makefile(s) shall not override  
190 macro definitions from source (1) or source (2).

191 Before the makefile(s) are read, all of the `make` utility command-line options  
192 (except `-f` and `-p`) and `make` utility command-line macro definitions (except any  
193 for the **MAKEFLAGS** macro), not already included in the **MAKEFLAGS** macro, shall be  
194 added to the **MAKEFLAGS** macro. Other implementation-defined options and mac-  
195 ros may also be added to the **MAKEFLAGS** macro. If this modifies the value  
196 **MAKEFLAGS** macro, or, if the **MAKEFLAGS** macro is modified at any subsequent time,  
197 the **MAKEFLAGS** environment variable shall be modified to match the new value  
198 of the **MAKEFLAGS** macro.

199 Before the makefile(s) are read, all of the `make` utility command-line macro  
200 definitions (except the **MAKEFLAGS** macro or the **SHELL** macro) shall be added  
201 to the environment of `make`. Other implementation-defined variables may also be  
202 added to the environment of `make`.

203 ⇒ **6.2.7.7 Internal Macros.** *Change the description of \$< to:*

204       \$<    In an inference rule, the \$< macro shall evaluate to the file name  
205            whose existence allowed the inference rule to be chosen for the tar-  
206            get. In the .DEFAULT rule, the \$< macro shall evaluate to the current  
207            target name. The meaning of the \$< macro is otherwise  
208            unspecified.

209            For example, in the .c.a inference rule, \$< represents the prere-  
210            quisite .c file.



## **Section 7: Revisions to Language-Independent System Services**

1 *Editor's Note: Remove this section. It is no longer required due to the movement of*  
2 *APIs from this standard to POSIX.1 {8}.*



## Annex A (normative)

### Revisions to C Language Development Utilities Option

#### 1    **A.1 c89 – Compile Standard C programs**

2    ⇒ **A.1.7.1 c89 Standard Libraries.** *Change the description of `-lc` to:*

3           `-lc`    This library contains all mandatory (i.e., nonoptional) library    C  
4                    functions referenced in `<stdlib.h>`, `<stdio.h>`, `<time.h>`,    C  
5                    `<setjmp.h>`, `<signal.h>`, `<unistd.h>`, `<sys/types.h>`,  
6                    `<string.h>`, and `<ctype.h>`, except for those functions refer-  
7                    enced in `<math.h>`. There may be additional functions included;  
8                    section 2.9.3 of POSIX.1 {8} describes constants that indicate the  
9                    presence of optional facilities, and these constants can be used  
10                   with `getconf` to determine whether those functions are included  
11                   in the library accessed by `-lc`. For example, if an invocation of  
12                    `getconf _POSIX_VERSION`  
13                    exits with a status of zero, the library searched also shall include  
14                    all mandatory (nonoptional) functions defined by ISO/IEC 9945-    C  
15                    1:1990; if the status is nonzero, it is unspecified whether these  
16                    functions are available. An implementation shall not require this  
17                    operand to be present to cause a search of this library.

18    ⇒ **A.1.7.1 c89 Standard Libraries.** *Add to the end of the subclause:*    C

19            All other libraries that shall be specified when building a POSIX.1 {8} conform-    C  
20            ing application are those listed in POSIX.1 {8} subclause 2.7.3, Headers and    C  
21            Function Prototypes. *Editor's Note: The table referenced is in fact to be found*    C  
22            *in POSIX.1a draft 17 onwards.*    C

23 **Rationale:** Since Annex B is gone, all references to it have to be removed and a  
 24 more generic statement of the interaction with POSIX.1 {8} has been included.

### 25 **A.3 yacc – Yet another compiler compiler**

26 ⇒ **A.3.6.3.1 yacc Code File.** *Delete the second paragraph, which is:* B

27 The contents of the program section (see A.3.7.1.4) of the input file shall then B  
 28 be included. B

29 ⇒ **A.3.7.1.4 yacc Programs Section.** *Change this subclause to:* B

30 The *programs* section can include the definition of the lexical analyzer *yylex()* B  
 31 and any other functions; for example, those used in the actions specified in the B  
 32 grammar rules. It is unspecified whether the programs section precedes or fol- B  
 33 lows the semantic actions in the output file; therefore, if the application con- B  
 34 tains any macro definitions and declarations intended to apply to the code in B  
 35 the semantic actions, it shall place them within `%{ ... %}` in the declarations B  
 36 section. B

37 **Rationale:** The preceding changes are the result of interpretation request PASC B  
 38 1003.2-92 #93 submitted for IEEE Std 1003.2-1992. B

39 ⇒ **A.3.7.4 Interface to the Lexical Analyzer.** *In the third paragraph (the one B  
 40 beginning “If the token numbers ...”), change the sentence “All assigned token B  
 41 numbers shall be unique and distinct from the token numbers used for B  
 42 literals.” to:* B

43 All token numbers assigned by *yacc* shall be unique and distinct from the B  
 44 token numbers used for literals and user assigned tokens. B

45 **Rationale:** This change is the result of interpretation request PASC 1003.2-92 B  
 46 #104 submitted for IEEE Std 1003.2-1992. B



**Annex B**  
(normative)

**Revisions to C Language Bindings Option**

1 *Editor's Note: Replace the text of this entire annex with the following. (It is no*  
2 *longer required due to the movement of APIs from this standard to POSIX.1 {8}.*  
3 *Unlike Section 7, it is not being removed because we wish to avoid renumbering all*  
4 *of the following annexes.)*

5 This annex is unused.



**Annex C**  
(normative)

**Revisions to FORTRAN Development and Runtime  
Utilities Options**

- 1 There are no revisions to Annex C.



## Annex D (informative)

### Revisions to Bibliography

1 ⇒ **D Bibliography.** *Remove the entry for ISO/IEC 10646-1.*

2 **Rationale:** The entry for this standard has been moved into the normative  
3 references.

4 ⇒ **D Bibliography.** *Add the following entry in the proper order:*

5	{B90} RFC 2045, Freed, N., Borenstein, N. <i>Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies</i>	C
6		C
7	{B91} ISO/IEC 14652: 199?, <i>Functionality for internationalization—</i>	C
8	<i>Specification of cultural conventions</i>	C
9	{B92} ISO/IEC 15435: 199?, <i>Information technology—Internationalization APIs</i>	C
10	{B93} ISO/IEC 15897: 199?, <i>Information technology—Procedures for European</i>	C
11	<i>registration of cultural elements</i>	C



## Annex E (informative)

### Revisions to Rationale and Notes

1 ⇒ **E Rationale and Notes.** *Remove all references to the C-Language Binding*  
 2 *Option and {POSIX2\_C\_BIND} from this annex, or reword to indicate they have*  
 3 *moved to P1003.1a. Reword all references to language-independent functions* B  
 4 *in Chapter 7 to use the POSIX.1 {8} function names.* B

5 **Rationale:** Since Chapter 7 and Annex B are gone, all references to them have to B  
 6 be removed. B

7 ⇒ **E.2.2.2 General Terms.** *Add the following rationale text at the end of this*  
 8 *subclause, immediately preceding E.2.2.3.*

#### 9 ***Symbolic Links***

10 Symbolic link support was added to the first revision of this standard to  
 11 achieve synchronization with IEEE Std 1003.1-199x. This entailed a  
 12 significant number of small changes to many interfaces.

13 Because a symbolic link and its referenced object coexist in the file system  
 14 name space, confusion can arise in distinguishing between the link itself and  
 15 the referenced object. Historically, utilities and system calls have adopted  
 16 their own link following conventions in a somewhat ad hoc fashion. Rules for a  
 17 uniform approach are outlined here, although historical practice has been  
 18 adhered to as much as was possible. To promote consistent system use, user-  
 19 written utilities are encouraged to follow these same rules.

20 Symbolic links are handled either by operating on the link itself, or by operat-  
 21 ing on the object referenced by the link. In the latter case, an application or  
 22 system call is said to “follow” the link. Symbolic links may reference other  
 23 symbolic links, in which case links are dereferenced until an object that is not  
 24 a symbolic link is found, a symbolic link that references a file that does not  
 25 exist is found, or a loop is detected. (Current implementations do not detect  
 26 loops, but have a limit on the number of symbolic links that they will derefer-  
 27 ence before declaring it an error.)

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

28 There are four domains for which default symbolic link policy is established in  
 29 a system. In almost all cases, there are utility options that override this  
 30 default behavior. The four domains are as follows:

- 31 (1) Symbolic links specified to system calls that take file name arguments
- 32 (2) Symbolic links specified as command-line file name arguments to utilities  
 33 that are not performing a traversal of a file hierarchy
- 34 (3) Symbolic links referencing files not of type directory, specified to utilities  
 35 that are performing a traversal of a file hierarchy
- 36 (4) Symbolic links referencing files of type directory, specified to utilities that  
 37 are performing a traversal of a file hierarchy

### 38 *First Domain*

39 The first domain is not within the scope of this standard.

### 40 *Second Domain*

41 The reason this category is restricted to utilities that are not traversing the file  
 42 hierarchy is that some standard utilities take an option that specifies a hierarchi-  
 43 cal traversal, but by default operate on the arguments themselves. Generally,  
 44 users specifying the option for a file hierarchy traversal wish to operate on a sin-  
 45 gular, physical hierarchy, and therefore symbolic links, which may reference files  
 46 outside of the hierarchy, are ignored. For example, `chown owner file` is a different  
 47 operation from the same command with the `-R` option specified. In this example,  
 48 the behavior of the command `chown owner file` is described here, while the  
 49 behavior of the command `chown -R owner file` is described in the third and fourth  
 50 domains.

51 The general rule is that the utilities in this category follow symbolic links named  
 52 as arguments.

53 Exceptions in the second domain are:

- 54 — The `mv` and `rm` utilities do not follow symbolic links named as arguments,  
 55 but respectively attempt to rename or delete them.
- 56 — The `ls` utility is also an exception to this rule. For compatibility with his-  
 57 torical systems, when the `-R` option is not specified, the `ls` utility follows  
 58 symbolic links named as arguments if the `-L` option is specified or if the `-F`,  
 59 `-d`, or `-l` options are not specified. (If the `-L` option is specified, `ls` always  
 60 follows symbolic links; it is the only utility where the `-L` option affects its  
 61 behavior even though a tree walk is not being performed.)

62 All other standard utilities, when not traversing a file hierarchy, always follow  
 63 symbolic links named as arguments.

64 Historical practice is that the `-h` option is specified if standard utilities are to act  
 65 upon symbolic links instead of upon their targets. Examples of commands that  
 66 have historically had a `-h` option for this purpose are the `chgrp`, `chown`, `file`,  
 67 and `test` utilities.



68 *Third Domain*

69 The third domain is symbolic links, referencing files not of type directory,  
70 specified to utilities that are performing a traversal of a file hierarchy. (This  
71 includes symbolic links specified as command-line file name arguments or encoun-  
72 tered during the traversal.)

73 The intention of POSIX.2 is that the operation that the utility is performing is  
74 applied to the symbolic link itself, if that operation is applicable to symbolic links.  
75 The reason that the operation is not required is that symbolic links in some sys-  
76 tems do not have such attributes as a file owner, and therefore the `chown` opera-  
77 tion would be meaningless. If symbolic links on the system have an owner, it is  
78 the intention that the utility `chown` cause the owner of the symbolic link to  
79 change. If symbolic links do not have an owner, the symbolic link should be  
80 ignored. Specifically, by default, no change should be made to the file referenced  
81 by the symbolic link.

82 *Fourth Domain*

83 The fourth domain is symbolic links referencing files of type directory, specified to  
84 utilities that are performing a traversal of a file hierarchy. (This includes sym-  
85 bolic links specified as command-line file name arguments or encountered during  
86 the traversal.)

87 All standard utilities do not, by default, indirect into the file hierarchy referenced  
88 by the symbolic link. (POSIX.2 uses the informal term “physical walk” to describe  
89 this case. The case where the utility does indirect through the symbolic link is  
90 termed a “logical walk.”)

91 There are three reasons for the default to a physical walk.

- 92 — With very few exceptions, a physical walk has been the historical default on  
93 UNIX systems supporting symbolic links. Because some utilities (i.e., `rm`)  
94 must default to a physical walk, regardless, changing historical practice in  
95 this regard would be confusing to users and needlessly incompatible.
- 96 — For systems where symbolic links have the historical file attributes (i.e.,  
97 owner, group, mode), defaulting to a logical traversal would require the  
98 addition of a new option to the commands to modify the attributes of the  
99 link itself. This is painful and more complex than the alternatives.
- 100 — There is a security issue with defaulting to a logical walk. Historically, the  
101 command `chown -R user file` has been safe for the super-user because `setuid`  
102 and `setgid` bits were lost when the ownership of the file was changed. If the  
103 walk were logical, changing ownership would no longer be safe because a  
104 user might have inserted a symbolic link pointing to any file in the tree.  
105 Again, this would necessitate the addition of an option to the commands  
106 doing hierarchy traversal to not indirect through the symbolic links, and  
107 historical scripts doing recursive walks would instantly become security  
108 problems. While this is mostly an issue for system administrators, it is  
109 preferable to not have different defaults for different classes of users.

110 As consistently as possible, users may cause standard utilities performing a file  
 111 hierarchy traversal to follow any symbolic links named on the command line,  
 112 regardless of the type of file they reference, by specifying the `-H` (for “half logical”)  
 113 option. This option is intended to make the command-line name space look like  
 114 the logical name space.

115 As consistently as possible, users may cause standard utilities performing a file  
 116 hierarchy traversal to follow any symbolic links named on the command line as  
 117 well as any symbolic links encountered during the traversal, regardless of the  
 118 type of file they reference, by specifying the `-L` (for “logical”) option. This option is  
 119 intended to make the entire name space look like the logical name space.

120 For consistency, implementors are encouraged to use the `-P` (for “physical”) flag to  
 121 specify the physical walk in utilities that do logical walks by default for whatever  
 122 reason. The only standard utilities that require the `-P` option are `cd` and `pwd`; see C  
 123 the note below. C

124 When one or more of the `-H`, `-L`, and `-P` flags can be specified, the last one B  
 125 specified determines the behavior of the utility. This permits users to alias com-  
 126 mands so that the default behavior is a logical walk and then override that  
 127 behavior on the command line.

#### 128 *Exceptions in the Third and Fourth Domains*

129 To maintain compatibility with historical systems, the `ls` and `rm` utilities are  
 130 exceptions to these rules.

131 The `rm` utility never follows symbolic links and does not support the `-H`, `-L`, or `-P`  
 132 options.

133 The `ls` utility never follows symbolic links unless the `-L` option is specified, when  
 134 it follows all of the symbolic links, regardless of their type or if specified on the  
 135 command line or encountered in the traversal. The `ls` utility does not support  
 136 the `-H` and `-P` options.

137 POSIX.2 requires that the standard utilities `ls`, `find`, and `pax` detect infinite  
 138 loops when doing logical walks; i.e., a directory, or more commonly a symbolic  
 139 link, that refers to an ancestor in the current file hierarchy. If the file system  
 140 itself is corrupted, causing the infinite loop, it may be impossible to recover.  
 141 Because `find` and `ls` are often used in system administration and security appli-  
 142 cations, they should attempt to recover and continue as best as they can. The `pax`  
 143 utility should terminate because the archive it was creating is by definition cor-  
 144 rupted. Other, less vital, utilities should probably simply terminate as well.  
 145 Implementations are strongly encouraged to detect infinite loops in all utilities.

146 Historical practice is shown in Table E-100. The heading SVID3 stands for the  
 147 Third Edition of the System V Interface Definition {B37}.

148 Historically, several shells have had built-in versions of the `pwd` utility. In some  
 149 of these shells, `pwd` reported the physical path, and in others, the logical path. C  
 150 Implementations of the shell corresponding to this standard must report the logi- C  
 151 cal path by default. Earlier versions of this standard did not require the `pwd` util- C  
 152 ity to be a built-in utility. Now that `pwd` is required to set an environment C

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

153

**Table E-100 – Historical Practice for Symbolic Links**

Utility	SVID3	4.3BSD	4.4BSD	POSIX	Comments
154 cd				-L	Treat “.” logically
155 cd				-P	Treat “.” physically
156 chgrp			-H	-H	Follow command-line symlinks
157 chgrp			-h	-L	Follow symlinks
158 chgrp	-h			-h	Affect the symlink
159 chmod				-h	Affect the symlink
160 chmod			-H	-H	Follow command-line symlinks
161 chmod			-h	-L	Follow symlinks
162 chown			-H	-H	Follow command-line symlinks
163 chown			-h	-L	Follow symlinks
164 chown	-h			-h	Affect the symlink
165 cp			-H	-H	Follow command-line symlinks
166 cp			-h	-L	Follow symlinks
167 cpio	-L		-L		Follow symlinks
168 du			-H	-H	Follow command-line symlinks
169 du			-h	-L	Follow symlinks
170 file	-h			-h	Affect the symlink
171 find			-H	-H	Follow command-line symlinks
172 find			-h	-L	Follow symlinks
173 find	-follow		-follow	-follow	Follow symlinks
174 ln	-s	-s	-s	-s	Create a symbolic link
175 ls	-L	-L	-L	-L	Follow symlinks
176 ls				-H	Follow command-line symlinks
177 mv					Operates on the symlink
178 pax			-H	-H	Follow command-line symlinks
179 pax			-h	-L	Follow symlinks
180 pwd				-L	Printed path may contain symlinks
181 pwd				-P	Printed path will not contain symlinks
182 rm					Operates on the symlink
183 tar			-H		Follow command-line symlinks
184 tar		-h	-h		Follow symlinks
185 test	-h		-h	-h	Affect the symlink

187 variable in the current shell execution environment, it must be a built-in utility. C

188 C

189 The `cd` command is required, by default, to treat the string “.” logically. Implementors are required to support the `-P` flag in `cd` so that users can have their current environment handled physically. C  
C  
C

192 In 4.3BSD, `chgrp` during tree traversal changed the group of the symbolic link, not the target. Symbolic links in 4.4BSD do not have owner, group, mode, or other standard UNIX system file attributes.

195 The only significant work required for vendors to conform to this standard will be to add the `-H` and `-L` options to the eight standard utilities that will require them. C

197 ⇒ **E.2.5.2.2 LC\_COLLATE.** *Change the second-to-last paragraph to:*

198 The character (and collating element) order is defined by the order in which  
 199 characters and elements are specified between the `order_start` and `order_-`  
 200 `end` keywords. This character order is used in range expressions in REs (see  
 201 2.8). Weights assigned to the characters and elements define the collation  
 202 sequence; in the absence of weights, the character order is also the collation  
 203 sequence. For two elements that have the same primary, secondary, and terti-  
 204 ary weights, the character order is also the collation sequence.

205 ⇒ **E.3.6.2 Parameter Expansion.** *In Table E-1, change the fourth row as fol-*  
 206 *lows:*

	<i>parameter set and not null</i>	<i>parameter set but null</i>	<i>parameter unset</i>
$\$ \{ parameter=word \}$	substitute <i>parameter</i>	substitute null	assign <i>word</i>

211 **Rationale:** This change is the result of interpretation request PASC 1003.2-92  
 212 #48 submitted for IEEE Std 1003.2-1992.

213 ⇒ **E.4.48 pax Rationale.** *Replace the full rationale for `pax` with the following.* B

#### 214 **E.4.48 pax – Portable archive interchange**

215 The `pax` utility was commissioned for POSIX.2-1992. It represented a peaceful B  
 216 compromise between advocates of the historical `tar` and `cpio` utilities. B

217 A fundamental difference between `cpio` and `tar` was in the way directories were  
 218 treated. The `cpio` utility did not treat directories differently from other files, and  
 219 to select a directory and its contents required that each file in the hierarchy be  
 220 explicitly specified. For `tar`, a directory matched every file in the file hierarchy it  
 221 rooted.

222 The `pax` utility offers both interfaces; by default, directories map into the file  
 223 hierarchy they root. The `-d` option causes `pax` to skip any file not explicitly refer-  
 224 enced, as `cpio` historically did. The `tar`-style behavior was chosen as the default  
 225 because it was believed that this was the more common usage and because `tar` is  
 226 the more commonly available interface (being provided historically on both  
 227 System V and BSD implementations). B

228 The data interchange format specification originally published in Section 10 of B  
 229 POSIX.1 {8} required that processes with “appropriate privileges” always shall B  
 230 restore the ownership and permissions of extracted files exactly as archived. If  
 231 viewed from the historic equivalence between super-user and “appropriate

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

232 privileges,” there are two problems with this requirement. First, users running  
 233 as super-users may unknowingly set dangerous permissions on extracted files.  
 234 Second, it is needlessly limiting in that super-users cannot extract files and own  
 235 them as super-user unless the archive was created by the super-user. (It should  
 236 be noted that restoration of ownerships and permissions for the super-user, by  
 237 default, is historical practice in `cpio`, but not in `tar`.) In order to avoid these two  
 238 problems, the `pax` specification has an additional “privilege” mechanism, the `-p`  
 239 option. Only a `pax` invocation with the POSIX.1 {8} privileges needed, and which  
 240 has the `-p` option set using the `e` specification character, has the “appropriate  
 241 privilege” to restore full ownership and permission information.

242 Note also that POSIX.1 {8} Section 10.1 requires that the file ownership and access  
 243 permissions shall be set, on extraction, in the same fashion as the POSIX.1 {8}  
 244 `creat()` function when provided the mode stored in the archive. This means that  
 245 the file creation mask of the user is applied to the file permissions.

246 Users should note that directories may be created by `pax` while extracting files C  
 247 with permissions that are different from those that existed at the time the archive C  
 248 was created. When extracting sensitive information into a directory hierarchy C  
 249 that no longer exists, users are encouraged to set their file creation mask C  
 250 appropriately to protect these files during extraction. C

251 The table of contents output is written to standard output to facilitate pipeline  
 252 processing.

253 B

254 The one pathname per line format of standard input precludes pathnames con-  
 255 taining `<newline>`s. Although such pathnames violate the portable filename  
 256 guidelines, they may exist and their presence may inhibit usage of `pax` within  
 257 shell scripts. This problem is inherited from historical archive programs. The  
 258 problem can be avoided by listing filename arguments on the command line  
 259 instead of on standard input.

260 A pre-1992 draft had hard links displaying for all pathnames. This was removed  
 261 because it complicates the output of the case where `-v` is not specified and does  
 262 not match historical `cpio` usage. The hard-link information is available in the `-v`  
 263 display.

264 B

265 The archive formats inherited from POSIX.1 {8} have certain restrictions that have B  
 266 been brought along from historical usage. For example, there are restrictions on B  
 267 the length of pathnames stored in the archive. When `pax` is used in copy (`-rw`) B  
 268 mode (copying directory hierarchies), the ability to use extensions from the B  
 269 `-x pax` format overcomes these restrictions. B

270 The default `blocksize` value of 5120 B for `cpio` was selected because it is one of  
 271 the standard block-size values for `cpio`, set when the `-B` option is specified. (The  
 272 other default block-size value for `cpio` is 512 B, and this was considered to be too  
 273 small.) The default block value of 10 240 B for `tar` was selected because that is  
 274 the standard block-size value for BSD `tar`. The maximum block size of 32 256 B  
 275 ( $2^{15}$ -512 B) is the largest multiple of 512 B that fits into a signed 16 b tape

276 controller transfer register. There are known limitations in some historical sys-  
 277 tems that would prevent larger blocks from being accepted. Historical values  
 278 were chosen to improve compatibility with historical scripts using `dd` or similar  
 279 utilities to manipulate archives. Also, default block sizes for any file type other  
 280 than character special file has been deleted from POSIX.2 as unimportant and not  
 281 likely to affect the structure of the resulting archive.

282 Implementations are permitted to modify the block-size value based on the  
 283 archive format or the device to which the archive is being written. This is to pro-  
 284 vide implementations the opportunity to take advantage of special types of dev-  
 285 ices, and it should not be used without a great deal of consideration because it  
 286 will almost certainly decrease archive portability.

287 The intended use of the `-n` option was to permit extraction of one or more files B  
 288 from the archive without processing the entire archive. This was viewed by the B  
 289 standard developers as offering significant performance advantages over histori- B  
 290 cal implementations. The `-n` option in pre-1992 drafts had three effects; the first B  
 291 was to cause special characters in patterns to not be treated specially. The second  
 292 was to cause only the first file that matched a pattern to be extracted. The third  
 293 was to cause `pax` to write a diagnostic message to standard error when no file was  
 294 found matching a specified pattern. Only the second behavior is retained by  
 295 POSIX.2, for many reasons. First, it is in general not acceptable for a single  
 296 option to have multiple effects. Second, the ability to make pattern matching  
 297 characters act as normal characters is useful for parts of `pax` other than file  
 298 extraction. Third, a finer degree of control over the special characters is useful  
 299 because users may wish to normalize only a single special character in a single  
 300 file name. Fourth, given a more general escape mechanism, the previous behavior  
 301 of the `-n` option can be easily obtained using the `-s` option or a `sed` script.  
 302 Finally, writing a diagnostic message when a pattern specified by the user is  
 303 unmatched by any file is useful behavior in all cases. In this version of POSIX.2, B  
 304 the `-n` was removed from the copy mode synopsis of `pax`; it is inapplicable B  
 305 because there are no *pattern* operands specified in this mode. B

306 There is another method than `pax` for copying subtrees in POSIX.2, described as B  
 307 part of the `cp` utility (see 4.13). Both methods are historical practice: `cp` provides  
 308 a simpler, more intuitive interface, while `pax` offers a finer granularity of control.  
 309 Each provides additional functionality to the other; in particular, `pax` maintains  
 310 the hard-link structure of the hierarchy while `cp` does not. It is the intention of  
 311 the standard developers that the results be similar (using appropriate option com-  
 312 binations in both utilities). The results are not required to be identical; there  
 313 seemed insufficient gain to applications to balance the difficulty of implementa-  
 314 tions having to guarantee that the results would be exactly identical.

315 A single archive may span more than one file. It is suggested that implementa- B  
 316 tions provide informative messages to the user on standard error whenever the B  
 317 archive file is changed.

318 The `-d` option (do not create intermediate directories not listed in the archive)  
 319 found in pre-1992 drafts was originally provided as a complement to the historical  
 320 `-d` option of `cpio`. It has been deleted.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

321 The `-s` option in pre-1992 drafts specified a subset of the substitution command  
 322 from the `ed` utility. As there was no reason for only a subset to be supported, the  
 323 `-s` option is now compatible with the current `ed` specification. Since the delimiter  
 324 can be any nonnull character, the following usage with single spaces is valid:

```
325     pax -s " foo bar " ...
```

326 The `-t` option (specify an implementation-defined identifier naming an input or  
 327 output device) found in pre-1992 drafts has been deleted because it is not histori-  
 328 cal practice and is of limited utility. In particular, historic versions of neither  
 329 `cpio` nor `tar` had the concept of devices that were not mapped into the file sys-  
 330 tem; if the devices are mapped into the file system, the `-f` option is sufficient.

331

B

332 The default behavior of `pax` with regard to file modification times is the same as  
 333 historical implementations of `tar`. It is not the historical behavior of `cpio`.

334 Because the `-i` option uses `/dev/tty`, utilities without a controlling terminal will  
 335 not be able to use this option. Implementations are allowed, but not required, to  
 336 keep track of interactively renamed files, allowing for the processing of links to  
 337 those files.

338 The `-y` option, found in pre-1992 drafts, has been deleted because a line contain-  
 339 ing a single period for the `-i` option has equivalent functionality. The special  
 340 lines for the `-i` option (a single period and the empty line) are historical practice  
 341 in `cpio`.

342 In pre-1992 drafts, an `-e charmap` option was included to increase portability of  
 343 files between systems using different coded character sets. This option was omit-  
 344 ted because it was apparent that consensus could not be formed for it. In this ver-  
 345 sion of POSIX.2, the use of UTF8 should be an adequate substitute.

B

B

B

B

346 The `-k` option was added to address international concerns about the dangers  
 347 involved in the character set transformations of `-e` (if the target character set  
 348 were different than the source, the file names might be transformed into names  
 349 matching existing files) and also was made more general to protect files  
 350 transferred between file systems with different `{NAME_MAX}` values (truncating a  
 351 filename on a smaller system might also inadvertently overwrite existing files).  
 352 As stated, it prevents any overwriting, even if the target file is older than the  
 353 source. This version of POSIX.2 adds more granularity of options to solve this  
 354 problem by introducing the `-o invalid=` option—specifically the UTF8 action.  
 355 (Note that an existing file that is named with a UTF8 encoding is still subject to  
 356 overwriting in this case. The `-k` option closes that loophole.)

B

B

B

B

357 It is almost certain that appropriate privileges will be required for `pax` to accom-  
 358 plish parts of this specification. Specifically, creating files of type block special or  
 359 character special, restoring file access times unless the files are owned by the user  
 360 (the `-t` option), or preserving file owner, group, and mode (the `-p` option) will all  
 361 probably require appropriate privileges.

362 Some of the file characteristics referenced in this standard may not be supported  
 363 by some archive formats. For example, neither the `tar` nor `cpio` formats contain

364 the file access time. For this reason, the `e` specification character has been pro-  
 365 vided, intended to cause all file characteristics specified in the archive to be  
 366 retained.

367 It is required that extracted directories, by default, have their access and  
 368 modification times and permissions set to the values specified in the archive.  
 369 This has obvious problems in that the directories are almost certainly modified  
 370 after being extracted and that directory permissions may not permit file creation.  
 371 One possible solution is to create directories with the mode specified in the  
 372 archive, as modified by the `umask` of the user, with sufficient permissions to allow  
 373 file creation. After all files have been extracted, `pax` would then reset the access  
 374 and modification times and permissions as necessary.

375 In read mode, implementations are permitted to overwrite files when the archive  
 376 has multiple members with the same name. This may fail, of course, if permis-  
 377 sions on the first version of the file do not permit it to be overwritten.

378 The `-p` (privileges) option was invented to reconcile differences between historical  
 379 `tar` and `cpio` implementations. In particular, the two utilities used `-m` in  
 380 diametrically opposed ways. The `-p` option also provides a consistent means of  
 381 extending the ways in which future file attributes can be addressed, such as for  
 382 enhanced security systems or high-performance files. There are two modes that  
 383 will be most commonly used:

384     `-p e`     “Preserve everything.” This would be used by the historical super-  
 385                 user, someone with all the appropriate privileges, to preserve all  
 386                 aspects of the files as they are recorded in the archive. The `e` flag is  
 387                 the sum of `o` and `p`, and other implementation-defined attributes.

388     `-p p`     “Preserve” the file mode bits. This would be used by the user with  
 389                 regular privileges who wished to preserve aspects of the file other  
 390                 than the ownership. The file times are preserved by default, but two  
 391                 other flags are offered to disable these and use the time of extraction.

392 The list-mode formatting description in 4.48.3.1 borrows heavily from the one  
 393 defined by the `printf` utility. However, since there is no separate operand list to  
 394 get conversion arguments, the format was extended to allow specifying the name  
 395 of the conversion argument as part of the conversion specification.

396 The `T` specifier allows time fields to be displayed in any of the date formats.     B  
 397 Unlike the `ls` utility, `pax` does not adjust the format when the date is less than     B  
 398 six months in the past. This makes parsing the output more predictable.             B

399 The `M` specifier handles the ten-character prefix field for type and permissions  
 400 used with `ls`.

401 The `D` specifier handles the ability to display the major/minor or file size, as with  
 402 `ls`, by using `%-8(size)D`.

403 The `L` specifier handles the `ls` display for symbolic links.



404 Conversion specifiers were added to generate existing known types used for `ls`. B  
 405 *Examples* B

406 To copy the contents of the current directory to tape drive 1, medium density  
 407 (assuming historical System V device naming procedures; the historical BSD dev-  
 408 ice name would be `/dev/rmt9`):

```
409     pax -w -f /dev/rmt/1m .
```

410 To copy the *olddir* directory hierarchy to *newdir*:

```
411     mkdir newdir
412     pax -rw olddir newdir
```

413 To read the archive `a.pax`, with all files rooted in `"/usr"` in the archive extracted  
 414 relative to the current directory:

```
415     pax -r -s '^/*usr/*,, ' -f a.pax
```

416 Using the option

```
417     -o listopt="%M %(atime)T %(size)D %(name)s"
```

418 overrides the default output description in Standard Output and instead writes

```
419     -rw-rw--- Jan 12 15:53 1492 /usr/foo/bar
```

420 Using the options

```
421     -o listopt='%L\t%(size)D\n%.7' \
422     -o listopt='(name)s\n%(ctime)T\n%T'
```

423 overrides the default output description in Standard Output and instead writes

```
424     /usr/foo/bar -> /tmp      1492
425     /usr/fo
426     Jan 12 1991
427     Jan 31 15:53
```

428 **Rationale for the New `pax` Interchange Format** B

429 The new POSIX data interchange format was developed primarily to satisfy inter-  
 430 national concerns that the `ustar` and `cpio` formats in POSIX.1 {8} did not provide  
 431 for file, user, and group names encoded in characters outside a subset of  
 432 ISO/IEC 646 {1}. The standard developers realized that this new POSIX data inter-  
 433 change format should be very extensible because there were other requirements  
 434 they foresaw in the near future:

- 435 — Support international character encodings and locale information
- 436 — Support security information (ACLs, etc.) emerging from POSIX security  
 437 working groups
- 438 — Support future file types, such as realtime or contiguous files
- 439 — Include data areas for implementation use
- 440 — Support systems with words larger than 32 b and timers with subsecond  
 441 granularity

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

442 The following were not goals for this format because these are better handled by  
443 separate utilities or are inappropriate for a portable format:

- 444 — Encryption
- 445 — Compression
- 446 — Data translation between locales and codesets
- 447 — I-node storage

448 The format chosen to support the goals is an extension of the `ustar` format,  
449 which has been moved into this standard from its original home in POSIX.1 {8}.  
450 Of the two formats, only the `ustar` format was selected for extensions because:

- 451 — It was easier to extend in an upward compatible way. It offered version  
452 flags and header block type fields with room for future standardization.  
453 The `cpio` format, while possessing a more flexible file naming methodology,  
454 could not be extended without breaking some theoretical implementation or  
455 using a dummy file name that could be a legitimate file name.
- 456 — Industry experience since the original “tar wars” fought in developing  
457 POSIX.1 {8} has clearly been in favor of the `ustar` format, which is gen-  
458 erally the default output format selected for `pax` implementations on new  
459 POSIX.2 systems.

460 The new format was designed with one additional goal in mind: reasonable  
461 behavior when an older `tar` or `pax` utility happened to read an archive. Since  
462 POSIX.1-1990 mandated that a “format-reading utility” had to treat unrecognized  
463 *typeflag* values as regular files, this allowed the format to include all the extended  
464 information in a pseudo-regular file that preceded each real file. An option is  
465 given that allows the archive creator to set up reasonable names for these files on  
466 the older systems. Also, the normative text suggests that reasonable file access  
467 values be used for this `ustar` header block. Making these header files inaccessi-  
468 ble for convenient reading and deleting would not be reasonable. File permissions  
469 of 600 or 700 are suggested.

470 The `ustar typeflag` field was used to accommodate the additional functionality of  
471 the new format rather than *magic* or *version* because POSIX.1-1990 (and, by refer-  
472 ence, the previous version of POSIX.2 `pax`), mandated the behavior of the format-  
473 reading utility when it encountered an unknown *typeflag*, but was silent about  
474 the other two fields.

475 A good deal of the complexity of this new format is found in its relation to the ori-  
476 ginal `ustar` format. If the backwards compatibility goal had been abandoned,  
477 none of the text relating the precedence of `ustar` fields to extended header  
478 records would have been required. A format that consisted entirely of extended  
479 header records followed by data records could have been designed. However, the  
480 standard developers believed that the new format should have some basis in an  
481 existing format, if only to avoid yet another complete invention as part of the  
482 standardization process.

483 Early drafts of the first revision to this standard contained a proposed archive for-  
484 mat that was based on compatibility with the standard for tape files (ISO 1001,

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

485 similar to the format used historically on many mainframes and minicomputers).  
486 This format was overly complex and required considerable overhead in volume  
487 and header records. Furthermore, the standard developers felt that it would not  
488 be acceptable to the community of POSIX developers, so it was later changed to be  
489 a format more closely related to historical practice on POSIX systems.

490 The *prefix* and *name* split of pathnames in *ustar* was replaced by the single *path*  
491 extended header record for simplicity.

492 The concept of a global extended header (*typeflag g*) was controversial. If this  
493 were applied to an archive being recorded on magnetic tape, a few unreadable  
494 blocks at the beginning of the tape could be a serious problem; a utility attempt-  
495 ing to extract as many files as possible from a damaged archive could lose a large  
496 percentage of file header information in this case. However, if the archive were  
497 on a reliable medium, such as a CD-ROM, the global extended header offers con-  
498 siderable potential size reductions by eliminating redundant information. Thus,  
499 the text warns against using the global method for unreliable media and provides  
500 a method for implanting global information in the extended header for each file,  
501 rather than in the *typeflag g* records.

502 No facility for data translation or filtering on a per-file basis is included because  
503 the standard developers could not invent an interface that would allow this in an  
504 efficient manner. If a filter, such as encryption or compression, is to be applied to  
505 all the files, it is more efficient to apply the filter to the entire archive as a single  
506 file. The standard developers considered interfaces that would invoke a shell  
507 script for each file going into or out of the archive, but the system overhead in this  
508 approach was considered to be too high.

509 One such approach would be to have *filter=* records that give a pathname for  
510 an executable. When the program is invoked, the file and archive would be open  
511 for standard input/output and all the header fields would be available as environ-  
512 ment variables or command-line arguments. The standard developers did discuss  
513 such schemes, but they were omitted from the standard due to concerns about  
514 excessive overhead. Also, the program itself would need to be in the archive if it  
515 were to be used portably.

516 There is currently no portable means of identifying the character set(s) used for a  
517 file in the file system. Therefore, *pax* has not been given a mechanism to gen-  
518 erate *charset* records automatically. The only portable means of doing this is for  
519 the user to write the archive using the *-o charset=string* command-line option.  
520 This assumes that all of the files in the archive use the same encoding. The  
521 “implementation defined” text is included to allow for a system that can identify  
522 the encodings used for each of its files.

523 The table of standards that accompanies the *charset* record description is ack-  
524 nnowledged to be very limited. Only a limited number of character set standards is  
525 reasonable for maximal interchange. Any character set is, of course, possible by  
526 prior agreement. It was suggested that EBCDIC be listed, but it was omitted  
527 because it is not defined by a formal standard. Formal standards, and then only  
528 those with reasonably large followings, can be included here, simply as a matter  
529 of practicality. The *<value>*s represent names of officially registered character

530 sets in the format required by ISO 2375 {B5}.

531 The normal comma-or-blank-separated-list rules are not followed in the case of  
532 keyword options to allow ease of argument parsing for `getopts`.

533 Further information on character encodings is in the following Rationale for  
534 Archive Character Set Encoding/Decoding.

535 The standard developers have reserved keyword name space for vendor exten-  
536 sions. It is suggested that the format to be used is:

537 *VENDOR.keyword*

538 where *VENDOR* is the name of the vendor or organization in all uppercase letters.  
539 It is further suggested that the *keyword* following the period be named differently  
540 than any of the standard keywords so that it could be used for future standardiza-  
541 tion, if appropriate, by omitting the *VENDOR* prefix.

542 The *<length>* field in the extended header record was included to make it simpler  
543 to step through the records, even if a record contains an unknown format (to a  
544 particular `pax`) with complex interactions of special characters. It also provides a  
545 minor integrity checkpoint within the records to aid a program attempting to  
546 recover files from a damaged archive.

547 There are no extended header versions of the *devmajor* and *devminor* fields  
548 because the unspecified-format `ustar` header field should be sufficient. If they  
549 are not, vendor-specific extended keywords (such as *VENDOR.devmajor*) should  
550 be used.

551 Device and i-number labeling of files was not adopted from `cpio`; files are inter-  
552 changed strictly on a symbolic name basis, as in `ustar`.

553 This version of POSIX.2 contains only namespace placeholders for security and  
554 realtime extensions. The POSIX working groups responsible for those areas are  
555 expected to amend this standard to provide additional details. It is currently  
556 unknown whether they would prescribe a single string of text or would allocate  
557 keywords at a finer granularity, such as *realtime.foo* or *security.bar*.

558 The POSIX security working group has not yet populated its “security.” name  
559 space. When it amends this standard, the POSIX security working group will  
560 presumably define the relationship between its records [which will probably  
561 define some sort of access control list (ACL)] and the modes and permissions found  
562 in the `ustar` headers. Vendor-specific extended keywords (such as  
563 *VENDOR.security*) should be used for any implementation-specific security  
564 arrangements.

565 Just as with the `ustar` format descriptions, the new format makes no special  
566 arrangements for multivolume archives. Each of the `pax` archive types is  
567 assumed to be inside a single POSIX file and splitting that file over multiple  
568 volumes (diskettes, tape cartridges, etc.), processing their labels, and mounting  
569 each in the proper sequence are considered to be implementation details that can-  
570 not be described portably. Perhaps the POSIX system administration working  
571 group will provide portable solutions for this.

572 The `pax` format is intended for interchange, not only for backup on a single (fam-  
573 ily of) systems. It is not as densely packed as might be possible for backup:

- 574 — It contains information as coded characters that could be coded in binary.
- 575 — It identifies extended records with name fields that could be omitted in  
576 favor of a fixed-field layout.
- 577 — It translates names into a portable character set and identifies locale-  
578 related information, both of which are probably unnecessary for backup.

579 The requirements on restoring from an archive are slightly different from the his-  
580 torical wording, allowing for nonmonolithic privilege to bring forward as much as  
581 possible. In particular, attributes such as “high performance file” might be  
582 broadly but not universally granted while `set-user-ID` or `chown()` might be much  
583 more restricted. There is no implication in this standard that the security infor-  
584 mation be honored after it is restored to the file hierarchy, in spite of what might  
585 be improperly inferred by the silence on that topic. That is a topic for another  
586 standard.

587 Links are recorded in the fashion described here because a link can be to any file  
588 type. It is desirable in general to be able to restore part of an archive selectively  
589 and restore all of those files completely. If the data is not associated with each  
590 link, it is not possible to do this. However, the data associated with a file can be  
591 large, and when selective restoration is not needed, this can be a significant bur-  
592 den. The archive is structured so that files that have no associated data can  
593 always be restored by the name of any linkname of any link, and the user may  
594 choose whether data is recorded with each instance of a file that contains data.  
595 The format permits mixing of both types of links in a single archive; this can be  
596 done for special needs, and `pax` is expected to interpret such archives on input  
597 properly, despite the fact that there is no `pax` option that would force this mixed  
598 case on output. (When `-o linkdata` is used, the output must contain the dupli-  
599 cate data, but the implementation is free to include it or omit it when `-o link-`  
600 `data` is not used.)

601 The time values are included as extended header records for those implementa-  
602 tions needing more than the eleven octal digits allowed by the `ustar` format.  
603 Even though some implementations can support finer file-time granularities than  
604 seconds, the normative text requires support only for seconds since the Epoch  
605 because POSIX.1 {8} states them that way. The `ustar` format includes only  
606 `mtime`; the new format adds `atime` and `ctime` for symmetry. The `atime` access  
607 time restored to the file system will be affected by the `-p a` and `-p e` options.  
608 The `ctime` creation time (actually i-node modification time) is described with  
609 “appropriate privilege” so that it can be ignored when writing to the file system.  
610 POSIX does not provide a portable means to change file creation time. Nothing is  
611 intended to prevent a nonportable implementation of `pax` from restoring the  
612 value.

613 The `gid`, `size`, and `uid` extended header records were included to allow expansion B  
614 beyond the sizes specified in the regular `tar` header. New file system archi- B  
615 tectures are emerging that will exhaust the 12-digit `size` field. There are probably B  
616 not many systems requiring more than 8 digits for user and group IDs, but the B

617 extended header values were included for completeness, allowing overrides for all B  
618 of the decimal values in the `tar` header. B

619 The standard developers intended to describe the effective results of `pax` with  
620 regard to file ownerships and permissions; implementations are not restricted in  
621 timing or sequencing the restoration of such, provided the results are as specified.

622 Much of the text describing the extended headers refers to use in “write or copy  
623 modes.” The copy-mode references are due to the normative text: “The effect of  
624 the copy shall be as if the copied files were written to an archive file and then sub-  
625 sequently extracted . . . .” There is certainly no way to test whether `pax` is actu-  
626 ally generating the extended headers in copy mode, but the effects must be as if it  
627 had.

### 628 **Rationale for `pax` Archive Character Set Encoding/Decoding**

629 There is a need to exchange archives of files between systems of different native  
630 codesets. File names, group names, and user names must be preserved to the ful-  
631 lest extent possible when an archive is read on the receiving platform. Transla-  
632 tion of the contents of files is not within the scope of the `pax` utility.

633 There will also be the need to represent glyphs that are not available on the  
634 receiving platform. (A *glyph* is commonly called a character, but without any  
635 reference to a specific encoding of that character. The term *glyph* refers to the  
636 symbol itself.) These unsupported glyphs cannot be automatically folded to the  
637 local set of glyphs due to the chance of collisions. This could result in overwriting  
638 previous extracted files from the archive or pre-existing files on the system.

639 For these reasons, the codeset used to represent glyphs within the extended  
640 header records of the `pax` archive must be sufficiently rich to handle all commonly  
641 used character sets. The fields requiring translation include, at a minimum, file  
642 names, user names, group names, and link pathnames. The POSIX security group  
643 and other working groups may specify other extended header records requiring  
644 similar treatment and implementations may wish to have localized extended key-  
645 words that use nonportable characters.

646 The standard developers considered the following options:

- 647 — The archive creator specifies the well-defined name of the source codeset.  
648 The receiver must then recognize the codeset name and perform the  
649 appropriate translations to the destination codeset.
- 650 — The archive creator includes within the archive the character mapping  
651 table for the source codeset used to encode extended header records. The  
652 receiver must then read the character mapping table and perform the  
653 appropriate translations to the destination codeset.
- 654 — The archive creator translates the extended header records in the source  
655 codeset into a canonical form. The receiver must then perform the  
656 appropriate translations to the destination codeset.

657 The approach that incorporates the name of the source codeset poses the problem  
658 of codeset name registration, and makes the archive useless to `pax` archive

659 decoders that do not recognize that codeset.

660 Because parts of an archive may be corrupted, the standard developers felt that  
661 including the character map of the source codeset was too fragile. The loss of this  
662 one key component could result in making the entire archive useless. (The differ-  
663 ence between this and the global extended header decision was that the latter has  
664 a workaround—duplicating extended header records on unreliable media—but  
665 this would be too burdensome for large character set maps.)

666 Both of the above approaches also put an undue burden on the `pax` archive  
667 receiver to handle the cross-product of all source and destination codesets.

668 To simplify the translation from the source codeset to the canonical form and from  
669 the canonical form to the destination codeset, the standard developers decided  
670 that the internal representation should be a stateless encoding. A stateless  
671 encoding is one where each codepoint has the same meaning, without regard to  
672 the decoder being in a specific state. An example of a stateful encoding would be  
673 the Japanese Shift-JIS; an example of a stateless encoding would ISO/IEC 646 {1}  
674 (equivalent to 7 b ASCII).

675 For these reasons, the standard developers decided to adopt a canonical format  
676 for the representation of file information strings. The obvious, well-endorsed can-  
677 didate is ISO/IEC 10646 {10} (based in part on Unicode), which can be used to  
678 represent the glyphs of virtually all standardized character sets. The standard  
679 developers initially agreed upon using UCS2 (16 b Unicode) as the internal  
680 representation. This repertoire of glyphs provides a sufficiently rich set to  
681 represent all commonly-used codesets.

682 However, the standard developers found that the 16 b Unicode representation had  
683 some problems. It forced the issue of standardizing byte ordering. The 2 B length  
684 of each character made the extended header records twice as long for the case of  
685 strings coded entirely from historical 7 b ASCII. For these reasons, the standard  
686 developers chose the UTF8 (File-System Safe Universal Translation Format)  
687 defined in ISO/IEC 10646 {10}. This multibyte representation encodes UCS2 or  
688 UCS4 characters reliably and deterministically, eliminating the need for a canoni-  
689 cal byte ordering. In addition, NUL octets and other characters possibly confusing  
690 to POSIX file systems do not appear, except to represent themselves. It was real-  
691 ized that certain national codesets take up more space after the encoding, due to  
692 their placement within the UCS range; it was felt that the usefulness of the encod-  
693 ing of the names outweighs the disadvantage of size increase for file, user, and  
694 group names.

695 The encoding of UTF8 is as follows:

696	<b>UCS4 Hex Encoding</b>	<b>UTF8 Binary Encoding</b>				
697	00000000-0000007F	0xxxxxxx				
698	00000080-000007FF	110xxxxx	10xxxxxx			
699	00000800-0000FFFF	1110xxxx	10xxxxxx	10xxxxxx		
700	00010000-001FFFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx	
701	00200000-03FFFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx
702	04000000-7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

703 where each `x` represents a bit value from the character being translated.

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.

## 704 **Rationale for the `ustar` Interchange Format**

B

705 The description of the `ustar` format reflects numerous enhancements over pre-  
706 1988 versions of the historical `tar` utility. The goal of these changes was not only  
707 to provide the functional enhancements desired, but also to retain compatibility  
708 between new and old versions. This compatibility has been retained. Archives  
709 written using the old archive format are compatible with the new format.

710 Implementors should be aware that the previous file format did not include a  
711 mechanism to archive directory type files. For this reason, the convention of  
712 using a file name ending with slash was adopted to specify a directory on the  
713 archive.

714 The total size of the name and prefix fields have been set to meet the minimum  
715 requirements for `{PATH_MAX}`. If a pathname will fit within the name field, it is  
716 recommended that the pathname be stored there without the use of the prefix  
717 field. Although the name field is known to be too small to contain `{PATH_MAX}`  
718 characters, the value was not changed in this version of the archive file format to  
719 retain backward compatibility, and instead the *prefix* was introduced. Also,  
720 because of the earlier version of the format, there is no way to remove the restric-  
721 tion on the *linkname* field being limited in size to just that of the *name* field.

722 The *size* field is required to be meaningful in all implementation extensions,  
723 although it could be zero. This is required so that the data blocks can always be  
724 properly counted.

725 It is suggested that if device special files need to be represented that cannot be  
726 represented in the standard format that one of the extension types ('A'-'Z') be  
727 used, and that the additional information for the special file be represented as  
728 data and be reflected in the size field.

729 Attempting to restore a special file type, where it is converted to ordinary data  
730 and conflicts with an existing file name, need not be specially detected by the util-  
731 ity. If run as an ordinary user, `pax` should not be able to overwrite the entries in,  
732 for example, `/dev` in any case (whether the file is converted to another type or  
733 not). If run as a privileged user, it should be able to do so, and it would be con-  
734 sidered a bug if it did not. The same is true of ordinary data files and similarly  
735 named special files; it is impossible to anticipate the needs of the user (who could  
736 really intend to overwrite the file), so the behavior should be predictable (and  
737 thus regular) and rely on the protection system as required.

738 The value '7' in the typeflag field is intended to define how contiguous files can be  
739 stored in a `ustar` archive. POSIX.1 {8} does not require the contiguous file exten-  
740 sion, but does define a standard way of archiving such files so that all conforming  
741 systems can interpret these file types in a meaningful and consistent manner. On  
742 a system that does not support extended file types, the `pax` utility should do the  
743 best it can with the file and go on to the next.

744 The file protection modes are those conventionally used by the `ls` utility. This is  
745 extended beyond the usage in POSIX.2 to support the "shared text" or "sticky" bit.  
746 It is intended that the conformance document should not document anything  
747 beyond the existence of and support of such a mode. Further extensions are

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



748 expected to these bits, particularly with overloading the set-user-ID and set-  
749 group-ID flags.

750 *Rationale for the cpio Interchange Format* B

751 The reference to appropriate privilege in the `cpio` format refers to an error on  
752 standard output; the `ustar` format does not make comparable statements.

753 The model for this format was the historical System V `cpio -c` data interchange  
754 format. This model documents the portable version of the `cpio` format and not  
755 the binary version. It has the flexibility to transfer data of any type described  
756 within POSIX.1 {8}, yet is extensible to transfer data types specific to extensions  
757 beyond POSIX.1 {8} (e.g., or contiguous files). Because it describes existing prac-  
758 tice, there is no question of maintaining upward compatibility.

759 **cpio Header**

760 There has been some concern that the size of the `c_ino` field of the header is too  
761 small to handle those systems that have very large i-node numbers. However, the  
762 `c_ino` field in the header is used strictly as a hard link resolution mechanism for  
763 archives. It is not necessarily the same value as the i-node number of the file in  
764 the location from which that file is extracted.

765 The name `c_magic` is based on historical usage.

766 **cpio File Name**

767 For most historical implementations of the `cpio` utility, {PATH\_MAX} octets can  
768 be used to describe the pathname without the addition of any other header fields  
769 (the NUL character would be included in this count). {PATH\_MAX} is the  
770 minimum value for pathname size, documented as 256 B in Section 2 of  
771 POSIX.1 {8}. However, an implementation may use `c_namesize` to determine the  
772 exact length of the pathname. With the current description of the `cpio` header,  
773 this pathname size can be as large as a number that is described in six octal  
774 digits.

775 Two values are documented under the `c_mode` field values to provide for extensi-  
776 bility for known file types:

777     0110 000     Reserved for contiguous files. The implementation may treat the  
778                    rest of the information for this archive like a regular file. If this  
779                    file type is undefined, the implementation may create the file as a  
780                    regular file.

781     0140 000     Reserved for sockets. If this type is undefined on the target sys-  
782                    tem, the implementation may decide to ignore this file type and  
783                    output a warning message.

784 This provides for extensibility of the `cpio` format while allowing for the ability to  
785 read old archives. Files of an unknown type may be read as “regular files” on  
786 some implementations. On a system that does not support extended file types,  
787 the `pax` utility should do the best it can with the file and go on to the next.

788 In POSIX.1 {8}, the symbolic link value was reserved, but this has been deleted in  
789 light of support for symbolic links elsewhere in this standard.

790 ⇒ **E.5.10 ex Rationale.** *Replace the full rationale for ex with the following.* B

### 791 **E.5.10 ex – Text editor**

792 The `ex/vi` specification is based on the historical practice found in the 4BSD and  
793 System V implementations of `ex` and `vi`. A freely redistributable implementation  
794 of `ex/vi`, which is tracking this specification fairly closely, and demonstrates the  
795 intended changes between historical implementations and this specification, may  
796 be obtained from Keith Bostic (`bostic@cs.berkeley.edu`) or by anonymous  
797 FTP from:

```
798 ftp.cs.berkeley.edu:ucb/4bsd/nvi.tar.gz
```

799 A “restricted editor” (both the historical `red` utility and modifications to `ex`) were  
800 considered and rejected for inclusion. Neither option provided the level of secu-  
801 rity that users might expect.

#### 802 **E.5.10.1 Synopsis**

803 There is no additional rationale provided for this subclause.

#### 804 **E.5.10.2 Description**

805 It is recognized that `ex` visual mode and related features would be difficult, if not  
806 impossible, to implement satisfactorily on a block-mode terminal, or a terminal  
807 without any form of cursor addressing; thus, it is not a mandatory requirement  
808 that such features should work on all terminals. It is the intention, however, that  
809 an `ex` implementation should provide the full set of capabilities on all terminals  
810 capable of supporting them.

#### 811 **E.5.10.3 Options**

812 The `-c` replacement for `+command` was inspired by the `-e` option of `sed`. Histori-  
813 cally, all such commands (see `edit` and `next` as well) were executed from the last  
814 line of the edit buffer. This meant, for example, that `+/pattern` would fail unless  
815 the `wrapscan` option was set. This standard requires conformance to historical  
816 practice. Historically, some implementations restricted the `ex` commands that  
817 could be listed as part of the command-line arguments. For consistency, this  
818 standard does not permit these restrictions.

819 Historically, the `ex` and `vi` utilities accepted a `-l` option, which set the `lisp` and  
820 `showmatch` edit options. The `-l` option was omitted because it was difficult to  
821 justify the inclusion of programming-language dependent features. Similarly, the  
822 `lisp` edit option was omitted.

823 In historical implementations of the editor, the `-R` option (and the `readonly` edit  
824 option) only prevented overwriting of files; appending to files was still permitted,  
825 mapping loosely into the `cs`h `noclobber` variable. Some implementations, how-  
826 ever, have not followed this semantic, and `readonly` does not permit appending  
827 either. This standard follows the latter practice, believing that it is a more obvi-  
828 ous and intuitive meaning of `readonly`.

829 The `-s` option (and its obsolescent single-hyphen form) suppresses all interactive  
830 user feedback and is useful for editing scripts in batch jobs. The list of specific  
831 effects is historical practice. The terminal type “incapable of supporting open and  
832 visual modes” has historically been named “dumb.”

833 The `-t` option was required because the `ctags` utility appears in POSIX.2 and the  
834 option is available in all historical implementations of `ex`.

835 Historically, the `ex` and `vi` utilities accepted a `-x` option, which did encryption  
836 based on the algorithm found in the historical `crypt` utility. The `-x` option for  
837 encryption, and the associated `crypt` utility, were omitted because the algorithm  
838 used was not specifiable and the export control laws of some nations make it  
839 difficult to export cryptographic technology. In addition, it did not historically  
840 provide the level of security that users might expect.

#### 841 **E.5.10.4 Operands**

842 There is no additional rationale provided for this subclause.

#### 843 **E.5.10.5 External Influences**

##### 844 **E.5.10.5.1 Standard Input**

845 An end-of-file condition is not equivalent to an end-of-file character. A common  
846 end-of-file character, `<control-D>`, is historically an `ex` command.

847 There was no maximum line length in historical implementations of `ex`. C  
848 Specifically, as it was parsed in chunks, the addresses had a different maximum C  
849 length than the filenames. Further, the maximum line buffer size was declared C  
850 as `{BUFSIZ}`, which was different lengths on different systems. This version of C  
851 this standard selected the value of `{LINE_MAX}` to impose a reasonable restriction C  
852 on portable usage of `ex` and to aid test-suite writers in their development of real- C  
853 istic tests that exercise this limit. C

##### 854 **E.5.10.5.2 Input Files**

855 It was an explicit decision by the standard developers that a `<newline>` charac-  
856 ter be added to any file lacking one. It was believed that this feature of `ex` and `vi`  
857 was relied on by users in order to make text files lacking a trailing `<newline>`  
858 more portable. It is recognized that this will require a user specified option or  
859 extension for implementations that permit `ex` and `vi` to edit files of type other  
860 than text if such files are not otherwise identified by the system. It was agreed  
861 that the ability to edit files of arbitrary type can be useful, but it was not con-  
862 sidered necessary to mandate that an `ex` or `vi` implementation be required to

863 handle files other than text files.

864 The paragraph in the Input Files subclause, “By default, . . .,” is intended to close  
 865 a long-standing security problem in `ex` and `vi`, that of the “modeline” or “mode-  
 866 lines” edit option. This feature allows any line in the first or last five lines of the  
 867 file containing the strings `ex:` or `vi:` (and, apparently, `ei:` or `vx:`) to be a line  
 868 containing editor commands, and `ex` interprets all the text up to the next `:` or  
 869 `<newline>` as a command. Consider the consequences, for example, of an  
 870 unsuspecting user using `ex` or `vi` as the editor when replying to a mail message  
 871 in which a line such as

```
872     ex:! rm -rf *:
```

873 appeared in the signature lines. The standard developers believed strongly that  
 874 an editor should not by default interpret any lines of a file. Vendors are strongly  
 875 urged to delete this feature from their implementations of `ex` and `vi`.

### 876 E.5.10.5.3 Environment Variables

877 There is no additional rationale provided for this subclause.

### 878 E.5.10.5.4 Asynchronous Events

879 The intention of the phrase “complete write” is that the entire edit buffer be writ-  
 880 ten to stable storage. The note regarding temporary files is intended for imple- C  
 881 mentations that use temporary files to back edit buffers unnamed by the user. C

882 Historically, `SIGQUIT` was ignored by `ex`, but was the equivalent of `Q` in visual  
 883 mode; i.e., it exited visual mode and entered `ex` mode. This standard permits, but  
 884 does not require, this behavior. Historically, `SIGINT` was often used by `vi` users  
 885 to terminate text input mode (`<control-C>` is often easier to enter than `<ESC>`).  
 886 Some implementations of `vi` alerted the terminal on this event, and some did not.  
 887 This standard requires that `SIGINT` behave identically to `<ESC>`, and that the ter-  
 888 minal not be alerted.

889 Historically, suspending the `ex` editor during text input mode was similar to `SIG-` C  
 890 `INT`, as completed lines were retained, but any partial line discarded, and the edi-  
 891 tor returned to command mode. This standard is silent on this issue; implemen-  
 892 tations are encouraged to follow historical practice, where possible.

893 Historically, the `vi` editor did not treat `SIGTSTP` as an asynchronous event, and it  
 894 was therefore impossible to suspend the editor in visual text input mode. There  
 895 are two major reasons for this. The first is that `SIGTSTP` is a broadcast signal on  
 896 UNIX systems, and the chain of events where the shell execs an application that  
 897 then execs `vi` usually caused confusion for the terminal state if `SIGTSTP` was  
 898 delivered to the process group in the default manner. The second was that most  
 899 implementations of the UNIX *curses* package are not reentrant, and the receipt of  
 900 `SIGTSTP` at the wrong time will cause them to crash. This standard is silent on  
 901 this issue; implementations are encouraged to treat suspension as an asynchro-  
 902 nous event if possible.

903 Historically, modifications to the edit buffer made before SIGINT interrupted an C  
 904 operation were retained; i.e., anywhere from zero to all of the lines to be modified C  
 905 might have been modified by the time the SIGINT arrived. These changes were C  
 906 not discarded by the arrival of SIGINT. This standard permits this behavior, not- C  
 907 ting that the `undo` command is required to be able to undo these partially com- C  
 908 pleted commands.

909 The action taken for signals other than SIGINT, SIGCONT, SIGHUP, and SIGTERM C  
 910 is unspecified because some implementations attempt to save the edit buffer in a C  
 911 useful state when other signals are received.

## 912 **E.5.10.6 External Effects**

### 913 **E.5.10.6.1 Standard Output**

914 There is no additional rationale provided for this subclause.

### 915 **E.5.10.6.2 Standard Error**

916 For `ex/vi`, diagnostic messages are those messages reported as a result of a failed C  
 917 attempt to invoke `ex` or `vi`, such as invalid options or insufficient resources, or an C  
 918 abnormal termination condition. Diagnostic messages should not be confused C  
 919 with the error messages generated by inappropriate or illegal user commands.

### 920 **E.5.10.6.3 Output Files**

921 There is no additional rationale provided for this subclause.

## 922 **E.5.10.7 Extended Description**

### 923 **E.5.10.7.1 `ex` and `vi` Initialization**

924 If an `ex` command (other than `cd`, `chdir`, or `source`) has a file name argument, C  
 925 one or both of the alternate and current pathnames will be set. Informally, they C  
 926 are set as follows: C

927 (1) If the `ex` command is one that replaces the contents of the edit buffer, C  
 928 and it succeeds, the current pathname will be set to the file name argu- C  
 929 ment (the first file name argument in the case of the next command) and C  
 930 the alternate pathname will be set to the previous current pathname, if C  
 931 there was one. C

932 (2) In the case of the file read/write forms of the read and write commands, if C  
 933 there is no current pathname, the current pathname will be set to the file C  
 934 name argument. C

935 (3) Otherwise, the alternate pathname will be set to the file name argument. C

936 For example, `:edit foo` and `:recover foo`, when successful, set the current C  
 937 pathname, and, if there was a previous current pathname, the alternate path- C  
 938 name. The commands `:write !command` and `:edit set` neither the current or C  
 939 alternate pathnames. If the `:edit foo` command were to fail for some reason, C

940 the alternate pathname would be set. The `read` and `write` commands set the C  
941 alternate pathname to their *file* argument, unless the current pathname is not  
942 set, in which case they set the current pathname to their *file* arguments. The  
943 alternate pathname was not historically set by the `:source` command. This  
944 standard requires conformance to historical practice. Implementations adding  
945 commands that take file names as arguments are encouraged to set the alternate  
946 pathname as described here.

947 Historically, `ex` and `vi` read the `.exrc` file in the `$HOME` directory twice, if the  
948 editor was executed in the `$HOME` directory. This standard prohibits this  
949 behavior.

950 Historically, the historical 4BSD `ex` and `vi` read the `$HOME` and local `.exrc` files  
951 if they were owned by the real ID of the user, or the `sourceany` option was set,  
952 regardless of other considerations. This was a security problem because it is pos-  
953 sible to put normal UNIX commands inside a `.exrc` file. This standard does not  
954 specify the `sourceany` option, and historical implementations are encouraged to  
955 delete it.

956 The `.exrc` files must be owned by the real ID of the user, and not writeable by  
957 anyone other than the owner. The appropriate privileges exception is intended to  
958 permit users to acquire special privileges, but continue to use the `.exrc` files in  
959 their home directories.

960 System V release 3.2 and later `vi` implementations added the option `[no]exrc`.  
961 The behavior is that local `.exrc` files are read only if the `exrc` option is set. The  
962 default for the `exrc` option was off, so by default, local `.exrc` files were not read.  
963 The problem this was intended to solve was that System V permitted users to give  
964 away files, so there is no possible ownership or writeability test to ensure that the  
965 file is safe. This is still a security problem on systems where users can give away  
966 files, but there is nothing additional that this standard can do. The  
967 implementation-defined exception is intended to permit groups to have local  
968 `.exrc` files that are shared by users, by creating pseudo-users to own the shared  
969 files.

970 This standard does not mention system-wide `ex` and `vi` startup files. While they  
971 exist in several implementations of `ex` and `vi`, they are not present in any imple-  
972 mentations considered historical practice by this standard. Implementations that  
973 have such files should use them only if they are owned by the real user ID or an  
974 appropriate user (e.g., `root` on UNIX systems) and if they are not writeable by any  
975 user other than their owner. System-wide startup files should be read before the  
976 `EXINIT` variable, `$HOME/.exrc` or local `.exrc` files are evaluated.

977 Historically, any `ex` command could be entered in the `EXINIT` variable or the  
978 `.exrc` file, although ones requiring that the edit buffer already contain lines of  
979 text generally caused historical implementations of the editor to drop core. This  
980 standard requires that any `ex` command be permitted in the `EXINIT` variable and  
981 `.exrc` files, for simplicity of specification and consistency, although many of them  
982 will obviously fail under many circumstances.

983 The initialization of the contents of the edit buffer uses the phrase “the effect C  
984 shall be” with regard to various `ex` commands. The intent of this phrase is that

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

985 edit buffer contents loaded during the initialization phase not be lost; i.e., loading  
 986 the edit buffer should fail if the `.exrc` file read in the contents of a file and did C  
 987 not subsequently write the edit buffer. An additional intent of this phrase is to  
 988 specify that the initial current line and column is set as specified for the indivi-  
 989 dual `ex` commands.

990 Historically, the `-t` option behaved as if the tag search were a *+command*; i.e., it C  
 991 was executed from the last line of the file specified by the tag. This resulted in C  
 992 the search failing if the pattern was a forward search pattern and the `wrapscan` C  
 993 edit option was not set. This standard does not permit this behavior, requiring C  
 994 that the search for the tag pattern be performed on the entire file, and, if not C  
 995 found, that the current line be set to a more reasonable location in the file. C

996 Historically, the empty edit buffer presented for editing when a file was not  
 997 specified by the user was unnamed. This is permitted by the standard, however,  
 998 implementations are encouraged to provide users a temporary file name for this  
 999 buffer because it permits them the use of `ex` commands that use the current path-  
 1000 name during temporary edit sessions.

1001 Historically, the file specified using the `-t` option was not part of the current  
 1002 argument list. This practice is permitted by the standard, however, implementa-  
 1003 tions are encouraged to include its name in the current argument list for con-  
 1004 sistency.

1005 Historically, the `-c` command (or *+command*) was generally not executed until a C  
 1006 file that already exists was edited. This standard requires conformance to this C  
 1007 historical practice. Commands that could cause the `-c` command to be executed C  
 1008 include the `ex` commands `edit`, `next`, `recover`, `rewind`, and `tag`, and the `vi` C  
 1009 commands `<control-^>` and `<control-]>`. Historically, reading a file into an C  
 1010 edit buffer did not cause the `-c` command to be executed (even though it might set C  
 1011 the current pathname) with the exception that it did cause the `-c` command to be C  
 1012 executed if: the editor was in `ex` mode, the edit buffer had no current pathname, C  
 1013 the edit buffer was empty, and no read commands had yet been attempted. For C  
 1014 consistency and simplicity of specification, this standard does not permit this C  
 1015 behavior. C

1016 Historically, the `-r` option was the same as a normal edit session if there was no  
 1017 recovery information available for the file. This allowed users to enter “`vi -r`  
 1018 `*.c`” and recover whatever files were recoverable. In some implementations,  
 1019 recovery was attempted only on the first file named, and the file was not entered  
 1020 into the argument list; in others, recovery was attempted for each file named. In  
 1021 addition, some historical implementations ignored `-r` if `-t` was specified or did  
 1022 not support command-line file arguments with the `-t` option. For consistency and C  
 1023 simplicity of specification, this standard disallows these special cases, and  
 1024 requires that recovery be attempted the first time each file is edited.

1025 C

1026 Historically, `vi` initialized the ``` and `'` marks, but `ex` did not. This meant that if  
 1027 the first command in `ex` mode was “visual,” or if an `ex` command was executed  
 1028 first (e.g., `vi +10 file`), `vi` was entered without the marks being initialized.  
 1029 Because the standard developers believed the marks to be generally useful, and

1030 for consistency and simplicity of specification, this standard requires that they  
 1031 always be initialized if in open or visual mode, or if in `ex` mode and the edit buffer  
 1032 is not empty. Not initializing it in `ex` mode if the edit buffer is empty is historical  
 1033 practice, however it has always been possible to set (and use) marks in empty edit  
 1034 buffers in open and visual mode edit sessions.

### 1035 **E.5.10.7.2 Addressing**

1036 Historically, `ex` and `vi` accepted the additional addressing forms `\/` and `\?`.  
 1037 They were equivalent to `//` and `??`, respectively. They are not required by this  
 1038 standard, mostly because nobody can remember if they ever did anything dif-  
 1039 ferent historically or not.

1040 Historically, `ex` and `vi` permitted an address of zero for several commands, and  
 1041 permitted the `%` address in empty files for others. For consistency, this standard  
 1042 requires support for the former in the few commands where it makes sense and  
 1043 disallows it otherwise. In addition, because this standard requires that `%` be logi-  
 1044 cally equivalent to `1, $`, it is also supported where it makes sense and disallowed  
 1045 otherwise.

1046 Historically, the `%` address could not be followed by further addresses. For con- C  
 1047 sistency and simplicity of specification, this standard requires that additional C  
 1048 addresses be supported. C

1049 All of the following are valid addresses:

1050	+++	Three lines after the current line
1051	<i>/pattern/</i> -	One line before the next occurrence of <i>pattern</i>
1052	-2	Two lines before the current line
1053	3 ---- 2	Line one (note intermediate negative address)
1054	1 2 3	Line six

1055 Any number of addresses can be provided to commands taking addresses; e.g.,  
 1056 `1,2,3,4,5p` prints lines 4 and 5, because two is the greatest valid number of  
 1057 addresses accepted by the `print` command. This, in combination with the semi-  
 1058 colon delimiter, permits users to create commands based on ordered patterns in  
 1059 the file. For example, the command `3;/foo/;+2print` will display the first line  
 1060 after line 3 that contains the pattern `foo`, plus the next two lines. Note that the  
 1061 address “3;” must be evaluated before being discarded because the search origin  
 1062 for the `/foo/` command depends on this.

1063 Historically, values could be added to addresses by including them after one or  
 1064 more `<blank>` characters; e.g., `3 - 5p` wrote the seventh line of the file, and  
 1065 `/foo/ 5` was the same as `/foo/+5`. However, only absolute values could be  
 1066 added; e.g., `5 /foo/` was an error. This standard requires conformance to histor-  
 1067 ical practice. Address offsets are separately specified from addresses because they  
 1068 could historically be provided to visual mode search commands.

1069 Historically, any missing addresses defaulted to the current line. This was true  
 1070 for leading and trailing comma-delimited addresses, and for trailing semicolon-



1071 delimited addresses. For consistency, this standard requires it for leading semi-  
1072 colon addresses as well.

1073 Historically, `ex` and `vi` accepted the `^` character as both an address and as a flag  
1074 offset for commands. In both cases it was identical to the “-” character. This  
1075 standard does not require or prohibit this behavior.

1076 Historically, the enhancements to BREs could be used in addressing: e.g., `~`, `\<`,  
1077 and `\>`. This standard requires conformance to historical practice; i.e., that RE  
1078 usage be consistent, and that RE enhancements be supported wherever REs are  
1079 used.

### 1080 **E.5.10.7.3 `ex` Command-Line Parsing**

1081 Historical `ex` command parsing was even more complex than that described by  
1082 this standard. This standard requires the subset of the command parsing that  
1083 the standard developers believed was documented and that users could reason-  
1084 ably be expected to use in a portable fashion, and that was historically consistent  
1085 between implementations. (The discarded functionality is obscure, at best.) His-  
1086 torical implementations will require changes in order to comply with this stan-  
1087 dard; however, users are not expected to notice any of these changes. Most of the  
1088 complexity in `ex` parsing is to handle three special termination cases:

- 1089 (1) The `!`, `global`, `v`, and the filter versions of the `read` and `write` com-  
1090 mands are delimited by `<newline>s` (they can contain vertical-line char-  
1091 acters that are usually shell pipes).
- 1092 (2) The `ex`, `edit`, `next` and `visual` in open and visual mode commands all  
1093 take `ex` commands, optionally containing vertical-line characters, as  
1094 their first arguments.
- 1095 (3) The `s` command takes an RE as its first argument, and uses the delimit-  
1096 ing characters to delimit the command.

1097 Historically, vertical-line characters in the *+command* argument of the `ex`, `edit`, C  
1098 `next`, `vi`, and `visual` commands, and in the *pattern* and *replacement* parts of the C  
1099 `s` command, did not delimit the command, and in the filter cases for `read` and C  
1100 `write`, and the `!`, `global`, and `v` commands, they did not delimit the command at C  
1101 all. For example, the following commands are all valid:

```
1102      :edit +25|s/abc/ABC/ file.c
1103      :s|/PIPE/
1104      :read !spell % | columnate
1105      :global/pattern/p|l
1106      :s/a/b|s/c/d|set
```

1107 Historically, empty or `<blank>`-filled lines in `.exrc` files and sourced files (as  
1108 well as **EXINIT** variables and `ex` command scripts) were treated as default com- C  
1109 mands; i.e., `print` commands. This standard specifically requires that they be C  
1110 ignored when encountered in `.exrc` and sourced files to eliminate a common C  
1111 source of new-user error.

- 1112 Historically, `ex` commands with multiple adjacent (or `<blank>` separated) verti-  
 1113 cal lines were handled oddly when executed from `ex` mode. For example, the com-  
 1114 mand “`||| <carriage-return>`”, when the cursor was on line 1, displayed lines  
 1115 2, 3, and 5 of the file. In addition, the command “`|`” would only display the line  
 1116 after the next line, instead of the next two lines. The former worked more logi-  
 1117 cally when executed from `vi` mode, and displayed lines 2, 3, and 4. This standard  
 1118 requires the `vi` behavior, i.e., a single default command and line number incre-  
 1119 ment for each command separator, and trailing `<newline>` characters after  
 1120 vertical-line separators are discarded.
- 1121 Historically, `ex` permitted a single extra colon as a leading command character;  
 1122 e.g., `:g/pattern/:p` was a valid command. This standard generalizes this to  
 1123 require that any number of leading colon characters be stripped.
- 1124 Historically, any prefix of the `delete` command could be followed without inter-  
 1125 vening `<blank>`s by a flag character because in the command “`d p`”, `p` is inter-  
 1126 preted as the buffer `p`. This standard requires conformance to historical practice.
- 1127 Historically, the `k` command could be followed by the mark name without inter-  
 1128 vening `<blank>` characters. This standard requires conformance to historical  
 1129 practice.
- 1130 Historically, the `s` command could be immediately followed by flag and option  
 1131 characters; e.g., `s/e/E/|s|sgc3p` was a valid command. However, flag charac- C  
 1132 ters could not stand alone; e.g., the commands “`sp`” and “`s l`” would fail, while C  
 1133 the command “`sgp`” and “`s gl`” would succeed. (Obviously, the `#` flag character C  
 1134 was used as a delimiter character if it followed the command.) Another issue was C  
 1135 that option characters had to precede flag characters even when the command C  
 1136 was fully specified; e.g., the command “`s/e/E/pg`” would fail, while the command C  
 1137 “`s/e/E/gp`” would succeed. This standard requires conformance to historical C  
 1138 practice.
- 1139 Historically, the first command name that had a prefix matching the input from C  
 1140 the user was the executed command; e.g., `ve`, `ver`, and `vers` all executed the `ver-`  
 1141 `sion` command. Commands were in a specific order, however, so that a matched  
 1142 append, not abbreviate. This standard requires conformance to historical prac-  
 1143 tice. The restriction on command search order for implementations with exten-  
 1144 sions is to avoid the addition of commands such that the historical prefixes would  
 1145 fail to work portably.
- 1146 Historical implementations of `ex` and `vi` did not correctly handle multiple `ex`  
 1147 commands, separated by vertical-line characters, that entered or exited visual  
 1148 mode or the editor. Because implementations of `vi` exist that do not exhibit this  
 1149 failure mode, this standard does not permit it.
- 1150 The requirement that alphabetic command names consist of all following alpha- C  
 1151 betic characters up to the next nonalphabetic character means that alphabetic C  
 1152 command names must be separated from their arguments by one or more nonal- C  
 1153 phabetic characters, normally a `<blank>` or `!` character, except as specified for  
 1154 the exceptions, the `delete`, `k`, and `s` commands. C

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1155 Historically, the repeated execution of the `ex` default print commands  
 1156 (`<control-D>`, `eof`, `<newline>`, `<carriage-return>`) erased any prompting  
 1157 character and displayed the next line(s) without scrolling the terminal; i.e.,  
 1158 immediately below any previously displayed lines. This provided a cleaner  
 1159 presentation of the lines in the file for the user. This standard does not require  
 1160 this behavior because it may be impossible in some situations; however, imple-  
 1161 mentations are strongly encouraged to provide this semantic if possible.

1162 Historically, it was possible to change files in the middle of a command, and have  
 1163 the rest of the command executed in the new file, e.g.,

```
1164      :edit +25 file.c|s/abc/ABC/|l
```

1165 was a valid command, and the substitution was attempted in the newly edited  
 1166 file. This standard requires conformance to historical practice. The following  
 1167 commands are examples that exercise the `ex` parser:

```
1168      echo 'foo|bar' > file1; echo 'foo/bar' > file2;
1169      vi
1170      :edit +1|s|/PIPE/|w file1| e file2|1 | s|//SLASH/|wq
```

1171 Historically, there was no protection in editor implementations to avoid `ex glo-` C  
 1172 `bal`, `v`, `@`, or `*` commands changing edit buffers during execution of their associ- C  
 1173 ated commands. Because this would almost invariably result in catastrophic C  
 1174 failure of the editor, and implementations exist that do exhibit these problems, C  
 1175 this standard requires that changing the edit buffer during a `global` or `v` com- C  
 1176 mand, or during a `@` or `*` command for which there will be more than a single exe- C  
 1177 cution, be an error. Implementations supporting multiple edit buffers simultane- C  
 1178 ously are strongly encouraged to apply the same semantics to switching between C  
 1179 buffers as well. C

1180 The `ex` command quoting required by this standard is a superset of the quoting in  
 1181 historical implementations of the editor. For example, it was not historically pos-  
 1182 sible to escape a `<blank>` character in a file name; e.g., `:edit foo\\\ bar`  
 1183 would report that too many file names had been entered for the `edit` command,  
 1184 and there was no method of escaping a `<blank>` in the first argument of an `edit`,  
 1185 `ex`, `next`, or `visual` command at all. This standard extends historical practice,  
 1186 requiring that quoting behavior be made consistent across all `ex` commands,  
 1187 except for the `map`, `unmap`, `abbreviate`, and `unabbreviate` commands, which  
 1188 historically used `<control-V>` instead of backslashes for quoting. For those four  
 1189 commands, this standard requires conformance to historical practice.

1190 Backslash quoting in `ex` is nonintuitive. Backslash escapes are ignored unless  
 1191 they escape a special character; e.g., when performing file argument expansion,  
 1192 the string `\\%` is equivalent to `\%`, not `\<current pathname>`. This can be confus-  
 1193 ing for users because backslash is usually one of the characters that causes shell  
 1194 expansion to be performed, and therefore shell quoting rules must be taken into  
 1195 consideration. Generally, quoting characters are only considered if they escape a  
 1196 special character, and a quoting character must be provided for each layer of pars-  
 1197 ing for which the character is special. As another example, only a single  
 1198 backslash is necessary for the `\l` sequence in substitute replacement patterns,  
 1199 because the character `l` is not special to any parsing layer above it.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1200 <Control-V> quoting in `ex` is slightly different from backslash quoting. In the  
 1201 four commands where <control-V> quoting applies (abbreviate, unabbrevi-  
 1202 ate, `map` and `unmap`), any character may be escaped by a <control-V> whether  
 1203 it would have a special meaning or not. This standard requires conformance to  
 1204 historical practice.

1205 Historical implementations of the editor did not require delimiters within charac-  
 1206 ter classes to be escaped; e.g., the command `:s/[ ]//` on the string `xxx/yyy`  
 1207 would delete the `/` from the string. This standard disallows this historical prac-  
 1208 tice for consistency and because it places a large burden on implementations by  
 1209 requiring that knowledge of REs be built into the editor parser.

1210 Historically, quoting <newline> characters in `ex` commands was handled incon- C  
 1211 sistently. In most cases, the <newline> always terminated the command, C  
 1212 regardless of any preceding escape character, because backslash characters did C  
 1213 not escape <newline> characters for most `ex` commands. However, some `ex` C  
 1214 commands (e.g., `s`, `map`, and abbreviation) permitted <newline>s to be escaped C  
 1215 (although in the case of `map` and abbreviation, <control-V> characters escaped C  
 1216 them instead of backslashes). This was true in not only the command line but C  
 1217 also `.exrc` and `sourced` files. For example, the command C

```
1218     map = foo<control-V><newline>bar C
```

1219 would succeed, although it was sometimes difficult to get the <control-V> and C  
 1220 the inserted <newline> passed to the `ex` parser. For consistency and simplicity C  
 1221 of specification, this standard requires that it be possible to escape <newline> C  
 1222 characters in `ex` commands at all times, using backslashes for most `ex` com- C  
 1223 mands, and using <control-V> characters for the `map` and abbreviation com- C  
 1224 mands. For example, the command `print<newline>list` is required to be C  
 1225 parsed as the single command `print<newline>list`. While this differs from C  
 1226 historical practice, the standard developers believed it unlikely that any script or C  
 1227 user depended on the historical behavior. C

1228 Historically, an error in a command specified using the `-c` option did not cause C  
 1229 the rest of the `-c` command(s) to be discarded. This standard disallows this for C  
 1230 consistency with mapped keys, the `@`, `global`, `source`, and `v` commands, the C  
 1231 `EXINIT` environment variable, and the `.exrc` files. C

#### 1232 **E.5.10.7.4 `ex` Input Editing**

1233 One of the common uses of the historical `ex` editor is over slow network connec-  
 1234 tions. Editors that run in canonical mode can require far less traffic to and from,  
 1235 and far less processing on, the host machine, as well as more easily supporting  
 1236 block-mode terminals. For these reasons, this standard requires that `ex` be  
 1237 implemented using canonical mode input processing, as was done historically.

1238 The POSIX.1 {8} standard does not require the historical 4BSD input editing char-  
 1239 acters “word erase” or “literal next.” For this reason, it is unspecified how they  
 1240 are handled by `ex`, although they must have the required effect. Implementations  
 1241 that resolve them after the line has been ended using a <newline> or  
 1242 <control-M> character, and implementations that rely on the underlying system  
 1243 terminal support for this processing, are both conforming. Implementations are

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1244 strongly urged to use the underlying system functionality, if at all possible, for  
1245 compatibility with other system text input interfaces.

1246 Historically, when the *eof* character was used to decrement the autoindent level,  
1247 the cursor moved to display the new end of the autoindent characters, but did not  
1248 move the cursor to a new line, nor did it erase the `<control-D>` character from  
1249 the line. This standard does not specify that the cursor remain on the same line  
1250 or that the rest of the line is erased; however, implementations are strongly  
1251 encouraged to provide the best possible user interface; i.e., the cursor should  
1252 remain on the same line, and any `<control-D>` character on the line should be  
1253 erased.

1254 The POSIX.1 {8} standard does not require the historical 4BSD input editing char-  
1255 acter “reprint,” traditionally `<control-R>`, which redisplayed the current input  
1256 from the user. For this reason, and because the functionality cannot be imple-  
1257 mented after the line has been terminated by the user, this standard makes no  
1258 requirements about this functionality. Implementations are strongly urged to  
1259 make this historical functionality available, if possible.

1260 Historically, `<control-Q>` did not perform a literal next function in *ex*, as it did  
1261 in *vi*. This standard requires conformance to historical practice to avoid breaking  
1262 historical *ex* scripts and `.exrc` files. C  
C

#### 1263 **E.5.10.7.4.1 *eof***

1264 Whether the *eof* character immediately modifies the autoindent characters in the  
1265 prompt is left unspecified so that implementations can conform in the presence of  
1266 systems that do not support this functionality. Implementations are encouraged  
1267 to modify the line and redisplay it immediately, if possible. C

1268 The specification of the handling of the *eof* character differs from historical prac- C  
1269 tice only in that *eof* characters are not discarded if they follow normal characters C  
1270 in the text input. Historically, they were always discarded. C

#### 1271 **E.5.10.7.4.2 `<newline>`**

1272 There is no additional rationale provided for this subclause.

#### 1273 **E.5.10.7.4.3 `<control-V>`**

1274 There is no additional rationale provided for this subclause.

#### 1275 **E.5.10.7.4.4 `<control-W>`**

1276 There is no additional rationale provided for this subclause.

### 1277 **E.5.10.7.5 *ex* Command Descriptions**

1278 Historically, several commands (e.g., `global`, `v`, `visual`, `s`, `write`, `wq`, `yank`, `!`,  
1279 `<`, `>`, `&`, and `~`) were executable in empty files (i.e., the default address(es) were 0), C  
1280 or permitted explicit addresses of 0 (e.g., 0 was a valid address, or `0,0`, was a C  
1281 valid range). Addresses of 0, or command execution in an empty file, make sense C

1282 only for commands that add new text to the edit buffer or write commands C  
1283 (because users may wish to write empty files). This standard requires this C  
1284 behavior for such commands and disallows it otherwise, for consistency and sim-  
1285 plicity of specification.

1286 A *count* to an `ex` command has been historically corrected to be no greater than  
1287 the last line in a file; e.g., in a five line file, the command `1,6print` would fail,  
1288 but the command `1print300` would succeed. This standard requires conform-  
1289 mance to historical practice.

1290 Historically, the use of flags in `ex` commands could be obscure. General historical  
1291 practice was as described by this standard, but there were some special cases.  
1292 For example, the `list`, `number`, and `print` commands ignored trailing address  
1293 offsets; e.g., “`3p +++#`” would display line 3, and 3 would be the current line after  
1294 the execution of the command. The `open` and `visual` commands ignored both  
1295 the trailing offsets and the trailing flags. Also, *flags* specified to the `open` and  
1296 `visual` commands interacted badly with the `list` edit option, and setting and  
1297 then unsetting it during the `open/visual` session would cause `vi` to stop displaying  
1298 lines in the specified format. For consistency and simplicity of specification, this  
1299 standard does not permit any of these exceptions to the general rule.

1300 This standard uses the word “copy” in several places when discussing buffers.  
1301 This is not intended to imply implementation.

1302 Historically, `ex` users could not specify numeric buffers because of the ambiguity  
1303 this would cause; e.g., in the command `3 delete 2`, it is unclear if 2 is a buffer  
1304 name or a *count*. This standard requires conformance to historical practice by  
1305 default, but does not preclude extensions.

1306 Historically, the contents of the unnamed buffer were frequently discarded after  
1307 commands that did not explicitly affect it; for example, when using the `edit` com-  
1308 mand to switch files. For consistency and simplicity of specification, this standard  
1309 does not permit this behavior.

1310 The `ex` utility did not historically have access to the numeric buffers, and, furth-  
1311 ermore, deleting lines in `ex` did not modify their contents. For example, if, after  
1312 doing a `delete` in `vi`, the user switched to `ex`, did another `delete`, and then  
1313 switched back to `vi`, the contents of the numeric buffers would not have changed.  
1314 This standard requires conformance to historical practice. Numeric buffers are  
1315 described in the `ex` portion of this standard in order to confine the description of  
1316 buffers to a single location in this standard.

1317 The metacharacters that trigger shell expansion in *file* arguments match histori-  
1318 cal practice, as does the method for doing shell expansion. Implementations wish-  
1319 ing to provide users the flexibility to alter the set of metacharacters are  
1320 encouraged to provide a `shellmeta` string edit option.

1321 Historically, `ex` commands executed from `vi` refreshed the screen when it did not  
1322 strictly need to do so; e.g., `!:date > /dev/null` does not require a screen  
1323 refresh because the output of the UNIX `date` command requires only a single line  
1324 of the screen. This standard requires that the screen be refreshed if it has been  
1325 overwritten, but makes no requirements as to how an implementation should

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

1326 make that determination. Implementations may prompt and refresh the screen  
1327 regardless.

1328 The following table is a condensed version of information contained in the norma-  
1329 tive text. It is presented here to facilitate the review of the editor options that  
1330 affect, or are affected by, `ex` commands or addresses. Edit options such as `list`  
1331 and `number`, which affect all commands that display lines, are not exhaustively  
1332 listed.

<u>ex Command</u>	<u>Editor Option</u>	
1334 /	ignorecase, magic, wrapscan	
1335 ?	ignorecase, magic, wrapscan	
1336 !	autoprint, autowrite, shell, warn, readonly, writeany	C C
1338 #, number	list	
1339 <	autoprint, tabstop, shiftwidth	
1340 >	autoprint, tabstop, shiftwidth	
1341 <control-D>	scroll	
1342 append	autoindent, number, shiftwidth	
1343 change	autoindent, number, shiftwidth	
1344 copy	autoprint	
1345 delete	autoprint	
1346 global	ignorecase, magic, report	
1347 insert	autoindent, number, shiftwidth	
1348 join	autoprint	
1349 list	number	
1350 map	remap	
1351 move	autoprint	
1352 next	autowrite, readonly, writeany	
1353 print	list, number	
1354 put	autoprint	
1355 read	autoprint, shell	
1356 rewind	autowrite, readonly, writeany	
1357 s	autoprint, ignorecase, magic	
1358 shell	shell	
1359 stop	autowrite, readonly, writeany	
1360 suspend	autowrite, readonly, writeany	
1361 tag	autoprint, autowrite, taglength, tags, readonly, writeany	
1362 undo	autoprint	
1364 v	ignorecase, magic, report	
1365 visual	window	
1366 write	readonly, shell, writeany	
1367 xit	readonly, writeany	
1368 z	scroll, window	

1369 **E.5.10.7.5.1 abbreviate**

1370 Historical practice was that characters that were entered as part of an abbrevia-  
 1371 tion replacement were subject to `map` expansions, the `showmatch` edit option,  
 1372 further abbreviation expansions, etc.; i.e., they were logically pushed onto the ter-  
 1373 minal input queue, and were not a simple replacement. This standard requires  
 1374 conformance to historical practice. Historical practice was that whenever a non-  
 1375 word character (that had not been escaped by a `<control-V>`) was entered after  
 1376 a word character, `vi` would check for abbreviations. The check was based on the  
 1377 type of the character entered before the word character of the word/nonword pair  
 1378 that triggered the check. The word character of the word/nonword pair that trig- C  
 1379 gered the check and all characters entered before the trigger pair that were of C  
 1380 that type were included in the check, with the exception of `<blank>s`, which  
 1381 always delimited the abbreviation.

1382 This means that, for the abbreviation to work, the *lhs* must end with a word char-  
 1383 acter, there can be no transitions from word to nonword characters (or vice-versa)  
 1384 other than between the last and next-to-last characters in the *lhs*, and there can  
 1385 be no `<blank>` characters in the *lhs*. In addition, because of the historical quoting  
 1386 rules, it was impossible to enter a literal `<control-V>` in the *lhs*. This standard  
 1387 requires conformance to historical practice. Historical implementations did not  
 1388 inform users when abbreviations that could never be used were entered; imple-  
 1389 mentations are strongly encouraged to do so.

1390 For example, the following abbreviations will work:

```
1391      :ab (p REPLACE
1392      :ab p  REPLACE
1393      :ab ((p REPLACE
```

1394 The following abbreviations will not work:

```
1395      :ab (  REPLACE
1396      :ab (pp REPLACE
```

1397 Historical practice is that words on the `vi` colon command line were subject to  
 1398 abbreviation expansion, including the arguments to the `abbrev` (and more  
 1399 interestingly) the `unabbrev` command. Because there are implementations that  
 1400 do not do abbreviation expansion for the first argument to those commands, this  
 1401 is permitted, but not required, by this standard. However, the following  
 1402 sequence:

```
1403      :ab foo bar
1404      :ab foo baz
```

1405 resulted in the addition of an abbreviation of `baz` for the string `bar` in historical  
 1406 `ex/vi`, and the sequence:

```
1407      :ab foo1 bar
1408      :ab foo2 bar
1409      :unabbreviate foo2
```

1410 deleted the abbreviation `foo1`, not `foo2`. These behaviors are not permitted by  
 1411 this standard because they clearly violate the expectations of the user.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



1412 It was historical practice that <control-V>, not backslash, characters be inter-  
 1413 preted as escaping subsequent characters in the abbreviate command. This  
 1414 standard requires conformance to historical practice; however, it should be noted  
 1415 that an abbreviation containing a <blank> will never work.

#### 1416 **E.5.10.7.5.2 append**

1417 Historically, any text following a vertical-line command separator after an C  
 1418 append, change, or insert command became part of the insert text. For exam- C  
 1419 ple, in the command: C

1420 :g/pattern/append|stuff1 C

1421 a line containing the text stuff1 would be appended to each line matching pat- C  
 1422 tern. It was also historically valid to enter: C

1423 :append|stuff1 C

1424 stuff2 C

1425 . C

1426 and the text on the ex command line would be appended along with the text C  
 1427 inserted after it. There was an historical bug, however, that the user had to enter C  
 1428 two terminating lines (the “.” lines) to terminate text input mode in this case. C  
 1429 This standard requires conformance to historical practice, but disallows the his- C  
 1430 torical need for multiple terminating lines. C

#### 1431 **E.5.10.7.5.3 args**

1432 There is no additional rationale provided for this subclause.

#### 1433 **E.5.10.7.5.4 change**

1434 See E.5.10.7.5.2. C

1435 Historical practice for cursor positioning after the change command when no text  
 1436 is input, is as described in this standard. However, one System V implementation  
 1437 (version SVR4.0) is known to have been modified such that the cursor is positioned  
 1438 on the first address specified, and not on the line before the first address. This  
 1439 standard disallows this modification for consistency.

1440 Historically, the change command did not support buffer arguments, although C  
 1441 some implementations allow the specification of an optional buffer. This behavior C  
 1442 is neither required nor disallowed by this standard. C

#### 1443 **E.5.10.7.5.5 chdir**

1444 A common extension in ex implementations is to use the elements of a cdpath C  
 1445 edit option as prefix directories for path arguments to chdir that are relative C  
 1446 pathnames and that do not have . or .. as their first component. Elements in C  
 1447 the cdpath edit option are colon separated. The initial value of the cdpath edit C  
 1448 option is the value of the shell CDPATH environment variable. This feature was C  
 1449 not included in this standard because it does not exist in any of the implementa- C  
 1450 tions considered historical practice by this standard. C

1451 **E.5.10.7.5.6 copy**

1452 Historical implementations of `ex` permitted copies to lines inside of the specified  
 1453 range; e.g., `:2,5copy3` was a valid command. This standard requires confor-  
 1454 mance to historical practice.

1455 **E.5.10.7.5.7 delete**

1456 This standard requires support for the historical parsing of a delete command fol-  
 1457 lowed by flags, without any intervening `<blank>s`. For example:

1458     `ldp`  
 1459     `ldelep` Deletes the first line and prints the line that was second.  
 1460     `ld p` Deletes the first line, saving it in buffer `p`.  
 1461     `ld p11` (Pee-one-ell.) Deletes the first line, saving it in buffer `p`, and listing  
 1462     the line that was second.

1463 **E.5.10.7.5.8 edit**

1464 Historically, any `ex` command could be entered as a *+command* argument to the `edit` C  
 1465 command, although some (e.g., `insert` and `append`) were known to confuse  
 1466 historical implementations. For consistency and simplicity of specification, this  
 1467 standard requires that any command be supported as an argument to the `edit`  
 1468 command.

1469 Historically, the command argument was executed with the current line set to the  
 1470 last line of the file, regardless of whether the `edit` command was executed from  
 1471 visual mode or not. This standard requires conformance to historical practice.

1472 Historically, the *+command* specified to the `edit` and `next` commands was delim-  
 1473 ited by the first `<blank>` character, and there was no way to quote them. For  
 1474 consistency, this standard requires that the usual `ex` backslash quoting be pro-  
 1475 vided.

1476 Historically, specifying the *+command* argument to the `edit` command required a  
 1477 file name to be specified as well; e.g., `:edit +100` would always fail. For con- C  
 1478 sistency and simplicity of specification, this standard does not permit this usage C  
 1479 to fail for that reason. C

1480 Historically, only the cursor position of the last file edited was remembered by the  
 1481 editor. This standard requires that this be supported; however, implementations C  
 1482 are permitted to remember and restore the cursor position for any file previously  
 1483 edited.

1484 **E.5.10.7.5.9 file**

1485 Historical versions of the `ex` editor `file` command displayed a current line and  
 1486 number of lines in the edit buffer of 0 when the file was empty, while the `vi`  
 1487 `<control-G>` command displayed a current line and number of lines in the edit  
 1488 buffer of 1 in the same situation. This standard does not permit this discrepancy,  
 1489 instead requiring that a message be displayed indicating that the file is empty.

1490 **E.5.10.7.5.10 global**

1491 The two-pass operation of the `global` and `v` commands is not intended to imply  
1492 implementation, only the required result of the operation.

1493 The current line and column are set as specified for the individual `ex` commands.  
1494 This requirement is cumulative; i.e., the current line and column must track  
1495 across all the commands executed by the `global` or `v` commands.

1496 **E.5.10.7.5.11 insert**

1497 See E.5.10.7.5.2. C

1498 Historically, `insert` could not be used with an address of zero; i.e., not when the  
1499 edit buffer was empty. This standard requires that this command behave con-  
1500 sistentlly with the `append` command. C

1501 **E.5.10.7.5.12 join**

1502 The action of the `join` command in relation to the special characters is only  
1503 defined for the POSIX Locale because the correct amount of white space after a  
1504 period varies; in Japanese none is required, in French only a single space, and so  
1505 on.

1506 **E.5.10.7.5.13 list**

1507 The historical output of the `list` command was potentially ambiguous. The stan-  
1508 dard developers believed correcting this to be more important than adhering to  
1509 historical practice, and this standard requires unambiguous output.

1510 **E.5.10.7.5.14 map**

1511 Historically, command mode maps only applied to command names; e.g., if the  
1512 character `x` was mapped to `y`, the command `fx` searched for the `x` character, not  
1513 the `y` character. This standard requires this behavior. Historically, entering  
1514 `<control-V>` as the first character of a `vi` command was an error. Several  
1515 implementations have extended the semantics of `vi` such that `<control-V>`  
1516 means that the subsequent command character is not mapped. This is permitted,  
1517 but not required, by this standard. Regardless, using `<control-V>` to escape the  
1518 second or later character in a sequence of characters that might match a com- C  
1519 mand map, or any character in text input mode, is historical practice, and stops  
1520 the entered keys from matching a map. This standard requires conformance to  
1521 historical practice.

1522 Historical implementations permitted digits to be used as a command map *lhs*,  
1523 but then ignored the map. This standard requires that the mapped digits not be  
1524 ignored.

1525 The historical implementation of the `map` command did not permit command  
1526 maps that were more than a single character in length if the first character was  
1527 printable. This behavior is permitted, but not required, by this standard.

1528 Specifications of “function keys” in the `map` command were omitted because the  
 1529 historical specification of such was too simple to be generally useful in a portable  
 1530 manner. Historical practice is that a `#` followed by a number mapped to that  
 1531 number function key; e.g., `#3` was function key 3 for the current terminal, as well  
 1532 as being accessible using the keys `#` and `3`. Implementations have extended this  
 1533 semantic to permit users to specify things like `#up` and `#page_forward` as well.  
 1534 These extensions are permitted, but not required, by this standard.

1535 Historically, mapped characters were remapped unless the `remap` edit option was  
 1536 not set, or the prefix of the mapped characters matched the mapping characters;  
 1537 e.g., in the `map`

```
1538     :map ab abcd
```

1539 the characters `ab` were used as is and were not remapped, but the characters `cd`  
 1540 were mapped if appropriate. This can cause infinite loops in the `vi` mapping  
 1541 mechanisms. This standard requires conformance to historical practice, and that  
 1542 such loops be interruptible.

1543 Text input maps had the same problems with expanding the *lhs* for the `ex map!`  
 1544 and `unmap!` command as did the `ex abbreviate` and `unabbreviate` commands.  
 1545 See the Rationale for the `ex abbreviate` command (E.5.10.7.5.1). This standard  
 1546 requires similar modification of some historical practice for the `map` and `unmap`  
 1547 commands, as described for the `abbreviate` and `unabbreviate` commands.

1548 Historically, maps that were subsets of other maps behaved differently depending  
 1549 on the order in which they were defined. For example:

```
1550     :map! ab     short
1551     :map! abc   long
```

1552 would always translate the characters `ab` to `short`, regardless of how fast the  
 1553 characters `abc` were entered. If the entry order was reversed:

```
1554     :map! abc   long
1555     :map! ab    short
```

1556 the characters `ab` would cause the editor to pause, waiting for the completing `c`  
 1557 character, and the characters might never be mapped to `short`. For consistency  
 1558 and simplicity of specification, this standard requires that the shortest match be  
 1559 used at all times.

1560 The length of time the editor spends waiting for the characters to complete the *lhs*  
 1561 is unspecified because the timing capabilities of systems are often inexact and  
 1562 variable, and it may depend on other factors such as the speed of the connection.  
 1563 The time should be long enough for the user to be able to complete the sequence,  
 1564 but not long enough for the user to have to wait. Some implementations of `vi`  
 1565 have added a `keytime` option, which permits users to set the number of 0,1 s the  
 1566 editor waits for the completing characters. Because mapped terminal function  
 1567 and cursor keys tend to start with an `<ESC>` character, and `<ESC>` is the key end-  
 1568 ing `vi` text input mode, maps starting with `<ESC>` characters are generally  
 1569 exempted from this timeout period, or, at least timed out differently.

1570 **E.5.10.7.5.15 mark**

1571 Historically, users were able to set the “previous context” marks explicitly. In  
 1572 addition, the `ex` commands `''` and `'\`` and the `vi` commands `''`, `\``, `''`, and `'\``  
 1573 all referred to the same mark. In addition, the previous context marks were not  
 1574 set if the command with which the address setting the mark was associated,  
 1575 failed. This standard requires conformance to historical practice. Historically, if  
 1576 marked lines were deleted, the mark was also deleted, but would reappear if the  
 1577 change was undone. This standard requires conformance to historical practice.

1578 The description of the special events that set the ``` and `'` marks matches histori-  
 1579 cal practice. For example, historically the command `/a/,/b/` did not set the ```  
 1580 and `'` marks, but the command `/a/,/b/delete` did.

1581 **E.5.10.7.5.16 move**

1582 There is no additional rationale provided for this subclause.

1583 **E.5.10.7.5.17 next**

1584 Historically, any `ex` command could be entered as a *+command* argument to the C  
 1585 `next` command, although some (e.g., `insert` and `append`) were known to confuse C  
 1586 historical implementations. This standard requires that any command be permit- C  
 1587 ted and that it behave as specified. The `next` command can accept more than one C  
 1588 file, so usage such as

```
1589     next `ls [abc]*`
```

1590 is valid; it need not be valid for the `edit` or `read` commands, for example,  
 1591 because they expect only one file name.

1592 Historically, the `next` command behaved differently from the `:rewind` command  
 1593 in that it ignored the force flag if the `autowrite` flag was set. For consistency,  
 1594 this standard does not permit this behavior.

1595 Historically, the `next` command positioned the cursor as if the file had never been  
 1596 edited before, regardless. This standard does not permit this behavior, for con- C  
 1597 sistency with the `edit` command.

1598 Implementations wanting to provide a counterpart to the `next` command that  
 1599 edited the previous file have used the command `prev[ious]`, which takes no *file*  
 1600 argument. This standard does not require this command.

1601 **E.5.10.7.5.18 number**

1602 There is no additional rationale provided for this subclause.

1603 **E.5.10.7.5.19 open**

1604 Historically, the `open` command would fail if the `open edit` option was not set.  
 1605 This standard does not mention the `open edit` option and does not require this  
 1606 behavior. Some historical implementations do not permit entering open mode  
 1607 from open or visual mode, only from `ex` mode. For consistency, this standard does

1608 not permit this behavior.

1609 Historically, entering `open` mode from the command line (i.e., `vi +open`) resulted  
1610 in anomalous behaviors; e.g., the `ex` file and set commands, and the `vi` command  
1611 `<control-G>` did not work. For consistency, this standard does not permit this  
1612 behavior.

1613 Historically, the `open` command only permitted `/` characters to be used as the  
1614 search pattern delimiter. For consistency, this standard requires that the search  
1615 delimiters used by the `s`, `global`, and `v` commands be accepted as well.

#### 1616 **E.5.10.7.5.20** `preserve`

1617 The `preserve` command does not historically cause the file to be considered  
1618 unmodified for the purposes of future commands that may exit the editor. This  
1619 standard requires conformance to historical practice.

1620 Historical documentation stated that mail was not sent to the user when `preserve`  
1621 was executed; however, historical implementations did send mail in this case.  
1622 This standard requires conformance to the historical implementations.

#### 1623 **E.5.10.7.5.21** `print`

1624 The writing of NUL by the `print` command is not specified as a special case  
1625 because the standard developers did not want to require `ex` to support NUL char-  
1626 acters. Historically, characters were displayed using the ARPA standard map-  
1627 pings, which are as follows:

- 1628 (1) Printable characters are left alone.
- 1629 (2) Control characters less than `\177` are represented as `^` followed by the  
1630 character offset from the `@` character in the ASCII map; e.g., `\007` is  
1631 represented as `^G`.
- 1632 (3) `\177` is represented as `^` followed by `?`.

1633 The display of characters having their eighth bit set was less standard. Existing  
1634 implementations use hex (`0x00`), octal (`\000`) and a meta-bit display. (The latter  
1635 displayed bytes that had their eighth bit set as the two characters “M-,” followed  
1636 by the seven-bit display as described above.) The latter probably has the best  
1637 claim to historical practice because it was used for the `-v` option of 4BSD- and  
1638 4BSD-derived versions of the `cat` utility since 1980.

1639 No specific display format is required by this standard.

1640 Explicit dependence on the ASCII character set has been avoided where possible,  
1641 hence the use of the phrase an “implementation-defined multicharacter sequence”  
1642 for the display of nonprintable characters in preference to the historical usage of,  
1643 for instance, `^I` for `<tab>`. Implementations are encouraged to conform to histor-  
1644 ical practice in the absence of any strong reason to diverge.

1645 Historically, all `ex` commands beginning with the letter `p` could be entered using  
1646 capitalized versions of the commands; e.g., `P[rint]`, `Pre[serve]`, and `Pu[t]` were  
1647 all valid command names. This standard permits, but does not require, this

1648 historical practice because capital forms of the commands are used by some imple-  
1649 mentations for other purposes.

1650 **E.5.10.7.5.22 put**

1651

C

1652 Historically, an `ex put` command, executed from open or visual mode, was the  
1653 same as the open or visual mode `P` command, if the buffer was named and was cut  
1654 in character mode, and the same as the `p` command if the buffer was named and  
1655 cut in line mode. If the unnamed buffer was the source of the text, the entire line  
1656 from which the text was taken was usually put, and the buffer was handled as if  
1657 in line mode, but it was possible to get extremely anomalous behavior. In addi-  
1658 tion, using the `Q` command to switch into `ex` mode, and then doing a `put` often  
1659 resulted in errors as well, such as appending text that was unrelated to the (sup-  
1660 posed) contents of the buffer. For consistency and simplicity of specification, this  
1661 standard does not permit these behaviors. All `ex put` commands are required to  
1662 operate in line mode, and the contents of the buffers are not altered by changing  
1663 the mode of the editor.

1664 **E.5.10.7.5.23 quit**

1665 There is no additional rationale provided for this subclause.

1666 **E.5.10.7.5.24 read**

1667 Historically, an `ex read` command executed from open or visual mode, executed  
1668 in an empty file, left an empty line as the first line of the file. For consistency and  
1669 simplicity of specification, this standard does not permit this behavior. Histori-  
1670 cally, a `read` in open or visual mode from a program left the cursor at the last line  
1671 read in, not the first. For consistency, this standard does not permit this  
1672 behavior.

C

C

1673 Historical implementations of `ex` were unable to undo `read` commands that read  
1674 from the output of a program. For consistency, this standard does not permit this  
1675 behavior.

1676 Historically, the `ex` and `vi` message after a successful `read` or `write` command  
1677 specified “characters,” not “bytes.” This standard requires that the number of  
1678 bytes be displayed, not the number of characters, because it may be difficult in  
1679 multibyte implementations to determine the number of characters read. Imple-  
1680 mentations are encouraged to clarify the message displayed to the user.

1681 Historically, reads were not permitted on files other than type regular, except that  
1682 FIFO files could be read (probably only because they did not exist when `ex` and `vi`  
1683 were originally written). Because the historical `ex` evaluated `read!` and `read !`  
1684 equivalently, there can be no optional way to force the `read`. This standard per-  
1685 mits, but does not require, this behavior.

1686 **E.5.10.7.5.25 recover**

1687 Some historical implementations of the editor permitted users to recover the edit  
 1688 buffer contents from a previous edit session, and then exit without saving those  
 1689 contents (or explicitly discarding them). The intent of this standard in requiring  
 1690 that the edit buffer be treated as already modified is to prevent this user error.

1691 **E.5.10.7.5.26 rewind**

1692 Historical implementations supported the `rewind` command when the user was  
 1693 editing the first file in the list; i.e., the file that the `rewind` command would edit.  
 1694 This standard requires conformance to historical practice.

1695 **E.5.10.7.5.27 s**

1696 Historically, `ex` accepted an `r` option to the `s` command. The effect of the `r` option  
 1697 was to use the last RE used in any command as the pattern, the same as the `~`  
 1698 command. The `r` option is not required by this standard. Historically, the `c` and  
 1699 `g` options were toggled; e.g., the command `:s/abc/def/` was the same as  
 1700 `s/abc/def/ccccgggg`. For simplicity of specification, this standard does not  
 1701 permit this behavior. C  
C

1702 Historically, the `edcompatible` edit option made the values of the `c` and `g`  
 1703 suffixes remembered instead of reinitializing them to “off” for each `s` command.  
 1704 The single special case was that they were always reinitialized to zero if the pat-  
 1705 tern and replacement strings were specified. This standard does not specify this  
 1706 behavior or the `edcompatible` edit option.

1707 The tilde command is often used to replace the last search RE. For example, in  
 1708 the sequence

```
1709     s/red/blue/
1710     /green
1711     ~
```

1712 the `~` command is equivalent to:

```
1713     s/green/blue/
```

1714 Historically, `ex` accepted all of the following forms:

```
1715     s/abc/def/
1716     s/abc/def
1717     s/abc/
1718     s/abc
```

1719 This standard requires conformance to this historical practice.

1720 The `s` command presumes that the `^` character only occupies a single column in  
 1721 the display. Much of the `ex` and `vi` specification presumes that the `<space>`  
 1722 character only occupies a single column in the display. There are no known char-  
 1723 acter sets for which this is not true.

1724 Historically, the final column position for the substitute commands was based on  
 1725 previous column movements; a search for a pattern followed by a substitution



1726 would leave the column position unchanged, while a `0` command followed by a  
1727 substitution would change the column position to the first nonblank. For con-  
1728 sistency and simplicity of specification, this standard requires that the final  
1729 column position always be set to the first nonblank.

1730 **E.5.10.7.5.28 set**

1731 Historical implementations redisplayed all of the options for each occurrence of  
1732 the `all` keyword. This standard permits, but does not require, this behavior.

1733 **E.5.10.7.5.29 shell**

1734 There is no additional rationale provided for this subclause.

1735 **E.5.10.7.5.30 source**

1736 Source commands can be nested to arbitrary depths, and should be limited only  
1737 by system resources.

1738 **E.5.10.7.5.31 suspend**

1739 There is no additional rationale provided for this subclause.

1740 **E.5.10.7.5.32 tag**

1741 No requirement is made as to where `ex` and `vi` shall look for the file referenced by  
1742 the `tag` entry. Historical practice has been to look for the path found in the `tags`  
1743 file, based on the current directory. A useful extension found in some implemen-  
1744 tations is to look based on the directory containing the `tags` file that held the  
1745 entry, as well. No requirement is made as to which reference for the `tag` in the  
1746 `tags` file is used. This is deliberate, in order to permit extensions such as multiple  
1747 entries in a `tags` file for a `tag`.

1748 Because users often specify many different `tags` files, some of which need not be  
1749 relevant or exist at any particular time, this standard requires that error mes-  
1750 sages about problem `tags` files be displayed only if the requested `tag` is not found,  
1751 and then, only once for each time that the `tag` edit option is changed.

1752 The requirement that the current edit buffer be unmodified is only necessary if  
1753 the file indicated by the `tag` entry is not the same as the current file (as defined by  
1754 the current pathname). Historically, the file would be reloaded if the file name  
1755 had changed, as well as if the file name was different from the current pathname.  
1756 For consistency and simplicity of specification, this standard does not permit this  
1757 behavior, requiring that the name be the only factor in the decision.

1758 Historically, `vi` only searched for `tags` in the current file from the current cursor  
1759 to the end of the file, and therefore, if the `wraps` option was not set, `tags`  
1760 occurring before the current cursor were not found. This standard considers this  
1761 a bug, and implementations are required to search for the first occurrence in the  
1762 file, regardless.

1763 **E.5.10.7.5.33 unabbreviate**

1764 There is no additional rationale provided for this subclause.

1765 **E.5.10.7.5.34 undo**

1766 The `undo` description deliberately uses the word “modified.” The `undo` command  
1767 is not intended to `undo` commands that replace the contents of the edit buffer,  
1768 such as `edit`, `next`, `tag`, or `recover`.

1769 Cursor positioning after the `undo` command was inconsistent in the historical `vi`,  
1770 sometimes attempting to restore the original cursor position (`global`, `undo`, and  
1771 `v` commands), and sometimes, in the presence of maps, placing the cursor on the  
1772 last line added or changed instead of the first. This standard requires a  
1773 simplified behavior for consistency and simplicity of specification.

1774 **E.5.10.7.5.35 unmap**

1775 There is no additional rationale provided for this subclause.

1776 **E.5.10.7.5.36 version**

1777 The `version` command cannot be exactly specified since there is no widely  
1778 accepted definition of what the version information should contain. Implementa-  
1779 tions are encouraged to do something reasonably intelligent.

1780 **E.5.10.7.5.37 visual**

1781 There is no additional rationale provided for this subclause.

1782 **E.5.10.7.5.38 write**

1783 Historically, the `ex` and `vi` message after a successful `read` or `write` command  
1784 specified “characters”, not “bytes.” This standard requires that the number of  
1785 bytes be displayed, not the number of characters because it may be difficult in  
1786 multibyte implementations to determine the number of characters written.  
1787 Implementations are encouraged to clarify the message displayed to the user.

1788 Implementation-defined tests are permitted so that implementations can make  
1789 additional checks; e.g., for locks or file modification times.

1790 Historically, attempting to append to a nonexistent file caused an error. It has  
1791 been left unspecified in this standard to permit implementations to let the write  
1792 succeed, so that the `append` semantics are similar to those of the historical `csh`. C

1793 Historical `vi` permitted empty edit buffers to be written. However, since the way  
1794 `vi` got around dealing with “empty” files was to always have a line in the edit  
1795 buffer, no matter what, it wrote them as files of a single, empty line. This stan-  
1796 dard does not permit this behavior.

1797 Historically, `ex` restored standard output and standard error to their values as of  
1798 when `ex` was invoked, before writes to programs were performed. This could dis-  
1799 turb the terminal configuration as well as be a security issue for some terminals.

- 1800 This standard does not permit this, requiring that the program output be cap-  
1801 tured and displayed as if by the `ex print` command.
- 1802 **E.5.10.7.5.39 `xit`**
- 1803 There is no additional rationale provided for this subclause.
- 1804 **E.5.10.7.5.40 `yank`**
- 1805 There is no additional rationale provided for this subclause.
- 1806 **E.5.10.7.5.41 `z`**
- 1807 Historically, the line count was set to the value of the `scroll` option if the type  
1808 character was end-of-file. This feature was broken on most historical implemen-  
1809 tations long ago, however, and is not documented anywhere. For this reason, this  
1810 standard is resolutely silent.
- 1811 Historically, the `z` command was `<blank>`-sensitive and “`z +`” and “`z -`” did dif- C  
1812 ferent things than “`z+`” and “`z-`” because the type could not be distinguished from C  
1813 a flag. (The commands “`z .`” and “`z =`” were historically invalid.) This standard C  
1814 requires conformance to this historical practice. C
- 1815 Historically, the `z` command was further `<blank>`-sensitive in that the *count* C  
1816 could not be `<blank>`-delimited; e.g., the commands “`z= 5`” and “`z- 5`” were also C  
1817 invalid. Because the *count* is not ambiguous with respect to either the type char- C  
1818 acter or the flags, this is not permitted by this standard. C
- 1819 **E.5.10.7.5.42 `!`**
- 1820 Historically, `ex` filter commands only read the standard output of the commands,  
1821 letting standard error appear on the terminal as usual. The `vi` utility, however,  
1822 read both standard output and standard error. This standard requires the latter  
1823 behavior for both `ex` and `vi`, for consistency.
- 1824 **E.5.10.7.5.43 `<`**
- 1825 Historically, it was possible to add shift characters to increase the effect of the C  
1826 command; e.g., `<<< outdented` (or `>>> indented`) the line(s) 3 levels of indentation C  
1827 instead of the default 1. This standard requires conformance to historical prac- C  
1828 tice. C
- 1829 **E.5.10.7.5.44 `>`**
- 1830 See E.5.10.7.5.43. C
- 1831 **E.5.10.7.5.45 `<control-D>`**
- 1832 Historically, the `<control-D>` command erased the prompt, providing the user  
1833 with an unbroken presentation of lines from the edit buffer. This is not required  
1834 by this standard; implementations are encouraged to provide it if possible. His-  
1835 torically, the `<control-D>` command took, and then ignored, a count. This

1836 standard does not permit this behavior.

1837 **E.5.10.7.5.46 =**

1838 Historically, the `ex =` command, when executed in `ex` mode in an empty edit C  
 1839 buffer, reported 0, and from open or visual mode, reported 1. For consistency and C  
 1840 simplicity of specification, this standard does not permit this behavior. C

1841 **E.5.10.7.5.47 @**

1842 Historically, `ex` did not correctly handle the inclusion of text input commands  
 1843 (i.e., append, insert, and change) in executed buffers. This standard does not  
 1844 permit this exclusion for consistency.

1845 Historically, the logical contents of the buffer being executed did not change if the  
 1846 buffer itself were modified by the commands being executed; i.e., buffer execution  
 1847 did not support self-modifying code. This standard requires conformance to his-  
 1848 torical practice.

1849 Historically, the `@` command took a range of lines, and the `@` buffer was executed  
 1850 once per line, with the current line (`.`) set to each specified line. This standard  
 1851 requires conformance to historical practice.

1852 Some historical implementations did not notice if errors occurred during buffer  
 1853 execution. This, coupled with the ability to specify a range of lines for the `ex @`  
 1854 command, makes it trivial to cause them to drop core. This standard requires  
 1855 that implementations stop buffer execution if any error occurs, if the specified line  
 1856 doesn't exist, or if the contents of the edit buffer itself are replaced (e.g., the buffer  
 1857 executes the `ex :edit` command).

1858 **E.5.10.7.6 REs**

1859 Historical practice is that the characters in the replacement part of the last `s`  
 1860 command; i.e., those matched by entering a `~` in the RE were not further expanded  
 1861 by the RE engine. So, if the characters contained the string `a.`, they would match  
 1862 `a` followed by `.`, and not `a` followed by any character. This standard requires con-  
 1863 formance to historical practice.

1864 **E.5.10.7.7 Replacement Strings**

1865 An example of case conversion with the `s` command:

```
1866      :p
1867      The cat sat on the mat.
1868      :s/\<.at\>/\u&/gp
1869      The Cat Sat on the Mat.
1870      :s/S\(.*\)M/S\U\1\eM/p
1871      The Cat SAT ON THE Mat.
```

1872 **E.5.10.7.8 Edit Options**

1873 The following paragraphs describe the historical behavior of some edit options C  
 1874 that were not, for whatever reason, included in the POSIX.2 standard. Implemen- C  
 1875 tations are strongly encouraged to only use these names if the functionality C  
 1876 described here is fully supported. C

1877 `beautify`

1878 The historical `beautify` edit option behaved as follows: In `ex` mode, keys  
 1879 that were not already specially handled, that were less than an ASCII space  
 1880 or were the `<DEL>` (`\177`) key, and were neither a `<tab>` nor a `<form-`  
 1881 `feed>`, and were read in from an `ex` script file, were discarded. When the  
 1882 first `<control-H>` was discarded a message was written to the terminal.  
 1883 Quoting (with a `\`) would keep the keys from being discarded.

1884 In `open` or `visual` mode, keys that were not already specially handled, that  
 1885 were less than an ASCII space or were the `<DEL>` (`\177`) key, and were nei-  
 1886 ther a `<tab>` nor a `<form-feed>`, and were entered in input mode (either  
 1887 to the edit buffer or to the colon command line), were discarded. Quoting  
 1888 (using a `<control-V>`) would keep the keys from being discarded.

1889 For various reasons, among them internationalization concerns, this stan-  
 1890 dard does not require the `beautify` option.

1891 `directory`

1892 The `directory` edit option historically specified the pathname of the direc-  
 1893 tory where temporary files (although not the backup file used for recovery) C  
 1894 were created by `ex` or `vi`. This option was omitted from this standard  
 1895 because the default value was always implementation specific.

1896 `edcompatible`

1897 The `edcompatible` edit option historically caused the `c` and `g` suffixes to  
 1898 the `s` command to be remembered, instead of initializing them to unset for  
 1899 each new `s` command. (Note that specifying both the pattern and replace-  
 1900 ment strings to the `s` command reset the `c` and `g` suffixes as well.) This  
 1901 option was omitted from this standard because it was not believed to be  
 1902 widely used, or generally useful.

1903 `extended`

1904 The `extended` edit option has been used in some implementations of `vi` to  
 1905 provide EREs instead of BREs. This option was omitted from this standard  
 1906 because it is not widespread historical practice.

1907 `flash`

1908 The `flash` edit option historically caused the screen to flash instead of  
 1909 beeping on error. This option was omitted from this standard because it is  
 1910 not found in some historical implementations.

1911 `hardtabs`

1912 The `hardtabs` edit option historically defined the number of columns  
 1913 between hardware tab settings. This option was omitted from this stan-  
 1914 dard because it was believed to no longer be generally useful.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 1915 `lisp`
- 1916 The `lisp` edit option historically altered the behavior of the `autoindent`
- 1917 edit option and the `(, )`, `{, }`, `[[, and ]]` commands to match the LISP
- 1918 language. In addition, there was a command `=` (`reindent`) that was avail-
- 1919 able only in `lisp` mode. This option was omitted from this standard
- 1920 because it was difficult to justify the inclusion of programming-language
- 1921 dependent features.
- 1922 `modeline`
- 1923 The `modeline` (sometimes named `modelines`) edit option(s) historically
- 1924 caused `ex` or `vi` to read the five first and last lines of the file for editor com-
- 1925 mands. This option is a security problem, and vendors are strongly
- 1926 encouraged to delete it from historical implementations.
- 1927 `open`
- 1928 The `open` edit option historically disallowed the `ex open` and `visual com-` C
- 1929 mands. This edit option was omitted from this standard because these C
- 1930 commands are required by this standard. C
- 1931 `optimize`
- 1932 The `optimize` edit option historically expedited text throughput by setting
- 1933 the terminal to not do automatic carriage returns when printing more than
- 1934 one logical line of output. This option was omitted from this standard
- 1935 because it was intended for terminals without addressable cursors, which
- 1936 are rarely, if ever, still used.
- 1937 `redraw`
- 1938 The `redraw` edit option historically simulated an intelligent terminal on a
- 1939 dumb terminal. This option was omitted from this standard because it was
- 1940 intended for terminals which are rarely, if ever, still used.
- 1941 `ruler`
- 1942 The `ruler` edit option has been used in some implementations of `vi` to
- 1943 present a current row/column ruler for the user. This option was omitted
- 1944 from this standard because it is not widespread historical practice.
- 1945 `sourceany`
- 1946 The `sourceany` edit option historically caused `ex` or `vi` to source startup
- 1947 files that were owned by users other than the user running the editor. This
- 1948 option is a security problem, and vendors are strongly encouraged to
- 1949 remove it from their implementations.
- 1950 `timeout`
- 1951 The `timeout` edit option historically enabled the (now standard) feature of
- 1952 only waiting for a short period before returning keys that could be part of a
- 1953 macro. This feature was omitted from this standard because its behavior is
- 1954 now standard, it is not widely useful, and it was rarely documented.
- 1955 `verbose`
- 1956 The `verbose` edit option has been used in some implementations of `vi` to
- 1957 cause `vi` to output error messages for common errors; e.g., attempting to
- 1958 move the cursor past the beginning or end of the line instead of only

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 1959 alerting the screen. (The historical `vi` only alerted the terminal and  
 1960 presented no message for such errors. The historical editor option `terse`  
 1961 did not select when to present error messages, it only made existing error  
 1962 messages more or less verbose.) This option was omitted from this standard  
 1963 because it is not widespread historical practice; however, implementors are  
 1964 encouraged to use it if they wish to provide error messages for naive users.
- 1965 `wraplen`  
 1966 The `wraplen` edit option has been used in some implementations of `vi` to  
 1967 specify an automatic margin measured from the left margin instead of from  
 1968 the right margin. This is useful when multiple screen sizes are being used C  
 1969 to edit a single file. This option was omitted from this standard because it C  
 1970 is not widespread historical practice; however, implementors are  
 1971 encouraged to use it if they add this functionality.
- 1972 **E.5.10.7.8.1 autoindent**
- 1973 Historically, the command `0a` did not do any autoindentation, regardless of the  
 1974 current indentation of line 1. This standard requires that any indentation  
 1975 present in line 1 be used.
- 1976 **E.5.10.7.8.2 autoprint**
- 1977 Historically, the `autoprint` edit option was not completely consistent or based  
 1978 solely on modifications to the edit buffer. Exceptions were the `read` command C  
 1979 (when reading from a file, but not from a filter), the `append`, `change`, `insert`,  
 1980 `global`, and `v` commands, all of which were not affected by `autoprint`, and the  
 1981 `tag` command, which was affected by `autoprint`. This standard requires confor-  
 1982 mance to historical practice.
- 1983 Historically, the `autoprint` option only applied to the last of multiple commands C  
 1984 entered using vertical-bar delimiters; e.g. `delete<newline>` was affected by C  
 1985 `autoprint`, but `delete|version<newline>` was not. This standard requires C  
 1986 conformance to historical practice. C
- 1987 **E.5.10.7.8.3 autowrite**
- 1988 Appending the `!` character to the `ex next` command to avoid performing an  
 1989 automatic write was not supported in historical implementations. This standard  
 1990 requires that the behavior match the other `ex` commands for consistency.
- 1991 **E.5.10.7.8.4 errorbells**
- 1992 There is no additional rationale provided for this subclause.
- 1993 **E.5.10.7.8.5 exerc**
- 1994 There is no additional rationale provided for this subclause.

1995 **E.5.10.7.8.6 ignorecase**

1996 Historical implementations of case-insensitive matching (the `ignorecase` edit  
1997 option) lead to counterintuitive situations when uppercase characters were used  
1998 in range expressions. Historically, the process was as follows:

- 1999 (1) Take a line of text from the edit buffer C  
2000 (2) Convert uppercase to lowercase in text line  
2001 (3) Convert uppercase to lowercase in REs, except in character class  
2002 specifications  
2003 (4) Match REs against text

2004 This would mean that, with `ignorecase` in effect, the text

2005 `The cat sat on the mat`

2006 would be matched by

2007 `/^the/`

2008 but not by

2009  `/^[A-Z]he/`

2010 For consistency with other commands implementing REs, this standard does not  
2011 permit this behavior.

2012 **E.5.10.7.8.7 list**

2013 There is no additional rationale provided for this subclause.

2014 **E.5.10.7.8.8 magic**

2015 There is no additional rationale provided for this subclause.

2016 **E.5.10.7.8.9 msg**

2017 There is no additional rationale provided for this subclause.

2018 **E.5.10.7.8.10 number**

2019 There is no additional rationale provided for this subclause.

2020 **E.5.10.7.8.11 paragraphs**

2021 Earlier versions of this standard made the default `paragraphs` and `sections`  
2022 edit options implementation-defined, arguing they were historically oriented to  
2023 the UNIX system `troff` text formatter, and a “portable user” could use the `{, },`  
2024 `[[, ]], (, and )` commands in open or visual mode and have the cursor stop in  
2025 unexpected places. This version of the standard specifies their values in the  
2026 POSIX Locale because the unusual grouping (they only work when grouped into  
2027 two characters at a time) means that they cannot be used for general purpose  
2028 movement, regardless.



2029 **E.5.10.7.8.12 prompt**

2030 There is no additional rationale provided for this subclause.

2031 **E.5.10.7.8.13 readonly**

2032 Implementations are encouraged to provide the best possible information to the  
 2033 user as to the readonly status of the file, with the exception that they should not  
 2034 consider the current special privileges of the process. This provides users a safety  
 2035 net because they must force the overwrite of readonly files, even when running  
 2036 with additional privileges.

2037 The `readonly` edit option specification largely conforms to historical practice. C  
 2038 The only difference is that historical implementations did not notice that the user C  
 2039 had set the `readonly` edit option in cases where the file was already marked C  
 2040 `readonly` for some reason, and would therefore reinitialize the `readonly` edit C  
 2041 option the next time the contents of the edit buffer were replaced. This behavior C  
 2042 is disallowed by this standard. C

2043 **E.5.10.7.8.14 remap**

2044 There is no additional rationale provided for this subclause.

2045 **E.5.10.7.8.15 report**

2046 The requirement that lines copied to a buffer interact differently than deleted C  
 2047 lines is historical practice. For example, if the `report` edit option is set to 3, C  
 2048 deleting 3 lines will cause a report to be written, but 4 lines must be copied before C  
 2049 a report is written. C

2050 The requirement that the `ex global`, `v`, `open`, `undo`, and `visual` commands C  
 2051 present reports based on the total number of lines added or deleted during the C  
 2052 command execution, and, that commands executed by the `global` and `v` com- C  
 2053 mands not present reports, is historical practice. This standard extends historical C  
 2054 practice by requiring that buffer execution be treated similarly. The reasons for C  
 2055 this are two-fold. Historically, only the report by the last command executed from C  
 2056 the buffer would be seen by the user, as each new report would overwrite the last. C  
 2057 In addition, the standard developers believed that buffer execution had more in C  
 2058 common with `global` and `v` commands than it did with other `ex` commands, and C  
 2059 should behave similarly, for consistency and simplicity of specification. C

2060 **E.5.10.7.8.16 scroll**

2061 There is no additional rationale provided for this subclause.

2062 **E.5.10.7.8.17 sections**

2063 See E.5.10.7.8.11.

2064 **E.5.10.7.8.18 shell**

2065 There is no additional rationale provided for this subclause.

2066 **E.5.10.7.8.19 shiftwidth**

2067 There is no additional rationale provided for this subclause.

2068 **E.5.10.7.8.20 showmatch**

2069 The length of time the cursor spends on the matching character is unspecified  
 2070 because the timing capabilities of systems are often inexact and variable. The  
 2071 time should be long enough for the user to notice, but not long enough for the user  
 2072 to become annoyed. Some implementations of `vi` have added a `matchtime` option  
 2073 that permits users to set the number of 0,1 s intervals the cursor pauses on the  
 2074 matching character.

2075 **E.5.10.7.8.21 showmode**

2076 The `showmode` option has been used in some historical implementations of `ex` and  
 2077 `vi` to display the current editing mode when in open or visual mode. The editing C  
 2078 modes have generally included “command” and “input,” and sometimes other  
 2079 modes such as “replace” and “change.” The string was usually displayed on the  
 2080 bottom line of the screen at the far right hand corner. In addition, a preceding \*  
 2081 character often denoted if the contents of the edit buffer had been modified. The  
 2082 latter display has sometimes been part of the `showmode` option, and sometimes  
 2083 based on another option. This option was not available in the 4BSD historical  
 2084 implementation of `vi`, but was viewed as generally useful, particularly to novice  
 2085 users, and is required by this standard.

2086 The `smd` shorthand for the `showmode` option was not present in all historical C  
 2087 implementations of the editor. This standard requires it, for consistency. C

2088 Not all historical implementations of the editor displayed a mode string for com- C  
 2089 mand mode, differentiating command mode from text input mode(s) by the C  
 2090 absence of a mode string. This standard permits this behavior for consistency C  
 2091 with historical practice, but implementations are encouraged to provide a display C  
 2092 string for both modes. C

2093 **E.5.10.7.8.22 slowopen**

2094 Historically the `slowopen` option was automatically set if the terminal baud rate  
 2095 was less than 1200 baud, or if the baud rate was 1200 baud and the `redraw`  
 2096 option was not set. The `slowopen` option had two effects. First, when inserting  
 2097 characters in the middle of a line, characters after the cursor would not be pushed  
 2098 ahead, but would appear to be overwritten. Second, when creating a new line of  
 2099 text, lines after the current line would not be scrolled down, but would appear to  
 2100 be overwritten. In both cases, ending text input mode would cause the screen to  
 2101 be refreshed to match the actual contents of the edit buffer. Finally, terminals  
 2102 that were sufficiently intelligent caused the editor to ignore the `slowopen` option.  
 2103 This standard permits most historical behavior, extending historical practice to

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2104 require `slowopen` behaviors if the `edit` option is set by the user.

2105 **E.5.10.7.8.23 `tabstop`**

2106 Tabstops are not related to the configured tabstops of the terminal hardware.

2107 **E.5.10.7.8.24 `taglength`**

2108 There is no additional rationale provided for this subclause.

2109 **E.5.10.7.8.25 `tags`**

2110 The default path for tags files is left unspecified as implementations may have  
2111 their own tags implementations that do not correspond to the historical ones. The  
2112 default tags option value should probably at least include the file `./tags`.

2113 **E.5.10.7.8.26 `term`**

2114 Historical implementations of `ex` and `vi` ignored changes to the `term` edit option  
2115 after the initial terminal information was loaded. This is permitted by this stan-  
2116 dard; however, implementations are encouraged to permit the user to modify their  
2117 terminal type at any time.

2118 **E.5.10.7.8.27 `terse`**

2119 Historically, the `terse` edit option optionally provided a shorter, less descriptive  
2120 error message, for some error messages. This is permitted, but not required, by  
2121 this standard. Historically, most common visual mode errors (e.g., trying to move  
2122 the cursor past the end of a line) did not result in an error message, but simply  
2123 alerted the terminal. Implementations wishing to provide messages for novice  
2124 users are urged to do so based on the edit option `verbose`, and not `terse`.

2125 **E.5.10.7.8.28 `warn`**

2126 There is no additional rationale provided for this subclause.

2127 **E.5.10.7.8.29 `window`**

2128 In historical implementations, the default for the `window` edit option was based  
2129 on the baud rate as follows:

2130 (1) If the baud rate was less than 1200, the edit option `w300` set the window  
2131 value; e.g., the line:

2132 `set w300=12`

2133 would set the `window` option to 12 if the baud rate was less than 1200.

2134 (2) If the baud rate was equal to 1200, the edit option `w1200` set the window  
2135 value.

2136 (3) If the baud rate was greater than 1200, the edit option `w9600` set the  
2137 window value.

2138 The `w300`, `w1200`, and `w9600` options do not appear in this standard because of  
2139 their dependence on specific baud rates.

2140 In historical implementations, the size of the window displayed by various com-  
2141 mands was related to, but not necessarily the same as, the `window edit` option.  
2142 For example, the size of the window was set by the `ex` command `visual 10`, but  
2143 it did not change the value of the `window edit` option. However, changing the  
2144 value of the `window edit` option did change the number of lines that were  
2145 displayed when the screen was repainted. This standard does not permit this  
2146 behavior in the interests of consistency and simplicity of specification, and  
2147 requires that all commands that change the number of lines that are displayed do  
2148 it by setting the value of the `window edit` option.

#### 2149 **E.5.10.7.8.30** `wrapmargin`

2150 Historically, the `wrapmargin` option did not affect maps inserting characters that  
2151 also had associated counts; e.g., “`:map K 5aABC DEF.`” Unfortunately, there are  
2152 widely used maps that depend on this behavior. For consistency and simplicity of  
2153 specification, this standard does not permit this behavior.

2154

C

2155 Historically, `wrapmargin` was calculated using the column display width of all  
2156 characters on the screen. For example, an implementation using `^I` to represent  
2157 `<tab>s` when the `list edit` option was set, where `^` and `I` each took up a single  
2158 column on the screen, would calculate the `wrapmargin` based on a value of 2 for  
2159 each `<tab>` character. The `number edit` option similarly changed the effective  
2160 length of the line as well. This standard requires conformance to historical prac-  
2161 tice.

#### 2162 **E.5.10.7.8.31** `wrapsan`

2163 There is no additional rationale provided for this subclause.

#### 2164 **E.5.10.7.8.32** `writeany`

2165 There is no additional rationale provided for this subclause.

### 2166 **E.5.10.8 Exit Status**

2167 There is no additional rationale provided for this subclause.

### 2168 **E.5.10.9 Consequences of Errors**

2169 There is no additional rationale provided for this subclause.

2170	⇒ <b>E.5.7 ctags Rationale.</b> <i>Change the seventh paragraph (the one beginning</i>	B
2171	<i>“Historically, . . .”) to:</i>	B
2172	Historically, the tags file has been used only by <code>ex</code> and <code>vi</code> . However, the for-	B
2173	mat of the tags file has been published to encourage other programs to use the	B
2174	tags in new ways. The format allows either search patterns or line numbers to	B
2175	find the identifiers because the historical <code>vi</code> recognizes either. The <code>ctags</code>	B
2176	utility does not produce the format using line numbers because it is not useful	B
2177	following any source file changes that add or delete lines. The documented	B
2178	search patterns match historical practice. It should be noted that literal lead-	B
2179	ing circumflex or trailing dollar-sign characters in the search pattern will only	B
2180	behave correctly if anchored to the beginning of the line or end of the line by	B
2181	an additional circumflex or dollar-sign character.	B
2182	⇒ <b>E.5.18 more Rationale.</b> <i>Replace the full rationale for <code>more</code> with the follow-</i>	B
2183	<i>ing.</i>	B
2184	<i>Editor’s Note: Only the portions changed from the 1992 standard are diff-</i>	B
2185	<i>marked.</i>	B
2186	<b>E.5.18 more — Display files on a page-by-page basis</b>	B
2187	The <code>more</code> utility, available in BSD and BSD-derived systems, was chosen as the	B
2188	prototype for the POSIX.2 file display program since it is more widely available	B
2189	than either the public-domain program <code>less</code> or than <code>pg</code> , a pager provided in	B
2190	System V. The 4.4BSD <code>more</code> is the model for the features selected; it is almost	B
2191	fully upward compatible from the 4.3BSD version in wide use and has become	B
2192	more amenable for <code>vi</code> users. Several features originally derived from various file	B
2193	editors, found in both <code>less</code> and <code>pg</code> , have been added to this specification as they	B
2194	have proved extremely popular with users.	B
2195	There are inconsistencies between <code>more</code> and <code>vi</code> that result from historical prac-	B
2196	tice. For example, the single-character commands <code>h</code> , <code>f</code> , <code>b</code> , and <code>&lt;space&gt;</code> are	B
2197	screen movers in <code>more</code> , but cursor movers in <code>vi</code> . These inconsistencies were	B
2198	maintained because the cursor movements are not applicable to <code>more</code> and the	B
2199	powerful functionality achieved without the use of the control key justifies the	B
2200	differences.	B
2201	The tags interface has been included in a program that is not a text editor	B
2202	because it promotes another degree of consistent operation with <code>vi</code> . It is conceiv-	B
2203	able that the paging environment of <code>more</code> would be superior for browsing source	B
2204	code files in some circumstances.	B
2205	The operating mode referred to for block-mode terminals effectively adds a <code>&lt;new-</code>	B
2206	<code>line&gt;</code> to each synopsis line that currently has none. So, for example,	B
2207	<code>d&lt;newline&gt;</code> would page one screenful. The mode could be triggered by a	B
2208	command-line option, environment variable, or some other method. The details	B
2209	are not imposed by POSIX.2 because there are so few systems known to support	B
2210	such terminals. Nevertheless, it was considered that all systems should be able	B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2211	to support <code>more</code> given the exception cited for this small community of terminals	B
2212	because, in comparison to <code>vi</code> , the cursor movements are few and the command set	B
2213	relatively amenable to the optional <code>&lt;newline&gt;s</code> .	B
2214	Some versions of <code>more</code> provide a shell escaping mechanism similar to the <code>ex !</code>	B
2215	command. The standard developers did not consider that this was necessary in a	B
2216	paginator, particularly given the wide acceptance of multiple window terminals	B
2217	and job control features. (They chose to retain such features in the editors and	B
2218	<code>mailx</code> because the shell interaction also gives an opportunity to modify the edit-	B
2219	ing buffer, which is not applicable to <code>more</code> ).	B
2220	The <code>-p</code> (position) option replaces the <code>+</code> command because of the Utility Syntax	B
2221	Guidelines. In early drafts, it took a <i>pattern</i> argument, but historical <code>less</code> pro-	B
2222	vided the more general facility of a command. It would have been desirable to use	B
2223	the same <code>-c</code> as <code>ex</code> and <code>vi</code> , but the letter was already in use.	B
2224	When the standard input is not a terminal, only the <code>-s</code> filter-modification option	B
2225	is effective. This is historical practice.	B
2226	The text stating “from a nonrewindable stream ... implementations may limit the	
2227	amount of backwards motion supported” would allow an implementation that per-	
2228	mitted no backwards motion beyond text already on the screen. It was not possi-	
2229	ble to require a minimum amount of backwards motion that would be effective for	
2230	all conceivable device types. The implementation should allow the user to back	
2231	up as far as possible, within device and reasonable memory allocation constraints.	
2232	<i>Examples</i>	
2233	The <code>-p</code> option allows arbitrary commands to be executed at the start of each file.	
2234	Examples are	
2235	<code>more -p G file1 file2</code>	Examine each file starting with its last
2236		screenful.
2237	<code>more -p 100 file1 file2</code>	Examine each file starting with line 100 as
2238		the first line of the screen.
2239	<code>more -p /100 file1 file2</code>	Examine each file starting with the first line
2240		containing the string 100.
2241	Historically, nonprintable characters were displayed using the ARPA standard	B
2242	mappings, which are as follows:	B
2243	(1) Printable characters are left alone.	B
2244	(2) Control characters less than <code>\177</code> are represented as <code>^</code> followed by the	B
2245	character offset from the <code>@</code> character in the ASCII map; e.g., <code>\007</code> is	B
2246	represented as <code>^G</code> .	B
2247	(3) <code>\177</code> is represented as <code>^</code> followed by <code>?</code> .	B
2248	The display of characters having their eighth bit set was less standard. Existing	B
2249	implementations use hex ( <code>0x00</code> ), octal ( <code>\000</code> ) and a meta-bit display. (The latter	B
2250	displayed characters with their eighth bit set as the two characters “M-”, followed	B
2251	by the seven bit display as described previously.) The latter probably has the best	B

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 2252 claim to historical practice because it was used with the `-v` option of 4BSD and B  
 2253 4BSD derived versions of the `cat` utility since 1980. B
- 2254 No specific display format is required by this standard. Implementations are B  
 2255 encouraged to conform to historic practice in the absence of any strong reason to B  
 2256 diverge. B
- 2257 ⇒ **E.5.35 vi Rationale.** *Replace the full rationale for `vi` with the following.* B

### 2258 **E.5.35 vi – Screen-oriented (visual) display editor**

- 2259 Major portions of the `vi` clause point to the `ex` clause to avoid inadvertent diver-  
 2260 gence. While `ex` and `vi` have historically been implemented as a single utility,  
 2261 this is not required by this standard. See the rationale for the `ex` utility (E.5.10)  
 2262 for more information on `vi`.

#### 2263 **E.5.35.1 Synopsis**

- 2264 There is no additional rationale provided for this subclause.

#### 2265 **E.5.35.2 Description**

- 2266 It is recognized that portions of `vi` would be difficult, if not impossible, to imple-  
 2267 ment satisfactorily on a block-mode terminal, or a terminal without any form of  
 2268 cursor addressing, thus it is not a mandatory requirement that such features  
 2269 should work on all terminals. It is the intention, however, that a `vi` implementa-  
 2270 tion should provide the full set of capabilities on all terminals capable of support-  
 2271 ing them.

#### 2272 **E.5.35.3 Options**

- 2273 There is no additional rationale provided for this subclause.

#### 2274 **E.5.35.4 Operands**

- 2275 There is no additional rationale provided for this subclause.

#### 2276 **E.5.35.5 External Influences**

##### 2277 **E.5.35.5.1 Standard Input**

- 2278 Historically, `vi` exited immediately if the standard input was not a terminal.  
 2279 This standard permits, but does not require, this behavior.
- 2280 An end-of-file condition is not equivalent to an end-of-file character. A common  
 2281 end-of-file character, `<control-D>`, is historically a `vi` command.

2282 **E.5.35.5.2 Input Files**

2283 There is no additional rationale provided for this subclause.

2284 **E.5.35.5.3 Environment Variables**

2285 There is no additional rationale provided for this subclause.

2286 **E.5.35.5.4 Asynchronous Events**

2287 There is no additional rationale provided for this subclause.

2288 **E.5.35.6 External Effects**2289 **E.5.35.6.1 Standard Output**

2290 The text in the standard output subclause reflects the usage of the verb “display”  
 2291 in this clause; some implementations of `vi` use standard output to write to the  
 2292 terminal, but POSIX.2 does not require that to be the case.

2293 **E.5.35.6.2 Standard Error**

2294 There is no additional rationale provided for this subclause.

2295 **E.5.35.6.3 Output Files**

2296 There is no additional rationale provided for this subclause.

2297 **E.5.35.7 Extended Description**

2298 Historically, implementations reverted to open mode if the terminal was incapa-  
 2299 ble of supporting full visual mode. This standard requires this behavior. Histori-  
 2300 cally, the open mode of `vi` behaved roughly equivalently to the visual mode, with  
 2301 the exception that only a single physical line from the edit buffer was kept current  
 2302 at any time. This line was normally displayed on the next to last line of a termi-  
 2303 nal with cursor addressing (and the last line performed its normal visual func-  
 2304 tions for line-oriented commands and messages). In addition, some few com-  
 2305 mands behaved differently in open mode than in visual mode. This standard  
 2306 requires conformance to historical practice. The following list is a condensed ver-  
 2307 sion of the information contained in the normative text. It is entered here so that  
 2308 the basic information about open mode is available in a single place.

2309 `[count]z`, `[count]control-F`, `[count]<control-B>`

2310 The `z` command has a different synopsis in open mode than in visual mode.  
 2311 The `z`, `<control-F>`, and `<control-B>` commands all behave identically,  
 2312 displaying zero or more lines before and after the current line, with the  
 2313 current line surrounded by hyphens.

2314 `<control-D>`2315 Write the next `scroll` edit option value lines, update the current line.



2316 <control-U>  
 2317 Update the current line, do nothing else.

2318 <control-E>, <control-Y>  
 2319 Do nothing.

2320 <control-L>  
 2321 Clear the screen and redisplay the current line.

2322 H, L, M  
 2323 Move to the first nonblank of the current line and do nothing else.

2324 Historically, `ex` and `vi` implementations have expected text to proceed from left to  
 2325 right and from top to bottom. There is no requirement in this standard that this  
 2326 be the case. The specification was deliberately written using words like “before,”  
 2327 “after,” “first,” and “last” in order to permit implementations to support the  
 2328 natural text order of the language.

2329 Historically, lines past the end of the edit buffer were marked with single tilde (~)  
 2330 characters; i.e., if the one-based display was 20 lines in length, and the last line of  
 2331 the file was on line one, then lines 2–20 would contain only a single ~ character.

2332 Historically, the `vi` editor attempted to display only complete lines at the bottom  
 2333 of the screen (it did display partial lines at the top of the screen). If a line was too  
 2334 long to fit in its entirety at the bottom of the screen, the screen lines where the  
 2335 line would have been displayed were displayed as single @ characters, instead of  
 2336 displaying part of the line. This standard permits, but does not require, this  
 2337 behavior. Implementations are encouraged to attempt always to display a complete  
 2338 line at the bottom of the screen when doing scrolling or screen positioning by  
 2339 physical lines.

2340 Historically, lines marked with @ were also used to minimize output to dumb ter-  
 2341 minals over slow lines; i.e., changes local to the cursor were updated, but changes  
 2342 to lines on the screen that were not close to the cursor were simply marked with  
 2343 an @ sign instead of being updated to match the current text. This standard per-  
 2344 mits, but does not require this feature because it is used ever less frequently as  
 2345 terminals become smarter and connections are faster.

#### 2346 **E.5.35.7.1 `ex` and `vi` Initialization**

2347 Historically, `vi` always had a line in the edit buffer, even if the edit buffer was  
 2348 “empty.” For example:

- 2349 (1) The `ex` command = executed from visual mode wrote “1” when the buffer  
 2350 was empty.
- 2351 (2) Writes from visual mode of an empty edit buffer wrote files of a single  
 2352 character (a <newline>), while writes from `ex` mode of an empty edit  
 2353 buffer wrote empty files.
- 2354 (3) Put and read commands into an empty edit buffer left an empty line at  
 2355 the top of the edit buffer.

- 2356 For consistency, this standard does not permit any of these behaviors.
- 2357 Historically, `vi` did not always return the terminal to its original modes; for C  
 2358 example, `ICRNL` was modified if it was not originally set. This standard does not C  
 2359 permit this behavior. C
- 2360 **E.5.35.7.2 `vi` Command Descriptions**
- 2361 Motion commands are among the most complicated aspects of `vi` to describe.  
 2362 With some exceptions, the text region and buffer type effect of a motion command  
 2363 on a `vi` command are described on a case-by-case basis. The descriptions of text  
 2364 regions in this standard are not intended to imply direction; i.e., an inclusive  
 2365 region from line  $n$  to line  $n + 5$  is identical to a region from line  $n + 5$  to line  $n$ .  
 2366 This is of more than academic interest—movements to marks can be in either  
 2367 direction, and, if the `wrapscreen` option is set, so can movements to search points.  
 2368 Historically, lines are always stored into buffers in text order; i.e., from the start  
 2369 of the edit buffer to the end. This standard requires conformance to historical  
 2370 practice.
- 2371 Historically, command counts were applied to any associated motion, and were  
 2372 multiplicative to any supplied motion count. For example, `2cw` is the same as  
 2373 `c2w`, and `2c3w` is the same as `c6w`. This standard requires this behavior.
- 2374 Historically, `vi` commands that used `bigwords`, `words`, `paragraphs`, and `sentences`  
 2375 as objects treated groups of empty lines, or lines that contained only `<blank>`  
 2376 characters, inconsistently. Some commands treated them as a single entity, while  
 2377 others treated each line separately. For example, the `w`, `W`, and `B` commands  
 2378 treated groups of empty lines as individual words; i.e., the command would move  
 2379 the cursor to each new empty line. The `e` and `E` commands treated groups of  
 2380 empty lines as a single word; i.e., the first use would move past the group of lines.  
 2381 The `b` command would just beep at the user, or if done from the start of the line  
 2382 as a motion command, fail in unexpected ways. If the lines contained only (or  
 2383 ended with) `<blank>` characters, the `w` and `W` commands would just beep at the  
 2384 user, the `E` and `e` commands would treat the group as a single word, and the `B`  
 2385 and `b` commands would treat the lines as individual words. For consistency and  
 2386 simplicity of specification, this standard requires that all `vi` commands treat  
 2387 groups of empty or `<blank>`-filled lines as a single entity, and that movement C  
 2388 through lines ending with `<blank>` characters be consistent with other move- C  
 2389 ments. C
- 2390 Historically, `vi` documentation indicated that any number of double quotes were  
 2391 skipped after punctuation marks at sentence boundaries, however, implementa-  
 2392 tions only skipped single quotes. This standard requires both to be skipped.
- 2393 Historically, the first and last characters in the edit buffer were word boundaries.  
 2394 This historical practice is required by this standard.
- 2395 Historically, `vi` attempted to update the minimum number of columns on the  
 2396 screen possible, which could lead to misleading information being displayed. This  
 2397 standard makes no requirements other than that the current character being  
 2398 entered is displayed correctly, leaving all other decisions in this area up to the  
 2399 implementations.

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2400 Historically, lines were arbitrarily folded between columns of any characters that  
 2401 required multiple column positions on the screen, with the exception of tabs,  
 2402 which terminated at the right-hand margin. This standard permits the former  
 2403 and requires the latter. Implementations that do not arbitrarily break lines  
 2404 between columns of characters that occupy multiple column positions should not  
 2405 permit the cursor to rest on a column that does not contain any part of a  
 2406 character.

2407 The historical `vi` had a problem in that all movements were by physical lines, not  
 2408 by logical, or screen, lines. This is often the right thing to do; e.g., single line  
 2409 movements, such as `j` or `k`, should work on physical lines. Commands like `dj`, or  
 2410 `j.`, where `.` is a change command, only make sense for physical lines. It is not,  
 2411 however, the right thing to do for screen motion or scrolling commands like  
 2412 `<control-D>`, `<control-F>`, and `H`. If the window is fairly small, using physical  
 2413 lines in these cases can result in completely random motion; e.g., `lcontrol-D`  
 2414 can result in a completely changed screen, without any overlap. This is clearly  
 2415 not what the user wanted. The problem is even worse in the case of the `H`, `L`, and  
 2416 `M` commands—as they position the cursor at the first nonblank of the line, they  
 2417 may all refer to the same location in large lines, and will result in no movement at  
 2418 all.

2419 In addition, if the line is larger than the screen, using physical lines can make it  
 2420 impossible to display parts of the line—there are not any commands that do not  
 2421 display the beginning of the line in historical `vi`, and if both the beginning and  
 2422 end of the line cannot be on the screen at the same time, the user suffers. Finally,  
 2423 the page and half-page scrolling commands historically moved to the first non-  
 2424 `<blank>` character in the new line. If the line is approximately the same size as  
 2425 the screen, this is inadequate because the cursor before and after a `<control-D>`  
 2426 command will refer to the same location on the screen.

2427 Implementations of `ex` and `vi` exist that do not have these problems because the  
 2428 relevant commands (`<control-B>`, `<control-D>`, `<control-F>`, `<control-U>`,  
 2429 `<control-Y>`, `<control-E>`, `H`, `L`, and `M`) commands operate on logical screen  
 2430 lines, not physical edit buffer lines.

2431 This standard does not permit this behavior by default because the standard  
 2432 developers believed that users would find it too confusing. However, historical  
 2433 practice has been relaxed. For example, `ex` and `vi` historically attempted, albeit  
 2434 sometimes unsuccessfully, to never put part of a line on the last lines of a screen;  
 2435 e.g., if a line would not fit in its entirety, no part of the line was displayed, and  
 2436 the screen lines corresponding to the line contained single `@` characters. This C  
 2437 behavior is permitted, but not required by this standard, so that it is possible for  
 2438 implementations to support long lines in small screens more reasonably without  
 2439 changing the commands to be logically (instead of physically) oriented. This stan- C  
 2440 dard also permits implementations to refuse to edit any edit buffer containing a  
 2441 line that will not fit on the screen in its entirety.

2442 The display area (e.g., the value of the `window` edit option) has historically been  
 2443 “grown,” or expanded, to display new text when local movements are done in  
 2444 displays where the number of lines displayed is less than the maximum possible.  
 2445 Expansion has historically been the first choice, when the target line is less than

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2446 the maximum possible expansion value away. Scrolling has historically been the  
 2447 next choice, done when the target line is less than half a display away, and other-  
 2448 wise, the screen was redrawn. There were exceptions, however, in that `ex` com-  
 2449 mands generally always caused the screen to be redrawn. This standard does not  
 2450 specify a standard behavior because there may be external issues such as connec-  
 2451 tion speed, the number of characters necessary to redraw as opposed to scroll, or  
 2452 terminal capabilities that implementations will have to accommodate.

2453 The current line in this standard maps one-to-one to a physical line in the file.  
 2454 The current column does not. There are two different column values that are  
 2455 described by this standard. The first is the current column value as set by many  
 2456 of the `vi` commands. This value is remembered for the lifetime of the editor. The  
 2457 second column value is the actual position on the screen where the cursor rests.  
 2458 The two are not always the same. For example, when the cursor is backed by a  
 2459 multicolumn character, the actual cursor position on the screen has historically  
 2460 been the last column of the character in command mode, and the first column of  
 2461 the character in input mode.

2462 Commands that set the current line, but that do not set the current cursor value,  
 2463 (e.g., `j` and `k`) attempt to get as close as possible to the remembered column posi-  
 2464 tion, so that the cursor tends to restrict itself to a vertical column as the user  
 2465 moves around in the edit buffer. This standard requires conformance to historical  
 2466 practice, requiring that the physical location of the cursor on the screen be  
 2467 adjusted from the current column value as necessary to support this historical  
 2468 behavior.

2469 Historically, only a single line (and for some terminals, a single line minus 1 C  
 2470 column) of characters could be entered by the user for the line oriented com- C  
 2471 mands; i.e., `:`, `!`, `/`, or `?`. This standard permits, but does not require, this limita- C  
 2472 tion. C

2473 Historically, “soft” errors in `vi` caused the terminal to be alerted, but no error C  
 2474 message was displayed. As a general rule, no error message was displayed for C  
 2475 errors in command execution in `vi`, when the error resulted from the user C  
 2476 attempting an invalid or impossible action, or when a searched-for object was not C  
 2477 found. Examples of soft errors included `h` at the left margin, `<control-B>` or `[[` C  
 2478 at the beginning of the file, `2G` at the end of the file, etc. In addition, errors such C  
 2479 as `%`, `]]`, `}`, `)`, `N`, `n`, `f`, `F`, `t`, and `T` failing to find the searched-for object were soft as C  
 2480 well. Less consistently, `/` and `?` displayed an error message if the pattern was not C  
 2481 found, `/`, `?`, `N`, and `n` displayed an error message if no previous RE had been C  
 2482 specified, and `;` did not display an error message if no previous `f`, `F`, `t`, or `T` com- C  
 2483 mand had occurred. Also, behavior in this area might reasonably be based on a C  
 2484 run-time evaluation of the speed of a network connection. Finally, some imple- C  
 2485 mentations have provided error messages for soft errors in order to assist naive C  
 2486 users, based on the value of a `verbose` edit option. This standard does not list C  
 2487 specific errors for which an error message shall be displayed. Implementations C  
 2488 should conform to historical practice in the absence of any strong reason to C  
 2489 diverge. C

2490 The following table is a condensed version of information contained in the norma-  
 2491 tive text. It is presented here to facilitate the review of the editor options that

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

2492 affect, or are affected by, vi commands.

2493	<u>vi Command</u>	<u>Editor Options</u>
2494	!	autowrite, shell, warn, writeany
2495	(, ), {, }	paragraphs, sections
2496	/, ?, N, n	ignorecase, magic, wrapscan
2497	<, >	shiftwidth, tabstop
2498	A, a, I, i, O, o	autoindent, showmatch, wrapmargin
2499	C, c, R, r, S, s	autoindent, showmatch, wrapmargin
2500	ZZ	readonly, writeany
2501	[[, ]]	sections
2502	<control-B>	window
2503	<control-D>	scroll
2504	<control-F>	window
2505	<control-T>	shiftwidth, tabstop
2506	<control-U>	scroll
2507	<control-]>	autowrite, tag, taglength, writeany
2508	z	window

2509 The = (reindent) command was omitted because it was LISP language-specific,  
 2510 and LISP language support was omitted from this standard. (See E.5.10 for more  
 2511 information).

#### 2512 **E.5.35.7.2.1 <control-B>**

2513 The <control-B> and <control-F> commands historically considered it an  
 2514 error to attempt to page past the beginning or end of the file, whereas the  
 2515 <control-D> and <control-U> commands simply moved to the beginning or  
 2516 end of the file. For consistency, this standard requires the latter behavior for all  
 2517 four commands. All four commands still consider it an error if the current line is  
 2518 at the beginning (<control-B>, <control-U>) or end (<control-F>,  
 2519 <control-D>) of the file. Historically, the <control-B> and <control-F>  
 2520 commands skip two lines in order to include overlapping lines when a single com-  
 2521 mand is entered. This makes less sense in the presence of a *count*, as there will  
 2522 be, by definition, no overlapping lines. The actual calculation used by historical  
 2523 implementations of the vi editor for <control-B> was:

2524  $((\text{current first line}) - \text{count} \times (\text{window edit option})) + 2$

2525 and for <control-F> was:

2526  $((\text{current first line}) + \text{count} \times (\text{window edit option})) - 2$

2527 This calculation does not work well when intermixing commands with and  
 2528 without counts; e.g., 3<control-F> is not equivalent to entering the <control-F>  
 2529 command three times, and is not reversible by entering the <control-B> com-  
 2530 mand three times. For consistency with other vi commands that take counts,  
 2531 this standard requires a different calculation.

2532 **E.5.35.7.2.2** <control-D>

2533 See 5.35.7.2.1. The 4BSD and System V implementations of `vi` differed on the ini-  
2534 tial value used by the `scroll` command. 4BSD used:

2535  $((\text{window edit option}) + 1) / 2$

2536 while System V used the value of the `scroll edit option`. The System V version  
2537 is specified by this standard because the standard developers believed that it was  
2538 more intuitive and permitted the user a method of setting the scroll value initially  
2539 without also setting the number of lines that are displayed.

2540 **E.5.35.7.2.3** <control-E>

2541 See E.5.35.7.2.1. Historically, the <control-E> and <control-Y> commands  
2542 considered it an error if the last and first lines, respectively, were already on the  
2543 screen. This standard requires conformance to historical practice.

2544 Historically, the <control-E> and <control-Y> commands had no effect in C  
2545 open mode. For simplicity and consistency of specification, this standard requires C  
2546 that they behave as usual, albeit with a single line screen. C

2547 **E.5.35.7.2.4** <control-F>

2548 See E.5.35.7.2.1.

2549 **E.5.35.7.2.5** <control-G>

2550 There is no additional rationale provided for this subclause.

2551 **E.5.35.7.2.6** <control-H>

2552 There is no additional rationale provided for this subclause.

2553 **E.5.35.7.2.7** <newline>

2554 There is no additional rationale provided for this subclause.

2555 **E.5.35.7.2.8** <control-L>

2556 The historical <control-L> command refreshed the screen exactly as it was sup-  
2557 posed to be currently displayed, replacing any @ characters for lines that had been  
2558 deleted but not updated on the screen (see 5.35.7.2) with refreshed @ characters.  
2559 The intent of the <control-L> command is to refresh when the screen has been  
2560 accidentally overwritten; e.g., by a write by another user, or modem noise.

2561 **E.5.35.7.2.9** <control-P>

2562 There is no additional rationale provided for this subclause.

2563 **E.5.35.7.2.10 <control-R>**

2564 The historical <control-R> command redisplayed only when necessary to  
 2565 update lines that had been deleted but not updated on the screen and that were  
 2566 flagged with @ characters (see 5.35.7). There is no requirement that the screen be  
 2567 in any way refreshed if no lines of this form are currently displayed. This stan-  
 2568 dard permits implementations to extend this command to refresh lines on the  
 2569 screen flagged with @ characters because they are too long to be displayed in the  
 2570 current framework; however, the current line and column need not be modified.

2571 **E.5.35.7.2.11 <control-U>**

2572 See E.5.35.7.2.1 and E.5.35.7.3.2.

2573 **E.5.35.7.2.12 <control-Y>**

2574 See E.5.35.7.2.1 and E.5.35.7.2.3.

2575 **E.5.35.7.2.13 <control-^>**

2576 There is no additional rationale provided for this subclause.

2577 **E.5.35.7.2.14 <ESC>**

2578 Historically, an escape character, optionally preceded by a count, in command  
 2579 mode alerted the terminal, but an escape character preceded by part of a com-  
 2580 mand did not. For example, 33c<ESC> is a partial command and is silently can-  
 2581 celled, but 33<ESC> must alert the terminal.

2582 Historically, half entered [[, ]], or zz commands were not cancelled by <ESC>;  
 2583 the terminal was alerted instead. For consistency and simplicity of specification,  
 2584 the standard does not permit this exception.

2585 Historically, a leading <ESC> in a vi command was not an error when it resulted C  
 2586 from a map expansion, and historical macros are known to depend on this feature. C  
 2587 This standard requires this behavior. C

2588 **E.5.35.7.2.15 <control- ]>**

2589 Historically, the first non-<blank> character at or after the cursor was the first  
 2590 character, and all subsequent characters that were word characters, up to the end  
 2591 of the line, were included. For example, with the cursor on the leading space or  
 2592 on the # character in the text “ #bar@”, the tag was #bar. On the character b it  
 2593 was bar, and on the a, it was ar. This standard requires this behavior.

2594 **E.5.35.7.2.16 <space>**

2595 There is no additional rationale provided for this subclause.

**2596 E.5.35.7.2.17 !**

2597 Historically, the <, >, and ! commands considered most cursor motions other than  
2598 line oriented motions an error; for example, the command >/foo<CR> succeeded,  
2599 while the command >1 failed, even though the text region described by the two  
2600 commands might be identical. For consistency, all three commands only consider  
2601 entire lines and not partial lines, and the region is defined as any line that con-  
2602 tains a character that was specified by the motion.

**2603 E.5.35.7.2.18 \$**

2604 There is no additional rationale provided for this subclause.

**2605 E.5.35.7.2.19 %**

2606 Other matching characters have been left implementation-defined in order to per-  
2607 mit implementations to support the historical LISP options, and to allow exten-  
2608 sions such as matching < and > for searching HTML, or #ifdef, #else, and  
2609 #endif for searching C source.

**2610 E.5.35.7.2.20 &**

2611 This standard requires that any c and g flags specified to the previous substitute  
2612 command be ignored; however, the r flag may still apply, if supported by the  
2613 implementation.

**2614 E.5.35.7.2.21 '**

2615 There is no additional rationale provided for this subclause.

**2616 E.5.35.7.2.22 `**

2617 There is no additional rationale provided for this subclause.

**2618 E.5.35.7.2.23 [[**

2619 The [[, ]], (, ), {, and } commands are all affected by “section boundaries,” but  
2620 in some historical implementations not all of the commands recognize the same  
2621 section boundaries. This is a bug, not a feature, and a unique section-boundary  
2622 algorithm was not described for each command. One special case that is  
2623 preserved is that the sentence command moves to the end of the last line of the  
2624 edit buffer while the other commands go to the beginning, in order to preserve the  
2625 traditional character cut semantics of the sentence command. Historically, vi  
2626 section boundaries at the beginning and end of the edit buffer were the first non-  
2627 blank character on the first and last lines of the edit buffer if one exists; other-  
2628 wise, the last character of the first and last lines of the edit buffer if one exists;  
2629 otherwise, the first and last lines of the edit buffer. To increase consistency with  
2630 other section locations, this has been simplified by this standard to the first char-  
2631 acter of the first and last lines of the edit buffer, or the first and the last lines of  
2632 the edit buffer if they are empty.



2633 Sentence boundaries were problematic in the historical `vi`. They were not only  
2634 the boundaries as defined for the section and paragraph commands, but they were  
2635 the first nonblank character that occurred after those boundaries, as well.

2636 Historically, the `vi` section commands were documented as taking an optional  
2637 window size as a count preceding the command. This was not implemented in  
2638 historical versions, so this standard requires that the count repeat the command,  
2639 for consistency with other `vi` commands.

2640 **E.5.35.7.2.24** ] ]

2641 See E.5.35.7.2.23.

2642 **E.5.35.7.2.25** ^

2643 There is no additional rationale provided for this subclause.

2644 **E.5.35.7.2.26** \_

2645 There is no additional rationale provided for this subclause.

2646 **E.5.35.7.2.27** (

2647 See E.5.35.7.2.23.

2648 **E.5.35.7.2.28** )

2649 See E.5.35.7.2.23.

2650 **E.5.35.7.2.29** {

2651 See E.5.35.7.2.23.

2652 **E.5.35.7.2.30** }

2653 See E.5.35.7.2.23.

2654 **E.5.35.7.2.31** |

2655 There is no additional rationale provided for this subclause.

2656 **E.5.35.7.2.32** ,

2657 There is no additional rationale provided for this subclause.

2658 **E.5.35.7.2.33** .

2659 Historically, mapped commands other than text input commands could not be  
2660 repeated using the period command. This standard requires conformance to his-  
2661 torical practice.

2662 The restrictions on the interpretation of special characters (e.g., `<control-H>`) in  
2663 the repetition of text input mode commands is intended to match historical

2664 practice. For example, given the input sequence

```
2665 iab<control-H><control-H><control-H>def<escape>
```

2666 the user should be informed of an error when the sequence is first entered, but  
2667 not during a command repetition. The character <control-T> is specifically  
2668 exempted from this restriction. Historical implementations of vi ignored  
2669 <control-T> characters that were input in the original command during com-  
2670 mand repetition. This standard prohibits this behavior.

2671 **E.5.35.7.2.34 /**

2672 Historically, commands did not affect the line searched to or from if the motion  
2673 command was a search (/, ?, N, n) and the final position was the start/end of the  
2674 line. There were some special cases and vi was not consistent. This standard  
2675 does not permit this behavior, for consistency. Historical implementations per-  
2676 mitted, but were unable to handle searches as motion commands that wrapped  
2677 (i.e., due to the edit option wrapscan) to the original location. This standard  
2678 requires that this behavior be treated as an error.

2679 Historically, the syntax /RE/O was used to force the command to cut text in line  
2680 mode. This standard requires conformance to historical practice.

2681

C

2682 Historically, in open mode, a z specified to a search command redisplayed the  
2683 current line instead of displaying the current screen with the current line  
2684 highlighted. For consistency and simplicity of specification, this standard does  
2685 not permit this behavior.

2686 Historically, trailing z commands were permitted and ignored if entered as part of  
2687 a search used as a motion command. For consistency and simplicity of  
2688 specification, this standard does not permit this behavior.

C

C

C

2689 **E.5.35.7.2.35 0**

2690 There is no additional rationale provided for this subclause.

2691 **E.5.35.7.2.36 :**

2692 Historically, vi implementations restricted the commands that could be entered  
2693 on the colon command line (e.g., append and change), and some other commands  
2694 were known to cause them to fail catastrophically. For consistency, this standard  
2695 does not permit these restrictions. When executing an ex command by entering  
2696 :, it is not possible to enter a <newline> as part of the command because it is  
2697 considered the end of the command. A different approach is to enter ex command  
2698 mode by using the vi Q command (and later resuming visual mode with the ex vi  
2699 command). In ex command mode, the single-line limitation does not exist. So, for  
2700 example, the following is valid:

```
2701 Q %s/break here/break\  
2702 here/ vi
```

C

C

2703 This standard requires that, if the `ex` command overwrites any part of the screen  
 2704 that would be erased by a refresh, `vi` pause for a character from the user. Historically,  
 2705 this character could be any character; e.g., a character input by the user  
 2706 before the message appeared, or even a mapped character. This is probably a  
 2707 bug, but implementations that have tried to be more rigorous by requiring that  
 2708 the user enter a specific character, or that the user enter a character after the  
 2709 message was displayed, have been forced by user indignation back into historical  
 2710 behavior. This standard requires conformance to historical practice.

2711 **E.5.35.7.2.37 ;**

2712 There is no additional rationale provided for this subclause.

2713 **E.5.35.7.2.38 <**

2714 See E.5.35.7.2.17 and E.5.35.7.3.4. Historically, the `<` and `>` commands some-  
 2715 times moved the cursor to the first nonblank (e.g., if the command was repeated  
 2716 or with `_` as the motion command), and sometimes left it unchanged. This stan-  
 2717 dard does not permit this inconsistency, requiring instead that the cursor always  
 2718 move to the first nonblank.

2719 Historically, the `<` and `>` commands did not support buffer arguments, although C  
 2720 some implementations allow the specification of an optional buffer. This behavior C  
 2721 is neither required nor disallowed by this standard. C

2722 **E.5.35.7.2.39 >**

2723 See E.5.35.7.2.17, E.5.35.7.2.38, and E.5.35.7.3.4.

2724 **E.5.35.7.2.40 ?**

2725 See E.5.35.7.2.34.

2726 **E.5.35.7.2.41 @**

2727 Historically, buffers could execute other buffers, and loops, infinite and otherwise,  
 2728 were possible. This standard requires conformance to historical practice. The  
 2729 *\*buffer* syntax of `ex` is not required in `vi`, because it is not historical practice and  
 2730 has been used in some `vi` implementations to support additional scripting  
 2731 languages.

2732 Historically, `vi` only supported the `@@` syntax for repeating the last buffer execu-  
 2733 tion. This standard requires that `vi` support the additional `ex` syntax `@*` as well,  
 2734 for consistency.

2735 **E.5.35.7.2.42 ~**

2736 Historically, the `~` command ignored any associated *count*, and acted only on the  
 2737 characters in the current line. For consistency with other `vi` commands, this  
 2738 standard requires that an associated *count* act on the next *count* characters, and  
 2739 that the command move to subsequent lines if warranted by *count*, to make it

2740 possible to modify large pieces of text in a reasonably efficient manner. There  
 2741 exist `vi` implementations that optionally require an associated motion command  
 2742 for the `~` command. Implementations supporting this functionality are  
 2743 encouraged to base it on the `tildedop` edit option and handle the text regions  
 2744 and cursor positioning identically to the `yank` command.

2745 **E.5.35.7.2.43 a**

2746 Historically, *counts* specified to the `A`, `a`, `I`, and `i` commands repeated the input of  
 2747 the first line *count* times, and did not repeat the subsequent lines of the input  
 2748 text. This standard requires that the entire text input be repeated *count* times.

2749 **E.5.35.7.2.44 A**

2750 See E.5.35.7.2.43.

2751 **E.5.35.7.2.45 b**

2752 Historically, `vi` became confused if word commands were used as motion com-  
 2753 mands in empty files. This standard requires that this be an error. Historical  
 2754 implementations of `vi` had a large number of bugs in the word movement com-  
 2755 mands, and they varied greatly in behavior in the presence of empty lines,  
 2756 “words” made up of a single character, and lines containing only `<blank>` charac-  
 2757 ters. For consistency and simplicity of specification, this standard does not permit  
 2758 this behavior.

2759 **E.5.35.7.2.46 B**

2760 See E.5.35.7.2.45.

2761 **E.5.35.7.2.47 c**

2762 There is no additional rationale provided for this subclause.

2763 **E.5.35.7.2.48 c**

2764 Some historical implementations of the `C` command did not behave as described  
 2765 by this standard when the `$` key was remapped because they were implemented  
 2766 by pushing the `$` key onto the input queue and reprocessing it. This standard  
 2767 does not permit this behavior.

2768 Historically, the `C`, `S`, and `s` commands did not copy replaced text into the numeric  
 2769 buffers. For consistency and simplicity of specification, this standard requires  
 2770 that they behave like their respective `c` commands in all respects.

2771 **E.5.35.7.2.49 d**

2772 Historically, lines in open mode that were deleted were scrolled up, and an @ C  
 2773 glyph written over the beginning of the line. In the case of terminals that are C  
 2774 incapable of the necessary cursor motions, the editor erased the deleted line from C  
 2775 the screen. This standard requires conformance to historical practice; i.e., if the C

- 2776 terminal cannot display the @ character, the line cannot remain on the screen. C
- 2777 **E.5.35.7.2.50 D**
- 2778 Some historical implementations of the D command did not behave as described  
2779 by this standard when the \$ key was remapped because they were implemented  
2780 by pushing the \$ key onto the input queue and reprocessing it. This standard  
2781 does not permit this behavior.
- 2782 **E.5.35.7.2.51 e**
- 2783 See E.5.35.7.2.45.
- 2784 **E.5.35.7.2.52 E**
- 2785 See E.5.35.7.2.45.
- 2786 **E.5.35.7.2.53 f**
- 2787 There is no additional rationale provided for this subclause.
- 2788 **E.5.35.7.2.54 F**
- 2789 There is no additional rationale provided for this subclause.
- 2790 **E.5.35.7.2.55 G**
- 2791 There is no additional rationale provided for this subclause.
- 2792 **E.5.35.7.2.56 H**
- 2793 There is no additional rationale provided for this subclause.
- 2794 **E.5.35.7.2.57 i**
- 2795 See E.5.35.7.2.43.
- 2796 **E.5.35.7.2.58 I**
- 2797 See E.5.35.7.2.43.
- 2798 **E.5.35.7.2.59 J**
- 2799 An historical oddity of vi is that the commands J, 1J, and 2J are all equivalent.  
2800 This standard requires conformance to historical practice.
- 2801 The vi J command is specified in terms of the ex join command with an ex com-  
2802 mand *count* value. The address correction for a count that is past the end of the  
2803 edit buffer is necessary for historical compatibility for both ex and vi.

2804 **E.5.35.7.2.60** L

2805 There is no additional rationale provided for this subclause.

2806 **E.5.35.7.2.61** m

2807 Historical practice is that only lower-case letters, plus ` and ', could be used to  
 2808 mark a cursor position. This standard requires conformance to historical practice,  
 2809 but encourages implementations to support other characters as marks as well.

2810 **E.5.35.7.2.62** M

2811 There is no additional rationale provided for this subclause.

2812 **E.5.35.7.2.63** n

2813 Historically, the N and n commands could not be used as motion components for C  
 2814 the c command. With the exception of the "cN" command, which worked if the C  
 2815 search crossed a line boundary, the text region would be discarded, and the user C  
 2816 would not be in text input mode. For consistency and simplicity of specification, C  
 2817 this standard does not permit this behavior. C

2818 **E.5.35.7.2.64** N

2819 See E.5.35.7.2.63. C

2820 **E.5.35.7.2.65** o

2821 Historically, *counts* to the O and o commands were used as the number of physical  
 2822 lines to open, if the terminal was dumb and the `slowopen` option was not set.  
 2823 This was intended to minimize traffic over slow connections and repainting for  
 2824 dumb terminals. This standard does not permit this behavior, requiring that a  
 2825 *count* to the `open` command behave as for other text input commands. This  
 2826 change to historical practice was made for consistency, and because a superset of  
 2827 the functionality is provided by the `slowopen` edit option.

2828 **E.5.35.7.2.66** o

2829 See E.5.35.7.2.65.

2830 **E.5.35.7.2.67** p

2831 Historically, *counts* to the p and P commands were ignored if the buffer was a line  
 2832 mode buffer, but were (mostly) implemented as described in this standard if the  
 2833 buffer was a character mode buffer. Because implementations exist that do not  
 2834 have this limitation, and because pasting lines multiple times is generally useful,  
 2835 this standard requires that *count* be supported for all p and P commands.

2836 Historical implementations of vi were widely known to have major problems in  
 2837 the p and P commands, particularly when unusual regions of text were copied into  
 2838 the edit buffer. The standard developers viewed these as bugs, and they are not  
 2839 permitted for consistency and simplicity of specification.

2840 Historically, a `P` or `p` command (or an `ex put` command executed from open or C  
 2841 visual mode) executed in an empty file, left an empty line as the first line of the C  
 2842 file. For consistency and simplicity of specification, this standard does not permit C  
 2843 this behavior. C

2844 **E.5.35.7.2.68 P**

2845 See E.5.35.7.2.67.

2846 **E.5.35.7.2.69 Q**

2847 There is no additional rationale provided for this subclause.

2848 **E.5.35.7.2.70 r**

2849 Historically, the `r` command did not correctly handle the *erase* and *word erase*  
 2850 characters as arguments, nor did it handle an associated *count* greater than 1  
 2851 with a `<carriage-return>` argument, for which it replaced *count* characters  
 2852 with a single `<newline>`. This standard does not permit these inconsistencies.

2853 Historically, the `r` command permitted the `<control-V>` escaping of entered  
 2854 characters, such as `<ESC>` and `<carriage-return>`; however, it required two  
 2855 leading `<control-V>` characters instead of one. This standard requires that this  
 2856 be changed for consistency with the other text input commands of `vi`.

2857 Historically, it is an error to enter the `r` command if there are less than *count*  
 2858 characters at or after the cursor in the line. While a reasonable and unambiguous  
 2859 extension would be to permit the `r` command on empty lines, it would require that  
 2860 too large a *count* be adjusted to match the number of characters at or after the  
 2861 cursor for consistency, which is sufficiently different from historical practice to be  
 2862 avoided. This standard requires conformance to historical practice.

2863 **E.5.35.7.2.71 R**

2864

C

2865 Historically, if there were autoindent characters in the line on which the `R` com-  
 2866 mand was run, and `autoindent` was set, the first `<newline>` character would be  
 2867 properly indented and no characters would be replaced by the `<newline>` charac-  
 2868 ter. Each additional `<newline>` character, would replace *n* characters, where *n*  
 2869 was the number of characters that were needed to indent the rest of the line to  
 2870 the proper indentation level. This behavior is a bug and is not permitted by this  
 2871 standard.

2872 **E.5.35.7.2.72 s**

2873 See E.5.35.7.2.48.

2874 **E.5.35.7.2.73 s**

2875 See E.5.35.7.2.48.

2876 **E.5.35.7.2.74 t**

2877 There is no additional rationale provided for this subclause.

2878 **E.5.35.7.2.75 T**

2879 There is no additional rationale provided for this subclause.

2880 **E.5.35.7.2.76 u**

2881 Historical practice for cursor positioning after undoing commands was mixed. In  
2882 most cases, when undoing commands that affected a single line, the cursor was  
2883 moved to the start of added or changed text, or immediately after deleted text.  
2884 However, if the user had moved from the line being changed, the column was  
2885 either set to the first nonblank, returned to the origin of the command, or  
2886 remained unchanged. When undoing commands that affected multiple lines or  
2887 entire lines, the cursor was moved to the first character in the first line restored.  
2888 As an example of how inconsistent this was, a search, followed by an `o` text input  
2889 command, followed by an `undo` would return the cursor to the location where the  
2890 `o` command was entered, but a `cw` command followed by an `o` command followed  
2891 by an `undo` would return the cursor to the first nonblank of the line. This stan-  
2892 dard requires the most useful of these behaviors, and discards the least useful, in  
2893 the interest of consistency and simplicity of specification.

2894 **E.5.35.7.2.77 U**

2895 There is no additional rationale provided for this subclause.

2896 **E.5.35.7.2.78 w**

2897 See E.5.35.7.2.45.

2898 **E.5.35.7.2.79 W**

2899 See E.5.35.7.2.45.

2900 **E.5.35.7.2.80 x**

2901 There is no additional rationale provided for this subclause.

2902 **E.5.35.7.2.81 X**

2903 There is no additional rationale provided for this subclause.



**2904 E.5.35.7.2.82 y**

2905 Historically, the `yank` command did not move to the end of the motion if the  
2906 motion was in the forward direction. It moved to the end of the motion if the  
2907 motion was in the backward direction, except for the `_` command, or for the `G` and  
2908 `'` commands when the end of the motion was on the current line. This was  
2909 further complicated by the fact that for a number of motion commands, the `yank`  
2910 command moved the cursor but did not update the screen; e.g., a subsequent com-  
2911 mand would move the cursor from the end of the motion, even though the cursor  
2912 on the screen had not reflected the cursor movement for the `yank` command. This  
2913 standard requires that all `yank` commands associated with backward motions  
2914 move the cursor to the end of the motion for consistency, and specifically, to make  
2915 `'` commands as motions consistent with search patterns as motions.

**2916 E.5.35.7.2.83 Y**

2917 Some historical implementations of the `Y` command did not behave as described  
2918 by this standard when the `_` key was remapped because they were implemented  
2919 by pushing the `_` key onto the input queue and reprocessing it. This standard  
2920 does not permit this behavior.

**2921 E.5.35.7.2.84 z**

2922 Historically, the `z` command always redrew the screen. This is permitted but not  
2923 required by this standard, because of the frequent use of the `z` command in mac-  
2924 ros such as “`map n nz.`” for screen positioning, instead of its use to change the  
2925 screen size. The standard developers believed that expanding or scrolling the  
2926 screen offered a better interface for users. The ability to redraw the screen is  
2927 preserved if the optional new window size is specified, and in the `<control-L>`  
2928 and `<control-R>` commands.

2929 The semantics of `z^` are confusing at best. Historical practice is that the screen  
2930 before the screen that ended with the specified line is displayed. This standard  
2931 requires conformance to historical practice.

2932 Historically, the `z` command would not display a partial line at the top or bottom  
2933 of the screen. If the partial line would normally have been displayed at the bot-  
2934 tom of the screen, the command worked, but the partial line was replaced with `@`  
2935 characters. If the partial line would normally have been displayed at the top of  
2936 the screen, the command would fail. For consistency and simplicity of  
2937 specification, this standard does not permit this behavior.

2938 Historically, the `z` command with a line specification of 1 ignored the command.  
2939 For consistency and simplicity of specification, this standard does not permit this  
2940 behavior.

2941 Historically, the `z` command did not set the cursor column to the first nonblank  
2942 for the `^` character if the first screen was to be displayed, and was already  
2943 displayed. For consistency and simplicity of specification, this standard does not  
2944 permit this behavior.

2945 **E.5.35.7.2.85 ZZ**

2946 There is no additional rationale provided for this subclause.

2947 **E.5.35.7.3 Input Mode Commands**

2948 Historical implementations of `vi` did not permit the the user to erase more than a  
 2949 single line of input, or to use normal erase characters such as *line erase*, *word*  
 2950 *erase*, and *erase* to erase autoindent characters. As there exist implementations  
 2951 of `vi` that do not have these limitations, both behaviors are permitted, but only  
 2952 historical practice is required. In the case of these extensions, `vi` is required to  
 2953 pause at the autoindent and previous line boundaries. C

2954 Historical implementations of `vi` updated only the portion of the screen where the  
 2955 current cursor character was displayed. For example, consider the `vi` input keys-  
 2956 trokes:

2957 `iabcd<escape>0C<tab>`

2958 Historically, the `<tab>` character would overwrite the characters `abcd` when it  
 2959 was displayed. Other implementations replace only the `a` character with the  
 2960 `<tab>`, and then push the rest of the characters ahead of the cursor. Both imple-  
 2961 mentations have problems. The historical implementation is probably visually  
 2962 nicer for the above example; however, for the keystrokes

2963 `iabcd<ESC>0R<tab><ESC>`

2964 the historical implementation results in the string `bcd` disappearing and then  
 2965 magically reappearing when `<ESC>` is entered. This standard requires the former  
 2966 behavior when overwriting erase-columns; i.e., overwriting characters that are no  
 2967 longer logically part of the edit buffer, and the latter behavior otherwise.

2968 Historical implementations of `vi` discarded the `<control-D>` and `<control-T>`  
 2969 characters when they were entered at places where their command functionality  
 2970 was not appropriate. This standard requires that the `<control-T>` functionality  
 2971 always be available, and that `<control-D>` be treated as any other key when not  
 2972 operating on autoindent characters.

2973 **E.5.35.7.3.1 NUL**

2974 Some historical implementations of `vi` limited the number of characters entered  
 2975 using the NUL input character to 256 bytes. This standard permits this limita- C  
 2976 tion; however, implementations are encouraged to remove this limit. C

2977 **E.5.35.7.3.2 <control-D>**

2978 See E.5.35.7.3.4. The hidden assumptions in the `<control-D>` command (and in  
 2979 the `vi` autoindent specification in general) is that `<space>` characters take up a  
 2980 single column on the screen and that `<tab>` characters are comprised of an  
 2981 integral number of `<space>` characters.

2982 **E.5.35.7.3.3 <control-H>**

2983 There is no additional rationale provided for this subclause.

2984 **E.5.35.7.3.4 <newline>**

2985 Implementations are permitted to rewrite autoindent characters in the line when  
2986 <newline>, <carriage-return>, <control-D>, and <control-T> are  
2987 entered, or when the shift commands are used, because historical implementa-  
2988 tions have both done so and found it necessary to do so. For example, a  
2989 <control-D> when the cursor is preceded by a single <tab>, with tabstop set  
2990 to 8, and shiftwidth set to 3, will result in the tab being replaced by several  
2991 <space> characters.

2992 **E.5.35.7.3.5 <control-T>**

2993 See E.5.35.7.3.4. Historically, <control-T> only worked if no non-<blank>  
2994 characters had yet been input in the current input line. In addition, the charac-  
2995 ters inserted by <control-T> were treated as autoindent characters, and could  
2996 not be erased using normal user erase characters. Because implementations exist  
2997 that do not have these limitations, and as moving to a column boundary is gen-  
2998 erally useful, this standard requires that both limitations be removed.

2999 **E.5.35.7.3.6 <control-U>**

3000 There is no additional rationale provided for this subclause.

3001 **E.5.35.7.3.7 <control-V>**

3002 Historically, vi used ^V, regardless of the value of the literal-next character of the  
3003 terminal. This standard requires conformance to historical practice.

3004 The uses described for <control-V> can also be accomplished with <control-  
3005 Q>, which is useful on terminals that use <control-V> for the down-arrow func-  
3006 tion. However, most historical implementations use <control-Q> for the *termios*  
3007 START character, so the editor will generally not receive the <control-Q> unless  
3008 stty ixon mode is set to off. (In addition, some historical implementations of vi  
3009 explicitly set ixon mode to on, so it was difficult for the user to set it to off.) Any  
3010 of the command characters described in POSIX.2 can be made ineffective by their  
3011 selection as *termios* control characters, using the stty utility or other methods  
3012 described in POSIX.1 {8}.

3013 **E.5.35.7.3.8 <control-W>**

3014 There is no additional rationale provided for this subclause.

3015 **E.5.35.7.3.9 <ESC>**

3016 Historically, SIGINT alerted the terminal when used to end input mode. This  
3017 behavior is permitted, but not required, by this standard.

3018 **E.5.35.8 Exit Status**

3019 There is no additional rationale provided for this subclause.

3020 **E.5.35.9 Consequences of Errors**

3021 There is no additional rationale provided for this subclause.

## **Annex F** (informative)

### **Revisions to Portability Considerations**

1 ⇒ **F Portability Considerations.** *Remove all references to the C-Language*  
2 *Binding Option and {POSIX2\_C\_BIND} from this annex, or reword to indicate*  
3 *they have moved to P1003.1a.*

4 **Rationale:** Since Annex B is gone, all references to it have to be removed.



## **Annex G** (informative)

### **Revisions to Sample National Profile**

1 ⇒ **G Sample National Profile.** *Remove all references to the C-Language Bind-*  
2 *ing Option and {POSIX2\_C\_BIND} from this annex, or reword to indicate they*  
3 *have moved to P1003.1a.*

4 **Rationale:** Since Annex B is gone, all references to it have to be removed.





## Annex H (informative)

### Balloting Instructions

1 This annex will not appear in the final standard. It is included in the draft to provide  
2 instructions for balloting that cannot be separated easily from the main document,  
3 as a cover letter might.

4 If you have received a copy of this draft before July 1999, it is important  
5 that you read this annex, whether you are an official member of the  
6 P1003.2b Balloting Group or not; comments on this draft are welcomed  
7 from all interested technical experts. **Your ballot is due to the IEEE  
8 office by \_\_\_ July 1999. This is not the date to postmark it—it is the  
9 date of receipt.**

#### 10 **Summary of Draft 12 Instructions**

11 This is the second “recirculation draft” of P1003.2b. The recirculation procedure  
12 is described in this annex. For this recirculation, we are accepting objections  
13 against any normative changes that occurred from Draft 11 to Draft 12 and the  
14 contents of the Unresolved Objections List, provided as a separate document from  
15 the draft.

16 Send your ballot and/or comments to:

17 IEEE Standards Office  
18 Computer Society Secretariat  
19 ATTN: P1003.2b Ballot  
20 P.O. Box 1331  
21 445 Hoes Lane  
22 Piscataway, NJ 08855-1331

23 It would also be very helpful if you sent us your ballot in machine-readable form.  
24 Your official ballot must be returned via mail to the IEEE office; if we receive only  
25 the e-mail or diskette version, that version will not count as an official document.  
26 However, the online version would be a great help to ballot resolution. Please e-  
27 mail to both of the following addresses:

28 [Don.Cragun@eng.sun.com](mailto:Don.Cragun@eng.sun.com)  
29 [nick@usenix.org](mailto:nick@usenix.org)

30 or IBM PC 3.5-inch diskette (plain text file), or Sun-style QIC-24 cartridge tapes to:

Copyright © 1999 IEEE. All rights reserved.  
This is an unapproved IEEE Standards Draft, subject to change.

31 Don Cragun  
32 Sun Microsystems, Inc.  
33 M/S UMPK17-307  
34 901 San Antonio Road  
35 Palo Alto, CA 94303

36 Some degree of judgment is required in determining what actually changed in  
37 Draft 12. Use the diff marks as a guide, but they will frequently mark text that  
38 has no real normative changes. Please limit your objections to the actual  
39 changes: for example, if we change the `foo -x` option to `-y`, don't use that as an  
40 opportunity to object that we have no `-z` option. Your objection should only  
41 address why the `x` to `y` change is a problem. (We have been balloting for a long  
42 time now and it is time to tighten the consensus and finish this up.) If you find  
43 problems unrelated to changes, submit them as comments and they will be con-  
44 sidered seriously in that category. Thanks for your cooperation on this.

#### 45 **Background on Balloting Procedures**

46 The Balloting Group consists of many technical experts who are members of the  
47 IEEE or the IEEE Computer Society; enrollment of individuals in this group has  
48 already been closed. There are also a few "parties of interest" who are not  
49 members of the IEEE or the Computer Society. Members of the Balloting Group  
50 are required to return ballots within the balloting period. Other individuals who  
51 may happen to read this draft are also encouraged to submit comments concern-  
52 ing this draft. The only real difference between members of the Balloting Group  
53 and other individuals submitting ballots is that *affirmative* ballots are only  
54 counted from Balloting Group members who are also IEEE or Computer Society  
55 members. (There are minimum requirements for the percentages of ballots  
56 returned and for affirmative ballots out of that group.) However, objections and  
57 nonbinding comments must be resolved if received from any individual, as fol-  
58 lows:

- 59 (1) Some objections or comments will result in changes to the standard. This  
60 will occur either by the publication of a list of changes or by the republi-  
61 cation of an entire draft. The objections/comments are reviewed by a  
62 team from the P1003.2 working group, consisting of the Chair, Vice  
63 Chair, the Chair of PASC, and one or more Technical Reviewers. The  
64 Technical Reviewers each have subject matter expertise in a particular  
65 area and are responsible for objection resolution in one or more sections.
- 66 (2) Other objections/comments will not result in changes.
  - 67 (a) Some are misunderstandings or cover portions of the document  
68 (front matter, informative annexes, rationale, editorial matters,  
69 etc.) that are not subject to balloting.
  - 70 (b) Others are so vaguely worded that it is impossible to determine  
71 what changes would satisfy the objector. These are referred to as  
72 *Unresponsive*. (The Technical Reviewers will make a reasonable  
73 effort to contact the objector to resolve this and get a newly worded  
74 objection.) Further examples of unresponsive submittals are those

75 not marked as either *Objection* or *Comment*; those that do not iden-  
 76 tify the portion of the document that is being objected to (each objec-  
 77 tion must be separately labeled); those that object to material in a  
 78 recirculation that has not changed and do not cite an unresolved  
 79 objection; those that do not provide specific or general guidance on  
 80 what changes would be required to resolve the objection.

- 81 (c) Finally, others are valid technical points, but they would result in  
 82 decreasing the consensus of the Balloting Group. (This judgment is  
 83 made based on other ballots and on the experiences of the working  
 84 group through almost five years of work and fifteen drafts preceding  
 85 this one.) These are referred to as *Unresolved Objections*. Sum-  
 86 maries of unresolved objections and their reasons for rejection are  
 87 maintained throughout the balloting process, are circulated to  
 88 members of the Balloting Group for their consideration, and are  
 89 presented to the IEEE Standards Board when the final draft is  
 90 offered for approval. Unresolved objections are only circulated to  
 91 the balloting group when they are presented by members of the bal-  
 92 loting group or by parties of interest. Unsolicited correspondence  
 93 from outside these two groups may result in draft changes, but are  
 94 not recirculated to the balloting group members.

95 Please ensure that you correctly characterize your ballot by providing one of the  
 96 following:

- 97 (1) Your IEEE member number  
 98 (2) Your IEEE Computer Society affiliate number  
 99 (3) If (1) or (2) don't apply, a statement that you are a "Party of Interest"

## 100 **Ballot Resolution**

101 The general procedure for resolving ballots is:

- 102 (1) The balloting cuts off on \_\_\_ July 1999. This is a receipt date at the  
 103 IEEE, not a postmark date. (Please do not telephone or FAX on \_\_\_ July  
 104 1999 and say that your specific comments will come later; late-arriving  
 105 comments will not be considered as objections.) We will accept comments  
 106 after that date, including direct e-mail to the working group officers or  
 107 the Technical Reviewers, but they will be treated as comments only—not  
 108 objections. And we don't guarantee a written response to these late sub-  
 109 missions.
- 110 (2) The ballots are put online and distributed to the Technical Reviewers.
- 111 (3) If a ballot contains an objection, the balloter will be contacted individu-  
 112 ally by telephone, letter, or e-mail and the corrective action to be taken  
 113 will be described (or negotiated). The personal contact will most likely  
 114 not occur if the objection is very simple and obvious to fix or the balloter  
 115 cannot be reached after a few reasonable attempts. Repeated failed  
 116 attempts to elicit a response from a balloter may result in an objection  
 117 being considered unresponsive, based on the judgment of the working

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- 118 group chair. Once all objections in a ballot have been resolved, it  
119 becomes an affirmative ballot.
- 120 (4) If any objection cannot be resolved, the entire ballot remains negative.
- 121 (5) Once more than seventy-five percent of the ballots received (that had  
122 voted either affirmative or negative) have been turned affirmative, two  
123 lists are published to the entire balloting group: the detailed list of  
124 approved changes and the list of unresolved objections, along with our  
125 reasons for rejecting them. This is known as a *recirculation*. You have  
126 minimum of ten days (after an appropriate time to ensure the mail got  
127 through) to review these two lists and take one of the following actions:
- 128 (a) Do nothing; your ballots will continue to be counted as we have  
129 classified them, based on items (3) and (4).
- 130 (b) Explicitly change your negative ballot to affirmative by agreeing to  
131 remove all of your objections from the unresolved list.
- 132 (c) Explicitly change your affirmative ballot to negative based on your  
133 disapproval of either of the two lists you reviewed. If an issue is not  
134 on one of the two lists, new objections about this are not allowed.  
135 Negative ballots that come in on recirculations cannot be cumula-  
136 tive. They shall repeat any objections that the balloter considers  
137 unresolved from the previous recirculation. Ballots that simply say  
138 “and all the unresolved objections from last time” will be declared  
139 unresponsive. Ballots that are silent will be presumed to fully  
140 replace the previous ballot, and all objections not mentioned on the  
141 most current ballot will be considered as successfully resolved.
- 142 (6) The list of changes will frequently be a new draft document with the  
143 changes integrated. This is not a requirement, however, and a small  
144 number of changes may prompt merely a change list approach to recircu-  
145 lation.
- 146 (7) A copy of all your objections and our resolutions will be mailed to you.  
147 You can receive the full package of all resolutions from all ballots by con-  
148 tacting the IEEE Standards Office (who will probably charge you for the  
149 copying involved). If you don’t agree with one of our resolutions and  
150 haven’t been contacted personally before you receive this list, please  
151 accept our apologies and submit a new ballot against the new draft dur-  
152 ing the recirculation period.
- 153 (8) If at the end of the recirculation period there remain greater than  
154 seventy-five percent affirmative ballots, and no new objections have been  
155 received, a new draft is prepared that incorporates all the changes. This  
156 draft and the unresolved objections list go to the IEEE Standards Board  
157 for approval. If the changes cause too many ballots to slip back into  
158 negative status, another resolution and recirculation cycle begins.

## 159 **Balloting Guidelines**

160 This section consists of guidelines on how to write and submit the most effective  
161 ballot possible. The activity of resolving balloting comments is difficult and time  
162 consuming. Poorly constructed comments can make that even worse.

163 We have found several things that can be done to a ballot that make our job more  
164 difficult than it needs to be, and likely will result in a less than optimal response  
165 to ballots that do not follow the form below. Thus it is to your advantage, as well  
166 as ours, for you to follow these recommendations and requirements.

167 If a ballot that significantly violates the guidelines described in this section comes  
168 to us, we will determine that the ballot is unresponsive, and simply ignore all the  
169 material in it.

170 Secondly, objections that don't contain a specification so that the correction to  
171 resolve the objection "can be readily determined" are also unresponsive and will  
172 be ignored.

173 (If we do recognize a ballot that is generally "unresponsive," we will try to inform  
174 the balloter as soon as possible so he/she can correct it, but it is ultimately the  
175 balloter's responsibility to assure the ballot is responsive.)

176 Typesetting is not particularly useful to us. And please do not send handwritten  
177 ballots. Typewritten (or equivalent) is fine, and if some font information is lost it  
178 will be restored by the Technical Editor in any case. If you use `nroff`, you will  
179 include extraneous spacing and sometimes backspaces and overstrikes; if you  
180 really must use `nroff`, please turn off hyphenation and line adjusting:

```
181     .hy 0
182     .na
```

183 and run the output through `col -b` to remove all the overstrikes. (Also  
184 remember that backslashes and leading periods and apostrophes in your text will  
185 be treated impolitely by the `*roff` family). The ideal ballot is formatted as a "flat  
186 ASCII file," without any attempt at reproducing the typography of the draft and  
187 without embedded control characters or overstrikes; it is then printed in Courier  
188 (or some other typewriter-like) font for paper-mailing to the IEEE Standards  
189 Office and simultaneously e-mailed to the working group Chair.

190 Don't quote others' ballots. Cite them if you want to refer to another's ballot. If  
191 more than one person wants to endorse the same ballot, send just the cover sheets  
192 and one copy of the comments and objections. [Note to Institutional Representa-  
193 tives of groups like X/Open, OSF, UI, etc.: this applies to you, too. Please don't  
194 duplicate objection text with your members.] Multiple identical copies are easy to  
195 deal with, but just increase the paper volume. Multiple almost-identical ballots  
196 are a disaster because we can't tell if they are identical or not, and are likely to  
197 miss the subtle differences. Responses of the forms:

198 — "I agree with the item in <someone>'s ballot, but I'd like to see this done  
199 instead"

200 — "I am familiar with the changes to `f00` in <someone>'s ballot and I would  
201 object if this change is [or is not] included"

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

202 are very useful information to us. If we resolve the objection with the original  
203 balloter (the one whose ballot you are referencing), we will also consider yours to  
204 be closed, unless you specifically include some text in your objection indicating  
205 that should not be done.

206 Be very careful of “Oh, by the way, this applies <here> too” items, particularly if  
207 they are in different sections of the document that are likely to be seen by dif-  
208 ferent reviewers. They are probably going to be missed! Note the problem in the  
209 appropriate section, and cite the detailed description if it’s too much trouble to  
210 copy it. The reviewers don’t have time to read the whole ballot, and only read the  
211 parts that appear to apply to them. Particularly where definitions are involved,  
212 even if the change really belongs in one section but the relevant content is in  
213 another, an extra cross-reference would be indicated. If you wish to endorse  
214 someone else’s ballot, either in whole or part, be specific about whether you will  
215 be automatically satisfied if they are satisfied. If you will not necessarily be  
216 satisfied if they are, your ballot could be deemed unresponsive because it does not  
217 give achievable conditions under which your ballot could be converted to  
218 affirmative. You then must give the conditions under which you would be  
219 satisfied as well. If you would be satisfied in some areas and not in others, it is  
220 best to specifically point to each specific objection in the ballot you point to, giving  
221 the conditions for each.

222 Please consider this a new ballot that should stand on its own. Please do not  
223 make backward references to your ballots for previous drafts—include all the text  
224 you want considered here because the Technical Reviewer may not have your old  
225 ballot. And, the old section and line numbers won’t match up anyway. If one of  
226 your objections was not accepted exactly as you wanted, it will not be useful to  
227 send in the exact text you sent before; read the nearby Rationale section and come  
228 up with a more compelling (or clearly-stated) justification for the change.

229 Please be very wary about global statements, such as “all of the arithmetic func-  
230 tions need to be defined more clearly.” Unless you are prepared to cite specific  
231 instances of where you want changes made, with reasonably precise replacement  
232 language, your ballot will be considered unresponsive.

### 233 **Ballot Form**

234 The following form is recommended. We would greatly appreciate it if you sent  
235 the ballot in electronic form in addition to the required paper copy. Our policy is  
236 to handle all ballots online, so if you don’t send it to us that way, we have to type  
237 it in manually. For the last POSIX.2 ballot, only one or two balloters could not  
238 accommodate us on this and thus we had very little typing to do. See the first  
239 page of this Annex for the addresses and media. As you’ll see from the following,  
240 formatting a ballot that’s sent to us online is much simpler than a paper-only bal-  
241 lot.

242 The ballot should be page-numbered, and contain the name, e-mail address, and  
243 phone number(s) of the objector(s). (If you send us only a paper copy, make sure  
244 this information appears on every page; electronic ballots just need it once, in the  
245 beginning.) The lines before the first dashed line are a page header, and should  
246 only appear once on each page. Please leave adequate (at least one inch) margins

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

247 on both sides. Each objection/comment/editorial comment should be sequentially  
248 numbered, not in individual ranges [i.e., not Objection #1, Comment #1]

249 Since we deal with the ballots online, there is no longer any requirement to put  
250 only one objection or section per page.

251 Don't format the ballot as a letter or document with its *own* section numbers.  
252 These are simply confusing. As shown below, it is best if you cause each objection  
253 and comment to have a sequential number that we can refer to amongst ourselves  
254 and to you over the phone. Number sequentially from 1 and count objections,  
255 comments, and editorial comments the same; don't number each in its own range.  
256 If you don't do this, we'll number them ourselves, but you won't know what  
257 numbers we're using.

258 Please precede each objection/comment with a little code line (if you don't, we'll  
259 have to do it ourselves):

260 @ <section>.<clause> <code> <seqno>

261 where:

- 262 @ At sign in column 1 (which means no @'s in any other column 1's).
- 263 <section> The major section (chapter or annex) number or letter in column  
264 3. Use zero for Global or for something, like the front matter,  
265 that has no section or annex number.
- 266 <clause> The clause number (second-level header). Please do not go deeper  
267 than these two levels. In the text of your objection or comment,  
268 go as deep as you can in describing the location, but this code line  
269 uses two levels only.
- 270 <code> One of the following lowercase letters, preceded and followed by  
271 spaces:
- 272 o Objection.
  - 273 c Comment or Editorial Comment.
- 274 <seqno> A sequence number, counting all objections and comments in a  
275 single range.

276 **Objection:**

277 Balloter Name (202)555-1212 page x of nn.

278 E-Mail Address FAX: Fax Number

279 Balloter2 Name (303)555-1213

280 E-Mail Address2 FAX: Fax Number2

281

282 @ x.y o seq#

283 &lt;Seq#&gt; Sect x.y OBJECTION. page xxx, line zzz:

284 Problem:

285 A clear statement of the problem that is observed, sufficient for others to under-  
 286 stand the nature of the problem. Note that you should identify problems by sec-  
 287 tion, page, and line numbers. This may seem redundant, but if you transpose a  
 288 digit pair, we may get totally lost without a cross-check like this. Use the line  
 289 number where the problem starts, not just where the section itself starts; we  
 290 sometimes attempt to sort objections by line numbers to make editing more accu-  
 291 rate. If you are referring to a range of lines, please don't say "lines 1000ff;" use a  
 292 real range so we can tell where to stop looking. If you have access to the online  
 293 versions of a balloting draft, please do not send in a ballot that refers to the page  
 294 numbers in the nroff output version; use only the line and page numbers found  
 295 in the printed draft or the online PostScript draft. We will really love you if you  
 296 can manage to include enough context information in the problem statement  
 297 (such as the name of the utility) so we can understand it without having the draft  
 298 in our laps at the time. (It also helps you when we e-mail it back to you.) If you  
 299 are objecting to an action in the Unresolved Objections List, use the  
 300 section/page/line number reference for the appropriate place in the standard;  
 301 don't refer to the UOL except to cite its number and for clarification of your points.

302 Action:

303 A precise statement of the actions to be taken on the document to resolve the  
 304 objection above, which if taken verbatim will completely remove the objection.

305 If there is an acceptable range of actions, any of which will resolve the problem for  
 306 you if taken exactly, please indicate all of them. If we accept any of these, your  
 307 objection will be considered as resolved.

308 If the Action section is omitted or is vague in its solution, the objection will be  
 309 reclassified as a nonbinding comment. The Technical Reviewers, being human,  
 310 will give more attention to Actions that are well-described than ones that are  
 311 vague or imprecise. The best ballots of all have very explicit directions to substi-  
 312 tute, delete, or add text in a style consistent with the rest of the document, such  
 313 as:



314 Delete the sentence on lines 101-102:  
 315       "The implementation shall not ... or standard error."  
 316 On line 245, change "shall not" to "should not".  
 317 After line 103, add:  
 318       -r     Reverse the order of bytes read from the file.

319 **Some examples of poorly-constructed actions:**

320 Remove all features of this command that are not supported by BSD.  
 321 Add -i.  
 322 Make this command more efficient and reliable.  
 323 Use some other flag that isn't so confusing.  
 324 I don't understand this section.  
 325 Specify a value--I don't care what.

326 **Objection Example:**

327 Hal Jespersen                           (415) 364-3410                           page 3 of 17.  
 328 UUCP: hlj@Posix.COM FAX:     (415) 364-4498

329 -----

330 @ 2.6 o 23  
 331 23. Sect 2.6 OBJECTION. page 77, line 1217:

332 Problem:

333 The EDITOR environment variable is not used as stated  
 334 in my company. This description would cause hundreds  
 335 of my shell scripts to break.

336 Action:

337 Change the first sentence on line 1217 to:

338       The e-mail address of the editor of the user's  
 339       favorite POSIX standard.

340 -----

341 @ 3.1 o 24  
 342 24. Sect 3.1.6 OBJECTION. page 123, line 17:

343 Problem:

344 I support UO 3.01-999-6 concerning the objection to the  
 345 definition of "operator".  
 346 This definition would cause great hardship to the users  
 347 of the systems I develop.  
 348 I feel your rationale for rejection was inappropriate  
 349 because you overlooked the following technical points [etc.]...

350 Action:

351 Change the term "operator" to "operation-symbol" in this  
 352 definition and globally throughout Section 3.

353 **Comment:**

Copyright © 1999 IEEE. All rights reserved.  
 This is an unapproved IEEE Standards Draft, subject to change.

354

355

-----  
@ x.z c seq#

356

&lt;Seq#&gt; Sect x.z COMMENT. page xxx, line zzz:

357

A statement of a problem that you might want to be resolved by the reviewer, but which does not in any way affect whether your ballot is negative or positive. The form for objections is not required, but it increases the probability that your comment will have an effect on the final document.

358

359

360

361

Although there may be questions to you or responses on the topic, no changes in the drafts are required by a comment, although it will be looked at to determine whether the concern should be addressed. It is possible to abuse this rule and label all of your comments as objections, but it is a significant disservice to the individuals who are volunteering their time to address your concerns.

362

363

364

365

366

Remember that any issue concerning the pages preceding page 1 (the front matter), Rationale text with shaded margins, Annexes, NOTES in the text, footnotes, or examples will be treated as a nonbinding comment whether you label it that way or not, but it would help us if you'd label it correctly.

367

368

369

370

**Editorial Comment:**

371

372

-----  
@ x.z c seq#

373

&lt;Seq#&gt; Sect x.z EDITORIAL COMMENT. page xxx, line zzz:

374

These are for strictly editorial issues, where the technical meaning of the document is not changed. Examples are: typos; misspellings; English syntax or usage errors; appearances of lists or tables; arrangement of sections, clauses, and sub-clauses (except where the location of information changes the optionality of a feature). Marking these as comments but indicating that they are editorial speeds the process.

375

376

377

378

379

380

Please be aware that after balloting concludes the document will be subjected to more sets of editors at the IEEE and ISO who are empowered to make broad editorial changes and rewording (for example, to get the text ready for translation into French.)

381

382

383

384

Thank you for your cooperation in this important balloting process.

385

Don Cragun

## Alphabetic Topical Index

- 0 ... 226, 340  
     ... 224, 339  
 ! ... 167  
 < ... 168, 168  
 = ... 169  
 @ ... 169  
 ! ... 217  
 \$ ... 218  
 % ... 219  
 & ... 220  
 ' ... 220  
 ` ... 220  
 [[ ... 221  
 ]] ... 222  
 ^ ... 222  
     ... 223  
 ( ... 223  
 ) ... 223  
 | ... 223  
 . ... 224  
 / ... 225  
 < ... 227  
 : ... 227  
 ; ... 227  
 > ... 228  
 ? ... 228  
 @ ... 229  
 . ... 307  
 ! ... 317  
 < ... 317, 317  
 = ... 318  
 @ ... 318  
 ! ... 338  
 \$ ... 338  
 % ... 338  
 & ... 338  
 ' ... 338  
 [[ ... 338  
 ` ... 338  
     ... 339  
 ( ... 339  
     ) ... 339  
     . ... 339  
     ]] ... 339  
     ^ ... 339  
     | ... 339  
     / ... 340  
     : ... 340  
     < ... 341, 341  
     ; ... 341  
     ? ... 341  
     @ ... 341  
     @\* ... 341  
     \_ ... 223, 339  
     .. ... 307  
     \_\_\_ ... 355
- ### A
- a ... 229, 342  
 A ... 230, 342  
 abbrev ... 306  
 abbreviate ... 138, 144, 301-302, 306-307, 310  
 Abbreviations ... 6  
 access() ... 174  
 ACL ... 286  
 Addressing ... 133, 298  
 affirmative response ... 84  
 append ... 129, 138, 142, 145, 149, 171, 307-309, 311, 316, 318, 321, 340  
 appropriate privileges ... 89-90, 94-95, 199, 278-279, 281-282, 287, 291, 296  
 ar  
     – Create and maintain library archives ... 255  
         Standard Output ... 255  
         utility definition ... 255  
 archiving files (pax) ... 72  
 args ... 145, 307  
 ARGV ... 28  
 ARPA ... 312, 328  
 ASCII ... 204, 289, 312, 319, 328

at ... 119-120  
 – Execute commands at a later time ... 119  
 Options ... 119  
 Standard Error ... 120  
 Synopsis ... 119  
 utility definition ... 119

*atof()* ... 35

autoindent ... 321  
 ai ... 171

autoprint ... 321  
 ap ... 171

autowrite ... 321  
 aw ... 172

awk ... 27, 29-31, 33-35  
 – Pattern scanning and processing language ... 27  
 Arithmetic Functions ... 31  
 Expressions ... 28  
 Input/Output and General Functions ... 32  
 Lexical Conventions ... 33  
 Operands ... 27  
 Output Statements ... 30  
 Overall Program Structure ... 27  
 Regular Expressions ... 30  
 Standard Input ... 27  
 String Functions ... 31-32  
 User-Defined Functions ... 33  
 utility definition ... 27  
 Variable and Special Variables ... 29-30

## B

b ... 230, 342

B ... 230, 342

<backslash> ... 141

Balloting Instructions ... 355

basename ... 76

batch ... 120  
 – Execute commands at a later time ... 120  
 Description ... 120  
 <title> ... 120  
 utility definition ... 120

bc ... 35-36  
 – Arbitrary-precision arithmetic language ... 35  
 Grammar ... 36  
 Operations ... 35-36  
 utility definition ... 35

BEGIN ... 27

Bibliography ... 271

BINARY ... 89

BRE ... 49-50, 80, 84, 102-103, 106-107, 121, 172, 191, 194-195

BREs Matching Multiple Characters ... 15-16

BSD ... 278-279, 283, 292, 296, 302-303, 312, 324, 327, 329, 336  
 4.3 ... 277, 327  
 4.4 ... 277, 327

BUFSIZ ... 293

built-in ... 7, 30, 256, 258, 276-277

Built-In Utilities ... 6-7

## C

c ... 230, 342

C ... 231, 342

c89 ... 70, 182  
 – Compile Standard C programs ... 265  
 Standard Libraries ... 265  
 utility definition ... 265

CAE ... 10

case  
 Conditional Construct ... 22

cat ... 312, 329

cd ... 14, 22, 37-39, 276-277, 295  
 – Change working directory ... 37  
 Description ... 37  
 Environment Variables ... 39  
 Operands ... 38  
 Options ... 38  
 Synopsis ... 37  
 utility definition ... 37

**CDPATH**  
 variable ... 37-39, 307

CDPATH ... 39

CD-ROM ... 285

change ... 129, 138, 146, 149, 171, 307, 318, 321, 333, 340

Character Set ... 7, 10

Character Set Description File ... 7-8

CHARCLASS ... 14

CHARCLASS\_NAME\_MAX ... 11-13, 18

charmap ... 114

CHARMAP ... 8-9

charmap file ... 58

- chdir()* ... 37-38
  - chdir* ... 146, 295, 307
  - chgrp* ... 40, 274, 277
    - Change file group ownership ... 40
    - Options ... 40
    - Synopsis ... 40
    - utility definition ... 40
  - chmod* ... 41, 68
    - Change file modes ... 41
    - Options ... 41
    - Synopsis ... 41
    - utility definition ... 41
  - chown()* ... 287
  - chown* ... 42, 274-275
    - Change file ownership ... 42
    - Options ... 42
    - Synopsis ... 42
    - utility definition ... 42
  - c\_magic* ... 291
  - collating element order
    - definition of ... 6
  - collation sequence ... 5
  - Collation Sequence ... 12
  - COLUMNS**
    - variable ... 127, 178, 188
  - comm* ... 10
  - Configuration Values ... 17
  - conformance ... 2, 52-53, 182, 290, 292, 296-304, 306-312, 314, 317-318, 321, 326, 330, 332, 334, 336, 339-345, 347, 349
  - Conformance ... 2
  - confstr()* ... 56-57
  - contiguous file ... 290-291
  - <control-^> ... 216
  - <control-] > ... 217, 337
  - <control-^ > ... 337
  - <control-B> ... 212, 335
  - <control-D> ... 169, 213, 247, 317, 336, 348
  - <control-E> ... 213, 336
  - <control-F> ... 213, 336
  - <control-G> ... 214, 336
  - <control-H> ... 214, 248, 336, 349
  - <control-L> ... 215, 336
  - <control-P> ... 215, 336
  - <control-R> ... 215, 337
  - <control-T> ... 249, 349
  - <control-U> ... 216, 250, 337, 349
  - <control-V> ... 141, 250, 303, 349
  - <control-W> ... 141, 251, 303, 349
  - <control-Y> ... 216, 337
  - copy* ... 146, 308
  - cp* ... 43-44, 280
    - Copy files ... 43
    - Description ... 43-44
    - Options ... 44
    - Synopsis ... 43
    - utility definition ... 43
  - cpio* ... 71-72, 81-82, 96, 278-284, 286, 291
    - Archive Entry ... 97
    - c\_mode* File Modes ... 98
    - c\_mode* File Types ... 98
    - File Data ... 98
    - File Name ... 98, 291
    - format ... 96
    - Header ... 96, 291
    - Interchange Format ... 96
    - Special Entries ... 99
  - creat()* ... 279
  - crypt* ... 293
  - cs* ... 293, 316
  - csplit* ... 121
    - Split files based on context ... 121
    - Operands ... 121
    - utility definition ... 121
  - C Standard ... 12, 35
  - ctags* ... 122, 126, 161, 187, 196, 293, 327
    - Create a tags file ... 122
    - Description ... 122
    - Output Files ... 122
    - utility definition ... 122
  - <ctype.h> ... 265
  - cut* ... 10, 45
    - Cut out selected fields of each line of a file ... 45
    - Options ... 45
    - utility definition ... 45
- ## D
- d* ... 231, 342
  - D* ... 232, 343
  - date* ... 304
    - Write the date and time ... 46
    - Modified Field Descriptors ... 46
    - utility definition ... 46
  - dd* ... 46, 280
    - Convert and copy a file ... 46
    - Asynchronous Events ... 46
    - Description ... 46
    - utility definition ... 46

DEF ... 326  
 DEFAULT ... 261  
 DEL ... 319  
 delete ... 65, 67, 136, 147, 172, 300, 308  
 /dev ... 290  
 /dev/stdout ... 200-202, 204  
 /dev/tty ... 74-75, 84, 188, 198, 281  
 diff ... 10, 47  
   - Compare two files ... 47  
   -c or -C Output Format ... 48  
   Directory Comparison Format ... 47  
   Operands ... 47  
   Options ... 47  
   utility definition ... 47  
 directory entry ... 5, 38, 57, 101  
 dirname ... 76  
 Discard and refresh ... 194  
 Display position ... 196  
 document ... 2, 47, 108, 114-116, 186, 290-291, 299, 312, 317, 320, 327, 332, 339  
 Double Quotes ... 21  
 DSEMI ... 23  
 du ... 123  
   - Estimate file usage ... 123  
   Description ... 123  
   Options ... 123  
   Synopsis ... 123  
   utility definition ... 123

## E

e ... 232, 343  
 E ... 232, 343  
 EBCDIC ... 204, 285  
 ed ... 49, 51-54, 80, 281  
   - Edit text ... 49  
   Addressing ... 49  
   Asynchronous Events ... 49  
   Change Command ... 51  
   Commands ... 51  
   Global Command ... 51  
   Insert Command ... 52  
   Interactive Global Command ... 52  
   List Command ... 52  
   utility definition ... 49  
 ed.hup ... 51  
 edit ... 130, 137, 147, 155, 158-159, 162-163, 174, 216, 292, 297, 299, 301, 304, 308, 311, 316

Edit Options ... 170, 319  
 editor  
   ex ... 124  
**EDITOR**  
   variable ... 65, 188, 196  
 editor  
   full-screen (vi) ... 205  
 effective group ID ... 21, 109  
 effective user ID ... 21, 109  
 END ... 8-9, 28  
**ENV**  
   variable ... 21-22, 109  
 ENVIRON ... 28-29  
**environment**  
   variable ... 37  
 eof ... 140, 303  
 EOL ... 14  
 ERE ... 16, 30-31, 33-34, 84  
 EREs Matching Multiple Characters ... 16  
 errorbells ... 321  
   eb ... 172  
 <ESC> ... 216, 251, 337, 349  
 ESC ... 128, 144-145, 207, 210, 216, 239, 251, 294, 310, 337, 345, 348-349  
 /etc/magic ... 180, 184  
 ex ... 124-133, 135-147, 149-151, 153-155, 157, 159, 161, 163, 165, 167-175, 178-179, 191, 196, 205-207, 210, 214, 216-218, 220, 225, 227-228, 235, 238, 240-241, 244-245, 249, 292-321, 323-329, 331, 333-334, 340-341, 343, 345  
   - Text editor ... 124, 292  
   abbreviate command ... 144  
   abbreviate rationale ... 306  
   and vi Initialization ... 130, 207, 295, 331  
   append command ... 145  
   append rationale ... 307  
   args command ... 145  
   autoindent option ... 171  
   autoindent rationale ... 321  
   autoprint option ... 171  
   autoprint rationale ... 321  
   autowrite option ... 172  
   autowrite rationale ... 321  
   <backslash> command ... 141  
   change command ... 146  
   change rationale ... 307  
   chdir command ... 146  
   chdir rationale ... 307  
   ! command ... 167  
   < command ... 168, 168  
   = command ... 169

@ command ... 169  
 Command Descriptions ... 142, 303  
 Command-Line Parsing ... 135, 299  
 <control-D> command ... 169  
 <control-D> rationale ... 317  
 <control-V> command ... 141  
 <control-W> command ... 141  
 copy command ... 146  
 copy rationale ... 308  
 delete command ... 147  
 delete rationale ... 308  
 edit command ... 147  
 edit rationale ... 308  
 eof command ... 140  
 errorbells option ... 172  
 exrc option ... 172  
 file command ... 148  
 file rationale ... 308  
 global command ... 149  
 global rationale ... 309  
 ignorecase option ... 172  
 ignorecase rationale ... 322  
**Input Editing** ... 139, 302  
 insert command ... 150  
 insert rationale ... 309  
 join command ... 150  
 join rationale ... 309  
 list command ... 151  
 list option ... 172  
 list rationale ... 309  
 magic option ... 173  
 map command ... 151  
 map rationale ... 309  
 mark command ... 153  
 mark rationale ... 311  
 mesg option ... 173  
 move command ... 154  
 <newline> command ... 141  
 next command ... 154  
 next rationale ... 311  
 number command ... 155  
 number option ... 173  
 open command ... 155  
 open rationale ... 311  
 paragraphs option ... 173  
 paragraphs rationale ... 322  
 preserve command ... 155  
 preserve rationale ... 312  
 print command ... 156  
 print rationale ... 312  
 prompt option ... 174  
 put command ... 157  
 put rationale ... 313  
 quit command ... 157  
 rationale ... 292  
 ! rationale ... 317  
 < rationale ... 317, 317  
 = rationale ... 318  
 @ rationale ... 318  
 read command ... 157  
 readonly option ... 174  
 readonly rationale ... 323  
 read rationale ... 313  
 recover command ... 158  
 recover rationale ... 314  
 remap option ... 174  
 report option ... 174  
 report rationale ... 323  
 rewind command ... 158  
 rewind rationale ... 314  
 s command ... 159  
 scroll option ... 175  
 sections option ... 175  
 set command ... 160  
 set rationale ... 315  
 shell command ... 160  
 shell option ... 175  
 shiftwidth option ... 175  
 showmatch option ... 176  
 showmatch rationale ... 324  
 showmode option ... 176  
 showmode rationale ... 324  
 slowopen option ... 176  
 slowopen rationale ... 324  
 source command ... 161  
 source rationale ... 315  
 s rationale ... 314  
 suspend command ... 161  
 tabstop option ... 176  
 tag command ... 161  
 taglength option ... 176  
 tag option ... 176  
 tag rationale ... 315  
 tags rationale ... 325  
 term option ... 177  
 term rationale ... 325  
 terse option ... 177  
 terse rationale ... 325  
 unabbreviate command ... 162  
 undo command ... 162  
 undo rationale ... 316  
 unmap command ... 163  
 utility definition ... 124  
 version command ... 163  
 version rationale ... 316  
 visual command ... 163  
 warn option ... 177  
 window option ... 177  
 window rationale ... 325  
 wrapmargin option ... 177  
 wrapmargin rationale ... 326  
 wrapscan option ... 178  
 writeany option ... 178  
 write command ... 164

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- write rationale ... 316
  - xit command ... 165
  - yank command ... 165
  - z command ... 166
  - z rationale ... 317
  - Examine new file ... 195
  - Examine next file ... 195
  - Examine previous file ... 195
  - EXINIT**
    - variable ... 125, 127, 131-132, 139, 296, 299, 302
  - expand ... 179, 199
    - Convert tabs to spaces ... 179
    - Environment Variables ... 179
    - utility definition ... 179
  - expr ... 54
    - Evaluate arguments as an expression ... 54
    - Extended Description ... 54
    - Standard Output ... 54
    - utility definition ... 54
  - .exrc ... 125-127, 131-132, 139, 172, 296-297, 299, 302-303
  - exrc ... 321
    - ex ... 172
  - Extended Header ... 88
  - Extended Header File Times ... 92
  - Extended Header Keyword Precedence ... 91
- F**
- f ... 233, 343
  - F ... 233, 343
  - FIFO ... 47, 60, 68, 94-95, 98-99, 313
  - file
    - archiving (pax) ... 72
    - contiguous ... 290-291
    - copying (pax) ... 72
    - displaying (more) ... 185
    - high performance ... 290-291
    - interchanging (pax) ... 72
  - file ... 67, 130, 148, 164, 180-181, 184, 214, 274, 308
    - Determine file type ... 180
    - Description ... 180
    - Extended Description ... 181
    - Options ... 180
    - Standard Output ... 181
    - Synopsis ... 180
    - utility definition ... 180
  - FILENAME ... 28
  - file system ... 6, 19, 69, 71, 81-82, 90, 92, 273, 276, 281, 285, 287, 289
  - find ... 54-56, 276
    - Find files ... 54
    - Description ... 54
    - Operands ... 55-56
    - Options ... 55
    - Synopsis ... 54
    - utility definition ... 54
  - FNR ... 32
  - FTP ... 292
  - full-screen editor (vi) ... 205
- G**
- G ... 233, 343
  - General Terms ... 5, 273
  - getconf ... 18, 56, 265
    - Get configuration values ... 56
    - Operands ... 56
    - Standard Output ... 57
    - utility definition ... 56
  - getenv() ... 14
  - getopt() ... 17
  - getopts ... 286
  - global ... 137, 139, 144
  - global! ... 149
  - global ... 149, 159, 171, 174-175, 301-303, 309, 312, 321, 323
  - Go to beginning of file ... 193
  - Go to end-of-file ... 193
  - Go to tag ... 196
  - grep ... 10
- H**
- H ... 234, 343
  - hard link ... 5
  - head ... 10
  - Header Block ... 87
  - Help ... 192
  - Historical Practice for Symbolic Links ... 277
  - HOME**
    - variable ... 37, 51, 127, 131-132, 146, 172
  - SHOME**
    - variable ... 296



\$HOME/.exrc ... 296  
HTML ... 338

## I

i ... 234, 343  
I ... 234, 343  
iconv ... 115-118  
  — Convert file codesets ... 114-115  
  utility definition ... 115  
ICRNL ... 332  
IEEE ... 1, 5  
IEEE P1003.1 ... 1-2, 5  
IEEE P1003.1a ... 1-2, 5-6, 17, 19, 21, 56-57,  
  109, 273, 351, 353  
IEEE P1003.2 ... 1-2, 5, 45  
IEEE P1003.2b ... 1, 5, 45, 102  
IEEE Std 1003.1 ... 273  
IEEE Std 1003.1a ... 2  
IEEE Std 1003.1b ... 2  
IEEE Std 1003.2 ... 5, 15-17, 21-23, 25, 27,  
  36, 46, 48-49, 51-52, 54, 56, 62-63, 65, 67-  
  70, 101-102, 109-111, 113-114, 121, 123-  
  124, 184-185, 197-199, 253, 255-256, 266,  
  278  
IGNORE ... 258-259  
ignorecase ... 322  
  ic ... 172  
implementation defined ... 8, 11, 37-38, 44,  
  57-59, 70, 73-74, 79-86, 89-91, 93, 97-98,  
  105, 111, 115, 118, 141, 144, 146, 151, 153,  
  156, 164, 174, 176-177, 182-183, 187, 191-  
  192, 196, 198-199, 219, 248, 250-251, 260,  
  281-282, 285, 296, 312, 316, 322, 338  
i-node  
  in archive ... 284  
  modification time ... 287  
Input Mode Commands ... 245, 348  
insert ... 129, 138, 142, 149-150, 171, 307-  
  309, 311, 318, 321  
interchange  
  cpio ... 96  
  pax ... 86  
  tar ... 92  
Interface to the Lexical Analyzer ... 266  
Internal Macros ... 261  
Invoke editor ... 196  
IRV ... 89, 93, 95-96, 98

IS1 ... 9  
ISO 1001 ... 284  
ISO 2375 ... 286  
ISO 8859-1 ... 89  
ISO 8859-2 ... 89  
ISO/IEC 10646-1 ... 2, 271  
ISO/IEC 10646 ... 6-8, 58-59, 88-89, 289  
ISO/IEC 14652 ... 271  
ISO/IEC 15435 ... 271  
ISO/IEC 15897 ... 271  
ISO/IEC 646 ... 88-89, 91, 93-96, 98, 204,  
  283, 289  
ISO/IEC 9899 ... 12, 35  
ISO/IEC 9945-1 ... 265  
ISO/IEC 9945-2 ... 8-13, 46, 62, 65, 71, 114,  
  119-120, 179-180, 185, 199-201  
ISO-IR ... 89  
*is\_wctype()* ... 12

## J

J ... 235, 343  
job control ... 161, 328  
join ... 10, 150, 235, 309, 343

## K

KornShell ... 24, 39  
ksh ... 24

## L

L ... 235, 344  
**LANG**  
  variable ... 84, 116, 127, 188  
language binding ... 267  
LC\_ ... 84, 127, 188  
**LC\_ALL**  
  variable ... 84, 116, 127, 188  
**LC\_COLLATE**  
  variable ... 84, 127, 188  
LC\_COLLATE ... 6, 111-112, 278  
**LC\_CTYPE**  
  variable ... 84, 116, 127, 179, 189, 199,  
  253

LC\_CTYPE ... 9, 11-12, 112, 179, 199, 253

### LC\_MESSAGES

variable ... 84, 117, 127, 160, 189

LC\_MESSAGES ... 84

LC\_MONETARY ... 12-13

### LC\_TIME

variable ... 85

LC\_TIME ... 13

less ... 327-328

LINE\_MAX ... 126, 131, 293

### LINES

variable ... 127, 177, 189

*link()* ... 5, 57-58

LISP ... 320, 335, 338

list ... 136, 142, 151, 172, 304, 309, 322

### LISTER

variable ... 62, 65

List-Mode Format Specifications ... 82

ln ... 5, 19, 57-58

– Link files ... 57

Description ... 57

Operands ... 58

Options ... 58

Synopsis ... 57

utility definition ... 57

Locale ... 11

localedef ... 59

– Define locale environment ... 58

Consequences of Errors ... 59

Extended Description ... 59

Options ... 58

Synopsis ... 58

utility definition ... 58

Locale Definition ... 11

Locale Grammar ... 14

Locale Lexical Conventions ... 13

ls ... 60-62, 83, 85, 274, 276, 282, 290

– List directory contents ... 60

Description ... 60

Exit Status ... 62

Options ... 60-61

Standard Output ... 61-62

Synopsis ... 60

utility definition ... 60

## M

m ... 235, 344

M ... 236, 344

Macros ... 259-260

magic ... 173, 322

mailx ... 62-67, 155, 328

– Process Messages ... 62

Delete messages ... 65

Delete messages and display ... 65

Display header summary ... 66

Display list of folders ... 65

Edit messages ... 65

Edit message with full-screen editor ... 67

Environment Variables ... 62

Extended Description ... 63-64

Internal Variables ... 64

Process next specified message ... 66

Save messages ... 66

Undelete messages ... 66

utility definition ... 62

make ... 256-260

Environment Variables ... 257

Operands ... 257

Options ... 256

Output Files ... 257

Standard Output ... 257

utility definition ... 256

Makefile Execution ... 258

Makefile Syntax ... 258

### MAKEFLAGS

variable ... 256-257, 260

MAKEFLAGS ... 260

map ... 138-139, 151-152, 212, 301-302, 306, 309-310

mark ... 136, 153-154, 235, 311

Mark position ... 194

<math.h> ... 265

### MBOX

variable ... 64

mesg ... 173, 322

MIME ... 202, 204, 271

MIME-RFC ... 200-201

*mkdir()* ... 68, 73

mkdir ... 68-69

– Make directories ... 68

Description ... 68

Options ... 68

utility definition ... 68

*mkfifo()* ... 68

mkfifo ... 68-69

– Make Make FIFO special files ... 68

Description ... 68

Options ... 68

utility definition ... 68

mode bit ... 94

## MORE

variable ... 189

more ... 185-188, 190-192, 196-197, 327-328

- Display files on a page-by-page basis ... 185, 327
- discard and refresh command ... 194
- display position command ... 196
- editor command ... 196
- examine new file command ... 195
- examine next file command ... 195
- examine previous file command ... 195
- exit command ... 196
- go to beginning of file command ... 193
- go to end-of-file command ... 193
- go to tag command ... 196
- help command ... 192
- invoke editor command ... 196
- mark position command ... 194
- move backwards one screenful command ... 192
- move forwards one screenful command ... 192
- quit command ... 196
- rationale ... 327
- refresh the screen command ... 194
- repeat search command ... 195
- repeat search in reverse command ... 195
- return to mark command ... 194
- return to previous position command ... 194
- scroll backwards one-half screenful command ... 193
- scroll backwards one line command ... 193
- scroll forwards one-half screenful command ... 193
- scroll forwards one line command ... 192
- search backwards for pattern command ... 195
- search forwards for pattern command ... 194
- skip forwards one line ... 193
- utility definition ... 185

move ... 154, 311

M/S ... 318

MSE ... 12

mv ... 69, 274

- Move files ... 69
- Description ... 69
- utility definition ... 69

## N

n ... 236, 344

N ... 236, 344

NAME\_MAX ... 281

<newline> ... 141, 214, 249, 303, 336, 349

next ... 67, 130, 137-139, 154, 174, 292, 297, 299, 301, 308, 311, 316, 321

nice ... 199

**noclobber**

- variable ... 293

nonprintable ... 86, 105, 191

Normative References ... 2

NUL ... 247, 348

NUL ... 9, 12, 30, 90, 93-94, 96-97, 126, 247, 289, 291, 312, 348

number ... 136, 142, 155, 304, 311, 322, 326

nu ... 173

NUMBER ... 28

## O

o ... 237, 344

O ... 237, 344

obsolescent features ... 24, 124-125, 185-187, 198-199, 205-206, 293

Octet-Oriented `cpio` Archive Entry ... 97

od ... 70

- Dump files in various formats ... 70
- Extended Description ... 70
- Operands ... 70
- utility definition ... 70

OFS

- awk variable ... 30

OFS ... 29-30

open ... 124, 137, 142, 149, 155, 174-175, 304, 311-312, 320, 323, 344

## P

p ... 237, 344

P ... 238, 345

PAR ... 3

paragraphs ... 322

- para ... 173

Parameter Expansion ... 278

PASC ... 5, 15-17, 21-23, 25, 27, 36, 46, 48-49, 51-52, 54, 56, 62-63, 65, 67-70, 101-102, 109-111, 113, 121, 123-124, 185, 197-199, 253, 255-256, 266, 278

- paste ... 10
- patch ... 197-198
- Apply changes to files ... 197
  - Filename Determination ... 197
  - Options ... 197
  - utility definition ... 197
- PATH**
- variable ... 127
- PATH\_MAX ... 290-291
- Pathname Variable Values ... 19
- pattern matching notation ... 83-84, 280
- Pattern Matching Notation ... 23
- Patterns Matching a Single Character ... 23
- pax ... 19, 71-82, 84-86, 88-92, 94-95, 99, 276, 278-282, 284-291
- Portable archive interchange ... 71, 278
  - format ... 86
  - Format Archive Example ... 87
  - Interchange Format ... 86
  - rationale ... 278
  - utility definition ... 71
- pg ... 327
- pipe ... 191, 279, 299
- PIPE ... 299, 301
- Plan 9 ... 118
- popen()* ... 21
- Portability Considerations ... 351
- POSIX.1 ... 1-2, 5, 14, 16-17, 19, 21, 37-39, 47, 56-58, 69, 71-73, 82, 89-90, 92, 94-95, 97-98, 101, 109, 139, 161, 174, 202, 258, 263, 265-267, 273, 278-279, 283-284, 287, 290-292, 302-303, 349
- POSIX.2 ... 2, 12, 15, 17, 19, 52, 57, 74, 102, 108, 124, 185, 204, 275-276, 278, 280-281, 284, 286, 290, 293, 319, 327, 330, 349
- POSIX2\_C\_BIND ... 2, 19, 273, 351, 353
- POSIX2\_CHARCLASS\_NAME\_MAX ... 18
- POSIX2\_CHAR\_TERM ... 205
- POSIX2\_LOCALEDEF ... 115
- POSIX2\_SYMLINKS ... 17, 19
- POSIX Locale ... 11-13, 48, 120, 122, 139, 143, 151, 153, 173, 175, 177, 209, 309, 322
- \_POSIX\_VERSION ... 265
- PRECIOUS ... 259
- preserve ... 128, 155, 312
- print ... 30, 53, 65-66, 135-136, 142, 149, 156, 165, 172-173, 298-299, 304, 312, 317
- printf ... 184, 282
- prompt ... 174, 323
- prompting ... 63, 198
- put ... 142, 157, 313, 345
- PWD**
- variable ... 7, 14, 22, 37-39, 100, 109
- pwd ... 6-7, 22, 100, 276
- Return working directory name ... 100
  - Description ... 100
  - Environment Variables ... 100
  - Options ... 100
  - Synopsis ... 100
  - utility definition ... 100
- Q**
- Q ... 238, 340, 345
- quit ... 157, 164, 313
- Quit ... 196
- R**
- r ... 239, 345
- R ... 239, 345
- range expression ... 15
- RE/0 ... 340
- read ... 130, 137, 139, 142, 144, 157-158, 164, 171, 296, 299, 311, 313, 316, 321
- readonly ... 323
- ro ... 174
- real user ID ... 131-132, 296
- RE and Bracket Expression Grammar ... 16
- RE Bracket Expression ... 15
- recover ... 129-130, 155, 158, 297, 314, 316
- red ... 292
- Refresh the screen ... 194
- regcomp()* ... 15
- regular expression ... 10, 12, 15, 30-31
- Regular Expression Notation ... 15
- reindent ... 335
- RE Introduction ... 15
- remap ... 174, 323
- rename()* ... 69
- renice ... 199
- Set system scheduling priorities of running processes ... 198
  - Description ... 198
  - Operands ... 199
  - Options ... 199
  - Synopsis ... 198
  - utility definition ... 198

- Repeat search ... 195
  - Repeat search in reverse ... 195
  - REPLACE ... 306
  - Replacement Strings ... 170, 318
  - report ... 174, 323
  - Requirements ... 2
  - REs ... 170, 318
  - return ... 36
  - Return to mark ... 194
  - Return to previous position ... 194
  - Revisions to Bibliography ... 271
  - Revisions to C Language Bindings Option ... 267
  - Revisions to C Language Development Utilities Option ... 265
  - Revisions to Execution Environment Utilities ... 27
  - Revisions to FORTRAN Development and Runtime Utilities Options ... 269
  - Revisions to General ... 1
  - Revisions to Language-Independent System Services ... 263
  - Revisions to Portability Considerations ... 351
  - Revisions to Sample National Profile ... 353
  - Revisions to Shell Command Language ... 21
  - Revisions to Software Development Utilities Option ... 255
  - Revisions to Terminology and General Requirements ... 5
  - Revisions to User Portability Utilities Option ... 119
  - rewind ... 158, 297, 314
  - RFC ... 204, 271
  - rm ... 101, 274-276
    - Remove directory entries ... 101
    - Description ... 101
    - Exit Status ... 101
    - utility definition ... 101
  - rmdir()* ... 101
- S**
- s ... 159, 239, 314, 345
  - S ... 239, 346
  - Sample National Profile ... 353
  - SAT ... 318
  - SC2 ... 6
  - Scope ... 1-2
  - screenful ... 185-187, 189, 192-195, 327-328
  - scroll ... 323, 336
    - scr ... 175
  - Scroll backwards one-half screenful ... 193
  - Scroll backwards one line ... 193
  - Scroll backwards one screenful ... 192
  - Scroll forwards one-half screenful ... 193
  - Scroll forwards one line ... 192
  - Scroll forwards one screenful ... 192
  - Search backwards for pattern ... 195
  - Search forwards for pattern ... 194
  - sections ... 323
    - sect ... 175
  - sed ... 102-104, 108, 114, 280, 292
    - Stream editor ... 102
    - Addresses ... 102
    - Editing Commands ... 103
    - Extended Description ... 102
    - Input Files ... 102
    - REs ... 103
    - utility definition ... 102
  - session ... 18, 73, 131, 157, 210, 297-298, 304, 314
  - set ... 24, 160, 170, 315
    - Set/unset options and positional parameters ... 24
  - <setjmp.h> ... 265
  - set +x ... 24
  - sh ... 21, 24, 109, 175
    - Shell: the standard command language interpreter ... 109
    - Environment Variables ... 109
    - Operands ... 109
    - utility definition ... 109
  - SHELL**
    - variable ... 128, 175, 257, 260
  - shell ... 130, 160, 210, 227, 315, 324
    - sh ... 175
  - SHELL ... 257
  - shell execution environment ... 37
  - Shell Grammar Rules ... 22
  - Shell Introduction ... 21
  - shift ... 349
  - shiftwidth ... 324
    - sw ... 175
  - showmatch ... 324
    - sm ... 176

- showmode ... 324
    - smd ... 176
  - SIGCONT ... 128, 295
  - SIGHUP ... 126, 129, 131, 179, 206, 252, 295
  - SIGINT ... 46, 128, 294-295, 349
  - <signal.h> ... 265
  - SIGQUIT ... 49, 294
  - SIGTERM ... 129, 295
  - SIGTSTP ... 161, 294
  - SILENT ... 256, 259
  - S\_IRGRP ... 16, 94, 98
  - S\_IROTH ... 16, 94, 98
  - S\_IRUSR ... 16, 94, 98
  - S\_IRWXG ... 16, 68, 73
  - S\_IRWXO ... 16, 68, 73
  - S\_IRWXU ... 16, 68, 73, 99
  - S\_ISGID ... 80, 94, 98
  - S\_ISUID ... 80, 94, 98
  - S\_IWGRP ... 16, 94, 98
  - S\_IWOTH ... 16, 94, 98
  - S\_IWUSR ... 16, 94, 98
  - S\_IXGRP ... 94, 98
  - S\_IXOTH ... 94, 98
  - S\_IXUSR ... 94, 98
  - Skip forwards one line ... 193
  - slowopen ... 176, 324
  - sockets ... 291
  - source ... 126, 161, 295, 299, 302, 315
  - source code ... 5
  - <space> ... 217, 337
  - START ... 349
  - stat() ... 82
  - stat ... 89-90
  - stat() ... 95
  - State-Dependent Character Encodings ... 10
  - <stdio.h> ... 265
  - <stdlib.h> ... 265
  - Strictly Conforming POSIX.2 Application ... 2
  - STRING ... 27, 31, 33
  - <string.h> ... 265
  - strtod() ... 35
  - stty ... 140, 161, 214, 248, 250, 349
  - SUFFIXES ... 259
  - super-user ... 279
  - suspend ... 161, 210, 315
  - SVID3 ... 276-277
  - SVR4 ... 307
  - Symbolic Constants for Portability Specifications ... 18-19
  - Symbolic Limits ... 18
  - symbolic link
    - definition of ... 6
  - SYMLINK\_MAX ... 6
  - sysconf() ... 56
  - system() ... 21, 109, 258
  - System V ... 276, 278, 283, 291-292, 296, 307, 327, 336
  - <sys/types.h> ... 265
- ## T
- t ... 240, 346
  - T ... 240, 346
  - tabstop ... 325
    - ts ... 176
  - tag ... 126, 130, 133, 144, 161, 176, 217, 297, 315-316, 321
    - tags ... 176
  - taglength ... 325
    - tl ... 176
  - ./tags ... 325
  - tags ... 325
  - tail ... 10
  - talk ... 173
  - tar ... 71-72, 278-279, 281-282, 284, 287-288, 290
    - format ... 92
  - Target Rules ... 259
  - TERM**
    - variable ... 125, 128, 177, 189
  - term ... 177, 325
  - terse ... 177, 325
  - test ... 110, 274
    - Evaluate expression ... 110
    - Operands ... 110
    - utility definition ... 110
  - Test Methods ... 3
  - text file ... 84, 102, 116, 126, 129, 202, 293-294
  - THE ... 318
  - <time.h> ... 265
  - /tmp ... 77

**TMPDIR**

variable ... 77, 85  
 TOC ... 1, 5, 21, 27, 119, 255, 263, 265, 267,  
 269, 271, 273, 351, 353, 355  
 touch ... 119, 256  
 tr ... 12, 112, 114  
   – Translate characters ... 111  
   Extended Description ... 111-112  
   Options ... 111  
   Synopsis ... 111  
   utility definition ... 111  
 TRAILER ... 99  
 troff ... 322

**U**

u ... 240, 346  
 U ... 241, 346  
 UCS2 ... 289  
 UCS4 ... 289  
 UCS ... 2, 289  
 umask ... 200  
 unabbrev ... 306  
 unabbreviate ... 138, 162, 301-302, 310,  
 316  
 undefined ... 8-9, 17, 19, 30-31, 33, 36-37,  
 75-77, 93, 99, 103-104, 107, 115-116, 126,  
 146, 187, 190, 196, 206-207, 253, 256, 291  
 undelete ... 64  
 undo ... 153, 162, 174-175, 240, 295, 316,  
 323, 346  
 unexpand ... 179, 199  
   – Convert spaces to tabs ... 199  
   Environment Variables ... 199  
   utility definition ... 199  
 Unicode ... 289  
 <unistd.h> ... 265  
 UNIX ... 275, 277, 294, 296, 304, 322  
 unlink() ... 101  
 unmap ... 138, 152, 163, 301-302  
 unmap! ... 310  
 unmap ... 310, 316  
 unspecified ... 15, 21-22, 24, 29-31, 34-35,  
 37-39, 51, 59-60, 63-65, 68, 70, 73-75, 77,  
 81, 85-86, 94-96, 99-100, 104-107, 109, 113,  
 115, 117-119, 121, 125, 128-129, 132-133,  
 143, 145, 148, 152, 155-157, 161, 163-164,  
 174, 176-177, 183, 189-191, 195-196, 202,  
 207, 210-212, 215, 227, 244-247, 249, 257,  
 260-261, 265-266, 286, 295, 302-303, 310,  
 316, 324-325

**Usage**

Examples ... 118  
 /usr ... 283  
 ustar ... 71, 76, 81, 86-88, 90-92, 283-287,  
 290-291  
   Header Block ... 93  
   Interchange Format ... 92  
   mode Field ... 94  
 UTF8 ... 6, 78, 82, 86, 88-91, 281, 289  
   definition of ... 6  
 utilities  
   ar ... 255  
   at ... 119  
   awk ... 27  
   batch ... 120  
   bc ... 35  
   c89 ... 265  
   cd ... 37  
   chgrp ... 40  
   chmod ... 41  
   chown ... 42  
   cp ... 43  
   csplit ... 121  
   ctags ... 122  
   cut ... 45  
   date ... 46  
   dd ... 46  
   diff ... 47  
   du ... 123  
   ed ... 49  
   ex ... 124  
   expand ... 179  
   expr ... 54  
   file ... 180  
   find ... 54  
   getconf ... 56  
   iconv ... 115  
   ln ... 57  
   localedef ... 58  
   ls ... 60  
   mailx ... 62  
   make ... 256  
   mkdir ... 68  
   mkfifo ... 68  
   more ... 185  
   mv ... 69  
   od ... 70  
   patch ... 197  
   pax ... 71  
   pwd ... 100  
   renice ... 198  
   rm ... 101  
   sed ... 102  
   sh ... 109  
   test ... 110  
   tr ... 111

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.

- unexpand ... 199
  - uudecode ... 200
  - uuencode ... 201
  - vi ... 205
  - write ... 253
  - xargs ... 113
  - yacc ... 266
  - Utility Conventions ... 17
  - Utility Syntax Guidelines ... 17
  - uudecode ... 200-204
    - Decode a binary file ... 200
    - Description ... 200
    - Options ... 200
    - Standard Output ... 201
    - Synopsis ... 200
    - utility definition ... 200
  - uuencode ... 200-202, 204
    - Encode a binary file ... 201
    - Base64 Algorithm ... 202
    - Base64 Values ... 203
    - Description ... 201
    - Historical Algorithm ... 204
    - Operands ... 202
    - Options ... 202
    - Standard Output ... 202
    - Synopsis ... 201
    - utility definition ... 201
- ## V
- ~ ... 229, 341
  - version ... 163, 300, 316
  - vi ... 67, 124-125, 128, 131, 133, 143-144, 149, 152-154, 159, 172, 174-175, 196, 204-210, 212, 216, 225, 227, 230-232, 234, 236, 239, 243, 252, 292-300, 303-304, 306, 308-317, 319-321, 324-325, 327-345, 348-349
  - 0 command ... 226
  - Screen-oriented (visual) display editor ... 204-205, 329
  - a command ... 229
  - A command ... 230
  - A rationale ... 342
  - a rationale ... 342
  - B command ... 230
  - b command ... 230
  - B rationale ... 342
  - b rationale ... 342
  - c command ... 230
  - C command ... 231
  - ! command ... 217
  - \$ command ... 218
  - % command ... 219
  - & command ... 220
  - ' command ... 220
  - ` command ... 220
  - [[ command ... 221
  - ]] command ... 222
  - ^ command ... 222
  - ( command ... 223
  - ) command ... 223
  - command ... 223
  - | command ... 223
  - / command ... 225
  - : command ... 227
  - ; command ... 227
  - < command ... 227
  - ? command ... 228
  - > command ... 228
  - @ command ... 229
  - \_ command ... 223
  - . command ... 224
  - Command Descriptions ... 207, 332
  - <control-B> command ... 212
  - <control-B> rationale ... 335
  - <control-^> command ... 216
  - <control-]> command ... 217
  - <control-D> command ... 213, 247
  - <control-D> rationale ... 336, 348
  - <control-E> command ... 213
  - <control-E> rationale ... 336
  - <control-F> command ... 213
  - <control-F> rationale ... 336
  - <control-G> command ... 214
  - <control-H> command ... 214, 248
  - <control-L> command ... 215
  - <control-L> rationale ... 336
  - <control-P> command ... 215
  - <control-]> rationale ... 337
  - <control-R> command ... 215
  - <control-R> rationale ... 337
  - <control-T> command ... 249
  - <control-T> rationale ... 349
  - <control-U> command ... 216, 250
  - <control-U> rationale ... 337
  - <control-V> command ... 250
  - <control-V> rationale ... 349
  - <control-W> command ... 251
  - <control-Y> command ... 216
  - <control-Y> rationale ... 337
  - C rationale ... 342
  - d command ... 231
  - D command ... 232
  - d rationale ... 342
  - D rationale ... 343
  - E command ... 232
  - e command ... 232
  - E rationale ... 343
  - e rationale ... 343
  - <ESC> command ... 251
  - <ESC> command-mode command ... 216

Copyright © 1999 IEEE. All rights reserved.

This is an unapproved IEEE Standards Draft, subject to change.



- <ESC> rationale ... 337, 349
  - F command ... 233
  - f command ... 233
  - G command ... 233
  - H command ... 234
  - I command ... 234
  - i command ... 234
  - I rationale ... 343
  - i rationale ... 343
  - J command ... 235
  - J rationale ... 343
  - L command ... 235
  - m command ... 235
  - M command ... 236
  - m rationale ... 344
  - N command ... 236
  - n command ... 236
  - <newline> command ... 214, 249
  - <newline> rationale ... 349
  - n rationale ... 344
  - NUL command ... 247
  - O command ... 237
  - o command ... 237
  - O rationale ... 344
  - o rationale ... 344
  - p command ... 237
  - P command ... 238
  - p rationale ... 344
  - P rationale ... 345
  - Q command ... 238
  - rationale ... 329
  - ! rationale ... 338
  - % rationale ... 338
  - & rationale ... 338
  - [ rationale ... 338
  - ( rationale ... 339
  - ) rationale ... 339
  - ] rationale ... 339
  - rationale ... 339
  - / rationale ... 340
  - : rationale ... 340
  - ? rationale ... 341
  - @ rationale ... 341
  - < rationale ... 341, 341
  - . rationale ... 339
  - R command ... 239
  - r command ... 239
  - R rationale ... 345
  - r rationale ... 345
  - S command ... 239
  - s command ... 239
  - <space> command ... 217
  - T command ... 240
  - t command ... 240
  - u command ... 240
  - U command ... 241
  - u rationale ... 346
  - utility definition ... 205
  - ~ command ... 229
  - ~ rationale ... 341
  - w command ... 241
  - W command ... 242
  - W rationale ... 346
  - w rationale ... 346
  - x command ... 242
  - X command ... 243
  - Y command ... 243
  - y command ... 243
  - Y rationale ... 347
  - y rationale ... 347
  - z command ... 244
  - z rationale ... 347
  - ZZ command ... 245
- VISUAL**
- variable ... 67
  - visual ... 124, 137, 139, 142, 149, 163, 174-175, 299, 301, 304, 316, 320, 323
- W**
- w1200 ... 326
  - w300 ... 326
  - w ... 241, 346
  - W ... 242, 346
  - w9600 ... 326
  - warn ... 177, 325
  - wcswidth() ... 9
  - wcwidth() ... 9
  - WG15 ... 118, 201
  - WG2 ... 6
  - WIDTH ... 9
  - WIDTH\_DEFAULT ... 9
  - window ... 325
    - wi ... 177
  - W\_OK ... 174
  - WORD ... 22
  - wq ... 164
  - wrapmargin ... 326
    - wm ... 177
  - wrapscan ... 326
    - ws ... 178
  - write() ... 92
  - write ... 130, 137, 139, 144, 164-165, 172-173, 178, 296, 299, 303, 316
    - Write to another user ... 253
    - Environment Variables ... 253
    - utility definition ... 253

writeany ... 326  
wa ... 178

## X

x ... 242, 346  
X ... 243, 346  
xargs ... 114  
  – Construct argument list(s) and invoke  
    utility ... 113  
  Description ... 113  
  Options ... 113  
  Synopsis ... 113  
  utility definition ... 113  
xit ... 130, 165, 317  
X/Open ... 10  
XPG4 ... 12-13, 46, 114, 118

## Y

y ... 243, 347  
Y ... 243, 347  
yacc ... 266  
  – Yet another compiler compiler ... 266  
  Code File ... 266  
  Programs Section ... 266  
  utility definition ... 266  
yank ... 165, 175, 303, 317, 347  
yesexpr ... 84  
yylex() ... 266

## Z

z ... 166, 244, 317, 347  
ZZ ... 245, 348

