

STANDARDS PROJECT

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

Sponsor

**Technical Committee on Operating Systems
and Application Environments
of the
IEEE Computer Society**

**Work Item Number:
JTC1 22.43**

%

Abstract: IEEE Std 1003.2c is an amendment to IEEE Std 1003.2-1992. It defines security utilities to open systems for access control lists, separation of privilege (capabilities), mandatory access control, and information label mechanisms. | +

Keywords: access control lists, application portability, information labels, mandatory access control, capability, open systems, operating systems, portable application, POSIX, POSIX.2, privilege, security, user portability +

PSSG / D17 October 1997

**Copyright © 1997 by the Institute of Electrical and Electronics Engineers, Inc
345 East 47th Street,
New York, NY 10017, USA
All rights reserved.**

ISBN-xxxx-xxxxx-x

Library of Congress Catalog Number 90-xxxxx

IEEE Draft P1003.2c, Copyright © IEEE.

All Rights Reserved by IEEE.

The IEEE disclaims any responsibility or liability resulting from the placement and use of this document.

This copyrighted document may be downloaded for personal use by one (1) individual user.

No further copying or distribution is permitted without the express written permission or an appropriate license from the IEEE.

This is a withdrawn IEEE Standards Draft.

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
October 1997

XXXXXXX

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Foreword

NOTE: This foreword is not a normative part of the standard and is included for informative purposes only.

The purpose of this standard is to define a standard interface and environment for Computer Operating Systems that require a secure environment. The standard is intended for system implementors and application software developers. It is an extension to the IEEE P1003.2 (POSIX.2).

Organization of the Standard

The standard is divided into several parts:

- Revisions to the General Section (Section 1)
- Revisions to Terminology and General Requirements (Section 2)
- Revisions to Execution Environment Utilities (Section 4)
- Revisions to User Portability Utilities (Section 5)
- Access Control Lists (Section 8)
- Capability (Section 9) %
- Mandatory Access Control (Section 10) %
- Information Labeling (Section 11)
- Annex E - Revisions to Rationale and Notes
- Annex I - Ballot Instructions

Changes to the draft since the previous ballot are indicated by one of four marks in the right-hand margin. These change marks should aid the balloter in determining what has changed and therefore what is candidate text for comments and objections during this ballot. A bar ("|") indicates changes to the line between drafts 15 and 16. A plus ("+") indicates that text has been added in draft 16. A minus ("-") indicates that text present in that location in draft 15 has been deleted in draft 16. A percent ("%") indicates that a change was made at that location in draft 17. %

Conformance Measurement

In publishing this standard, both IEEE and the security working group simply intend to provide a yardstick against which various operating system implementations can be measured for conformance. It is not the intent of either IEEE or the

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

security working group to measure or rate any products, to reward or sanction any vendors of products for conformance or lack of conformance to this standard, or to attempt to enforce this standard by these or any other means. The responsibility for determining the degree of conformance or lack thereof with this standard rests solely with the individual who is evaluating the product claiming to be in conformance with this standard.

Extensions and Supplements to This Standard

Activities to extend this standard to address additional requirements can be anticipated in the future. This is an outline of how these extensions will be incorporated, and also how users of this document can keep track of that status. Extensions are approved as “Supplements” to this document, following the IEEE Standards Procedures. Approved Supplements are published separately and are obtained from the IEEE with orders for this document until the full document is reprinted and such supplements are incorporated in their proper positions.

If you have any questions regarding this or other POSIX documents, you may contact the the IEEE Standards Office by calling IEEE at:

1 (800) 678-IEEE - from within the US
1+ (908) 981-1393 from outside the US

to determine what supplements have been published. Published supplements are available for a modest fee.

Supplements are numbered in the same format as the main document with unique positions as either subsections or main sections. A supplement may include new subsections in various sections of the main document as well as new main sections. Supplements may include new sections in already approved supplements. However, the overall numbering shall be unique so that two supplements only use the same numbers when one replaces the other. Supplements may contain either required or optional facilities. Supplements may add additional conformance requirements (see POSIX.2, Implementation Conformance, 1.3) defining new classes of conforming systems or applications.

It is desirable, but perhaps not avoidable, that supplements do not change the functionality of the already defined facilities. Supplements are not used to provide a general update of the standard. A general update of the standard is done through the review procedure as specified by the IEEE.

If you have interest in participating in Portable Applications Standards Committee (PASC) working groups please send your name, address and phone number to the Secretary, IEEE Standards Board, Institute of Electrical and Electronics Engineers, Inc., P.O. Box 1331, 445 Hoes Lane, Piscataway, NJ 08855-1331, and ask to have your request forwarded to the chairperson of the appropriate PASC working group. If you have interest in participating in this work at the international level, contact your ISO/IEC national body.

Please report typographical errors and editorial changes for this draft standard directly to:

Casey Schaufler
Silicon Graphics
2011 North Shoreline Blvd.
P.O. Box 7311
Mountain View, CA 94039-7311
(415) 933-1634 (voice)
(415) 962-8404 (fax)
casey@sgi.com
Schaufler@DOCKMASTER.NCSC.MIL

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

IEEE Std 1003.2c was prepared by the security Working Group, sponsored by the Portable Applications Standards Committee (PASC) of the IEEE Computer Society.

Standards Subcommittee for PASC

Chair: Lowell Johnson
Treasurer: Barry Needham
Secretary: Charles Severence

1003.6 Working Group Officials

Chair: Lynne Ambuel
Technical Editor: Casey Schaufler

The following people have participated in the security Working Group.

Lynne Ambuel	Jeanne Baccash	Lee Badger
Martin Bailey	John-Olaf Bauner	D. Elliott Bell
Lowell Bogard	Kevin Brady	Joe Brame
Matthew Brisse	Joseph Bulger	Lisa Carnahan
Mark Carson	Charisse Castagnoli	Paul Close
Roland Clouse	Peter E. Cordsen	Janet Cugini
Anthony D'Alessandro	Daniel D. Daugherty	Manilal Daya
Ana Maria De Alvare'	Terence Dowling	Jack Dwyer
Maryland R. Edwards	Ron Elliott	Lloyd English
Jeremy Epstein	Frank Fadden	Kevin Fall
David Ferbrache	Carl Freeman	Mark Funkenhauser
Morrie Gasser	Gerald B. Green	John Griffith
Henry Hall	Craig Heath	Tom Houghton
Rand Hoven	Chris Hughes	Howard Israel
Paul A. Karger	Joseph Keenan	Jerry Keselman
Yvon Klein	Andy Kochis	Steve Kramer
Steven LaFountain	Danielle Lahmani	Jason Levitt
Warren E. Loper	Jeff Mainville	Doug Mansur
Richard E. Mcnaney	Chris Milsom	Mark Modig
Jim Moseman	Kevin V. Murphy	Greg Nuss
Rose Odonnell	Gary Oing	C. Larry Parker
Gordon Parry	Jeff Picciotto	Michael Ressler
David Rogers	Peter L. Rosencrantz	Shawn Rovanseck
Craig Rubin	Roman Saucedo	Stuart Schaeffer
Mark Schaffer	Casey Schaufler	Michael Schmitz
Larry Scott	Eric Shaffer	Olin Sibert
Rick Siebenaler	Alan Silverstein	Jon Spencer
Dennis Steinauer	Chris Steinbroner	Michael Steuerwalt
Doug Steves	Steve Sutton	W. Lee Terrell
Charlie Testa	Jeff Tofano	Brian Weis
Catherine West	Ken Witte	

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Information technology—Portable operating 1 system interface for computer environments

2 Section 1: Revisions to the General Section

3 ⇒ **1.1 Scope** *This scope is to be revised and integrated appropriately into the* %
4 *scope when POSIX.2c is approved:* %

5 This standard, P1003.2c/D17: October 1997 (POSIX.2c), defines four indepen- |
6 dent, security-related, optional sets of utilities. These interfaces will provide %
7 changes and additions to ISO/IEC 9945-2 (Shell and Utilities) as they are pub-
8 lished and approved. The sets of utilities for implementation are:

- 9 (1) Access Control Lists (ACL)
- 10 (2) Capability
- 11 (3) Mandatory Access Controls (MAC)
- 12 (4) Information Labeling (IL)

13 Each option defines new utilities, as well as security-related constraints for the
14 functions and utilities defined by other POSIX standards.

15 ⇒ **1.2 Normative References (POSIX.2: line 92)** *Modify normative reference 8%*
 16 *(IEEE Std 1003.1-1990) to refer to POSIX.1 as amended by P1003.1e.* %

17 ⇒ **1.3.1.3 Conforming Implementation Options (POSIX.2: line 172)** *Insert*
 18 *the following options in alphabetic order:*

19	{POSIX2_ACL}	The system supports the Access Control List Utilities	
20		Option (see Section 8).	
21	{POSIX2_CAP}	The system supports the Capability Utilities Option (see %	
22		Section 9).	
23	{POSIX2_INF}	The system supports the Information Label Utilities	
24		Option (see Section 11).	%
25	{POSIX2_MAC}	The system supports the Mandatory Access Control Utili-	
26		ties Option (see Section 10).	%

1 Section 2: Revisions to Terminology and General Requirements

2 ⇒ 2.2 Definitions

3 ⇒ **2.2.2 General Terms** *Delete 2.2.2.66 file access permissions Modify the con-*
4 *tents of subclause 2.2.2, General Terms, to add or modify the indicated*
5 *definitions in the correct sorted order [disregarding the subclause numbers*
6 *shown here.]*

7 **2.2.2.1 access ACL:** An access control list (ACL) which is used in making discre-
8 tionary access control decisions for an object. [POSIX.1e]

9 **2.2.2.2 access control:** The prevention of unauthorized access to objects by
10 processes and, conversely, the permitting of authorized access to objects by
11 processes. [POSIX.1e]

12 **2.2.2.3 access control list (ACL):** A discretionary access control entity associ-
13 ated with an object, consisting of a list of entries where each entry is a user
14 identifier coupled with a set of access permissions. [POSIX.1e]

15 **2.2.2.4 capability:** An attribute of a process that determines whether or not a
16 process has the appropriate privilege to perform a specific POSIX.1 action where
17 appropriate privilege is required. [POSIX.1e] –

18 **2.2.2.5 capability state:** A grouping of all of the flags defined by an implemen-
19 tation for a capability. [POSIX.1e]

20 **2.2.2.6 default ACL:** An ACL which is used in determining the initial discre-
21 tionary access control information for objects created. [POSIX.1e] –

22 **2.2.2.7 discretionary access control (DAC):** A means of determining and
23 enforcing access to objects based on the identity of the user, process, and/or
24 groups to which the objects belong. The controls are discretionary in the sense
25 that a subject with a certain access permission is capable of passing that

26 permission (perhaps indirectly) on to other subjects. [POSIX.1e]

27 **2.2.2.8 file access controls:** One standard file access control mechanism based
28 on file permission bits and two optional file access control mechanisms, based on
29 access control lists and mandatory access control labels, are defined by this stan-
30 dard.

31 **2.2.2.8.1 file access permissions**

32 This standard defines discretionary file access control on the basis of file permis-
33 sion bits as described below. The additional provision, access control lists, applies
34 only if {_POSIX2_ACL} is defined. The additional provision, mandatory access
35 control, applies only if {_POSIX2_MAC} is defined.

36 The file permission bits of a file contain read, write, and execute/search permis-
37 sions for a file owner class, file group class, and file other class. —

38 Implementations may provide *additional* or *alternate* file access control mechan-
39 isms, or both. An additional access control mechanism shall only further restrict
40 the access permissions defined by the file access control mechanisms described in
41 this section. An alternate access control mechanism shall:

- 42 (1) Specify file permission bits for the file owner class, file group class, and
43 file other class corresponding to the access permissions. —
- 44 (2) Be enabled only by explicit user action, on a per user basis by the file
45 owner or a user with the appropriate privilege.
- 46 (3) Be disabled for a file after the file permission bits are changed for that
47 file with the `chmod` utility. The disabling of the alternate mechanism
48 need not disable any additional mechanisms defined by an implementa-
49 tion.

50 Whenever a process requests file access permission for read, write, or
51 execute/search, if no additional mechanism denies access, access is determined as
52 follows:

53 If the process possesses appropriate privilege:

- 54 — If read, write, or directory search permission is requested, access is
55 granted.
- 56 — If execute permission is requested, access is granted if execute permission
57 is granted to at least one user by the file access permission bits.

58 Otherwise: Access is granted on the basis of the evaluation of the file per-
59 mission bits.

60 ⇒ **2.2.2.8.2 access control lists:** *Add this as a new concept.* %

61 The {_POSIX2_ACL} option provides an additional access control mechanism
62 by providing file access control based upon an access control list mechanism.
63 The additional provisions of this subclause apply only if {_POSIX2_ACL} is
64 defined. The interaction between file permission bits and the ACL mechanism
65 is defined such that a correspondence is maintained between them. The ACL
66 mechanism therefore enhances access control based upon the file permission
67 bits.

68 An ACL entry shall support at a minimum read, write, and execute/search per-
69 missions.

70 An ACL is set at file creation time. An additional *default ACL* can be associ-
71 ated with a directory; this is used in setting the ACL of any object created in
72 that directory. —

73 Each access mode requested shall be individually evaluated against the ACL.
74 A process is granted discretionary access to a file only if all individual
75 requested modes of access are granted or the process possesses appropriate
76 privileges.

77 If the process possesses appropriate privilege:

78 — If read, write or directory search permission is requested, access is granted.

79 — If execute permission is requested, access is granted if execute permission
80 is specified in at least one ACL entry.

81 Otherwise, access is granted on the basis of the evaluation of the ACL per-
82 missions. —

83 ⇒ **2.2.2.8.3 mandatory access control:** *Add this as a new concept.* %

84 The {_POSIX2_MAC} option provides utilities to an additional access control
85 mechanism based on the assignment of MAC labels to subjects and objects.
86 The provisions of this subclause only apply if {_POSIX2_MAC} is defined. |

87 The MAC mechanism permits or restricts access to an object by a process
88 based on a comparison of the MAC label of the process to the MAC label of the
89 object. A process can read an object if the process's MAC label dominates the
90 object's MAC label, and write an object if the process's MAC label is dominated
91 by the object's MAC label. However, an implementation may impose further
92 restrictions, permitting write access to objects only by processes with a MAC
93 label equivalent to that of the object. This standard does not define the domi-
94 nance and equivalence relationships, and thus does not define a particular
95 MAC policy.

96 MAC read access to an object by a process requires that the process's MAC
97 label dominate the object's MAC label or that the process possess appropriate
98 privilege.

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

99 MAC write access to an object by a process requires that the process's MAC
100 label be dominated by the object's MAC label or that the process possess
101 appropriate privilege.

102 Execute/search file access requires MAC read access to the file.

103 The MAC label of an object (including a process object) is set at creation time
104 to dominate the MAC label of the creating process. Although this allows crea-
105 tion of upgraded objects, this standard provides only interfaces which will
106 create objects with MAC labels equivalent to that of the creating process.
107 However, interfaces are provided to allow an appropriately privileged process
108 to upgrade existing objects.

109 ⇒ **2.2.2.8.4 evaluation of file access:** *Add this as a new concept.* —

110 Whenever a process requests file access, if no alternate access control mechan-
111 ism applies, then access shall be granted only if all the applicable POSIX.1
112 access control mechanisms and any additional access control mechanisms
113 grant the access.

114 **2.2.2.9 dominate:** An implementation-defined relation between the values of
115 MAC labels or implementation labels. [POSIX.1e] —

116 **2.2.2.10 file group class:** The property of a file indicating access permissions
117 for a process related to the process's group identification.

118 A process is in the file group class of a file if the process is not in the file owner
119 class and if the effective group ID or one of the supplementary group IDs of the
120 process matches the group ID associated with the file. [POSIX.1e]

121 **2.2.2.11 information label:** The representation of a security attribute of a sub-
122 ject or object that applies to the data contained in that subject or object and is not
123 used for mandatory access control. [POSIX.1e]

124 **2.2.2.12 information label floating:** The operation whereby one information
125 label is combined with another information label. The specific algorithm used to
126 define the result of a combination of two labels is implementation defined.
127 [POSIX.1e]

128 **2.2.2.13 MAC label:** The representation of a security attribute of a subject or
129 object which represents the sensitivity of the subject or object and is used for
130 mandatory access control decisions. The contents of MAC labels are
131 implementation-defined. [POSIX.1e]

- 132 **2.2.2.14 mandatory access control (MAC):** A means of determining and
 133 enforcing access to objects based on an implementation-defined security policy
 134 using MAC labels and the use of the implementation-defined dominate operator.
 135 The determinations are mandatory in the sense that that are always imposed by
 136 the system. [POSIX.1e]
- 137 **2.2.2.15 minimum ACL:** An ACL that contains only the required ACL entries.
 138 [POSIX.1e]
- 139 **2.2.2.16 principle of least privilege:** A security design principle that states
 140 that a process or program be granted only those privileges necessary to accom-
 141 plish its legitimate function, and only for the time that such privileges are actu-
 142 ally required. [POSIX.1e]
- 143 **2.2.2.17 required ACL entries:** The three ACL entries that must exist in every
 144 valid ACL. These entries are exactly one entry each for the owning user, the own-
 145 ing group, and other users not specifically enumerated in the ACL.
- 146 **2.2.2.18 security:** The set of measures defined within a system as necessary to
 147 adequately protect the information to be processed by the system. [POSIX.1e] –
- 148 ⇒ **2.3 Built-In Utilities** *Add the following entry to Table 2-3.*
- 149 getpcap
- 150 ⇒ **2.9 Dependencies on Other Standards**
- 151 ⇒ **2.9.1 Features Inherited From POSIX.1**
- 152 ⇒ **2.9.1.4 File Read, Write, and Creation (POSIX.2: line 3395)** *Replace line*
 153 *3395 in Section 2.9.1.4 with the following:*
- 154 (3) If {_POSIX_ACL} is in effect and {_POSIX_ACL_EXTENDED} is in
 155 effect for the directory that will contain the new file, the ACL shall be
 156 set as described in POSIX.1 {8}, section 23.1.4; otherwise the file per- %
 157 mission bits are set to:

158 ⇒ **2.9.1.4 File Read, Write, and Creation (POSIX.2: line 3403)** *Insert the fol-*
 159 *lowing after line 3403:*

160 (7) If `{_POSIX_CAP}` is in effect and `{_POSIX_CAP_PRESENT}` is in
 161 effect for the directory that will contain the new file, the permitted,
 162 inheritable and effective capability flags for all capabilities defined in
 163 the implementation shall be cleared.

164 (8) If `{_POSIX_INF}` is in effect and `{_POSIX_INF_PRESENT}` is in effect
 165 for the directory that will contain the new file, the information label of
 166 the file shall be set to an implementation-defined value which should
 167 be equivalent to the value returned by the POSIX.1 `{8}` `inf_default()` %
 168 function.

169 (9) If `{_POSIX_MAC}` is in effect and `{_POSIX_MAC_PRESENT}` is in
 170 effect for the directory that will contain the new file, the MAC label of
 171 the file shall be set to the MAC label of the creating process.

172 ⇒ **2.9.3 Concepts Derived from the Security Standard** *Add this as a new*
 173 *section.*

174 Some of the standard utilities specify that a utility performs actions equivalent
 175 to a POSIX.1 function. In POSIX.1e, if `{_POSIX_CAP}` is in effect, many func-
 176 tions are associated with specific capability overrides. The behavior of these
 177 functions is different between processes whose effective flag is set for the
 178 specific capability and processes whose effective flag is clear for the specific
 179 capability. Specific utility actions with respect to capabilities are unspecified
 180 for utilities in POSIX.2. The concept of user authorization to invoke privileged
 181 utility functions is left unspecified in POSIX.1e, and therefore utility enforce-
 182 ment of the authorization mechanism must remain implementation-defined in
 183 POSIX.2c.

184 ⇒ **2.13.2 Symbolic Constants for Portability Specifications (POSIX.2: line**
 185 **4238)** *Insert the following entries in alphabetical order in Table 2-19:*

186 **Table 2-19 - Optional Facility Configuration Values**

188		Name	Description	
189		{POSIX2_ACL}	The system supports the	
190			Access Control List Utilities	
191			Option (see Section 8).	
196		{POSIX2_CAP}	The system supports the	
194			Capability Utilities Option	%
195			(see Section 9).	
200		{POSIX2_INF}	The system supports the	
198			Information Label Utilities	
199			Option (see Section 11).	%
205		{POSIX2_MAC}	The system supports the	
202			Mandatory Access Control	
203			Utilities Option (see Section	%
204			10).	

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

1 **Section 4: Revisions to Execution Environment Utilities**

2 ⇒ **4.13.2 cp — Copy files — Description (POSIX.2: line 2659)** *Change* %
3 *"POSIX.1 {8}" to "POSIX.1 as amended by POSIX.1e {8}" on line 2659 in Sec-* %
4 *tion 4.13.2.*

5 ⇒ **4.13.3 cp — Copy files — Description (POSIX.2: line 2708)** *Replace line*
6 *2708 in Section 4.13.3 with the following:*

- 7 (3) The file permission bits and the S_ISUID and S_ISGID bits. |
8 (4) If {_POSIX_ACL} is defined and {_POSIX_ACL_EXTENDED} is in |
9 effect for the destination file, the ACLs. If this fails for any reason, cp |
10 shall write a diagnostic message to the standard error, do nothing |
11 more with the current *source_file* and go on with any remaining files.

12 ⇒ **4.39.2 ls — List directory contents — Description (POSIX.2: line 6024)**
13 *Insert the following sentence after line 6024 in Section 4.39.6.1:*

14 If {_POSIX_ACL} is defined and {_POSIX_ACL_EXTENDED} is in effect for |
15 the file, then the *<optional alternate access method flag>* shall be a plus sign
16 ("+").

17 ⇒ **4.43.2 mv — Move files — Description (POSIX.2: line 7129)** *Insert the fol-* |
18 *lowing entries after line 7129 in section 4.43.2:* |

- 19 (6) If {_POSIX_ACL} is defined and {_POSIX_ACL_EXTENDED} is in |
20 effect for *dest_file*, then the ACLs associated with the *dest_file* shall |
21 reflect the ACLs associated with the *source_file*. If this fails for any |
22 reason, mv shall write a diagnostic message to the standard error, do |
23 nothing more with the current *source_file* and go on with any remain- |
24 ing files. |
25 (7) If {_POSIX_MAC} is defined and {_POSIX_MAC_PRESENT} is in |
26 effect for *dest_file*, then the MAC label of *dest_file* shall be set to the |
27 MAC label of the invoking process. |

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

28 (8) If {_POSIX_INF} is defined, and {_POSIX_INF_PRESENT} is in effect
29 for *dest_file* the information label of *dest_file* shall dominate the infor-
30 mation label of the source file. |

31 (9) If {_POSIX_CAP} is defined and {_POSIX_CAP_PRESENT} is in effect
32 for *dest_file*, then all capabilities defined by the implementation shall
33 be cleared for *dest_file*.

34 ⇒ **4.48.2 pax — Portable Archive Interchange — Description**
35 **(POSIX.2: line 7671)** *Insert the following text in line 7671 before the word*
36 *"access" in section 4.48.2:*

37 access control lists, |

38 ⇒ **4.48.3 pax — Portable Archive Interchange — Description**
39 **(POSIX.2: line 7774)** *Insert the following after "bits (see 2.2.2.71)," in lines*
40 *7774-7775 in section 4.48.3:*

41 and access control lists, |

42 ⇒ **4.48.3 pax — Portable Archive Interchange — Description**
43 **(POSIX.2: line 7780)** *Insert the following entries after line 7780 in section*
44 *4.48.3:*

45 I Preserve information labels associated with files.

46 M Preserve mandatory access control labels associated with files.

47 C Preserve capability state associated with files.

1 **Section 5: POSIX.2— Revisions to User Portability Utilities**

2 ⇒ **5.2.2 at — Execute utilities at a later time — Description (POSIX.2: line**
3 **87)** *Insert the following after 'mask,' on line 85 in Section 5.2.2:*

4 , the process MAC label (if {_POSIX_MAC} is defined),
5 the process information label (if {_POSIX_INF} is defined), audit ID
6 of the parent job (if {_POSIX_AUD} is defined), the process's capability
7 state (if {_POSIX_CAP} is defined),

8 ⇒ **5.5.2 crontab — Schedule periodic background work — Description**
9 **(POSIX.2: line 489)** *Insert the following after line 489 in Section 5.5.2:*

10 The security attributes of the command executed from the *crontab* entry shall
11 be set as follows:

- 12 • If {_POSIX_MAC} is defined, a separate *crontab* entry shall be maintained |
13 for each MAC label at which the user invokes the *crontab* utility. The
14 MAC label of the environment shall be the MAC label of the process invok-
15 ing the *crontab* utility.
- 16 • If {_POSIX_INF} is defined, the information label of the environment shall |
17 be the information label of the process invoking the *crontab* utility.
- 18 • If {_POSIX_AUD} is defined, the audit ID of the environment shall be the |
19 audit ID of the process invoking the *crontab* utility.
- 20 • If {_POSIX_CAP} is defined, the value of the inheritable capability flags in |
21 the environment shall be implementation-defined.

22 Additional implementation-defined restrictions may be imposed when the periodi-
23 cally scheduled job is executed.

1

Section 8: Access Control Lists

2 This section describes utilities for the retrieval, modification, and manipulation of
3 access ACLs and default ACLs on specified objects.

4 Support for the utilities defined in this section is optional but shall be provided by
5 any implementation claiming conformance to the Access Control List Utilities
6 Option. Such an implementation shall provide all of the utilities as described in
7 this section.

8 8.1 getfacl — Get ACL Information

9 8.1.1 Synopsis

10 `getfacl [-d] [file...]` %

11 8.1.2 Description

12 The `getfacl` utility writes discretionary access control information associated
13 with the specified file(s) to standard output. If the `getconf` utility indicates that
14 `{_POSIX_ACL_EXTENDED}` is not in effect for a *file* then the standard discretion-
15 ary access permissions are interpreted as an ACL containing only the required
16 ACL entries.

17 8.1.3 Options

18 The `getfacl` utility shall conform to the utility argument syntax guidelines
19 described in 2.10.2.

20 `-d` The operation applies to the default ACL of a directory instead of the
21 access ACL. An error shall be generated if a default ACL cannot be
22 associated with *file*.

23 **8.1.4 Operands**

24 The following operand shall be supported by the implementation:

25 *file* A pathname of a file whose ACL shall be retrieved. If *file* is not
26 specified, or a *file* is specified as “–”, then `getfacl` shall read a list
27 of pathnames, each terminated by one <newline> character, from the
28 standard input. If a pathname read from standard input contains
29 only a <newline> character, the results are unspecified. –

30 **8.1.5 External Influences**

31 **8.1.5.1 Standard Input**

32 If no *file* operand is specified, or a *file* is specified as a “–”, then `getfacl` shall
33 read a list of zero or more pathnames from standard input. Otherwise, standard
34 input shall not be used.

35 **8.1.5.2 Input Files**

36 None.

37 **8.1.5.3 Environment Variables**

38 The following environment variables shall affect the execution of `getfacl`:

39 **LANG** This variable shall determine the locale to use for the locale
40 categories when both **LC_ALL** and the corresponding
41 environment variable (beginning with **LC_**) do not specify a
42 locale. See 2.6.

43 **LC_ALL** This variable shall determine the locale to be used to over-
44 ride any values for locale categories specified by the settings
45 of **LANG** or any environment variables beginning with **LC_**.

46 **LC_CTYPE** This variable shall determine the locale for this interpreta-
47 tion of sequences of bytes of text data as characters (e.g.,
48 single- versus multibyte characters in arguments and stan-
49 dard input).

50 **LC_MESSAGES** This variable shall determine the language in which mes-
51 sages should be written.

52 **8.1.5.4 Asynchronous Events**

53 Default.

54 8.1.6 External Effects

55 8.1.6.1 Standard Output

56 The `getfacl` utility writes to standard output a header followed by ACL entries
57 in the form described in 8.1.7. The ACL entries shall be written in the order in %
58 which they are evaluated when a discretionary access check is performed. If the
59 `-d` option is specified and no default ACL is associated with a *file*, then only the
60 header shall be written for that file.

61 The header shall be written in the following format:

62 "#file:%s\n#owner:%d\n#group:%d\n",<filename>,<uid>,<gid> |

63 Additional implementation-defined lines starting with a number sign (#) charac- -
64 ter may be added to the header after the lines specified above.

65 If more than one ACL is written to standard output, an empty line shall be writ-
66 ten to standard output before each header except the first. %

67 8.1.6.2 Standard Error

68 Used only for diagnostic messages.

69 8.1.6.3 Output Files

70 None.

71 8.1.7 Extended Description

72 The `getfacl` utility shall write ACL entries in the following form: %

73 <acl_entry> %

74 [<acl_entry>] ... %

75 Each <acl_entry> line shall contain one ACL entry with three required colon- %
76 separated fields: an ACL entry tag type, an ACL entry qualifier, and the discre- %
77 tionary access permissions. An implementation may define additional colon- %
78 separated fields after the required fields. Comments may be included on any %
79 <acl_entry> line. If a comment starts at the beginning of a line, then the entire %
80 line shall be interpreted as a comment. %

81 The first field contains the ACL entry tag type. This standard defines the follow- %
82 ing ACL entry tag type keywords, one of which shall appear in the first field: %

83 user A user ACL entry specifies the access granted to either the file %
84 owner or a specified user. %

85 group An group ACL entry specifies the access granted to either the file %
86 owning group or a specified group. %

87 other An other ACL entry specifies the access granted to any process %
88 that does not match any user, group, or implementation-defined %

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

89 ACL entries. %

90 mask A mask ACL entry specifies the maximum access which can be %
91 granted by any ACL entry except the user entry for the file owner %
92 and the other entry. %

93 An implementation may define additional ACL entry types. %

94 The second field contains the ACL entry qualifier (referred to in the remainder of %
95 this section as qualifier). This standard defines the following qualifiers: %

96 uid This qualifier specifies a user name or a user ID number. %

97 gid This qualifier specifies a group name or a group ID number. %

98 empty This qualifier specifies that no uid or gid information is to be applied %
99 to the ACL entry. An empty qualifier shall be represented by an %
100 empty string or by white space. %

101 An implementation may define additional qualifiers. %

102 The third field contains the discretionary access permissions. This standard %
103 defines the following symbolic discretionary access permissions: %

104 r Read access %

105 w Write access %

106 x Execute/search access %

107 – No access by this ACL entry. %

108 The discretionary access permissions field shall contain exactly one each of the %
109 following characters in the following order: r, w, and x. Each of these may be %
110 replaced by the “-” character to indicate no access. An implementation may define %
111 additional characters following the required characters that represent %
112 implementation-defined permissions. %

113 A user entry with an empty qualifier shall specify the access granted to the file %
114 owner. A user entry with a uid qualifier shall specify the access permissions %
115 granted to the user name matching the uid value. If the uid value does not match %
116 a user name, then the ACL entry shall specify the access permissions granted to %
117 the user ID matching the numeric uid value. %

118 A group entry with an empty qualifier shall specify the access granted to the file %
119 owning group. A group entry with a gid qualifier shall specify the access permis- %
120 sions granted to the group name matching the gid value. If the gid value does not %
121 match a group name, then the ACL entry shall specify the access permissions %
122 granted to the group ID matching the numeric gid value. %

123 The mask and other entries shall contain an empty qualifier. An implementa- %
124 tion may define additional ACL entry types that use the empty qualifier. %

125 A number-sign (#) starts a comment on an <acl_entry> line. A comment may start %
126 at the beginning of a line, after the required fields and after any implementation- %
127 defined, colon-separated fields. The end of the line denotes the end of the com- %
128 ment. %

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

129 If an ACL entry contains permissions that are not also contained in the mask %
 130 entry, then the output text form for that *<acl_entry>* line shall be displayed as %
 131 described above followed by a number-sign (#), the string "effective: ", and the %
 132 effective access permissions for that ACL entry. %

133 White space is permitted in *<acl_entry>* lines as follows: at the start of the line; %
 134 immediately before and after a ":" separator; immediately before the first %
 135 number-sign (#) character; at any point after the first number-sign (#) character. %

136 Comments shall have no effect on the discretionary access check of the object with%
 137 which they are associated. An implementation shall define whether or not com- %
 138 ments are stored with an ACL. %

139 If an implementation allows the colon character ":" to be present in an ACL entry %
 140 qualifier, then that implementation shall provide a method for distinguishing %
 141 between a colon character as a field separator in an ACL entry definition and a %
 142 colon character as a component of the ACL entry qualifier value.

143 **8.1.8 Exit Status**

144 The `getfacl` utility shall exit with one of the following values:

145	0	The ACL for the specified file(s) was successfully retrieved and written
146		to standard output.
147	>0	An error occurred.

148 **8.1.9 Consequence of Errors**

149 Default.

150 **8.2 setfacl — Set Access Control List**

151 **8.2.1 Synopsis**

152 `setfacl [-bdkn] [-m entries] [-M file1] [-x entries] [-X file2] [file...] %`

153 **8.2.2 Description**

154 The `setfacl` utility changes discretionary access control information associated
 155 with the specified file(s).

156 8.2.3 Options

157 The `setfacl` utility shall conform to the utility argument syntax guidelines
158 described in 2.10.2.

- 159 **-b** Remove all entries except the three required base entries.
- 160 **-d** The operation applies to the default ACL instead of the access
161 ACL. With this option, any *file* arguments must refer to files that
162 may have default ACLs (e.g., directories).
- 163 **-k** Delete any default ACLs on the specified files. It shall not be con-
164 sidered an error if one or more specified files could, but do not,
165 have a default ACL. An error shall be reported if one or more
166 specified files cannot have a default ACL. The **-k** option does not
167 require that the **-d** option be specified. If the **-k** option is
168 specified, but the **-d** option is not specified, then all other options
169 apply to the access ACL and not to the default ACL.
- 170 **-m entries** Modify the access or default ACL by adding new entries and
171 updating existing entries with the entries specified in *entries*. The
172 *entries* option argument is a list of comma-separated ACL entries. |
173 Each ACL entry shall be in the form described in 8.2.7.1. Permis- %
174 sions in each ACL entry shall be specified by either an absolute
175 value or a relative value. See 8.2.7 for a discussion of how absolute%
176 values and relative values are used.
- 177 **-M file1** Modify the access or default ACL by adding new entries and updat-
178 ing existing entries with the entries specified in the pathname
179 *file1*. If *file1* is specified as “-”, `setfacl` will read entries from the |
180 standard input.
- 181 **-n** Do not recalculate the permissions associated with the ACL mask
182 entry.
- 183 **-x entries** Remove the ACL entries specified in *entries* from the access or
184 default ACL of the specified files. The *entries* option argument is a
185 list of comma-separated ACL entries. Each ACL entry shall be in
186 the form described in 8.2.7.1. The permissions field and the %
187 preceding colon separator may be omitted from each ACL entry
188 specified in *entries*. If the permissions field is provided, then the
189 value of the permissions field shall be ignored during the process-
190 ing of the **-x** option.
- 191 **-X file2** Remove the ACL entries specified in the pathname *file2* from the
192 access or default ACL on the specified files. If *file2* is specified as
193 “-”, `setfacl` will read entries from the standard input. The per- |
194 missions field and the preceding colon separator may be omitted
195 from each ACL entry specified in *file2*. If the permissions field is
196 provided, then the value of the permissions field shall be ignored
197 during the processing of the **-X** option.

198 The `-b`, `-k`, `-m`, `-M`, `-x` and `-X` options shall be evaluated in the order in which
199 they are specified in the invocation of the utility.

200 **8.2.4 Operands**

201 The following operand shall be supported by the implementation:

202 *file* A pathname of a file on which the specified actions are performed.

203 **8.2.5 External Influences**

204 **8.2.5.1 Standard Input**

205 If no *file* operands are specified or if a *file* operand of “-” is specified, `setfacl` |
206 shall read the list of pathnames on which to operate from standard input. The
207 format used shall be:

208 “%s\n”, <*file*>

209 The results are unspecified if a pathname read from standard input contains a -
210 <newline> character.

211 If `-M file1` or `-X file2` are specified and either *file1* or *file2* is “-”, ACL entries are
212 read from standard input as specified in 8.2.5.2. %

213 Standard input shall not be read if a file operand is specified, no *file* operand is
214 “-”, no *file1* option argument is “-”, and no *file2* option argument is “-”.

215 The results are unspecified if more than one reference is made to standard input
216 by having no *file* operands, a *file* operand of “-”, a *file1* option argument of “-”, and
217 a *file2* option argument of “-”.

218 **8.2.5.2 Input Files**

219 When the `-M` option is specified, it shall be followed by a *file1* argument. The file |
220 specified by the *file1* operand shall contain one or more text form representations |
221 of an ACL entry. Each text form representation of an ACL entry shall be in the %
222 form described in 8.1.7. ACL entries shall be separated by <newline>. The
223 specified ACL entries shall be added to or updated in the access or default ACL(s)
224 on the specified file(s). The permissions of each ACL entry shall be specified by
225 either an absolute value or a relative value.

226 When the `-X` option is specified, it shall be followed by a *file2* argument. The file |
227 specified in the *file2* operand shall contain one or more text form representations |
228 of an ACL entry. Each text form representation of an ACL entry shall be in the %
229 form described in 8.1.7. ACL entries shall be separated by <newline>. The
230 specified ACL entries shall be removed from the access or default ACL(s) on the
231 specified file(s).

232 **8.2.5.3 Environment Variables**

233 The following environment variables shall affect the execution of `setfacl`:

234 **LANG** This variable shall determine the locale to use for the locale
235 categories when both **LC_ALL** and the corresponding
236 environment variable (beginning with **LC_**) do not specify a
237 locale. See 2.6.

238 **LC_ALL** This variable shall determine the locale to be used to over-
239 ride any values for locale categories specified by the settings
240 of **LANG** or any environment variables beginning with **LC_**.

241 **LC_CTYPE** This variable shall determine the locale for this interpreta-
242 tion of sequences of bytes of text data as characters (e.g.,
243 single- versus multibyte characters in arguments and input
244 files).

245 **LC_MESSAGES** This variable shall determine the language in which mes-
246 sages should be written.

247 **8.2.5.4 Asynchronous Events**

248 Default.

249 **8.2.6 External Effects**

250 **8.2.6.1 Standard Output**

251 None.

252 **8.2.6.2 Standard Error**

253 Used only for diagnostic messages.

254 **8.2.6.3 Output Files**

255 None.

256 **8.2.7 Extended Description**

257 In all cases, if the resulting access or default ACL would not be valid, then the –
258 utility shall fail for the current file and the access ACL, the default ACL, and the
259 file permission bits shall not be changed. This validity check is not performed
260 until all operations indicated by the specified options have been completed, i.e., at
261 any interim point in the manipulation of the ACL, the internal form of an ACL
262 may be "ill-formed," but it must be valid when the manipulations have been com-
263 pleted.

264 Two ACL entries shall be considered to match if their tag types are equal and
 265 their tag qualifiers are equal.

266 If the `-b` option is specified, then all entries other than the three required base
 267 entries shall be removed from the ACL. If the ACL contains a `mask` entry, then
 268 the permissions associated with the owning `group` entry in the resulting ACL
 269 shall be set to only those permissions associated with both the owning `group`
 270 entry and the `mask` entry of the current ACL.

271 If the `-m` option is specified, the access or default ACL associated with a file
 272 operand shall be modified by adding new entries and updating existing entries
 273 with each of the ACL entries in the ACL representation specified by *entries*.

274 If the `-M` option is specified, the access or default ACL associated with a file
 275 operand shall be modified by adding new entries and updating existing entries
 276 with each of the ACL entries contained within *file1*.

277 For both the `-m` and `-M` options, if permissions are specified by an absolute value
 278 and a matching entry is found, the entire new entry, including permissions, shall
 279 replace the current matched entry. If permissions are specified by an absolute
 280 value and a matching entry is not found in the ACL, then the entire new entry
 281 shall be added to the ACL. If permissions are specified by a relative value and a
 282 matching entry is found, the new permissions shall be computed by adding or
 283 removing the relative permissions to or from, as appropriate, the permissions in
 284 the matching entry. Permissions which are specified to be removed and which are
 285 not contained in the permissions of the matching ACL entry shall have no effect
 286 on the resulting permissions in the entry. The entire new entry, including the
 287 computed permissions, shall replace the current matched entry. If permissions
 288 are specified by a relative value and a matching entry is not found in the ACL,
 289 then the new entry containing only those permissions specifically granted by the
 290 relative value shall be added to the ACL. If no permissions are specified as being
 291 added to the entry or if the relative value specifies only the removal of permis-
 292 sions, then the new entry containing no permissions shall be added to the ACL.

293 For both the `-m` and `-M` options, if a `mask` entry is specified, then the permis-
 294 sions of the `mask` entry in the resulting ACL shall be set to the permissions in
 295 the specified ACL `mask` entry. If no `mask` entry is specified and the `-n` option is
 296 not specified, then the permissions of the resulting ACL `mask` entry shall be set
 297 to the union of the permissions associated with all entries which belong to the file
 298 group class in the resulting ACL after all `-b`, `-k`, `-m`, `-M`, `-x`, and `-X` operations
 299 have been performed. If no `mask` entry is specified and the `-n` option is specified,
 300 then the permissions of the resulting ACL `mask` entry shall remain unchanged
 301 from the existing ACL(s) associated with the *file* operands. If no `mask` entry is
 302 specified, the `-n` option is specified, and no ACL `mask` entry exists in the ACL
 303 associated with a file operand, then the `setfacl` utility shall write an error mes-
 304 sage to standard error and continue with the next file.

305 If the `-x` option is specified, those entries in the access or default ACL associated
 306 with a file operand which match entries in the ACL representation specified by
 307 *entries* shall be removed.

308 If the `-X` option is specified, those entries in the access or default ACL associated
 309 with a file operand which match ACL entries contained within *file2* shall be
 310 removed.

311 For both the `-x` and `-X` options, if a `mask` entry is specified, then the `mask` entry
 312 shall be removed from the existing ACL. If no `mask` entry is specified and the `-n`
 313 option is not specified, then the permissions of the resulting ACL `mask` entry
 314 shall be set to the union of the permissions associated with all entries which
 315 belong to the file group class in the resulting ACL after all `-b`, `-k`, `-m`, `-M`, `-x`, and
 316 `-X` operations have been performed. If no `mask` entry is specified and the `-n`
 317 option is specified, then the permissions of the resulting ACL `mask` entry shall
 318 remain unchanged in the ACL(s) associated with the *file* operands. %

319 **8.2.7.1 ACL Text Format** %

320 For both the `-m` and `-x` options, the `getfacl` utility shall accept a list of ACL %
 321 entries in the following form: %

322 `<acl_entry>[,<acl_entry>]...` %

323 Each `<acl_entry>` shall contain one ACL entry, as defined in 8.1.7, with two excep-%
 324 tions. %

325 The ACL entry tag type keyword shall appear in the first field in either its full %
 326 unabbreviated form or its single letter abbreviated form. The abbreviation for %
 327 user is “u”, the abbreviation for group is “g”, the abbreviation for other is “o”, %
 328 and the abbreviation for mask is “m”. An implementation may define additional %
 329 ACL entry tag type abbreviations. %

330 There are no exceptions for the second field in the short text form for ACLs. %

331 The discretionary access permissions shall appear in the third field. The symbolic %
 332 string shall contain at most one each of the following characters in any order: `r`, %
 333 `w`, and `x`; implementations may define additional characters that may appear in %
 334 any order within the string.

335 **8.2.8 Exit Status**

336 The `setfacl` utility shall exit with one of the following values:

337 0 Executed successfully and all requested changes were made.

338 >0 An error occurred.

339 **8.2.9 Consequence of Errors**

340 Default.

1

Section 9: Capability

%

2 This section describes utilities for the retrieval, modification, and manipulation of
3 the capability state of files, and the retrieval of the capability state for processes.

4 Three utilities are specified to support capability operations: `getfcap`,
5 `getpcap`, and `setfcap`.

6 Support for the utilities defined in this section is optional but shall be provided by
7 any implementation claiming conformance to the Capability Utilities Option.
8 Such an implementation shall provide all of the utilities as described in this sec-
9 tion.

10 9.1 `getfcap` — Get the Capability State of a File

11 9.1.1 Synopsis

12 `getfcap [-m | -M flag_spec] [target...]` %

13 9.1.2 Description

14 The `getfcap` utility writes the capability state of the specified target files to stan-
15 dard output.

16 9.1.3 Options

17 The `getfcap` utility shall conform to the utility argument syntax guidelines
18 described in 2.10.2.

19 The following options shall be supported by the implementation:

20 `-m` Produce output for only those capabilities that have at least one
21 flag set. The default is to produce output for all capabilities
22 defined by the implementation.

23 `-M flag_spec` Produce output only for those capabilities that have at least one
24 of the flags specified in *flag_spec* set. The default is to produce
25 output for all capabilities defined by the implementation.
26 *flag_spec* contains one or more character(s), each of which
27 represents a capability flag defined in the implementation:

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

28 e Specifies the effective capability flag.
 29 i Specifies the inheritable capability flag.
 30 p Specifies the permitted capability flag.
 31 Future revisions of this standard may use other lowercase
 32 letters in the portable filename character set in *flag_spec*.
 33 Uppercase letters in the portable filename character set are
 34 reserved for implementations to refer to implementation-defined
 35 capability flags.

36 **9.1.4 Operands**

37 The following operand shall be supported by the implementation:

38 *target* *Target* represents the pathname(s) of the file(s) whose capability
 39 state shall be displayed. If no *target* is specified or if a target
 40 operand is “–”, *getfcap* shall read a list of zero or more path-
 41 names, each terminated by one <newline> character, from stan-
 42 dard input. If a pathname read from standard input contains
 43 only a <newline> character, the results are unspecified. This list
 44 of pathnames read from standard input shall be terminated by
 45 end-of-file (EOF).

46 **9.1.5 External Influences**

47 **9.1.5.1 Standard Input**

48 The standard input shall be used only if no *target* operands are specified, or if a
 49 *target* operand is “–”. If the use of standard input is specified, *getfcap* shall
 50 read a list of zero or more pathnames from standard input.

51 **9.1.5.2 Input Files**

52 None.

53 **9.1.5.3 Environment Variables**

54 The following environment variables shall affect the execution of *getfcap*:

55 **LANG** This variable shall determine the locale to use for the locale
 56 categories when both **LC_ALL** and the corresponding
 57 environment variable (beginning with **LC_**) do not specify a
 58 locale. See 2.6.

83 9.1.8 Exit Status

84 The `getfcap` utility shall exit with one of the following values:

- | | | | |
|----|----|---|--|
| 85 | 0 | The capability state of the specified target was successfully reported. | |
| 86 | >0 | An error occurred. | |

87 9.1.9 Consequence of Errors

88 The file pathname that was being examined when the error occurred shall be
89 written to the error output file along with a brief error message. Processing of
90 that target shall immediately be terminated, and the command shall continue
91 with the next target, if specified.

92 9.2 getpcap — Get the Capability State of a Process

93 9.2.1 Synopsis

94 `getpcap [-m | -M flag_spec]` %

95 9.2.2 Description

96 The `getpcap` utility writes the capability state of the invoking process to stan-
97 dard output.

98 9.2.3 Options

99 The `getpcap` utility shall conform to the utility argument syntax guidelines
100 described in 2.10.2.

101 The following options shall be supported by the implementation:

102	-m	Produce output for only those capabilities that have at least one
103		flag set, and for only those capability attributes that are set. The
104		default is to produce output for all capabilities defined by the
105		implementation.

106	-M <i>flag_spec</i>	Produce output only for those capabilities that have at least one
107		of the flags specified in <i>flag_spec</i> set. The default is to produce
108		output for all capabilities defined by the implementation.
109		<i>flag_spec</i> contains one or more character(s), each of which
110		represents a capability flag defined in the implementation:

111	e	Specifies the effective capability flag.
-----	---	--

112	i	Specifies the inheritable capability flag.
-----	---	--

113 p Specifies the permitted capability flag.
 114 Future revisions of this standard may use other lowercase
 115 letters in the portable filename character set in *flag_spec*.
 116 Uppercase letters in the portable filename character set are
 117 reserved for implementations to refer to implementation-defined
 118 capability flags.

119 **9.2.4 Operands**

120 None.

121 **9.2.5 External Influences**

122 **9.2.5.1 Standard Input**

123 None.

124 **9.2.5.2 Input Files**

125 None.

126 **9.2.5.3 Environment Variables**

127 The following environment variables shall affect the execution of `getpcap`:

128	LANG	This variable shall determine the locale to use for the locale categories when both LC_ALL and the corresponding environment variable (beginning with LC_) do not specify a locale. See 2.6.
-----	-------------	---

129		
130	LC_ALL	This variable shall determine the locale to be used to override any values for locale categories specified by the settings of LANG or any environment variables beginning with LC_ .
131		

132	LC_CTYPE	This variable shall determine the locale for this interpretation of sequences of bytes of text data as characters (e.g., single versus multibyte characters in arguments).
133		
134		

135	LC_MESSAGES	This variable shall determine the language in which messages should be written.
136		
137		
138		
139		

140 **9.2.5.4 Asynchronous Events**

141 Default.

142 **9.2.6 External Effects**

143 **9.2.6.1 Standard Output**

144 The `getpcap` utility writes to standard output the textual representation of the |
145 capability state of the invoking process in the text form specified in 9.1. The fol- %
146 lowing format shall be used:

147 "%s:\n", <*process_cap*>

148 **9.2.6.2 Standard Error**

149 Used only for diagnostic messages.

150 **9.2.6.3 Output Files**

151 None.

152 **9.2.7 Extended Description**

153 None.

154 **9.2.8 Exit Status**

155 The `getpcap` utility shall exit with one of the following values:

156 0 The capability state of the specified target(s) was(were) successfully
157 reported.

158 >0 An error occurred.

159 **9.2.9 Consequence of Errors**

160 None.

161 **9.3 setfcap — Set Capability State of a File**

162 **9.3.1 Synopsis**

163 `setfcap -e state [-e state | -f state_file] ... [target...]` %

164 `setfcap -f state_file [-e state | -f state_file] ... [target...]` %

165 `setfcap state [target...]`

166 9.3.2 Description

167 The `setfcap` utility changes the capability state associated with the specified
168 file(s). The state to be set is defined by the `-e` option, the `-f` option, or the *state*
169 operand. If no `-e` options and no `-f` options are specified, the last form is assumed.%
170 The *state* operand's value consists of one or more capabilities as defined by the
171 text form representation specified in 9.1. If a complete capability state is %
172 specified, that capability state completely replaces any existing capability state on
173 the file. If only a partial capability state is specified, the old capability state will
174 be a modified to reflect the specification.

175 9.3.3 Options

176 The `setfcap` utility shall conform to the utility argument syntax guidelines
177 described in 2.10.2.

178 `-e state` Specify a partial or complete capability state, consisting of one or
179 more capability specifications, to be assigned to each of the
180 target(s). Capability specifications in *state* shall be separated by
181 a `<comma>` or `<newline>`. A null capability specification can
182 be specified by two adjacent `<newline>`s or `<comma>`s in *state*.
183 Null capability specifications shall be ignored. The capability
184 specification is described by the text form representation
185 specified in 9.1. %

186 `-f state_file` Read one or more capability specifications that represent a par-
187 tial or complete capability state from the file named by the path-
188 name *state_file*. Capability specifications in *state_file* shall be
189 terminated by a `<newline>`. A null capability specification can
190 be specified by an empty line in *state_file*. Null capability
191 specifications shall be ignored. If *state_file* is specified as a dash
192 “`-`”, `setfcap` shall read capability specifications from the stan-
193 dard input file. +

194 Multiple `-e` and `-f` options shall be accepted by the `setfcap` utility, in which +
195 case the capability specifications shall be processed in the order they are specified.

196 9.3.4 Operands

197 The following operands shall be defined by the implementation: |

198 *state* *State* represents the partial or complete capability state to be |
199 assigned. Capability specifications in *state* shall be separated by|
200 a `<comma>` or `<newline>`. A null capability specification can |
201 be specified by two adjacent `<newline>`s or `<comma>`s in *state*.|
202 Null capability specifications shall be ignored. The capability |
203 specification is described by the text form representation |
204 specified in 9.1. %

205 *target* *Target* represents the name(s) of the file(s) whose capability
 206 state is (are) to be modified. If no *target* is specified, or is “–”,
 207 and no *state_file* argument is specified as a dash “–”, *setfcap*
 208 shall read a list of pathnames, separated by one or more white
 209 space characters, from the standard input file, terminated by
 210 end-of-file (EOF).

211 **9.3.5 External Influences**

212 **9.3.5.1 Standard Input**

213 The *setfcap* utility shall read capability state information from standard input
 214 if a *state_file* option argument is “–” (see 9.4.3) and shall read pathnames from %
 215 standard input if a *target* operand is “–” or if no *target* operands are specified (see %
 216 9.4.4). Otherwise, standard input shall not be read. The results are unspecified if |
 217 more than one reference is made to standard input by having “–” as a statefile |
 218 option argument, having no operands, or by having “–” as a target operand.

219 **9.3.5.2 Input Files**

220 The *state_file* option argument shall be interpreted as the pathname of a file that
 221 contains a set of correct external representations of capability states, as described
 222 by the text form representation specified. –

223 **9.3.5.3 Environment Variables**

224 The following environment variables shall affect the execution of *setfcap*:

225 LANG	This variable shall determine the locale to use for the locale
226	categories when both LC_ALL and the corresponding
227	environment variable (beginning with LC_) do not specify a
228	locale. See 2.6.
229 LC_ALL	This variable shall determine the locale to be used to over-
230	ride any values for locale categories specified by the settings
231	of LANG or any environment variables beginning with LC_ .
232 LC_CTYPE	This variable shall determine the locale for this interpreta-
233	tion of sequences of bytes of text data as characters (e.g.,
234	single versus multibyte characters in arguments).
235 LC_MESSAGES	This variable shall determine the language in which mes-
236	sages should be written.

237 **9.3.5.4 Asynchronous Events**

238 Default.

239 **9.3.6 External Effects**

240 **9.3.6.1 Standard Output**

241 None.

242 **9.3.6.2 Standard Error**

243 Used only for diagnostic messages.

244 **9.3.6.3 Output Files**

245 None.

246 **9.3.7 Extended Description**

247 None.

248 **9.3.8 Exit Status**

249 The `setfcap` utility shall exit with one of the following values:

250 0 The specified capability state changes were successfully made.

251 >0 An error occurred.

252 **9.3.9 Consequences of Errors**

253 In the event of an error, the capability state of the target that caused the error
254 shall not be modified, and `setfcap` shall continue on to the next file. Each diag-
255 nostic message indicating that the capability state of a target could not be
256 changed shall include the name of the target.

1

Section 10: Mandatory Access Control

%

2 This section describes the utilities that shall be implemented on all systems that
3 claim conformance to the Mandatory Access Control Utilities Option.

4 Support for the utilities defined in this section is optional but shall be provided by
5 any implementation claiming conformance to the Mandatory Access Control Utilities
6 Option. Such an implementation shall provide all of the utilities as described
7 in this section.

8 Three utilities are specified to support mandatory access control. The `getfmac`
9 utility provides the means for a user to display the MAC label of a file. The `getp-`
10 `mac` utility provides the means for a user to display the MAC label of the current
11 process. The `setfmac` utility provides the means to set the MAC label of a file.

12 10.1 `getfmac` — Get the MAC Label of a File

13 10.1.1 Synopsis

14 `getfmac` [*file...*]

15 10.1.2 Description

16 The `getfmac` utility writes to standard output the text form of MAC labels. For
17 each *file* operand, the `getfmac` utility shall perform the equivalent to the
18 POSIX.1e `mac_get_file()` and `mac_to_text()` functions, and write the returned text
19 string to standard output.

20 The `getfmac` utility requires mandatory read access to each file for which the
21 label has been requested.

22 10.1.3 Options

23 None.

24 10.1.4 Operands

25 The following operand shall be supported by the implementation: |

26 *file* The pathname of a file whose MAC label is to be written. If *file* is not
27 specified, or a *file* is specified as a “–”, then `getfmac` shall read a list of
28 pathnames, each terminated by one <newline> character, from the stan-
29 dard input. –

30 10.1.5 External Influences

31 10.1.5.1 Standard Input

32 If no *file* operand is specified, or a *file* is specified as a “–”, then `getfmac` will
33 accept a list of zero or more pathnames, each terminated by one <newline> char-
34 acter, from the standard input.

35 10.1.5.2 Input Files

36 None.

37 10.1.5.3 Environment Variables

38 The following environment variables shall affect the execution of `getfmac`:

39 **LANG** This variable shall determine the locale to use for the locale
40 categories when both **LC_ALL** and the corresponding
41 environment variable (beginning with **LC_**) do not specify a
42 locale. –

43 **LC_ALL** This variable shall determine the locale to be used to over-
44 ride any values for locale categories specified by the settings
45 of **LANG** or any environment variables beginning with **LC_**.

46 **LC_CTYPE** This variable shall determine the locale for this interpreta-
47 tion of sequences of bytes of text data as characters (e.g.,
48 single- versus multibyte characters in arguments and stan- |
49 dard input).

50 **LC_MESSAGES** This variable shall determine the language in which mes- |
51 sages should be written.

52 10.1.5.4 Asynchronous Events

53 Default.

54 **10.1.6 External Effects**

55 **10.1.6.1 Standard Output**

56 The following format shall be used for each file processed: %

57 "%s:\t%s\n",<file name>,<file_MAC_label>

58 The output format of the <file_MAC_label> shall be suitable for re-input as the –
59 label operand to the setfmac utility on the same system. –

60 **10.1.6.2 Standard Error**

61 Used only for diagnostic messages.

62 **10.1.6.3 Output Files**

63 None.

64 **10.1.7 Extended Description**

65 None.

66 **10.1.8 Exit Status**

67 The getfmac utility shall exit with one of the following values:

68 0 The utility executed successfully.

69 >0 An error occurred.

70 **10.1.9 Consequence of Errors**

71 Default.

72 **10.2 getpmac — Get Text Form Of Current Process's MAC Label**

73 **10.2.1 Synopsis**

74 getpmac

75 **10.2.2 Description**

76 The `getpmac` utility writes the text form of the MAC label of the current process.

77 **10.2.3 Options**

78 None.

79 **10.2.4 Operands**

80 None.

81 **10.2.5 External Influences**

82 **10.2.5.1 Standard Input**

83 None.

84 **10.2.5.2 Input Files**

85 None.

86 **10.2.5.3 Environment Variables**

87 The following environment variables shall affect the execution of `getpmac`:

88	LANG	This variable shall determine the locale to use for the locale
89		categories when both LC_ALL and the corresponding
90		environment variable (beginning with LC_) do not specify a
91		locale. See 2.6. %
92	LC_ALL	This variable shall determine the locale to be used to over-
93		ride any values for locale categories specified by the settings
94		of LANG or any environment variables beginning with LC_-
95	LC_MESSAGES	This variable shall determine the language in which mes-
96		sages should be written.

97 **10.2.5.4 Asynchronous Events**

98 Default.

120 10.3.2 Description

121 The `setfmac` utility changes the MAC label of each specified file to the label
122 specified by *label*. For each *file* operand, the `setfmac` utility shall perform the
123 equivalent to the POSIX.1e `mac_from_text()` and `mac_set_file()` functions.

124 10.3.3 Options

125 None.

126 10.3.4 Operands

127 The following operands shall be supported by the implementation:

128 *label* The textual representation of the MAC label specified.

129 *file* The pathname of a file whose MAC label is to be changed. If *file* is not
130 specified, or a *file* is specified as a “–”, then `setfmac` shall read a list of
131 pathnames, each terminated by one <newline> character, from the stan-
132 dard input. If a pathname read from standard input contains only a
133 <newline> character, the results are unspecified.

134 10.3.5 External Influences

135 10.3.5.1 Standard Input

136 If no *file* operand is specified, or a *file* is specified as a “–”, then `setfmac` will
137 accept a list of one or more pathnames, each terminated by one <newline> charac-
138 ter, from the standard input. The results are unspecified if “–” is specified as a
139 file operand more than once.

140 10.3.5.2 Input Files

141 None.

142 10.3.5.3 Environment Variables

143 The following environment variables shall affect the execution of `setfmac`:

144	LANG	This variable shall determine the locale to use for the locale
145		categories when both LC_ALL and the corresponding
146		environment variable (beginning with LC_) do not specify a
147		locale. See 2.6. %
148	LC_ALL	This variable shall determine the locale to be used to over-
149		ride any values for locale categories specified by the settings
150		of LANG or any environment variables beginning with LC_ .

151 **LC_CTYPE** This variable shall determine the locale for this interpreta-
 152 tion of sequences of bytes of text data as characters (e.g.,
 153 single versus multibyte characters in arguments).
 154 **LC_MESSAGES** This variable shall determine the language in which mes- |
 155 sages should be written.

156 **10.3.5.4 Asynchronous Events**

157 Default.

158 **10.3.6 External Effects**

159 **10.3.6.1 Standard Output**

160 None.

161 **10.3.6.2 Standard Error**

162 Used only for diagnostic messages.

163 **10.3.6.3 Output Files**

164 None.

165 **10.3.7 Extended Description**

166 None.

167 **10.3.8 Exit Status**

168 The `setfmac` utility shall exit with one of the following values:

169	0	The utility executed successfully.
170	>0	An error occurred.

171 **10.3.9 Consequence of Errors**

172 Default.

1

Section 11: Information Labeling

%

2 This section describes utilities for the retrieval and manipulation of information
3 labels on specified objects.

4 Support for the utilities defined in this section is optional but shall be provided by
5 any implementation claiming conformance to the Information Label Option. Such
6 an implementation shall provided all of the utilities as described in this section.

7 Three utilities are specified to support information labeling. The `getfinf` utility
8 provides the means for a user to display the information label of a file. The `get-`
9 `pinf` utility provides the means for a user to display the information label of the
10 current process. The `setfinf` utility provides the means for a user to set the
11 information label of a file.

12 11.1 `getfinf` — Get File Information Label

13 11.1.1 Synopsis

14 `getfinf` [*file...*]

15 11.1.2 Description

16 The `getfinf` utility writes to standard output the text form of information labels.
17 For each *file* operand, the `getfinf` utility shall perform the equivalent to the
18 POSIX.1e `inf_get_file()` and `inf_to_text()` functions, and write the returned text
19 string to standard output.

20 11.1.3 Options

21 None.

22 11.1.4 Operands

23 The following operand shall be supported by the implementation:

24 *file* The pathname of a file whose information label is to be written. –

25 11.1.5 External Influences

26 11.1.5.1 Standard Input

27 If no *file* operand is specified, or a *file* is specified as a “–”, then `getfinf` shall
28 read a list of zero or more pathnames from standard input. Otherwise standard
29 input shall not be read. The results are unspecified if “–” is specified as a file
30 operand more than once.

31 11.1.5.2 Input Files

32 None.

33 11.1.5.3 Environment Variables

34 The following environment variables shall affect the execution of `getfinf`:

35 **LANG** This variable shall determine the locale to use for the locale
36 categories when both **LC_ALL** and the corresponding
37 environment variable (beginning with **LC_**) do not specify a
38 locale. See 2.6.

39 **LC_ALL** This variable shall determine the locale to be used to over-
40 ride any values for locale categories specified by the settings
41 of **LANG** or any environment variables beginning with **LC_**.

42 **LC_CTYPE** This variable shall determine the locale for this interpreta-
43 tion of sequences of bytes of text data as characters (e.g.,
44 single- versus multibyte characters in arguments and stan-
45 dard input).

46 **LC_MESSAGES** This variable shall determine the language in which mes-
47 sages should be written.

48 11.1.5.4 Asynchronous Events

49 Default.

50 **11.1.6 External Effects**

51 **11.1.6.1 Standard Output**

52 The following format shall be used for each *file* operand specified if multiple
53 operands are specified, or if the *file* operand is specified as “–”.

54 "%s:\t%s\n", <file_name>, <file_information_label>

55 The output format of the label shall be suitable for re-input as the *infile* –
56 operand to the *setfinf* utility on the same system. +

57 **11.1.6.2 Standard Error**

58 Used only for diagnostic messages.

59 **11.1.6.3 Output Files**

60 None.

61 **11.1.7 Extended Description**

62 None.

63 **11.1.8 Exit Status**

64 The *getfinf* utility shall exit with one of the following values:

65 0 The information labels associated with all specified files were success-
66 fully reported.

67 >0 An error occurred.

68 **11.1.9 Consequences of Errors**

69 Default.

70 **11.2 getpinf — Get Process Information Label**

71 **11.2.1 Synopsis**

72 `getpinf`

73 **11.2.2 Description**

74 The `getpinf` utility writes the information label associated with the current pro-
75 cess to standard output. Note that some floating policies may cause this label to
76 differ from that of the invoking process (e.g., command interpreter).

77 **11.2.3 Options**

78 None.

79 **11.2.4 Operands**

80 None.

81 **11.2.5 External Influences**

82 **11.2.5.1 Standard Input**

83 None.

84 **11.2.5.2 Input Files**

85 None.

86 **11.2.5.3 Environment Variables**

87 The following environment variables shall affect the execution of `getpinf`:

88	LANG	This variable shall determine the locale to use for the locale
89		categories when both LC_ALL and the corresponding
90		environment variable (beginning with LC_) do not specify a
91		locale. See 2.6.
92	LC_ALL	This variable shall determine the locale to be used to over-
93		ride any values for locale categories specified by the settings
94		of LANG or any environment variables beginning with LC_ —

95 **LC_MESSAGES** This variable shall determine the language in which mes-
96 sages should be written.

97 **11.2.5.4 Asynchronous Events**

98 Default.

99 **11.2.6 External Effects**

100 **11.2.6.1 Standard Output**

101 The following format shall be used:

102 "%s\n", <*process_information_label*>

103 The output format of the label shall be suitable for re-input as the *inflabel*
104 operand to the `setfinf` utility on the same system. +

105 **11.2.6.2 Standard Error**

106 Used only for diagnostic messages.

107 **11.2.6.3 Output Files**

108 None.

109 **11.2.7 Extended Description**

110 None.

111 **11.2.8 Exit Status**

112 The `getpinf` utility shall exit with one of the following values:

113 0 The information label associated with the invoking process was suc-
114 cessfully reported.

115 > 0 An error occurred.

116 **11.2.9 Consequences of Errors**

117 Default.

118 **11.3 setfinf — Change File Information Label**

119 **11.3.1 Synopsis**

120 `setfinf inflabel [file...]`

121 **11.3.2 Description**

122 The `setfinf` utility sets the information label associated with each of the
123 specified files to the specified information label.

124 **11.3.3 Options**

125 None.

126 **11.3.4 Operands**

127 The following operands shall be supported by the implementation:

128 *inflabel* The new information label to be associated with each of the specified
129 files.

130 *file* The pathname of a file whose information label is to be changed. –

131 **11.3.5 External Influences**

132 **11.3.5.1 Standard Input**

133 If no *file* operand is specified, or a *file* is specified as a “–”, then `setfinf` shall
134 read a list of zero or more pathnames from standard input. Otherwise standard
135 input shall not be read.

136 **11.3.5.2 Input Files**

137 None.

138 **11.3.5.3 Environment Variables**

139 The following environment variables shall affect the execution of `setfinf`:

140 **LANG** This variable shall determine the locale to use for the locale
141 categories when both **LC_ALL** and the corresponding
142 environment variable (beginning with **LC_**) do not specify a
143 locale. See 2.6.

144	LC_ALL	This variable shall determine the locale to be used to over-
145		ride any values for locale categories specified by the settings
146		of LANG or any environment variables beginning with LC_ .
147	LC_CTYPE	This variable shall determine the locale for this interpreta-
148		tion of sequences of bytes of text data as characters (e.g.,
149		single- versus multibyte characters in arguments and stan-
150		dard input).
151	LC_MESSAGES	This variable shall determine the language in which mes-
152		sages should be written.

153 **11.3.5.4 Asynchronous Events**

154 Default.

155 **11.3.6 External Effects**

156 **11.3.6.1 Standard Output**

157 None.

158 **11.3.6.2 Standard Error**

159 Used only for diagnostic messages.

160 **11.3.6.3 Output Files**

161 None.

162 **11.3.7 Extended Description**

163 None.

164 **11.3.8 Exit Status**

165 The `setfinf` utility shall exit with one of the following values:

166	0	The information labels associated with all specified files were success-
167		fully changed.
168	>0	An error occurred.

169 **11.3.9 Consequences of Errors**

170 Default.

Annex E (informative)

Revisions to the General Section

⇒ **E.2.9.3 Concepts Derived from the Security Standard Rationale** *Add this as a new section.*

This subclause was introduced to describe the relationship between capabilities defined in POSIX.2c and the power or trust associated with a user account, commonly called an authorization in security literature. The POSIX.1 and POSIX.2 standards purposely do not assume the traditional superuser model of trust (effective or real user ID 0), or any other model. Rather, the phrase “appropriate privilege” is included in POSIX.1 to allow for traditional POSIX implementations and trusted system implementations that can support POSIX.1 conforming applications.

The enforcement of power by trusted applications can be based on the identity of the invoking user or on the presence of a trusted program in the process chain that preceded the execution of the utility in the current process. The inheritance of trust, or the indication of a previous trusted process image, through the process chain is the motivation behind the inheritable capabilities flag in the POSIX.2c standard. The state of the user’s initial process inheritable flags at user authentication time is unspecified by POSIX.2c because there is no concept of a user account or a user authentication profile that would normally contain this information. Similarly, the manner in which an administrator or security officer assigns trust or power to user accounts is similarly unspecified by POSIX.2c.

Therefore, the constraints imposed by the utilities defined in POSIX.2 must be specified as behaving differently given “appropriate privilege” or “appropriate authorization.” Until the mechanisms for enforcement are specified by POSIX.2c, this concept must remain undefined.

The two major goals in determining changes to POSIX.2 were to define minimal changes, and to avoid requiring that particular utilities be added to the TCB.

The main reason for avoiding changes to POSIX.1 and POSIX.2 is the lack of consensus of existing practice. For example, some implementations add options to the `ls` utility to display security attributes, while others define new utilities. Even those which add options are inconsistent in the options and the

35 formats used. Additionally, there is no strong justification for adding options
 36 to some utilities. For example, options to test to compare MAC labels may be
 37 desirable, but it is not yet clear what tests are useful.

38 Putting particular utilities in the TCB causes a ripple effect, possibly forcing
 39 utilities like the shell into the TCB. This was considered undesirable. In
 40 order to avoid putting utilities into the TCB, the standard does not specify
 41 capabilities required by or used by particular utilities.

42 For example, the standard does *not* require that the `chown` utility use the
 43 `CAP_CHOWN` capability if the invoking process includes that capability in its
 44 inheritable set, nor does it require that the file containing the `chown` utility
 45 include that capability in its permitted set. Thus, inclusion of the
 46 `CAP_CHOWN` capability in the inheritable set may or may not cause the
 47 `chown` utility to use the capability. Another example is the `ps` utility, where
 48 the standard does not specify what capabilities may be required in order to get
 49 information about processes, nor the capability enforced. Neither does the
 50 standard specify whether any policy or capabilities are enforced by the `ps` util-
 51 ity (as in some historical implementations which use `/dev/kmem`), or by the
 52 underlying system (as in other implementations which use `/dev/proc` or simi-
 53 lar mechanisms).

54 Thus, rather than requiring specific capabilities, the standard makes the capa-
 55 bilities required by particular utilities implementation-defined. By not
 56 defining capabilities, the standard leaves open utility based authentication
 57 using mechanisms outside this standard.

58 ⇒ **E.2.13.2 Symbolic Constants for Portability Specifications**
 59 **(POSIX.2: line 2875)** *Insert the following after line 2875:*

60	<code>POSIX2_ACL</code>	See the rationale in E.8	
61	<code>POSIX2_CAP</code>	See the rationale in E.9	%
62	<code>POSIX2_INF</code>	See the rationale in E.11	%
63	<code>POSIX2_MAC</code>	See the rationale in E.10	%

64 ⇒ **E.4.6 chgrp — Change file group ownership** *Rationale for the lack of*
 65 *changes to this section in POSIX.2 is provided below:*

66 The `chgrp` utility refers to the `chown` function in POSIX.1 for its functional-
 67 ity. While this standard specifies the capabilities required for the function, it
 68 does not specify the capabilities requires for the utility.

69 ⇒ **E.4.7 chmod — Change file mode** *Rationale for the lack of changes to this*
70 *section in POSIX.2 is provided below:*

71 The `chmod` utility does *not* refer to the `chmod` function in POSIX.1 for the
72 exclusive definition of its functionality, but rather specifies “appropriate
73 privilege.” This standard does not define appropriate privilege for utilities.

74 ⇒ **E.4.8 chown — Change file ownership** *Rationale for the lack of changes to*
75 *this section in POSIX.2 is provided below:*

76 The `chown` utility refers to the `chown` function in POSIX.1 for its functional-
77 ity. While this standard specifies the capabilities required for the function, it
78 does not specify the capabilities requires for the utility.

79 ⇒ **E.4.13 cp — Copy files** *Rationale for changes to this section in POSIX.2 is pro-*
80 *vided below:*

81 The definition of `cp` calls for duplicating file permission bits when a new file is
82 created, and for propagating certain characteristics of files when the “-p”
83 option is given. ACLs are included in these propagated characteristics. ACLs
84 are not copied in absence of the “-p” option to maintain compatibility in the
85 cases where the ACL can not be copied.

86 Note that in the absence of the “-p” option, the `cp` utility already specifies that
87 new files are created using `open()`, specifying the file permission bits of the
88 source file in the *mode* argument, and POSIX.1e specifies the impact of default
89 ACLs on `open()`. The result being that if there is a default ACL on the destina-
90 tion directory, the resulting ACL on the destination file will be the default
91 ACL modified by the permission bits of the source file. This effectively will
92 limit access to the newly created file to the minimum of accesses specified in
93 the default ACL and the source file permissions. If the destination directory
94 does not have a default ACL, then the permission bits of the newly created file
95 will be the source file permission bits as modified by the `umask`.

96 Experience with historical operating systems has shown that it is important to
97 be able to specify whether the old ACL is copied (as when the `-p` option is
98 specified), or whether to apply the normal creation defaults (when the `-p`
99 option is not specified). While this standard does not require any particular
100 option, implementors are advised to add specific options to copy the old ACL
101 without copying the other attributes brought along when the “-p” option is
102 used.

103 No specific feature is provided for copying MAC labels, information labels, or
104 capabilities when the “-p” option is provided. Such a feature would require use
105 of appropriate privilege, which this standard avoids wherever possible.

106 It would appear feasible to specify the information label of the copied file.
107 However, that label may depend on the label of the invoking process and the

108 files being copied. Additionally, it may depend on the order in which the
109 operations are performed. For example, a request to copy several files could
110 cause the information label of the last file copied to float up to include all of the
111 previous files. Such a definition would be too complex to be of any use.

112 The only statement that is clearly true is that the information label of the des-
113 tination file will dominate the information label of the invoking process, the
114 information label of the source file, and the information label of the destination
115 file (if it already existed).

116 ⇒ **E.4.16 dd — Convert and copy a file** *Rationale for the lack of changes to this*
117 *section in POSIX.2 is provided below:*

118 The dd utility is a sophisticated copy tool frequently used for copying disks
119 and tapes. However, because it relies on the lower level primitives (such as
120 the open function in POSIX.1) no changes are required.

121 ⇒ **E.4.24 find — Find files** *Rationale for the lack of changes to this section in*
122 *POSIX.2 is provided below:*

123 Some consideration was given to adding options to find to locate files based
124 on MAC labels, information labels, capabilities, and ACLs. However, there was
125 no overwhelming evidence that such options are necessary. Furthermore, they
126 can be emulated using existing features of the find utility. For example, to
127 find a file based on the presence of a particular user in the file's ACL, use the
128 following statement:

129 *find directory -exec checkacl {} username \;*

130 where *directory* is the directory being searched, *username* is the user being
131 searched for, and checkacl is the following shell script:

```
132         getfacl $1 | grep -s $2 >/dev/null  
133         if [ $? -eq 0 ]  
134         then  
135             printf "%s0$1"  
136         fi
```

137 Similar scripts can be written to find files based on other attributes.

138 ⇒ **E.4.26 getconf — Get configuration values** *Rationale for the lack of changes*
139 *to this section in POSIX.2 is provided below:*

140 The description of the `getconf` utility does not list the configuration parame-
141 ters, but refers to the appropriate tables in POSIX.1. Because POSIX.1e is
142 updating the POSIX.1 tables, the additional configuration parameters are
143 included in the `getconf` utility by reference.

144 ⇒ **E.4.38 lp — Send files to a printer** *Rationale for why no changes were made*
145 *to this section in POSIX.2 is provided below:*

146 The ballot resolution group came to the conclusion that labeling of print output
147 is a policy issue, not an interface issue, and therefore the standard should not
148 require `lp` to print MAC or information labels.

149 There are several issues associated with `lp`: human readable label format,
150 what label(s) to print, suppression of human readable labels, and rejection of
151 spool requests.

152 The standard does not address the format of human readable labels because it
153 is not an interface issue. Further, there is no agreement on what is contained
154 in a MAC or information label, so any discussion of the human readable form
155 is pointless.

156 It is arguable whether the MAC label on printed output should be the MAC
157 label of the file being printed or the MAC label of the process making the print
158 request.

159 There is some sentiment that the information label of a file and/or process
160 making a print request should also be printed on each page. The standards
161 committee did not see a strong demand for this facility, and hence it is not
162 included in the standard.

163 Suppression of human readable labels can be considered an interface issue,
164 but it is not *required* by the TCSEC. Rather, it is *allowed* as an exception, pro-
165 viding that auditing is performed. While many implementations will provide
166 this exception, there was no consensus that it is required as an option, espe-
167 cially because the POSIX.2 standard explicitly states that the format of any
168 output is implementation defined.

169 In a secure system, `lp` may reject spool requests based on criteria such as the
170 physical location of the printer, security level of the requesting user, or time of
171 day. The standard provides a general allowance for any unspecified policy for
172 rejecting or cancelling print requests. The statement “..if such a device is not
173 available to the application...” allows a conforming system to refuse the
174 request.

175 ⇒ **E.4.39 ls — List directory contents** *Rationale for changes to this section in*
176 *POSIX.2 is provided below:*

177 There are two issues with regard to `ls`: handling of the alternate access
178 method flag, and displaying additional file information.

179 The access method flag is used to identify ACLs only. Consideration was given
180 to having it indicate MAC, information labels, and capabilities as well. How-
181 ever, on a system with MAC, all files will have a MAC label. Similarly, on a
182 system with information labels, all files will have information labels. For
183 these cases, the indicator would always be on, so it would provide no addi-
184 tional information. Privileges do not participate in access control decisions, so
185 it was decided that they should not be indicated by an access method flag.

186 The issue of extending `ls` to display additional information is difficult. The
187 utility `ls` already has a myriad of options, and many more would be needed for
188 security information. Also, the existing paradigm is one line per file. Given
189 the amount of security information (MAC label, information label, ACL, capa-
190 bilities) which can be on a file, restricting output to a single line is impractical.
191 Thus, no additional options are provided, but rather new utilities are added to
192 display the security relevant data.

193 ⇒ **E.4.43 mv — Move files** *Rationale for changes to this section in POSIX.2 is*
194 *provided below:*

195 When a file is being renamed within a file system, the POSIX.2 standard
196 specifies that an error is generated if the renaming fails. This is adequate
197 from the perspective of MAC and ACL. When a file is copied (as a result of a
198 movement to a different file system), the POSIX.2 standard specifies the file
199 characteristics which are copied with the file. Requiring that the file MAC
200 label, information label, and capabilities be retained would require a discussion
201 of appropriate privilege, which this standard avoids.

202 It would appear that the information label of the destination file could be
203 specified. However, the destination information label depends on whether the
204 destination file is on the same file system as the source file (in which case the
205 operation is a rename, most likely without any change in the information label
206 of the file), or on a separate file system (in which case the operation is a copy,
207 and the information label of the destination file is based on the information
208 label of the process which invoked `mv` as well as the information label of the
209 source file). Hence, this standard does not specify the information label of the
210 destination file.

211 ⇒ **E.4.44 nohup — Invoke a utility immune to hangups** *Rationale for the*
212 *lack of changes to this section in POSIX.2 is provided below:*

213 The nohup utility refers to specific permission bits when creating an output
214 file. However, because the bits are referred to as modifications to 2.9.1.4, the
215 other changes to 2.9.1.4 (to specify the file MAC label, etc.) are adequate.

216 ⇒ **E.4.48 pax — Portable archive interchange** *Rationale for changes to this*
217 *section in POSIX.2 is provided below:*

218 There are three classes of changes to the pax utility: changes to the user inter-
219 face, changes to the backup format, and use of appropriate privilege.

220 Extensions to the interface to restore security attributes are provided as part
221 of this standard. Capital letters were selected to avoid conflicts with other
222 specification characters. Note that using one or more of these specification
223 characters may not cause restoration of the corresponding security attributes,
224 because the pax utility may still require appropriate privilege. Rather than
225 defining a separate specification character for access control lists, restoration
226 of ACLs is included with file permission bits.

227 The pax utility specifies the interface for creating a backup. Definition of the
228 backup format is outside the scope of POSIX.2, and hence changes to the
229 backup format are outside the scope of this standard. Note that the two
230 backup formats referenced in POSIX.2 (tar and cpio) are not extensible to add
231 security attributes. Hence, the capability to restore security attributes is only
232 present if an implementation dependent backup format is used.

233 As with other utilities, this utility calls for appropriate privilege, which is not
234 further specified.

235 ⇒ **E.4.53 rm — Remove directory entries** *Rationale for the lack of changes to*
236 *this section in POSIX.2 is provided below:*

237 The rm utility refers to file permissions, but without specifying permission
238 bits. Additionally, it allows for arbitrary failure of the directory entry removal.
239 Thus, the definition of the utility is general enough that no change is required.

240 ⇒ **E.4.59 stty — Set the options for a terminal** *Rationale for the lack of*
241 *changes to this section in POSIX.2 is provided below:*

242 The stty utility exists only to provide access to the General Terminal Inter-
243 face. Thus, implementations should add restrictions to this utility consistent
244 with the restrictions placed on the GTI interfaces as noted previously.

245 ⇒ **E.4.62 test — Evaluate expression** *Rationale for the lack of changes to this*
246 *section in POSIX.2 is provided below:*

247 Some consideration was given to adding options to test to compare MAC
248 labels, test for presence of ACLs, etc. However, there was no overwhelming
249 evidence that such options are necessary.

250 ⇒ **E.5.2 at — Execute utilities at a later time** *Rationale for changes to this sec-*
251 *tion in POSIX.2 is provided below:*

252 Particular implementations may wish to restrict use of this utility, or to
253 require additional authorization checks when the job is actually run. The form
254 of any such restrictions is left implementation defined.

255 For example, the system could reauthorize the user before the job is executed
256 to verify that the user is still authorized to run at the MAC label and with the
257 capabilities in use at the time the job was queued.

258 ⇒ **E.5.3 batch** *Rationale for the lack of changes to this section in POSIX.2 is pro-*
259 *vided below:*

260 No changes are required for this utility, because batch is defined in terms of
261 at.

262 ⇒ **E.5.5 crontab — Schedule periodic background work** *Rationale for*
263 *changes to this section in POSIX.2 is provided below:*

264 Some implementations may wish to restrict use of this utility, or to require
265 additional authorization checks when the job is actually run. The form of any
266 such restrictions is left implementation defined.

267 For example, the system could reauthorize the user before the job is executed
268 to verify that the user is still authorized to run at the MAC label and with the
269 capabilities in use at the time the job was queued.

270 ⇒ **E.5.17 mesg — Permit or deny messages** *Rationale for the lack of changes*
271 *to this section in POSIX.2 is provided below:*

272 The POSIX.2 definition of this utility specifies “appropriate privilege.” This
273 standard does not define what capability is required.

274 ⇒ **E.5.19 newgrp — Change to a new group** *Rationale for changes to this sec-*
275 *tion in POSIX.2 is provided below:*

276 When *newgrp* changes the group identification, it is important to retain the
277 MAC label, information labels, and inherited capability flags along with the
278 remainder of the process environment.

279 On traditional implementations, the program requires capabilities in order to
280 change the group ID. Some implementations may require that the invoking
281 process also have capabilities before the utility is executed.

282 This utility may prompt the user for a password. This is in some sense a form
283 of identification. However, it is only changing the group ID for a user who has
284 already been identified. Hence, this standard does not require any additional
285 restrictions.

286 ⇒ **E.5.20 nice — Invoke a utility with an altered system scheduling prior-**
287 **ity** *Rationale for the lack of changes to this section in POSIX.2 is provided*
288 *below:*

289 The POSIX.2a definition of this utility specifies “appropriate privilege.” This
290 standard does not define what privilege is required.

291 ⇒ **E.5.23 ps — Report process status** *Rationale for the lack of changes to this*
292 *section in POSIX.2 is provided below:*

293 The POSIX.2 definition of this utility specifies “appropriate privilege.” This
294 standard does not define what privilege is required. For example, a privilege
295 may be required to see processes belonging to a different user or operating at
296 MAC labels other than the MAC label of the invoking process.

1 **E.8 Access Control Lists**

2 **E.8.1 User-Level Utilities**

3 Command line interfaces, i.e., utilities, are provided to examine and manipulate
4 ACL entries. There were several major decisions with the utility interfaces. The
5 following subsections explain the rationale for these decisions.

6 **E.8.1.1 Separate Utilities**

7 The functionality specified in the `getfacl` utility could be added to the `ls` utility.
8 However, the `ls` interface is already sufficiently complex and adding an ACL
9 display capability to `ls` would simply further complicate an overly complex

10 interface.

11 As an alternative, a single utility interface could be provided which would include
12 all of the optional and non-optional utility interfaces specified in this standard.
13 Separate `getfacl` and `setfacl` utilities were specified in order to provide a
14 more modular solution.

15 **E.8.1.2 Utility Names**

16 The names of the `getfacl` and `setfacl` utilities were chosen to be as descrip-
17 tive as possible of the operations performed by the utilities. The names were also
18 chosen to be consistent with the equivalent utilities in the other sections of the
19 standard.

20 **E.8.1.3 Ease of Use**

21 ACL entries are manipulated by specification of a new external representation or
22 by specification of changes to the existing external representation. The external
23 representation of an ACL entry is not trivial. One of the goals of the working
24 group is to encourage the use of ACLs. This goal is accomplished by making
25 design decisions that are biased towards ease of use. In order to make the utility
26 interface easier to use, the output of the `getfacl` utility for a single file can be
27 used as input to the `setfacl` utility. The `getfacl` utility can be used to list the
28 ACLs of multiple files. However, the resulting output could not be used directly as
29 input to the `setfacl` utility because the `getfacl` output would contain multiple
30 entries for the file owner, file group, and other. An attempt to use this as input to
31 `getfacl` would result in an error because the resulting ACL would not be valid
32 as defined by the `acl_valid()` function.

33 The ACL **mask** entry adds significant complexity to the `getfacl` and `setfacl`
34 utility interfaces. This complexity is especially obvious in the `setfacl` utility. In
35 keeping with the goal to provide interfaces which are relatively easy to use, the
36 `setfacl` utility provides a basic set of options to manipulate ACLs including the
37 **mask** entry. The function of automatically generating (or recalculating) the per-
38 missions for the **mask** entry was chosen as the default operation for `setfacl` in
39 order to allow most users to manipulate ACLs without requiring direct, conscious
40 manipulation of the mask. For those cases where the mask has specifically been
41 altered to limit the permissions granted by additional entries, the `-n` option is
42 provided to allow users to manipulate the ACL without affecting the mask. These
43 few operations for the mask should provide the basic capability of manipulating
44 ACLs in most environments. Certainly, additional options may be desirable, e.g.
45 an option to recalculate the mask but remove "extra" permissions that might be
46 granted to other entries by the mask recalculation. Such options were included in
47 a previous draft but were removed due to the overwhelming complexity which
48 they added to the interface. This includes the following:

- 49 (1) Default operation to recalculate the mask value but issue warning mes-
50 sages if any ACL entries might inadvertently grant additional access
51 based on the recalculation of the mask.

52 (2) A `-p` option to remove any permissions from ACL entries which are lim-
53 ited by the mask value.

54 (3) A `-c` option to always recalculate the mask regardless of the effect on the
55 effective permissions granted by ACL entries.

56 The intent of this interface is to provide a basic set of utilities for manipulating
57 ACLs. Implementations may certainly extend the utility interfaces with these or
58 other options.

59 It is expected that most implementations will provide more sophisticated ACL
60 editors to improve ease of use. The working group considered specifying this edi-
61 tor but concluded that such an editor would primarily be a screen oriented user
62 interface and should not be required of conforming implementations.

63 E.8.1.4 Utility Options

64 Picking utility options is never an easy task. Options and option characters were
65 selected which were, at least, moderately meaningful while maintaining con-
66 sistency with the use of option characters by other utilities in existing implemen-
67 tations. The rationale for the inclusion of each of the utility options, and the selec-
68 tion of the option characters, for the `setfacl` utility are:

69 **-b** This option provides a simple method to reset an ACL to the three base
70 entries (the owner, owning group, and other). This operation could be
71 accomplished by reading the current ACL associated with the file, remov-
72 ing all extended ACL entries, and then updating the ACL with the result.
73 However, resetting a file's ACL to the three base entries appears to be a
74 sufficiently significant and frequent operation as to justify an option to
75 quickly perform the operation. The `-b` option was chosen to indicate the
76 **base** ACL entries. The owning **group** entry is reset to the intersection of
77 the owning **group** entry and the **mask** entry by the `-b` option in order to
78 prevent an inadvertent increase in the effective permissions of the own-
79 ing group when removing the **mask** entry.

80 **-d** This option indicates that the requested operations are to be performed
81 on the directory's default ACL instead of the access ACL. This could be
82 implemented with another utility, e.g. a `setdefacl` utility; however,
83 this would result in a second utility with exactly the same options as
84 `setfacl`. Since default and access ACL are manipulated in exactly the
85 same manner and with the same entry validations, a single utility with
86 an option to select the type of ACL is moderately simpler.

87 **-k** The `-k` option entirely removes a default ACL from a directory. This
88 option is necessary, in addition to the `-b` option, because the `-b` option
89 only removes the extended entries and leaves the 3 base entries. The `-k`
90 will completely remove the default ACL from the directory. This option
91 could be implemented as a separate utility; however, keeping the `-k`
92 option would allow the option to be used in conjunction with other
93 options in order to provide more flexibility for the user.

94 The `-k` option is not allowed to operate on access ACLs since access ACLs
 95 must always contain the required ACL entries corresponding to the file
 96 class permissions. if the `-k` option were to be allowed on an access ACL,
 97 then the target file could potentially be left an inconsistent state, i.e. with
 98 NO file permission bits.

99 The `-k` option does not report an error if the `setfacl` utility is used to
 100 remove a default ACL from a directory which does not contain a default
 101 ACL. This is done in order to avoid potentially generating many errors
 102 when the utility is used in conjunction with the `find` utility to remove all
 103 default ACL recursively from all directories within a filesystem hierar-
 104 chy.

105 `-n` The `-n` option is used to indicate that a **mask** entry should not be gen-
 106 erated nor should the permissions associated with the **mask** entry be
 107 recalculated by the `setfacl` utility. A user (or an application) can set a
 108 value for the mask which may restrict the permissions granted by addi-
 109 tional ACL entries. The `-n` option allows the user to subsequently modify
 110 the ACL without automatically changing the **mask** entry and, thereby,
 111 inadvertently increasing the effective permissions of ACL entries. This
 112 option appears to be most useful when manipulating an ACL on a file
 113 whose permission bits are also being manipulated by `chmod`.

114 `-m` The `-m` option is used to update existing entries and to add new entries
 115 to the ACL.

116 `-M` The `-M` option is also used to update existing entries and to add new
 117 entries to an ACL. However, the `-M` option allows entries to be contained
 118 within a file (or to be obtained from standard input). This option is espe-
 119 cially useful when using the `getfacl` utility in conjunction with the
 120 `setfacl` utility to copy an ACL from one object to another; the output
 121 from one utility can be piped directly into the input of the other. While
 122 this capability could be obtained using the shell's back quote substitu-
 123 tion, this operation is expected to be frequent enough to justify an easy,
 124 direct method of specification.

125 `-x` The `-x` option is used to remove existing entries from an ACL. The letter
 126 "x" was chosen instead of "r" in order to not conflict with the use of "r" to
 127 indicate recursive operations in existing utilities.

128 `-X` The `-X` option is also used to remove existing entries from an ACL. How-
 129 ever, like the `-M` option, the `-X` option allows entries to be contained
 130 within a file (or to be obtained from standard input).

131 Originally, the `setfacl` utility also contained the `-i` and `-I` options to completely
 132 replace an ACL with an entirely new ACL. These options were removed because
 133 this operation is effectively provided by the combination of the `-b` option with the
 134 `-m` option. Adding the additional options of `-i` and `-I` was viewed as adding
 135 unnecessary options and complexity to the utility.

136 The working group also considered specifying a single remove option with special
 137 operands:

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
 Preliminary—Subject to Revision.

138 `-r entries` This option removes the named ACL entries.

139 `-r all` This option removed all the ACL entries.

140 `-r nonbase` This option removed all the ACL entries except the base ACL

141 entries.

142 The various `-r` options were considered to be inconsistent with other options to

143 the utility and the syntax was considered to be overly complex. Separate options

144 were chosen as being slightly less complex and better overall syntax.

145 It was suggested that the `getfacl` utility support the option of retrieving both

146 the default and access ACL in one invocation of the utility. Specifically, it was

147 suggested that this operation be the default operation of the utility and that the

148 utility support the `-a` option for retrieving only the access ACL. While adding this

149 feature to the utility is certainly a convenience for the user (instead of having to

150 invoke `getfacl` twice), it also adds complexity to the use of the utility. This

151 would be particularly apparent since the default ACL would only be retrieved

152 when the utility was used to retrieve the ACLs on directories. Likewise, the

153 option should actually be to retrieve all ACLs associated with the file (instead of

154 just the default and access ACLs). Thus, any implementation defined ACLs asso-

155 ciated with the file should also be retrieved. As a default operation, this would

156 add significant complexity to the interface. Also, adding this option would make

157 the `getfacl` utility less consistent with the `setfacl` utility unless `setfacl`

158 were to be modified to accept both the default and access ACLs as input. This

159 option would be best provided as an implementation defined extension to the `get-`

160 `facl` utility.

161 It was suggested that the `setfacl` utility allow an option to recursively set an

162 ACL throughout a filesystem hierarchy and to include an option to select the type

163 of files to which the ACL would be applied. These options are not provided in the

164 utility since the recursive selection of files based on type (as well as other criteria)

165 is provided via the `find` utility. Thus, the recursive setting of an ACL on selected

166 files can easily be accomplished via the combination of the `find` and `setfacl`

167 utilities. This facility is not provided within `setfacl` in order to avoid the dupli-

168 cation of function in different utilities.

169 **E.8.1.5 Evaluation Order of Option Characters**

170 There are two general possibilities for the processing order of the options that

171 may be specified in an invocation of the `setfacl` utility. The options may either

172 be processed in a well defined order as specified by the standard or the options

173 may be processed in the order of the occurrence of the options in the invocation of

174 the utility. The standard could easily specify the apparent "most logical" evalua-

175 tion order for the options (e.g., the `-k` and `-b` options first followed by the `-m` (`-M`)

176 and `-x` (`-X`) options). However, the options are processed in the order in which

177 they are specified by the user. This allows users the flexibility to determine the

178 order of the options to best meet their needs.

179 **E.8.1.6 ACL Entry Type Names**

180 The working group considered using a lower case version of the programmatic
181 ACL entry tag types as the ACL entry tag types for the first field of the external
182 ACL entry representation, e.g., ACL_USER_OBJ => **user_obj**. This option has
183 the advantage of being easier to parse, than using **user** for both the
184 ACL_USER_OBJ and ACL_USER ACL entries. However, this option does not
185 lead to easy aliases, e.g., **user_obj** is difficult to alias, but **user** can be aliased as
186 *u*. The working group felt that the availability of easy aliases outweighs the com-
187 plexity introduced in the parsing and decided not to use the lower case versions of
188 the programmatic ACL entry tag types.

189 **E.8.1.7 ACL Entry Permissions**

190 The `setfacl` utility allows ACL entry permissions to be specified as a symbolic
191 string with an absolute or relative value. The format of this symbolic string is dif-
192 ferent from the format used when specifying permissions in the `chmod()` utility.
193 The format for the permissions was chosen to allow the output of the `getfacl`
194 utility to be used without modification as input to the `setfacl` utility. This will
195 allow a user to easily copy an ACL from one object to other objects. This addi-
196 tional functionality was believed to be more important than maintaining complete
197 consistency with the format of the permissions in `chmod()`.

198 **E.8.1.8 Mask Entry Calculation**

199 The `getfacl` and `setfacl` utilities are designed to make the existence of the
200 ACL **mask** entry as transparent as possible to the users of the utilities. For most
201 ACL manipulations, the permissions can be specified by the ACL entries and the
202 permissions associated with the **mask** entry will be automatically calculated and
203 reset to be the logical union of all of the permissions of all ACL entries in the file
204 group class. While recalculation of the mask value is the default operation, the `-n`
205 option is provided for the `setfacl` utility in order to allow users the ability to
206 directly manipulate the mask value, as desired.

207 **E.8.1.8.1 Mask Calculation Algorithm and Unsafe Conditions**

208 In an earlier draft, the `setfacl` utility attempted to determine the caller's inten-
209 tions in changing ACL entries and would warn the user of situations which the
210 utility considered to be "unsafe" (i.e., when permissions could be granted to ACL
211 entries which the utility determined that the user might not be expecting). In
212 addition, the `-p` option was provided to actually remove such permissions from
213 the ACL and the `-c` option to always recalculate the mask regardless of the
214 utility's interpretation of the user's intentions. The mask recalculation algorithm
215 attempted to detect conditions which the utility considered unsafe entries while
216 minimizing, but not eliminating, false alarms. The algorithm also attempted to
217 limit the detection of unsafe cases to situations which were considered to be
218 highly unlikely user behavior.

219 The actual algorithm for detecting such “unsafe” conditions is described below for
220 reference. Implementations may choose to add options which incorporate this (or
221 a similar) algorithm in the `setfacl` utility. The algorithm specifies changes to
222 the ACL, but those changes would only be made permanent in the ACL after all
223 checks had been made and the operation determined to be “safe”. All operations
224 are performed only on ACL entries that are members of the file group class.

225 The algorithm is specified below in programmatic form:

- 226 (1) Retrieve the existing ACL of the object.
- 227 (2) Perform all requests to remove entries and requests to reduce the permis-
228 sions of existing entries.
- 229 (3) Calculate the union of the actual permissions of all remaining entries.
- 230 (4) Calculate the union of the effective permissions of all remaining entries.
- 231 (5) Determine which permissions differ between the actual and effective
232 rights (logical XOR of results of steps 3 and 4).
- 233 (6) Perform all requests to add new entries to the ACL and all requests to
234 increase the permissions of existing entries.
- 235 (7) Calculate the union of these newly granted permissions and the old effec-
236 tive permissions (step 4). This is the candidate new mask value.
- 237 (8) If there are any permissions in the candidate new mask that are also in
238 the permissions that differ between the original actual and effective
239 rights (step 5), applying the candidate new mask would unexpectedly
240 grant some new right that the user did not intend. Unless the user
241 specified one of the options `-c`, `-p`, or `-n`, this condition shall generate an
242 error and the ACL will not be modified. If this condition does not hold,
243 then apply the candidate new mask as the new mask.

244 The algorithm avoids false alarms that would occur if the new mask were simply
245 calculated to be the logical OR of all the entries of the new file group class.

246 The following is an example of how false alarms could be avoided.

247 Consider an ACL with the following entries:

```
248     user::rwx
249     mask::r-x
250     user:user1:rwx #effective:r-x
251     user:user2:rwx #effective:r-x
252     group::r-x
253     other::---
```

254 If user1’s permissions were changed to r-x permission, a simple recalculation
255 using `acl_calc_mask()` would result in changing the mask to rwx which would
256 inadvertently grant w permission to user2. However, the algorithm specified
257 above detects that removing user1’s w permission does not require altering the
258 mask.

259 The `-p` option was provided to allow the user to remove ineffective permissions.
260 The `-c` option was included in order to allow users to always request uncondi-
261 tional recomputation of the mask regardless of any unsafe conditions. As this
262 option could be quite dangerous, it was suggested that an implementation issue a
263 warning message if any unsafe conditions were detected.

264 Notice that the mask entry was only relevant when it had been “lowered” to actu-
265 ally reduce the permissions granted by one or more entries within the ACL. A
266 “lowered” mask could only occur for three reasons. The first reason is that the
267 mask may have been specified to a value less than some of the permissions in the
268 default ACL (such as execute permission on a data file) when an object was
269 created.

270 The second reason is that a program may have temporarily lowered the mask to
271 lock out other users from the file.

272 The third reason is that the user may have lowered the mask using the *chmod()*
273 utility explicitly.

274 In all cases, the user of the *setfacl* utility would need to know that the mask
275 had been lowered, understand why the mask had been lowered, and would be
276 required to be able to select the appropriate options for the utility in order to
277 achieve the desired results.

278 Since it had been suggested that implementations issue warning messages for the
279 detection of “unsafe” conditions, it was also suggested that such implementations
280 also would provide a `-s` option that would suppress the messages.

281 The detection of “unsafe” conditions and the attempted interpretation of the user’s
282 intention in manipulating the mask added significant complexity to the *setfacl*
283 interface. This type of operation and the resulting interface were considered
284 entirely too complex for users to understand or use effectively. As such, the `-c` and
285 `-p` options were removed; and the default operation was changed to simply recal-
286 culate the mask. Likewise, the mask recalculation was changed to be simply the
287 union of the permissions in all ACL entries within the file group class.

288 **E.8.1.8.2 Mask Calculation and *chmod***

289 It was considered to allow the ACL mask entry to be set only by the *chmod* utility
290 and not be modifiable by the *setfacl* utility. This restriction was rejected
291 because it would have made copying ACLs from one file to another too difficult by
292 requiring the use of the *chmod* utility as well as the ACL utilities.

1 **E.10 Capability**

%

2 **E.10.1 Capability-Related Utilities**

3 These utilities were determined to be the minimal set necessary for the determi-
4 nation and establishment of the capability attributes of files.

5 These is an argument that such utilities are administrative in nature and there-
6 fore outside of the scope of this working group. The working group noted, how-
7 ever, that installation scripts and programs are themselves portable applications
8 that will need to work across implementations, and these utilities will be required
9 to support them. In addition, should the argument be accepted, there are
10 sufficient reasons for adopting standards for these types of utilities to persuade
11 other existing working groups to just adopt them as part of their standard. Since
12 the only practical result of not specifying these utilities is to merely delay their
13 specification, we felt that they would best be specified by this committee.

14 Some responders to the initial ballot felt that there should be commands to assign
15 capabilities to users as well. Since POSIX does not yet specify a user database or
16 identification and authentication system, we felt that inclusion of such commands
17 was premature. In addition, a few people proposed examples of systems which
18 would conform to this standard where no capability data was directly associated
19 with users. For these cases, requiring such commands would be an undue burden
20 on the implementation. If an implementation does wish to assign capabilities to
21 users, however, we believe that extending the syntax of the two commands
22 presented here would be simple and straightforward.

23 **E.10.1.1 Get and Set the Capability State of a Subject or Object**

24 The `getfcap`, `getpcap` and `setfcap` utilities were included as a part of the
25 standard primarily to support the definition of a standard, cross-implementation
26 user interface for the administration of file capabilities. Increasingly, secure
27 applications will need to have a standard means of being installed so that opera-
28 tors that do not necessarily have a strong background in security can just run a
29 script supplied to them by a security administrator. In addition, one of the neces-
30 sary functions of a security administrator is to periodically check the security
31 related attributes of files and programs to ensure that they have not been tam-
32 pered with. The standardization of utilities that support setting and display of file
33 capability attributes is therefore considered to be necessary.

34 The grammar chosen for representing, setting and modifying capability states is a
35 modified version of that used by `chmod` for the symbolic representation of mode
36 bit operations. This representation was chosen because it is compact and is fami-
37 liar to current users of POSIX systems.

38 It was decided not to add to the functionality of existing utilities or system func-
39 tions in this area; specifically, the `stat()` system function and the `ls` utilities are
40 already overburdened and complex.

1 **E.11 Mandatory Access Control**

2 **E.11.1 General Overview**

3 The following utilities have been added to support mandatory access control
4 (MAC): `getfmac`, `setfmac`, and `getpmac`. These utilities were determined to
5 be the minimal set necessary for the determination and establishment of MAC
6 labels for files and processes. No utility is provided to allow users to modify the
7 MAC label of an executing process for two reasons: (1) there is no precedent in
8 POSIX.2 for utilities that modify the attributes (e.g., user ID or umask) of existing
9 processes, and (2) no compelling argument has ever been put forward for why
10 such a utility would be useful.

11 For each utility, the code is not derived from any existing system, and no source
12 code was examined. None of these interfaces depend on a precise definition of
13 what constitutes a MAC label provided. That is, the format of the MAC label
14 argument to `setfmac` is not specified, nor is the format of the MAC labels writ-
15 ten to standard out by `getfmac` and `getpmac`. This is because POSIX.1e does
16 not constrain implementations in terms of the allowable human-readable
17 representations of MAC labels. (As a practical matter, most *implementations*
18 probably should not constrain them: the human-readable representations of MAC
19 labels will typically be administrator defined.)

20 Because the precise format of the text representation of MAC labels is not
21 specified, both `getfmac` and `getpmac` only loosely specify the location of the
22 label within the output stream. Until standards for label syntax are specified,
23 utilities (especially standard utilities such as `awk` or `grep`) cannot parse the out-
24 put of these utilities. Therefore, the output of these utilities are primarily useful
25 only for display to users. Specifying a standard for the syntax of the text
26 representation of labels was considered, but rejected for inclusion in this stan-
27 dard.

28 Note that conforming implementations may choose to provide a more rigorously
29 specified output format to assist implementation-specific parsing utilities, or pro-
30 vide a visually more easily understood output format through the use of an addi-
31 tional argument.

32 **E.11.2 Separate Utilities**

33 The working group considered adding the functionality specified in the `getfmac`
34 to `ls`. However, the working group strongly feels that the `ls` interface is already
35 sufficiently complex and that adding MAC label display capabilities to `ls` would
36 further complicate an overly complex interface.

37 The working group also considered designing a single utility interface that
38 included all of the utility interfaces specified in this standard. However, one of
39 the goals of the working group is to produce a modular set of interfaces. Since the
40 working group felt that this solution does not fit into a modular model, the group
41 discarded this solution.

42 **E.11.3 Label Input and Output**

43 As noted above, the format of the labels produced by `getfmac`, `setfmac`, and
44 `getpmac` are not precisely defined. Nevertheless, in order to provide support for
45 portable applications, it was felt that these utilities should be required to inter-
46 operate. This is necessary to allow portable applications to use the output of the
47 utilities, and portable shell scripts to reuse the labels output from the utilities
48 (e.g., in save-alter-restore algorithms that temporarily modify file labels). There-
49 fore, the specification requires that the labels output by `getfmac` and `getpmac`
50 must be in a format suitable for re-input to the `setfmac` utility. Similarly, the
51 labels output by `getfmac` and `getpmac` must be suitable for re-input to the
52 `mac_from_text()` function defined in section 26.3.7 of POSIX.1e. Finally, we
53 require that the labels produced by the `mac_to_text()` defined in section 26.3.17 of
54 POSIX.1e. must be suitable for re-input to the `setfmac` utility. |

55 **E.11.4 Utility use of Capabilities** |

56 Actual implementation of utilities such as `setfmac` may require the utility to
57 possess appropriate privilege to perform its function, but this standard is mute on
58 whether privilege is required or the specific capabilities which may be required. |
59 This is an interface specification for the utilities. As such, the interface to these
60 utilities must be specified, as must their behavior. Their implementation, how-
61 ever, is outside the scope of this standard. A conforming implementation could
62 certainly implement these utilities without using POSIX interfaces. Whatever
63 (native) interfaces are used to implement the utility may not require capabilities, |
64 or if they do, they may not require POSIX capabilities. Therefore, specifying that
1 capabilities are required may, in at least some cases, be incorrect. %

2 **E.12 Information Labeling**

3 **E.12.1 General Overview**

4 The following utilities have been added to support information labeling: `get-`
5 `finf`, `setfinf`, and `getpinf`. These utilities were determined to be the
6 minimal set necessary for the determination and establishment of information
7 labels for files and processes. No utility is provided to allow users to modify the
8 information label of an executing process for two reasons: (1) there is no pre-
9 cedent in POSIX.2 for utilities that modify the attributes (e.g., user ID or umask)
10 of existing processes, and (2) no compelling argument has ever been put forward
11 for why such a utility would be useful.

12 For each utility, the code is not derived from any existing system, and no source
13 code was examined. None of these interfaces depend on a precise definition of
14 what constitutes an information label. That is, the format of the information
15 label argument to `setfinf` is not specified, nor is the format of the information
16 labels written to standard out by `getfinf` and `getpinf`. This is because
17 POSIX.1e does not constrain implementations in terms of the allowable human-

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

18 readable representations of information labels. (As a practical matter, most
19 *implementations* probably should not constrain them: the human-readable
20 representations of information labels will typically be administrator defined.)

21 Because the precise format of the text representation of information labels is not
22 specified, both `getfinf` and `getpinf` only loosely specify the location of the
23 label within the output stream. Until standards for label syntax are specified,
24 utilities (especially standard utilities such as `awk` or `grep`) cannot parse the out-
25 put of these utilities. Therefore, the output of these utilities are primarily useful
26 only for display to users. Specifying a standard for the syntax of the text
27 representation of labels was considered, but rejected for inclusion in this stan-
28 dard.

29 Note that conforming implementations may choose to provide a more rigorously
30 specified output format to assist implementation-specific parsing utilities, or pro-
31 vide a visually more easily understood output format through the use of an addi-
32 tional argument.

33 **E.12.2 Separate Utilities**

34 The working group considered adding the functionality specified in the `getfinf`
35 to `ls`. However, the working group strongly feels that the `ls` interface is already
36 sufficiently complex and that adding information label display capabilities to `ls`
37 would further complicate an overly complex interface.

38 The working group also considered designing a single utility interface that
39 included all of the utility interfaces specified in this standard. However, one of
40 the goals of the working group is to produce a modular set of interfaces. Since the
41 working group felt that this solution does not fit into a modular model, the group
42 discarded this solution.

43 **E.12.3 Label Input and Output**

44 As noted above, the format of the labels produced by `getfinf`, `setfinf`, and
45 `getpinf` are not precisely defined. Nevertheless, in order to provide support for
46 portable applications it was felt that these utilities must interoperate both with
47 themselves, and with the relevant functions defined in section 27 of POSIX.1e.
48 This is necessary to allow portable applications to use the labels produced by the
49 utilities, portable shell scripts to use the labels output by applications using the
50 applicable IL functions and to reuse the labels output from the utilities (e.g., in
51 save-alter-restore algorithms that temporarily modify file labels), etc. Therefore,
52 the specification requires that the labels output by `getfinf` and `getpinf` must
53 be in a format suitable for re-input to the `setfinf` utility and to the
54 `inf_from_text()` function defined in section 27.3.9 of POSIX.1e. Similarly, we
55 require that the labels produced by the `inf_to_text()` function defined in section
56 27.3.17 of POSIX.1e must be suitable for re-input to the `setfinf` utility.

Annex F
(informative)
Ballot Instructions

This annex will not appear in the final standard. It is included in the draft to provide instructions for balloting that cannot be separated easily from the main document, as a cover letter might.

It is important that you read this annex, whether you are an official member of the PSSG Balloting Group or not; comments on this draft are welcomed from all interested technical experts.

Summary of Draft 17 Instructions

This is a recirculation on the P1003.2c ballot. The procedure for a recirculation is described in this annex. Because this is a recirculation comments may only be provided concerning sections that have changed, sections affected by those changes, or on rejected comments from the previous ballot.

Send your ballot and/or comments to:

*IEEE Standards Office
Computer Society Secretariat
ATTN: PSSG Ballot (Carol Buonfiglio)
P.O. Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331*

It would also be very helpful if you sent us your ballot in machine-readable form. Your official ballot must be returned via mail to the IEEE office; if we receive only the e-mail or diskette version, that version will not count as an official document. However, the online version would be a great help to ballot resolution. Please send your e-mail copies to the following address:

casey@sgi.com

or you may send your files in ASCII format on DOS 3.5 inch formatted diskettes (720Kb or 1.4Mb), or Sun-style QIC-24 cartridge tapes to:

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Casey Schaufler
Silicon Graphics
2011 North Shoreline Blvd.
P.O. Box 7311
Mountain View, CA 94039-7311

Background on Balloting Procedures

The Balloting Group consists of approximately eighty technical experts who are members of the IEEE or the IEEE Computer Society; enrollment of individuals in this group has already been closed. There are also a few “parties of interest” who are not members of the IEEE or the Computer Society. Members of the Balloting Group are required to return ballots within the balloting period. Other individuals who may happen to read this draft are also encouraged to submit comments concerning this draft. The only real difference between members of the Balloting Group and other individuals submitting ballots is that *affirmative* ballots are only counted from Balloting Group members who are also IEEE or Computer Society members. (There are minimum requirements for the percentages of ballots returned and for affirmative ballots out of that group.) However, objections and nonbinding comments must be resolved if received from any individual, as follows:

- (1) Some objections or comments will result in changes to the standard. This will occur either by the republication of the entire draft or by the publication of a list of changes. The objections/comments are reviewed by a team from the POSIX Security working group, consisting of the Chair, Vice Chair, Technical Editor, and a group of Technical Reviewers. The Chair will act as the Ballot Coordinator. The Technical Reviewers each have subject matter expertise in a particular area and are responsible for objection resolution in one or more sections.
- (2) Other objections/comments will not result in changes.
 - (a) Some are misunderstandings or cover portions of the document (front matter, informative annexes, rationale, editorial matters, etc.) that are not subject to balloting.
 - (b) Others are so vaguely worded that it is impossible to determine what changes would satisfy the objector. These are referred to as *Unresponsive*. (The Technical Reviewers will make a reasonable effort to contact the objector to resolve this and get a newly worded objection.) Further examples of unresponsive submittals are those not marked as either *Objection*, *Comment*, or *Editorial*; those that do not identify the portion of the document that is being objected to (each objection must be separately labeled); those that object to material in a recirculation that has not changed and do not cite an unresolved objection; those that do not provide specific or general guidance on what changes would be required to resolve the objection.

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

- (c) Finally, others are valid technical points, but they would result in decreasing the consensus of the Balloting Group. (This judgment is made based on other ballots and on the experiences of the working group through over seven years of work and fifteen drafts preceding this one.) These are referred to as *Unresolved Objections*. Summaries of unresolved objections and their reasons for rejection are maintained throughout the balloting process and are presented to the IEEE Standards Board when the final draft is offered for approval. Summaries of all unresolved objections and their reason for rejection will also be sent to members of the Balloting Group for their consideration upon a recirculation ballot. (Unresolved objections are not circulated to the ballot group for a re-ballot.) Unresolved objections are only circulated to the balloting group when they are presented by members of the balloting group or by parties of interest. Unsolicited correspondence from outside these two groups may result in draft changes, but are not recirculated to the balloting group members.

Please ensure that you correctly characterize your ballot by providing one of the following:

- (1) Your IEEE member number
- (2) Your IEEE Computer Society affiliate number
- (3) If (1) or (2) don't apply, a statement that you are a "Party of Interest"

Ballot Resolution

The general procedure for resolving ballots is:

- (1) The ballots are put online and distributed to the Technical Reviewers.
- (2) If a ballot contains an objection, the balloter may be contacted individually by telephone, letter, or e-mail and the corrective action to be taken described (or negotiated). The personal contact will most likely not occur if the objection is very simple and obvious to fix or the balloter cannot be reached after a few reasonable attempts. Repeated failed attempts to elicit a response from a balloter may result in an objection being considered unresponsive, based on the judgment of the Ballot Coordinator. Once all objections in a ballot have been resolved, it becomes an affirmative ballot.
- (3) If any objection cannot be resolved, the entire ballot remains negative.
- (4) After the ballot resolution period the technical reviewers may choose to either *re-ballot* or *recirculate* the ballot, based on the status of the standard and the number and nature of outstanding (i.e., rejected or unresolved) objections. The ballot group may or may not be reformed at this time. If a *reballot* is chosen, the entire process of balloting begins anew. If a *recirculation* is chosen, only those portions affected by the previous ballot will be under consideration. This ballot falls into this latter category

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

- (5) On a *recirculation* ballot, the list of unresolved objections, along with the ballot resolution group's reasons for rejecting them will be circulated to the existing ballot group along with a copy of the document that clearly indicates all changes that were made during the last ballot period. You have a minimum of ten days (after an appropriate time to ensure the mail got through) to review these two documents and take one of the following actions:
 - (a) Do nothing; your ballots will continue to be counted as we have classified them, based on items (3) and (4).
 - (b) Explicitly change your negative ballot to affirmative by agreeing to remove all of your unresolved objections.
 - (c) Explicitly change your affirmative ballot to negative based on your disapproval of either of the two documents you reviewed. If an issue is not contained in an unresolved objection or is not the result of a change to the document during the last ballot resolution period, it is not allowed. Negative ballots that come in on recirculations cannot be cumulative. They shall repeat any objections that the balloter considers unresolved from the previous recirculation. Ballots that simply say "and all the unresolved objections from last time" will be declared unresponsive. Ballots that are silent will be presumed to fully replace the previous ballot, and all objections not mentioned on the most current ballot will be considered as successfully resolved.
- (6) Rather than reissue the entire document, a small number of changes may result in the issuance of a change list rather than the entire document during recirculation.
- (7) A copy of all your objections and our resolutions will be mailed to you.
- (8) If at the end of a recirculation period there remain greater than seventy-five percent affirmative ballots, and no new objections have been received, a new draft is prepared that incorporates all the changes. This draft and the unresolved objections list go to the IEEE Standards Board for approval. If the changes cause too many ballots to slip back into negative status, another resolution and recirculation cycle begins.

Balloting Guidelines

This section consists of guidelines on how to write and submit the most effective ballot possible. The activity of resolving balloting comments is difficult and time consuming. Poorly constructed comments can make that even worse.

We have found several things that can be done to a ballot that make our job more difficult than it needs to be, and likely will result in a less than optimal response to ballots that do not follow the form below. Thus it is to your advantage, as well as ours, for you to follow these recommendations and requirements.

If a ballot that significantly violates the guidelines described in this section comes to us, we may determine that the ballot is unresponsive.

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

If we recognize a ballot as “unresponsive,” we will try to inform the balloter as soon as possible so he/she can correct it, but it is ultimately the balloter’s responsibility to assure the ballot is responsive. Ballots deemed to be “unresponsive” may be ignored in their entirety.

Some general guidelines to follow before you object to something:

- (1) Read the Rationale section that applies to the troublesome area. In general there is a matching informative section in the Rationale Annex for each normative section of the standard. This rationale often explains why choices were made and why other alternatives were not chosen.
- (2) Read the Scope, section 1, to see what subset of functionality we are trying to achieve. This standard does not attempt to be everything you ever wanted for accomplishing secure software systems. If you feel that an additional area of system interface requires standardization, you are invited to participate in the security working group which is actively involved in determining future work.
- (3) Be cognizant of definitions in section 2. We often rely in the document on a precise definition from section 2 which may be slightly different than your expectation.

Typesetting is not particularly useful to us. Also please do not send handwritten ballots. Typewritten (or equivalent) is fine, and if some font information is lost it will be restored by the Technical Editor in any case. You may use any word processor to generate your objections but do not send [nt]roff (or any other word processor) input text. Also avoid backslashes, leading periods and apostrophes in your text as they will confuse our word processor during collation and printing of your comments. The ideal ballot is formatted as a “flat ASCII file,” without any attempt at reproducing the typography of the draft and without embedded control characters or overstrikes; it is then printed in Courier (or some other typewriter-like) font for paper-mailing to the IEEE Standards Office and simultaneously e-mailed to the Working Group Ballot Coordinator at the following email address.

casey@sgi.com

Don’t quote others’ ballots. Cite them if you want to refer to another’s ballot. If more than one person wants to endorse the same ballot, send just the cover sheets and one copy of the comments and objections. [Note to Institutional Representatives of groups like X/Open, OSF, UI, etc.: this applies to you, too. Please don’t duplicate objection text with your members.] Multiple identical copies are easy to deal with, but just increase the paper volume. Multiple almost-identical ballots are a disaster, because we can’t tell if they are identical or not, and are likely to miss the subtle differences. Responses of the forms:

- “I agree with the item in <someone>’s ballot, but I’d like to see this done instead”
- “I am familiar with the changes to foo in <someone>’s ballot and I would object if this change is [or is not] included”

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

are very useful information to us. If we resolve the objection with the original balloter (the one whose ballot you are referencing), we will also consider yours to be closed, unless you specifically include some text in your objection indicating that should not be done.

Be very careful of “Oh, by the way, this applies <here> too” items, particularly if they are in different sections of the document that are likely to be seen by different reviewers. They are probably going to be missed! Note the problem in the appropriate section, and cite the detailed description if it’s too much trouble to copy it. The reviewers don’t read the whole ballot. They only read the parts that appear in the sections that they have responsibility for reviewing. Particularly where definitions are involved, if the change really belongs in one section but the relevant content is in another, please include two separate comments/objections.

Please consider this a new ballot that should stand on its own. Please do not make backward references to your ballots for the previous draft. Include all the text you want considered here, because the Technical Reviewer will not have your old ballot. (The old section and line numbers won’t match up anyway.) If one of your objections was not accepted exactly as you wanted, it may not be useful to send in the exact text you sent before; read our response to your objection (you will receive these in a separate mailing) and the associated Rationale section and come up with a more compelling (or clearly-stated) justification for the change.

Please be very wary about global statements, such as “all of the arithmetic functions need to be defined more clearly.” Unless you are prepared to cite specific instances of where you want changes made, with reasonably precise replacement language, your ballot will be considered unresponsive.

Ballot Form

The following form is strongly recommended. We would greatly appreciate it if you sent the ballot in electronic form in addition to the required paper copy. Our policy is to handle all ballots online, so if you don’t send it to us that way, we have to type it in manually. See the first page of this Annex for the addresses and media. As you’ll see from the following, formatting a ballot that’s sent to us online is much simpler than a paper-only ballot.

The paper ballot should be page-numbered, and each page should contain the name, e-mail address, and phone number(s) of the objector(s). The electronic copy of the ballot should only have it once, in the beginning. Please leave adequate (at least one inch) margins on both sides.

Don’t format the ballot as a letter or document with its *own* section numbers. These are simply confusing. As shown below, it is best if you cause each objection and comment to have a sequential number that we can refer to amongst ourselves and to you over the phone. Number sequentially from 1 and count objections, comments, and editorial comments the same; don’t number each in its own range.

We recognize three types of responses:

Objection A problem that must be resolved to your satisfaction prior to your casting an "affirmative" vote for the document.

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Comment 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. Any response 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 non-binding comment whether you label it that way or not. (It would help us if you'd label it correctly.)

Editorial A problem that is strictly an editorial oversight and is not of a technical nature. Examples are: typos; misspellings; English syntax or usage errors; appearances of lists or tables; arrangement of sections, clauses, and subclauses (except where the location of information changes the optionality of a feature).

To help us in our processing of your objections and comments, we are requiring that all comments, objections and editorial comments meet the following specific format. (We know that the format defined below contains redundant information but it has become a de facto standard used by many different POSIX standard ballots. It is felt that it is better to continue to use this format with the redundancies rather than to create a new format just for 1003.1e and P1003.2c)

Separate each objection/comment with a line of dashes ("-"), e.g.,

Precede each objection/comment with two lines of identifying information:

The first line should contain:

`@ <section>. <clause> <code> <seqno>`

where:

- | | |
|------------------------------|--|
| <code>@</code> | At-sign in column 1 (which means no @'s in any other column 1's). |
| <code><section></code> | The major section (chapter or annex) number or letter in column 3. Use zero for Global or for something, like the frontmatter, that has no section or annex number. |
| <code><clause></code> | The clause number (second-level header). Please do not go deeper than these two levels. In the text of your objection or comment, go as deep as you can in describing the location, but this code line uses two levels only. |
| <code><code></code> | One of the following lowercase letters, preceded and followed by spaces: <ul style="list-style-type: none">o Objection.c Comment.e Editorial Comment. |

<seqno> A sequence number, counting all objections and comments in a single range.

The second line should contain:

<seqno>. Sect *<sectno>* *<type>*. page *<pageno>*, line *<lineno>*:

where:

<seqno> The sequence number from the preceding line
<sectno> The full section number. (Go as deep as you can in describing the location.)
<type> One of the following key words/phrases, preceded and followed by spaces:

OBJECTION

COMMENT

EDITORIAL COMMENT

<pageno> The page number from the document.

<lineno> The line number or range of line numbers that the object/comment relates to.

For each objection, comment, or editorial comment, you should provide a clear statement of the problem followed by the action required to solve that problem.

Problem:

A clear statement of the problem that is observed, sufficient for others to understand the nature of the problem. (Note that you should identify problems by section, page, and line numbers. This may seem redundant, but if you transpose a digit pair, we may get totally lost without a cross-check like this. Use the line number where the problem starts, not just where the section itself starts; we sometimes attempt to sort objections by line numbers to make editing more accurate. If you are referring to a range of lines, please don't say "lines 10xx;" use a real range so we can tell where to stop looking. Please try to include enough context information in the problem statement (such as the name of the function or command) so we can understand it without having the draft in our laps at the time. (It also helps you when we e-mail it back to you.)

Action:

A precise statement of the actions to be taken on the document to resolve the objection above, which if taken verbatim will completely remove the objection.

If there is an acceptable range of actions, any of which will resolve the problem for you if taken exactly, please indicate all of them. If we accept any of these, your objection will be considered as resolved.

If the Action section is omitted or is vague in its solution, the objection may be reclassified as a nonbinding comment. The Technical Reviewers, being human, will give more attention to Actions that are well-described than ones that are

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

vague or imprecise. The best ballots of all have very explicit directions to substitute, delete, or add text in a style consistent with the rest of the document, such as:

Delete the sentence on lines 101-102:

"The implementation shall not ... or standard error."

On line 245, change "shall not" to "should not".

After line 711, add:

-c Calculate the mask permissions and update the mask.

Some examples of poorly-constructed actions:

Remove all features of this command that are not supported by BSD.

Add -i.

Make this command more efficient and reliable.

Use some other flag that isn't so confusing.

I don't understand this section.

Specify a value--I don't care what.

Sample Response:

Joseph Balloter (999)123-4567 page 4 of 17.
EMAIL: jmb@mycomp.com FAX: (999)890-1234

@ 1.1 o 23

23. Sect 1.1 OBJECTION. page 7, line 9:

Problem:

The current draft describes one the mechanisms specified in it as "Least Privilege" which is incorrect. "Least Privilege" is a general principle related to access control rather than a mechanism. In fact, the definition given in the standard (p. 91, l. 274) calls it a principle rather than a mechanism.

Action:

Replace line 9 with: "(3) Enforcement of Least Privilege"

@ 3.1 o 24

24. Sect 3.1 OBJECTION. page 27, line 13:

Problem:

"during process of changing ACL" is vague.
Could be read as the duration from acl_read through acl_write.

Action:

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Should state "while ACL is being written (acl_write)".

@ 3.3 e 25

25. Sect 3.3.1 EDITORIAL COMMENT. page 29, line 68:

Problem:

The two previous sentences describe the "ACL_USER_OBJ entry" and the "ACL_GROUP_OBJ entry". Line 68 describes "ACL_OTHER_OBJ", the word "entry" should be added for consistency.

Action:

change "ACL_OTHER_OBJ" to "ACL_OTHER_OBJ entry"

Sample Response (continued):

Joseph Balloter (999)123-4567 page 5 of 17.
EMAIL: jmb@mycomp.com FAX: (999)890-1234

@ 4.5 c 26

26. Sect 4.5.1.1 COMMENT. page 92, line 836:

Problem:

There is no introduction to table 4-1.

Action:

Add before line 836 "The aud_ev_info_t structure shall contain at least the following fields:"

@ 6.5 o 27

27. Sect 6.5.7.2 OBJECTION. page 181, line 449-450:

Problem:

Can this "must" be tested ?

Is this really needed since the format of the label is undefined and no functions are provided to access the individual components (so that a comparison could be made). This seems to be a comment that could just as easily be applied to most other mac functions, say mac_freelabel for example.

Action:

Suggest either moving this into the MAC introductory section, striking or changing "must" to "should" or "are advised".

Thank you for your cooperation and assistance in this important balloting process.

Lynne M. Ambuel
Chair, POSIX Security Working Group

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Identifier Index

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Topical Index

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

Contents

SECTION	PAGE
Section 1: Revisions to the General Section	1
Section 2: Revisions to Terminology and General Requirements	3
Section 4: Revisions to Execution Environment Utilities	11
Section 5: POSIX.2— Revisions to User Portability Utilities	13
Section 8: Access Control Lists	15
8.1 getfacl — Get ACL Information	15
8.1.1 Synopsis	15
8.1.2 Description	15
8.1.3 Options	15
8.1.4 Operands	16
8.1.5 External Influences	16
8.1.6 External Effects	17
8.1.7 Extended Description	17
8.1.8 Exit Status	19
8.1.9 Consequence of Errors	19
8.2 setfacl — Set Access Control List	19
8.2.1 Synopsis	19
8.2.2 Description	19
8.2.3 Options	20
8.2.4 Operands	21
8.2.5 External Influences	21
8.2.6 External Effects	22
8.2.7 Extended Description	22
8.2.8 Exit Status	24
8.2.9 Consequence of Errors	24
Section 9: Capability	25
9.1 getfcap — Get the Capability State of a File	25
9.1.1 Synopsis	25
9.1.2 Description	25
9.1.3 Options	25
9.1.4 Operands	26
9.1.5 External Influences	26
9.1.6 External Effects	27
9.1.7 Extended Description	27
9.1.8 Exit Status	28
9.1.9 Consequence of Errors	28
9.2 getpcap — Get the Capability State of a Process	28
9.2.1 Synopsis	28
9.2.2 Description	28

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

SECTION		PAGE
	9.2.3 Options	28
	9.2.4 Operands	29
	9.2.5 External Influences	29
	9.2.6 External Effects	30
	9.2.7 Extended Description	30
	9.2.8 Exit Status	30
	9.2.9 Consequence of Errors	30
9.3	setfcap — Set Capability State of a File	30
	9.3.1 Synopsis	30
	9.3.2 Description	31
	9.3.3 Options	31
	9.3.4 Operands	31
	9.3.5 External Influences	32
	9.3.6 External Effects	33
	9.3.7 Extended Description	33
	9.3.8 Exit Status	33
	9.3.9 Consequences of Errors	33
Section 10:	Mandatory Access Control	35
10.1	getfmac — Get the MAC Label of a File	35
	10.1.1 Synopsis	35
	10.1.2 Description	35
	10.1.3 Options	35
	10.1.4 Operands	36
	10.1.5 External Influences	36
	10.1.6 External Effects	37
	10.1.7 Extended Description	37
	10.1.8 Exit Status	37
	10.1.9 Consequence of Errors	37
10.2	getpmac — Get Text Form Of Current Process's MAC Label	37
	10.2.1 Synopsis	37
	10.2.2 Description	38
	10.2.3 Options	38
	10.2.4 Operands	38
	10.2.5 External Influences	38
	10.2.6 External Effects	39
	10.2.7 Extended Description	39
	10.2.8 Exit Status	39
	10.2.9 Consequence of Errors	39
10.3	setfmac — Set the MAC Label of a File	39
	10.3.1 Synopsis	39
	10.3.2 Description	40
	10.3.3 Options	40
	10.3.4 Operands	40
	10.3.5 External Influences	40
	10.3.6 External Effects	41
	10.3.7 Extended Description	41

SECTION	PAGE
10.3.8 Exit Status	41
10.3.9 Consequence of Errors	41
Section 11: Information Labeling	43
11.1 getfinf — Get File Information Label	43
11.1.1 Synopsis	43
11.1.2 Description	43
11.1.3 Options	43
11.1.4 Operands	44
11.1.5 External Influences	44
11.1.6 External Effects	45
11.1.7 Extended Description	45
11.1.8 Exit Status	45
11.1.9 Consequences of Errors	45
11.2 getpinf — Get Process Information Label	46
11.2.1 Synopsis	46
11.2.2 Description	46
11.2.3 Options	46
11.2.4 Operands	46
11.2.5 External Influences	46
11.2.6 External Effects	47
11.2.7 Extended Description	47
11.2.8 Exit Status	47
11.2.9 Consequences of Errors	47
11.3 setfinf — Change File Information Label	48
11.3.1 Synopsis	48
11.3.2 Description	48
11.3.3 Options	48
11.3.4 Operands	48
11.3.5 External Influences	48
11.3.6 External Effects	49
11.3.7 Extended Description	49
11.3.8 Exit Status	49
11.3.9 Consequences of Errors	50
Annex E (informative) Revisions to the General Section	51
E.8 Access Control Lists	59
E.8.1 User-Level Utilities	59
E.10 Capability	67
E.10.1 Capability-Related Utilities	67
E.11 Mandatory Access Control	68
E.11.1 General Overview	68
E.11.2 Separate Utilities	68
E.11.3 Label Input and Output	69
E.11.4 Utility use of Capabilities	69
E.12 Information Labeling	69
E.12.1 General Overview	69
E.12.2 Separate Utilities	70

WITHDRAWN DRAFT. All Rights Reserved by IEEE.
Preliminary—Subject to Revision.

SECTION	PAGE
E.12.3 Label Input and Output	70
Annex F (informative) Ballot Instructions	71
Identifier Index	83
Topical Index	88