UNICOS® Multilevel Security (MLS) Feature User's Guide

SG–2111 10.0

New Features

UNICOS® Multilevel Security (MLS) Feature User's Guide SG–2111 10.0

This rewrite of the *UNICOS Multilevel Security (MLS) User's Guide* supports the 10.0 release of the UNICOS operating system.

- The term *Trusted UNICOS* has been replaced with *Cray ML-Safe* to indicate the configuration of the UNICOS system that most closely approximates the B1 evaluated configuration of UNICOS release 8.0.2.
- MLS feature capabilities may also be referred to in the UNICOS release 10.0 documentation set as *security enhancements*.

Record of Revision

Version	*Description*
6.0	December 1990 Original printing.
7.0	July 1992 Reprint with revisions. Information added to support the use of machine-generated password, the new ACL checking algorithm, and the use of the IP security option in networking. Various editoral changes were also made.
8.0	February 1994 Rewrite. Information added to support the UNICOS 8.0 release. See the New Features page for a list of changes.
8.3	January 1995 Editorial changes and information added to support UNICOS 8.3 technical changes. See the New Features page for a list of changes.
9.0	August 1995 Editorial and technical changes. See the New Features page for a list of technical changes.
9.1	November 1995 Editorial and technical changes. See the New Features page for more information.
10.0	November 1997 Editorial and technical changes. See the New Features page for more information.

Contents

Page

Page

Page

Figures

Page

Tables

Preface

This publication documents the UNICOS multilevel security (MLS) feature (also referred to as *security enhancements*) for the UNICOS 10.0 operating system running on Cray PVP computer systems. The Cray ML-Safe system configuration is also described in this manual.

This manual is written for two audiences. First, new users of the UNICOS MLS feature can use it as a stand-alone tutorial/reference manual. Second, the Cray Research Training department is using it as a training document to introduce the UNICOS MLS feature to new users (these users may include people who are becoming system or security administrators). Whenever possible, the text and the accompanying figure are presented in a two-page layout for easy use.

Note: In a future release of the UNICOS system, this manual (*UNICOS Multilevel Security (MLS) Feature User's Guide*, Cray Research publication SG–2111) may not be available as relevant chapters are merged into other UNICOS man pages or administrator guides.

The examples shown in this manual, especially in chapters 3, 4, and 7, build on information found in the preceding example. Studying an example out of order or out of context can cause problems when trying to understand the concept being presented.

Warning: The Cray ML-Safe configuration of the UNICOS operating system supports processing at multiple security labels and system administration using only nonsuper-user administrative roles. The Cray ML-Safe configuration consists of the subset of UNICOS software that offers these capabilities. The Cray ML-Safe name does not imply maintenance of the UNICOS 8.0.2 security evaluation.

This manual contains the following chapters, which are briefly described as follows:

Chapter	Description
Introduction	This chapter introduces the UNICOS MLS feature, defines basic security concepts, security terminology, the U.S. Department of Defense (DoD) criteria used to define security, and explains how the UNICOS MLS feature is implemented on Cray Research mainframes, including an overview of the Cray ML-Safe configuration of the UNICOS operating system.

Logging in and Using Passwords	This chapter describes the login and password procedures used on a UNICOS system. Descriptions are given for interactive logins on both UNICOS and Cray ML-Safe system configurations, logins using the SecurID card, and how to choose passwords. Also the login and password protection features available on a UNICOS system are described.
Using Access Controls Lists (ACL)	This chapter describes the discretionary access controls used by a UNICOS system. How to create, display, remove and duplicate access controls lists (ACL) for your files are explained. Examples are provided showing you how to use the `spacl`(1) and `spset`(1) commands. Also, the chapter explains how ACLs are check to grant or deny access.
Using Security Labels	This chapters describes the mandatory access controls used by a UNICOS system. The UNICOS security policy is defined, plus examples are provided showing you how to use the `setulvl`(1), and `setucmp`(1) commands to change your active security label.
Creating Directories and Files	This chapter describes the rules that must be followed when assigning security labels to files and directories, defines wildcard and multilevel directories, explains the use of these directories on a UNICOS system, provides examples of how to display a file's or directory's security attributes by using the `ls`(1) command, and how to create, remove, and archive files and directories.
Network Access	This chapter describes how the UNICOS MLS feature affects RQS. TCP/IP and NQS controls are explained in other manuals; cross-references are provided.
Miscellaneous Information	This chapter describes how the `cron`(8) and `su`(1) commands work on a UNICOS system. Also, cross-references are provided for finding MLS information for the tape subsystem, the Data Migration Facility (DMF), and Cray/REELlibrarian (CRL) features.

Overview of TCSEC Trusted System Divisions	This appendix provides and overview of the DoD security policy and accountability requirements of the *Trusted Computer System Evaluation Criteria* (TCSEC).

Related publications

The following documents contain additional information that may be helpful:

- *TCP/IP Network User's Guide*, Cray Research publication SG–2009, *publication number*
- *Tape Subsystem User's Guide*, Cray Research publication SG–2051
- *Cray/REELlibrarian (CRL) User's Guide*, Cray Research publication SG–2126
- *NQE Administration*, Cray Research publication SG–2150

Ordering Cray Research publications

The *User Publications Catalog*, Cray Research publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907, or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names. The following list describes the identifiers: 1 User commands 1B User commands ported from BSD 2 System calls 3 Library routines, macros, and opdefs 4 Devices (special files) 4P Protocols 5 File formats 7 Miscellaneous topics 7D DWB-related information 8 Administrator commands Some internal routines (for example, the `_assign_asgcmd_info`() routine) do not have man pages associated with them.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

The following machine naming conventions may be used throughout this document:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems that support this release
Cray MPP systems	All configurations of the CRAY T3D series. The UNICOS operating system is not supported on CRAY T3E systems. CRAY T3E systems run the UNICOS/mk operating system.
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release.

The default shell in the UNICOS and UNICOS/mk operating systems, referred to in Cray Research documentation as the *standard shell*, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS and UNICOS/mk operating systems also support the optional use of the C shell.

Cray UNICOS Version 10.0 is an X/Open Base 95 branded product.

Reader comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail at the following address:

  `publications@cray.com`

- Contact your customer service representative and ask that an SPR or PV be filed. If filing an SPR, use `PUBLICATIONS` for the group name, `PUBS` for the command, and `NO-LICENSE` for the release name.
- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of "Software Publications Group" in Eagan, Minnesota, at fax number +1–612–683–5599.

We value your comments and will respond to them promptly.

Introduction [1]

This manual describes the UNICOS multilevel security (MLS) feature (also referred to as *security enhancements*) and how you, as a nonadministrative user, can use it. This chapter describes the following:

- Basic security concepts
- The U.S. Department of Defense (DoD) criteria used to define security
- The divisions used by the DoD to classify trusted systems
- How the UNICOS MLS feature is implemented on Cray Research mainframes

Warning: In previous releases of the UNICOS operating system, the term *Trusted UNICOS* was used to refer to the configuration that most closely approximated the B1 evaluated configuration of UNICOS release 8.0.2. In the UNICOS 10.0 release, this configuration is referred to as the *Cray ML-Safe configuration* of the UNICOS operating system. Although the Cray ML-Safe configuration of the UNICOS operating system is not an evaluated product, this configuration fully supports all functionality described in the B1 evaluation criteria.

1.1 Introduction to security concepts

The following sections introduce concepts of a trusted (Cray ML-Safe) computer environment and the UNICOS MLS feature.

1.1.1 Concepts of computer security

On traditional UNIX systems, the integrity of both system and user data is preserved by using both simple passwords and permission mode bits. Security is usually left up to the individual user without much intervention on the part of a system administrator. Such a system does not provide many mechanisms for preventing deliberate destruction or corruption of data.

In some computing environments, such a system may not meet the need to protect data, and a trusted environment is used instead. A trusted computer environment should enforce policies that prevent the following (and are summarized in Figure 1, page 3):

- Unauthorized system access - A trusted system should have mechanisms that prevent users from bypassing the authentication process or guessing other user's passwords, plus provide the means necessary to detect such attempts.
- Unauthorized data object access - A trusted system should prevent a user without the correct permissions from gaining access to protected parts of the system (for example, a file, a terminal, or memory).
- Excessive use of system resources - A trusted system should deny a user the ability to monopolize system resources so that other users are denied the resources necessary to perform their work.
- Loss of system integrity - A trusted system should prevent a user from inadvertently or deliberately destroying a data object that is not his or hers to destroy.

The UNICOS MLS feature provides mechanisms not found on a non-MLS UNICOS system to protect both system integrity and sensitive information. The following sections describe the DoD specifications that shaped the UNICOS MLS feature and explain how the feature is implemented on a Cray Research computer system running the UNICOS system.

1.2 DoD criteria for trusted systems

Design specifications for the UNICOS MLS feature were derived from the *Department of Defense Trusted Computer System Evaluation Criteria* (also known as the *Orange book*). These criteria outline the system software capabilities needed to satisfy government security requirements and the divisions that classify computer systems according to how well the security criteria are met.

The evaluation criteria are divided into three categories: security policy, accountability, and assurance.

Computer systems are classified in one of four divisions, D through A, with A representing the most trusted system.

The following sections discuss the criteria in more detail. For more information on the four divisions, see Appendix A, page 97.

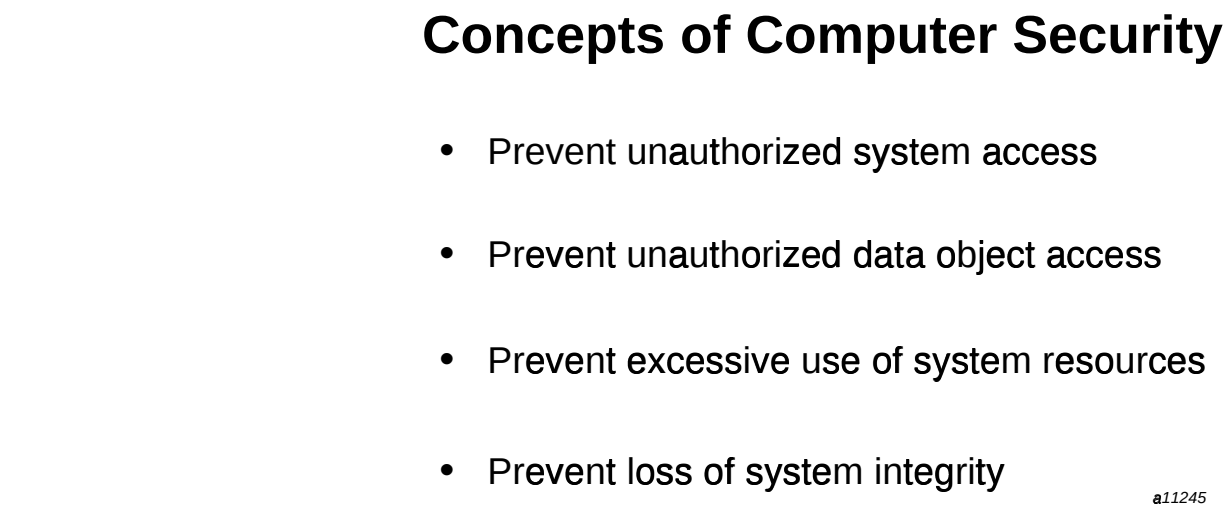

Figure 1. Concepts of computer security

1.2.1 Security policy

The DoD criteria state that a trusted system must enforce an explicit and well-defined security policy. A security policy is defined as the set of rules and practices by which a system regulates the processing of sensitive information. The criteria also state that each subject and object in the system be marked with a clearance or classification label, respectively, for controlling access.

System security can be defined by three security properties: simple security property, *-property (star property), and discretionary security property.

The simple security property states that no subject can read an object unless the subject's security level is greater than or equal to the object's security level and the subject's compartments are a superset of the object's compartments; this is referred to as *no read up*. The UNICOS MLS feature supports this property by using the mandatory access controls.

The *-property states that to write to an object, the object's current security level must be greater than or equal to the subject's security level and the object's compartments must be a superset of the subject's compartments. This property prevents a subject from writing information from one object into another object with a lower classification and is known as *no write down*. The UNICOS MLS feature supports the *-property by using the mandatory access controls.

The discretionary security policy allows a subject with the authority (for example, the owner of an object) to define both the subjects that can access an object and the mode of access. The UNICOS MLS feature supports the discretionary security policy by using discretionary access controls.

1.2.2 Accountability

Individual accountability is the key to securing and controlling any system that processes information on behalf of users or groups of users. The DoD criteria identify the following accountability requirements:

- Individual subjects must be identified and authenticated. The trusted system maintains the identification and authentication information; this information must then be associated with every security-relevant user activity in the trusted system.
- Security audit information must be selectively kept and protected so that events related to security can be traced to the responsible party. This requirement implies a need for a security logging device that permits regular surveillance of system security. It is also essential that an authorized agent (in most cases, the security administrator) be able to selectively access and evaluate security audit information. Of course, audit data must be protected from alteration and unauthorized destruction.

1.2.3 Assurance

The DoD assurance objective states that the computer system must contain hardware and software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces the security policy and accountability objectives. These mechanisms must be continuously protected against unauthorized changes. To satisfy the DoD evaluation criteria, the UNICOS system must guarantee that the security policy is effectively enforced and must be subjected to analysis and tests, the completeness of which can be assured.

1.3 The UNICOS implementation of MLS

The following topics are described in this section:

- An overview of a Cray ML-Safe configuration of the UNICOS system
- An overview of discretionary access control implementation

- An overview of mandatory access control implementation

1.3.1 Cray ML-Safe configuration of the UNICOS system

The Cray ML-Safe configuration of the UNICOS operating system supports processing at multiple security labels and system administration using only nonsuper-user administrative roles. The Cray ML-Safe configuration consists of the subset of UNICOS software that offers these capabilities. The Cray ML-Safe name does not imply maintenance of the UNICOS 8.0.2 security evaluation.

Warning: For the UNICOS 10.0 release, the functionality of the Trusted UNICOS system has been retained, but the `CONFIG_TRUSTED` option, which enforces the conformance to the strict B1 configuration, is no longer available. All references to the Trusted UNICOS system have been replaced by the *Cray ML-Safe configuration* in UNICOS 10.0 documentation.

A Cray ML-Safe configuration provides a fully functional, practical, and usable trusted operating system when integrated into a heterogeneous network set of Cray ML-Safe components. Contact your site security administrator to see if your site is using the Cray ML-Safe configuration.

If your site is using a Cray ML-Safe configuration, this means your system configuration is using many specific security mechanisms that enforce the requirements in the *Trusted Computer System Evaluation Criteria* (TCSEC). These mechanisms include many of mechanisms, such as security labels and access control lists (ACLs), plus the use of the privilege mechanism, multilevel directories (instead of wildcard directories), CrayML-Safe mail, and many other configuration options that are used to define a Cray ML-Safe environment.

In addition, the Cray ML-Safe configuration uses the basic security option (BSO) and common IP security option (CIPSO) at the IP layer to provide packet labeling across a network. This allows connections to other trusted systems in a heterogeneous set of Cray ML-Safe components. The following network services and protocols rely and build on BSO and CIPSO to maintain object labels across a network:

- TCP/IP
- RPC
- NFS
- NQE/NQS
- User sockets

- `telnet`(1B), `rlogin`(1B), `rexecd`(8), `rsh`(1), `rcp`(1), `ftp`(1B), `lpd`(8), and `finger`(1B)

Other major Cray Research products included in the set of Cray ML-Safe components are the GigaRing mechanism, the system workstation (SWS), the I/O subsystem model E (IOS-E), the SSD solid-state storage device model E (SSD-E), the operator workstation (OWS) model E, the maintenance workstation (MWS) model E, the tape subsystem, the Cray Data Migration Facility (DMF), and Cray/REELlibrarian (CRL).

In addition to this manual, you should also read the Cray ML-Safe and MLS (security enhancements) information in the following manuals:

- *TCP/IP Network User's Guide*, Cray Research publication SG–2009
- *Tape Subsystem User's Guide*, Cray Research publication SG–2051
- *NQE Administration*, Cray Research publication SG–2150
- *Cray/REELlibrarian (CRL) User's Guide*, Cray Research publication SG–2126

For the most part, the use of these Cray ML-Safe features should be transparent to you as a nonadministrative user within the restrictions of a Cray ML-Safe configuration of the UNICOS system. If you encounter difficulties, contact your security administrator. Throughout this manual, assume that the explanations apply to both the basic UNICOS configuration and the Cray ML-Safe configuration, unless otherwise indicated.

1.3.2 Discretionary access controls

Discretionary access controls are rules that define a subject's access to an object, based on the subject's identity and groups to which it belongs, and the object's ownership, permissions modes, and access control list (ACL). The mode permission bits (that is, read (r), write (w), and execute (x) bits) are masked against the ACL entries to allow the owner of a file to control both the subjects who can access the file and the type of access (rwx) granted. For a finer granularity of control than that offered by the mode permission bits, the file's owner can also explicitly deny a subject access to the file by making a null/none (n) entry in the ACL for that subject. See Section 3.3, page 55, for more information on this masking operation.

File owners usually establish the discretionary access rules for files they own; however file access is also governed by the mandatory access controls established by the security administrator.

Refer to the `spacl`(1) man page and chapter Chapter 3, page 37, for more information on how ACLs are used and examples that show how to create and maintain ACLs.

1.3.3 Mandatory access controls

Mandatory access controls are rules that control how users access a system in order to prohibit the unauthorized disclosure of any system or user data. The mandatory part of the definition comes from the fact that the enforcement of the controls is done by administrators and the system, and is not left up to the discretion of users as is done with discretionary access controls (described in the previous section).

The UNICOS system uses mandatory access rules to form the UNICOS security policy. This policy controls access based directly on a comparison of the subject's clearance and the object's classification.

Broadly stated, the UNICOS security policy enforces the following rules for nonadministrative users:

- A subject cannot read an object unless the subject's clearance is greater than or equal to the object's classification.
- A subject cannot write or append to an object unless the subject's clearance is equal to the object's classification.
- Any object created by a subject inherits a security label equal to that of the subject.
- A nonadministrative subject cannot lower its own clearance. On a Cray ML-Safe configuration, a nonadministrative subject cannot raise or lower its own clearance.
- A subject cannot change an object's classification.

A subject's clearance and an object's classification consist of security levels and compartments, which form security labels. The security label is the focus of the UNICOS security policy and is defined for you by your security administrator in the user database (UDB).

A security label should be thought of as a single entity and is treated as such when the UNICOS system uses it for access control decisions. However, many of the UNICOS MLS interfaces (that is, user commands) address the component parts (that is, the security level and security compartment) of the label. Because of this situation, it is easier to understand the UNICOS security policy if you

first understand the concepts of a security level and compartment and how they are used to form a security label.

1.3.3.1 Security levels

As stated earlier, a security label consists of a security level and a set of security compartments. For nonadministrative users, a security level can be a hierarchical value from 0 to 16, that indicates the classification of an object or the clearance of a user. System high and system low levels are also used. See Section 1.3.3.6, page 17, for more information on these levels.

Depending on how your UNICOS system is installed, default names are given to each level (`level0`, `level1`, `level2`, and so on). Your security administrator can change these names to accommodate the needs of your site. For example, `level0` could be named `unclassified`, `level1` could be named `classified`, `level2` could be named `secret`, and so on.

In the following example, assume that a user has been assigned a security level of 13 and the system's range is 0 to 16. According to the broadly stated rules in Section 1.3.3, page 7 (and ignoring the use of compartments), this user would be constrained by the following rules:

- The user can read objects with security levels of 13 or less, but cannot read an object with a security level of 14, 15, or 16.
- The user cannot write to an object unless the object's security level is also 13.
- All objects created by the user would be assigned security level 13.

Because of the hierarchical nature of security levels, only one security level can be active for a user at any time; an object can be labeled with only one security level.

1.3.3.2 Security compartments

A security compartment is a nonhierarchical value that indicates the type or topic of information contained in an object for which the subject is cleared. Unlike security levels, you can have more than one compartment active at a time and objects can have more than one compartment assigned to them.

The UNICOS system supports 63 compartments for site definition and use. Compartments can be used to separate groups working on different projects on a UNICOS system. For example, if group one is working on a propeller project, the security administrator could assign the `prop` compartment to all users in that group, while all of group two, working on weather forecasting, could be

assigned the `winds` compartment. A director of both projects could be assigned both compartments.

According to the broadly stated rules in Section 1.3.3, page 7 (and ignoring the use of security levels), users with these compartments would be allowed the following when attempting read access:

- A user with the `winds` compartment has read access to all objects whose compartment sets are a subset of `wind`. This means that a user with the `winds` compartment has read access to an object with the `winds` compartment or an object with an empty compartment set.
- A user with the `prop` compartment has read access to all objects whose compartment sets are a subset of `prop`. This means that a user with the `prop` compartment has read access to an object with the `prop` compartment or an object with an empty compartment set.
- A user with the compartments `winds` and `prop` has read access to an object with an empty compartment set, an object with the `winds` compartment, an object with the `prop` compartment, or an object with a compartment set that includes both `winds` and `prop`.

1.3.3.3 Definition of dominance

With security levels, the concept of equal or not equal is easy to apply, as levels are hierarchical. If a subject's security level is the same as an object's level, they are equal. If not, then one is less than or greater than the other.

Determining this type of relationship between compartments is not easy to do as compartments are not hierarchical. If you use the concept of sets, subsets, and supersets when comparing compartments, the job of comparing compartment sets becomes easier.

To avoid the comparison issues introduced by the hierarchical nature of levels versus the nonhierarchical nature of compartments, it is necessary to understand the concept of dominance between security labels.

A security label is said to *dominate* another security label if the first security label meets the following criteria:

- The first security label has a security level that is greater than or equal to the security level of the second security label.
- The first security label has a compartment set that is equal to or a superset of the compartment set of the second security label.

A security label is equal to another security label if both security labels have exactly the same security level and compartment set.

By using the concepts of a security label and dominance, the broadly stated rules in Section 1.3.3, page 7, translate as follows:

- A subject cannot read an object unless the security label of the subject dominates the security label of the object.
- A subject cannot write to an object unless the security label of the subject is equal to the security label of the object.
- When a subject creates an object, the object inherits the security label of the subject.
- A subject cannot change its own security label to a security label that does not dominate its current security label.
- A subject cannot change the security label of an object.

1.3.3.4 User's security label ranges

You can be assigned a range of security labels that you can use on a UNICOS system. This range is defined in the user database (UDB) by your security administrator in terms of a minimum security level, a maximum security level, a minimum compartment set, and an authorized set of compartments.

In the simplest configuration, the minimum security level and an empty compartment set form your minimum security label; the maximum security level and an empty compartment set form your maximum security label.

You are allowed to log into the system at any label that dominates your minimum label and is dominated by your maximum label. Once you have logged in, however, the subject created on your behalf can only have a single label at any given time.

In addition to your assigned label range, you are assigned a default login label by your security administrator. On a UNICOS system **not** running a Cray ML-Safe configuration, the default label defines the label at which you log in, assuming no other consideration is made. See Section 2.1, page 21, and Section 2.2, page 24, for more information on how your security label is determined.

It is possible to configure the UNICOS system to define a minimum compartment set or to use your default label as the minimum label. This allows your security administrator to define minimum user security labels with

nonempty compartment sets. See your security administrator for information on how your system is configured.

On a Cray ML-Safe configuration, the default label is not very useful, as your login label is based on the label of the socket connection itself and not any label requested by you. See Section 2.2, page 24, for more information on how your security label is determined on a Cray ML-Safe configuration.

1.3.3.5 Examples of security labels

In the previous sections, the concept of a security label was explained. The following sections provide some examples of how labels can be used to protect data on a UNICOS system. These examples show the following:

- The use of a label with an empty compartment set
- A system that uses labels that all contain the same level with different compartment sets
- A system that uses labels with different levels and compartment sets

1.3.3.5.1 Example of using hierarchical security labels

In Figure 2, data is stored in levels 0 through 5 with no compartments. For the following examples, security labels are represented as a level number and a set of compartments separated by a colon. For example, level 3 and a compartment set of AB would be represented as 3:AB.

In Figure 2, a user called Mary has a security label range that allows her to log in at any label from 0:NONE to 4:NONE. Mary's default security label is 3:NONE. Mary's default security label becomes her active label when she logs in.

> **Note:** On a Cray ML-Safe configuration, Mary may not be assigned her default security label when she logs in. Her label depends on the socket connection's security label through which she connects to the Cray ML-Safe configuration. See Section 2.2, page 24, for more information on how your security label is determined. For this example, assume that Mary logged in with a label of 3:NONE.

When Mary logs in at her default label, she can read and change data that has a security label of 3:NONE. She can also read data that has a label of levels 0:NONE, 1:NONE, 2:NONE, or 3:NONE. On a UNICOS MLS system, Mary can also raise her label to 4:NONE.

Note: On a Cray ML-Safe configuration, Mary cannot raise her security label after she logs in. Her process label range, which is derived from the label range of her connection and her UDB label range, allows her to work only at the label at which she logged in. For this example, assume that Mary logs out and logs in to change her security label.

If she raises her label, Mary can change data that has a security label of 4:NONE (although she cannot change data with a label of 3:NONE). She can also read data with a label that has a level of 0:NONE, 1:NONE, 2:NONE, 3:NONE, or 4:NONE.

Mary cannot raise her security label to 5:NONE, as it is out of her defined range. Also, once she raises her label from 3:NONE to 4:NONE, she cannot lower it back; she must log off and log back on to have a label of 3:NONE.

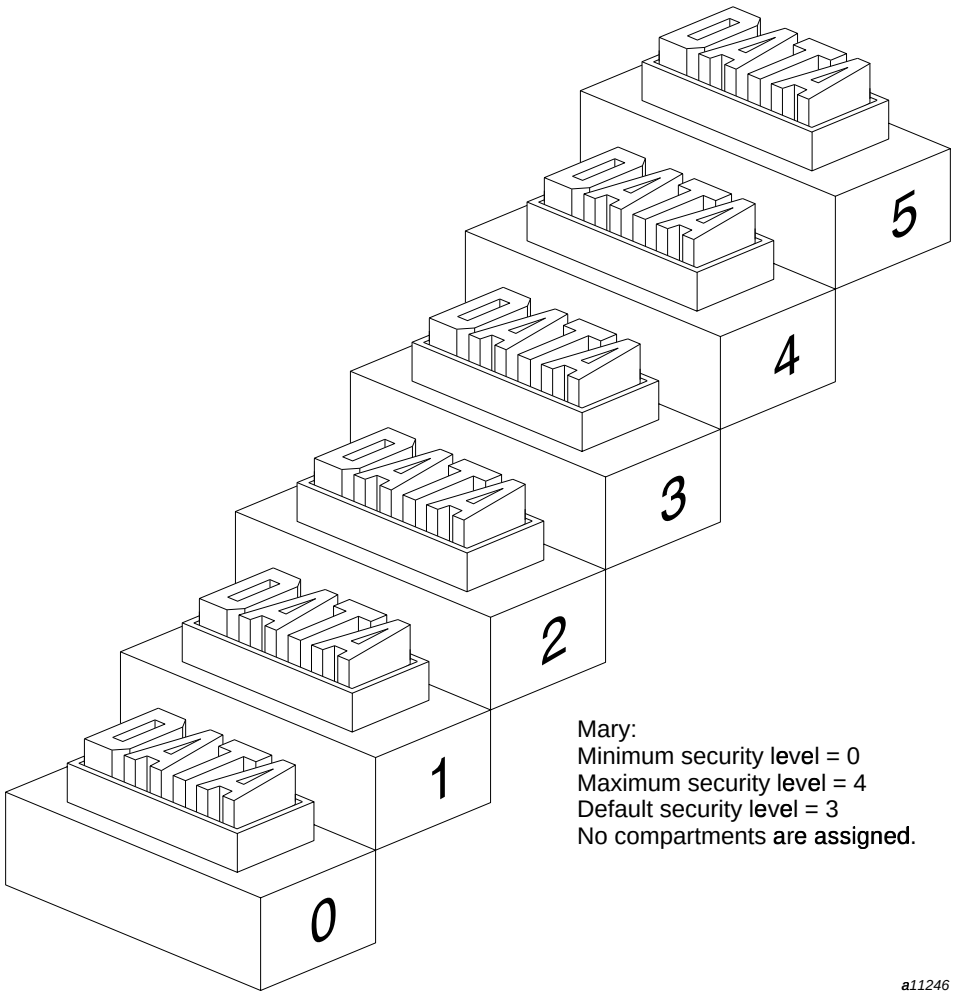

Figure 2. Example of using hierarchical security labels

1.3.3.5.2 Example of using nonhierarchical security labels

In Figure 3, data is labeled with compartments A (Data 1); B (Data 2); C (Data 3); A and B (Data 4); A and C (Data 5); B and C (Data 6); A, B, and C (Data 7); and with no compartments (Data 8); all the labels contain a security level of 0.

In Figure 3, page 14, a user named Mary has been assigned a security label range that includes 0:NONE to 0:AB. Her default security label is 0:B.

Mary can log in at a label of 0:NONE, 0:A, 0:B, or 0:AB. Mary can never log in with a label containing compartment C, because compartment C is not part of her authorized compartment set.

Note: Your security administrator can configure your system so that your default security label is treated as the minimum security label. If this is true for your site, then in this example Mary would not be allowed to log in with a label containing the empty compartment set or a compartment set containing only compartment A. Her only choices would be label of 0:B or 0:AB.

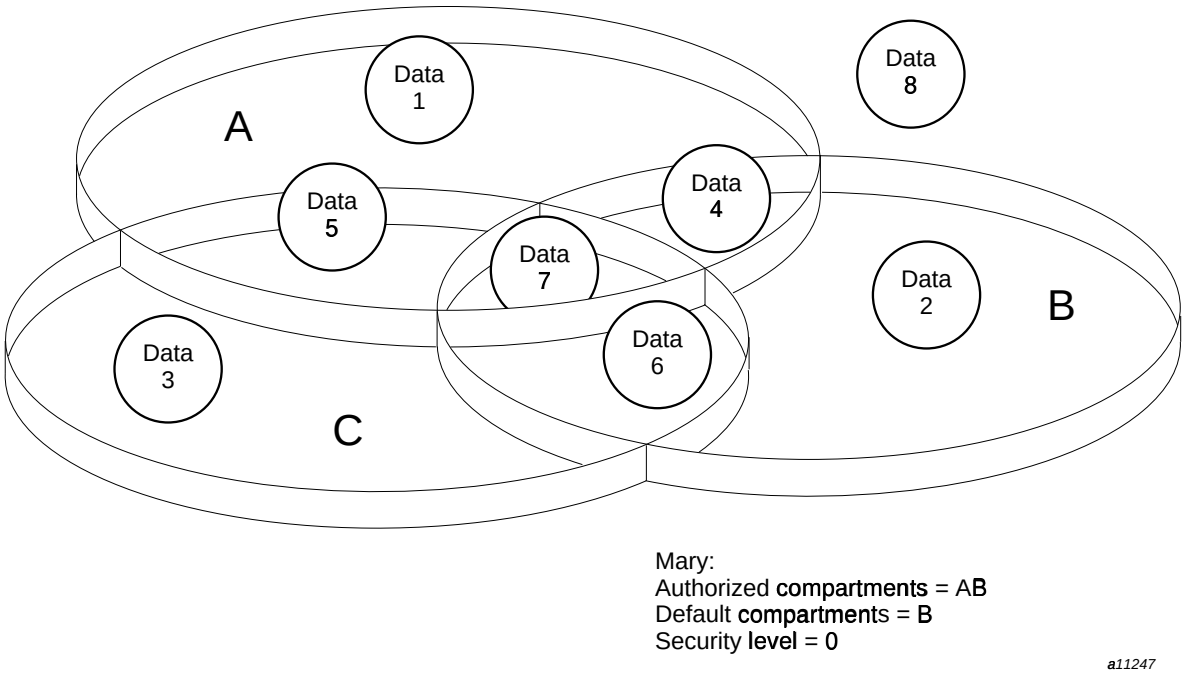

Figure 3. Example of using nonhierarchical labels

Table 1 shows all possible combinations of data that can be read or written (in Figure 3) by users having labels containing all possible combinations of compartments and a level of 0.

Table 1. Accessing data with nonhierarchical labels

Active label	Data that can be read	Data that can be written
0:A	Data 1,8	Data 1
0:B	Data 2,8	Data 2
0:C	Data 3,8	Data 3
0:AB	Data 1,2,4,8	Data 4
0:AC	Data 1,3,5,8	Data 5
0:BC	Data 2,3,6,8	Data 6
0:ABC	Data 1 - 8	Data 7
0:NONE	Data 8	Data 8

1.3.3.5.3 Example of security labels

Figure 4 shows a system on which labels can contain 0:ABC through 4:ABC. Two users, Jack and Jill, have the following security label ranges:

- Jack is allowed labels that range from 0:NONE to 3:AC.
- Jill is allowed labels that range from 0:NONE to 4:BC.

Jack's default label is 3:A and Jill's default label is 3:B. If Jill creates a data object at her default label (3:B), Jack cannot read the data object (assuming he is using his default label), as his label does not dominate the label of the data object created by Jill. The same restriction would apply to Jill if she tried to read a data object created by Jack. Her label (3:B) would not dominate the label of the security object created by Jack (3:A).

The label range defined for Jack allows him to login at a number of different labels (for example, 3:C, 2:C, 3:a, and 3:AC). Jill's label range allows her to log in at a number of different labels also (for example, 3:C, 4:C, 3:B, 4:B, 2:C, and 4:BC).

If Jill logged in at 3:C and created a file, Jack could read the file if he had logged in at 3:C or 3:AC. He could not read the file if he logged in at 3:A or 2:C, as neither of these labels dominates the label 3:C.

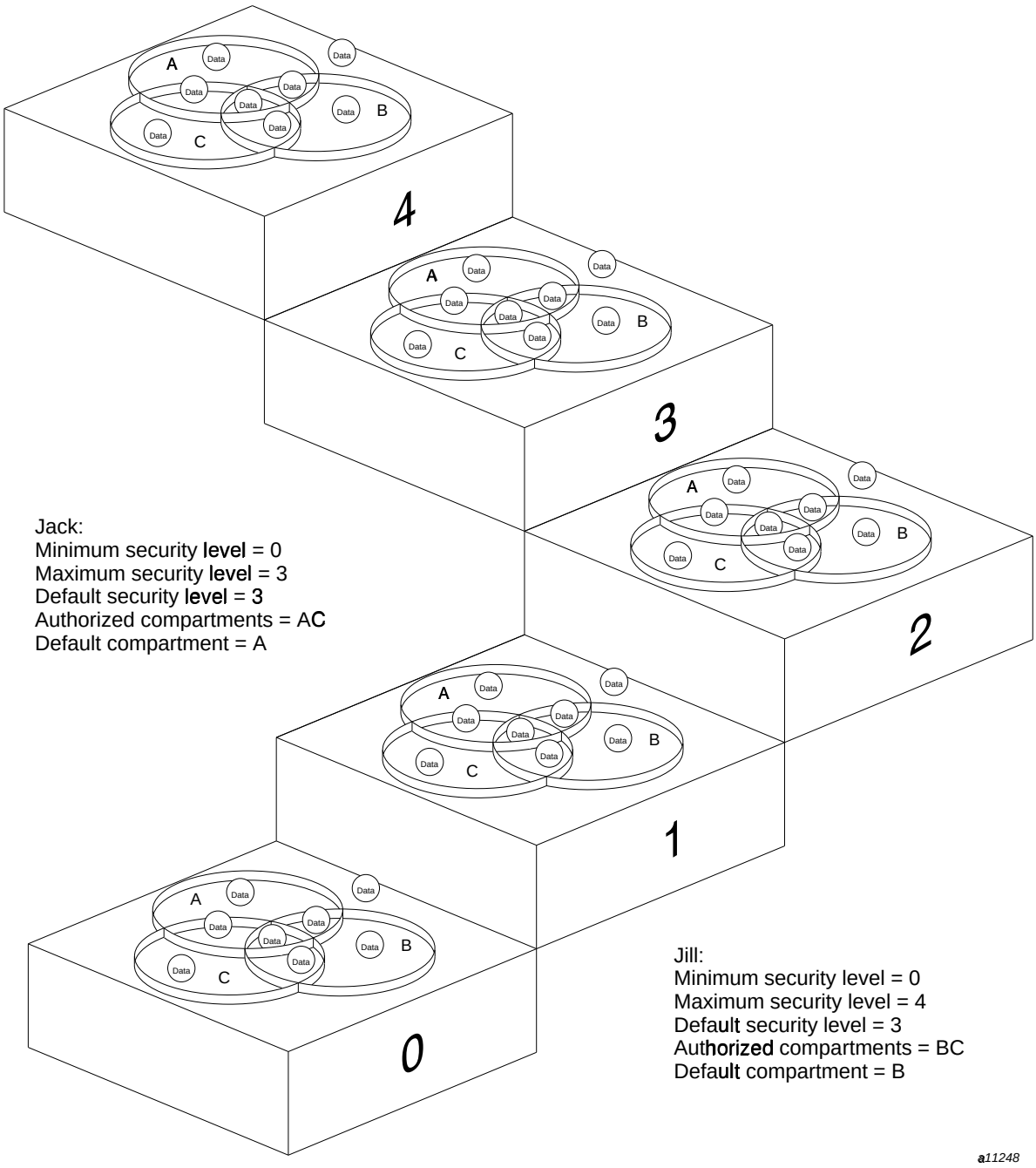

Figure 4. Example of security labels

If Jack logged in at 3:C and created a file, Jill could read the file if she logged in at 3:C, 4:C, or 4:BC, but not if she logged in at a label of 3:B, 4:B, or 2:C, as these files do not dominate the label 3:C.

The important concept to understand from these examples is that both components of the security label must dominate the components of the object's security label in order to read it. So, even if your security level is greater than the object's security level, if your compartment set is not a superset of the object's compartment set, you are not allowed read access to the object.

Although some UNICOS MLS configurations allow you to log in and then change your security label, you may not be able to obtain some labels without logging out and logging in again. For example, if Jill logs in at her default label (3:B), she could not change to a label of 3:C or 2:B, as neither of these labels dominates 3:B. She would have to log off and then log on at 3:C or 2:B.

1.3.3.6 System high and system low security labels

The UNICOS system uses the system high (`syshigh`) and system low (`syslow`) security labels to protect system data and software. These security labels fall outside the range of nonadministrative users.

The `syshigh` label is assigned to system-private databases (for example, `/etc/udb` or the audit log files) and is not dominated by any user security label. This means that system files protected by `syshigh` cannot be read or written to by an unauthorized user.

The `syslow` label is assigned to the majority of Cray ML-Safe binaries, and public databases and directories (for example, `/etc/passwd` or `/bin/cat`). The `syslow` label is dominated by all user labels, but is not equal to any of them. This means that system files protected by `syslow` can be read, but not written to, by unauthorized users, making them available to nonadministrative users, yet protecting the contents of the files.

1.3.4 Special security levels

Your UNICOS system may use a wildcard label (security level 63). This label allows directories to contain files at different security levels. The wildcard security mechanism cannot be used on a Cray ML-Safe configuration; multilevel directories (MLDs) must be used. See Section 5.1.3, page 71, for more information on MLDs.

Security level 51 indicates the system low (`syslow`) label and security level 54 indicates the system high (`syshigh`) label, as explained in the previous section.

1.3.5 Accountability objective

In the UNICOS system, individual accountability is achieved by the following means:

- A user authentication process, activated at the time of system access (for example, login or NQS submission), identifies and authenticates each user; minimum and maximum security levels, an active security level, active compartments, authorized compartments, an active and maximum integrity class, an active category, authorized categories, and a set of permissions granted by the security administrator define the security policy for each authenticated user.
- A security log, maintained by the system, produces an audit trail of user activity. The security log is accessed and processed by the administrative system utility `reduce`(8).

The UNICOS MLS feature provides a set of audit and monitoring utilities that process security log entries and check various other system occurrences to report on individually accountable actions, security policy violations, system integrity, and sensitive information handling. The audit and monitor utilities used are the `reduce`(8), `spfilck`(8), and `spcheck`(8) commands. See the appropriate man page for more information on these commands.

1.3.6 Assurance objective

In the UNICOS system, assurance measures are applied as follows:

- The reference monitor concept is applied within the UNICOS kernel and are always invoked. The reference monitor concept is explained in more detail in the following section.
- The least privilege principle is applied; details of how this principle is applied are found in Section 1.3.6.2, page 19.
- The UNICOS system provides the ability to separate administration functions through the use of categories. See Section 1.3.6.2, page 19.
- Provisions are made to document or audit the use of some covert channels. Information on covert channels can be found in *General UNICOS System Administration*, Cray Research publication SG–2301.

1.3.6.1 Reference monitor concept

The DoD evaluation criteria define a reference monitor concept as an access control concept that refers to an abstract machine that mediates all accesses made by subjects to objects. The reference monitor must be contained entirely in the kernel and must satisfy the following design requirements:

- It must always be invoked.
- It cannot be bypassed.
- It must be safe from tampering.

The UNICOS system supports the reference monitor concept through the entire set of kernel mechanisms that control access to objects, creation of objects and subjects, assignment and use of privilege, as well as the set of kernel mechanisms that maintain subject and object attributes, perform auditing functions, and preserve the isolation of the set of Cray ML-Safe components.

Because the reference monitor resides and executes entirely in the UNICOS kernel, it is always invoked. Because of its location, users cannot tamper with the reference monitor logic or data. It cannot be bypassed.

1.3.6.2 System management

System management is the set of security-related administrative and operational policies and procedures that are needed to maintain system security. One of the procedures needed for system security is the ability to separate operator and administrator functions. The following mechanisms support this need:

Mechanism	Description
The super-user mechanism	This mechanism is the traditional administrative policy that is enforced on UNICOS systems not using the MLS feature. The `root` user ID is used to administer the system. In a UNICOS MLS environment, the super user can override virtually all UNICOS MLS restrictions. This mechanism is often referred to as the `PRIV_SU` system.
The PAL-based privilege mechanism	This mechanism uses privilege assignment lists (PALs) to map active administrative categories to

a set of granular privileges that can be used for the duration of a command execution.

Warning: The UNICOS 10.0 system will support only the following configurations:

- A super-user (`PRIV_SU` enabled) system with PALs
- A nonsuper-user (`PRIV_SU` not enabled) system with PALs

Your site administrator determines which mechanism is used at your site. On a Cray ML-Safe configuration, only a strict PAL-based privilege environment can be used. In general, these mechanisms are used to regulate administrative work, so the impact on nonadministrative users is minimal. For more information on the implementation and use of the mechanisms, see *General UNICOS System Administration*, Cray Research publication SG–2301.

Logging in and Using Passwords [2]

The following chapter describes the login and password procedures used on a UNICOS system with the multilevel security (MLS) feature. The following topics are described:

- Logging in interactively on a UNICOS system
- Logging in interactively on a Cray ML-Safe system configuration
- Your SecurID card
- Choosing passwords
- Login and password protection features available

The following sections also apply to the `ftpd` and `rexecd` daemons (except for the SecurID information), as well as to `login`(1).

2.1 Interactive logins

Note: For information on Cray ML-Safe logins, see Section 2.2, page 24. Some of the information in this section does not apply to Cray ML-Safe logins.

To successfully access the UNICOS system in interactive mode, you must respond with the correct information to the system prompts shown in Example 1, page 23.

You must enter your user name after the `login:` prompt. After your name is entered, the system displays the `Password:` prompt, and you must then enter your correct password. The UNICOS system suppresses the display of your password characters. Your login access can be affected by checks made to the network access list (NAL) and workstation access list (WAL). For more information on the NAL and WAL, see the *TCP/IP Network User's Guide*, Cray Research publication SG–2009.

After successfully entering your name and password, the system displays your active security label, the date and time of day of your last successful login, and where your last valid login originated. If there were any unsuccessful login attempts since the last successful login, the system displays this information also.

If you know that these unsuccessful attempts were not yours, or if you notice any other unusual login problems (such as a successful login that was not performed by you), contact your security administrator immediately.

The final login step is the display of any system messages. A successful login indicates that you have been authenticated for access to the UNICOS system and that `login` has set your clearance. Your security clearance is determined as follows:

- The valid label range for your session is the intersection of your range from the user database (UDB) and the range on the socket connection. Depending on the configuration of your system, the default UDB label can be used as your minimum UDB label.
- If you request a security label, the requested label is the active label for the session. The requested label must be within the range of the session or access is denied.
- If you do not request a security label, your default label is used if it is within the label range of your session. The minimum label of the session is used if the default is not within the range of the session.

To display your current security login environment, use the `spget`(1) command, as shown in Example 1, page 23. The `spget` command, used without options, displays the following information:

- The `permits equal` line displays your security permissions, as defined in the UDB.
- The `security level is` line displays your active security level.
- The `maximum level is` line displays the maximum security level you can activate during your current session.
- The `minimum level is` line displays your minimum security level.

Example 1: Interactive login screen and the `spget` command

```
login:jack
Password:
Active label set to: level2,none

Last successful login was: Wed Oct 7 12:14:29 from training
followed by 2 failed attempts

Welcome to the 8.0 UNICOS system
$
$  spget
permits equal 00000
               none
security level is 2
               level2
maximum level is 16
               level16
minimum level is 0
               level0
authorized compartments are 010
               test
active compartments are 00
               none
integrity class is 0
               class0
maximum class is 0
               class0
active categories are 00
               none
authorized categories are 00000000000
               none
```

- The `authorized compartments are` line displays the maximum set of security compartments that you can activate during your current session.
- The `active compartments are` line displays your active compartments.
- The `integrity class is` line displays your active class. This value is not used.
- The `maximum class is` line displays your maximum class. This value is not used.

- The `active categories are` line displays your active category. If you use the `spget` command immediately after login (that is, before you use the `setucat` command to change your category), this line is your default category, as defined in the UDB.
- The `authorized categories are` line displays your authorized categories, as defined in the UDB.

You can also use the `-L` *requested_label* option of the `login` command to set your security label at the login prompt, where *requested_label* is the label you want as your active label for the session. The `-L` option of `login` cannot be successfully executed on a Cray ML-Safe configuration.

The following example shows Jack specifying a security label with the `-L` option of the `login` command. The active security label is displayed for all logins.

Example 2: Use of the `login -L` command

```
login:jack -L level1,comp24
Password:
Active label set to: level1,comp24

Last successful login was: Wed Oct 8 16:14:30 from training

Welcome to the 8.0 UNICOS system
$
```

If you specify a security label that is outside of your range (as determined by the UDB and NAL), your login request is denied.

2.2 Interactive Cray ML-Safe logins

Logging into a Cray ML-Safe configuration of the UNICOS operating system follows the same procedures outlined in the previous section for logging into a UNICOS system, but there are restrictions regarding your security label range. You cannot successfully use the `-L` option of the `login` command (explained in the previous section) on a Cray ML-Safe configuration.

On a Cray ML-Safe configuration, the IP security option (IPSO) protocol must, for the most part, be used for network connections to the Cray ML-Safe configuration (see Example 1, page 23). This example shows the IPSO connection between two Cray ML-Safe configurations, and between a

compartmented mode workstation (CMW) and one of the Cray ML-Safe configurations. Also shown, is a non-IPSO connection, which is explained later.

On a Cray ML-Safe configuration, your range is still determined by the combination of your UDB entry and range on the socket connection as explained in the previous section. On an IPSO connection, the socket range is determined when it is established, and the range is based on the label of the process that created the socket connection. If you use the `telnet`(1B) command to connect to a Cray ML-Safe configuration at a label outside of this defined range, you are denied access to the Cray ML-Safe configuration (that is, you never reach the `login` prompt sequence shown in the previous section).

On a Cray ML-Safe configuration, your minimum label, maximum label, and active label are all the same value. This means that whatever label you have when you log in is the only one allowed during that session. You cannot successfully use the `setulvl`(1) and `setucmp`(1) commands to change your label. To change your active label, you must log off completely and establish a socket connection at a different security label.

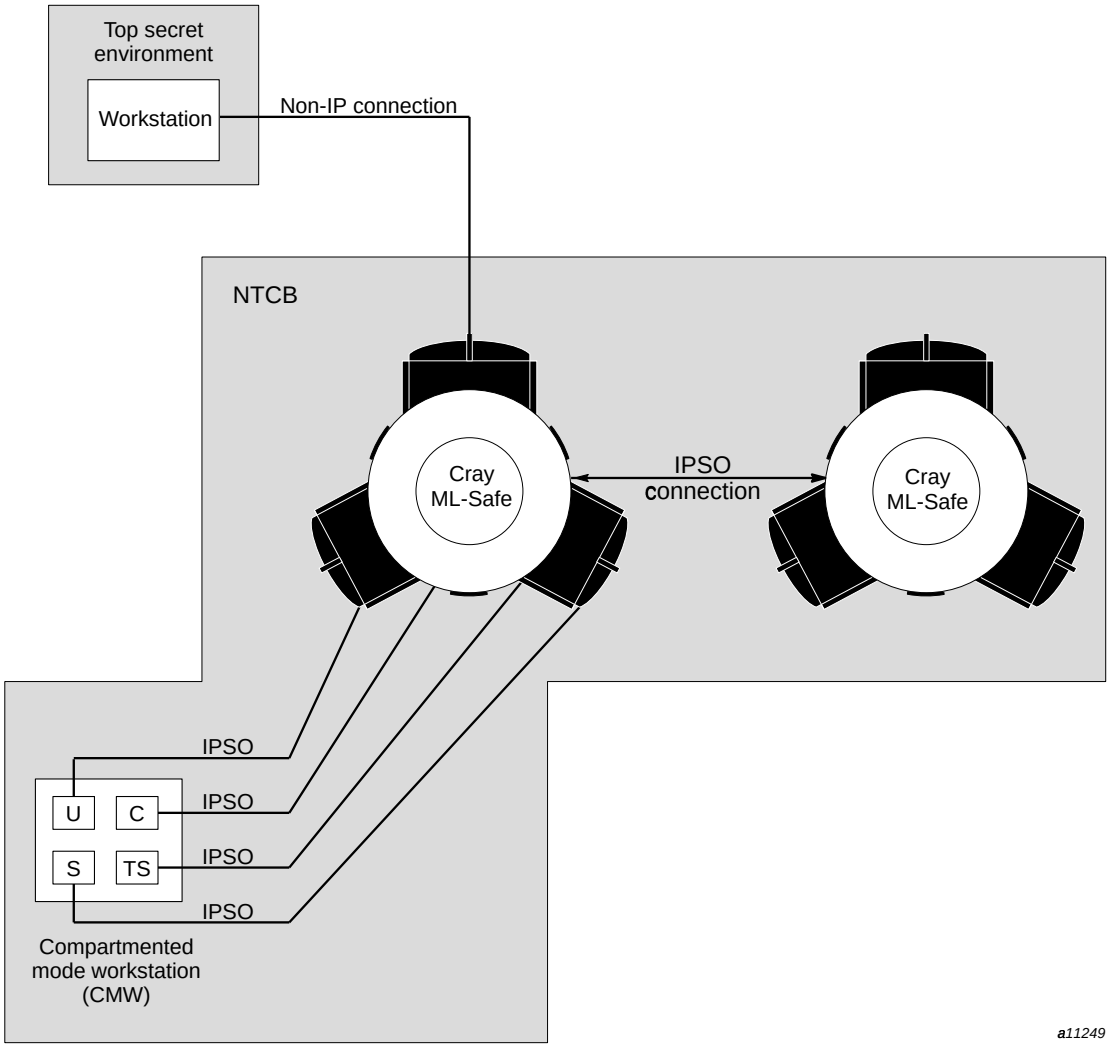

Figure 5. Logging in on a Cray ML-Safe system configuration

In Figure 5, the CMW has four windows, each having a different security label: u (unclassified) c (classified), s (secret), and ts (top secret). Each window has its own IPSO connection into a CrayML-Safe configuration of the UNICOS operating system; each connection can only function at the security label established when the socket was created.

There is limited support for the use of a single-label non-IPSO connection into a CrayML-Safe configuration. Packets on a non-IPSO connection are not labeled and the systems on a non-IPSO network cannot be trusted to label the information they contain.

Because of these limitations, all systems connected to a Cray ML-Safe configuration of the UNICOS system through an non-IPSO network must have the same label, and the network itself must be configured with that label. This allows the use of workstations that do not manage security labels as connections to a Cray ML-Safe configuration without compromising the labeled security of the Cray ML-Safe configuration.

Use of a non-IPSO connection is shown in Figure 5. In this example, a workstation is connected to a top secret, non-IPSO network. All communication on this network occurs at the top secret level, and all workstations on this network are both procedurally and physically secured for the top secret level. If a workstation that was at a secret level were configured into this network, the evaluated rating of the network would be compromised, as this workstation could not differentiate between secret and top secret information, allowing the user of the secret-rated workstation access to top secret information.

2.3 Displaying the operating system's MLS environment

The security level range and valid compartments for the UNICOS operating system are established by your security administrator. Only your security administrator can change these parameters, and all processes are restricted by them. As a user on a UNICOS system, you can work only within the range set for the operating system.

You can display the security environment for your operating system by using the `spget -s` command, as shown in Example 3.

Example 3: Displaying the operating system's security environment (`spget -s`)

```
$ spget -s
system minimum level is 0
               level0
system maximum level is 16
               level16
valid system compartments are 01777
               company
               mktg
               develop
               test
               train
               bnchk
               pubs
               techop
               corp
               cust
```

2.4 Remote logins with SecurID card

The SecurID card, manufactured by Security Dynamics, Inc., can be used on Cray Research systems for interactive logins from remote hosts. This authentication mechanism makes it harder to break into accounts because new passcodes are generated for each authentication and a passcode cannot be used more than one time.

Note: Use of SecurID is optional on a CrayML-Safe configuration of the UNICOS operating system.

Basically, the SecurID card is an electronic device, the size and shape of a credit card (see Figure 6 for an example of a SecurID card).

Figure 6. The SecurID card

For this type of SecurID card, a number appears on the card's LCD display; this number is unique to the card and changes at a predetermined rate. This number, along with your personal identification number (PIN) number, is used in the login sequence or at other times to identify you. It is very important that you do not disclose your PIN by letting others use it or writing it down, as this would allow anyone to use your card to gain access to the system.

The SecurID feature is optional on UNICOS and Cray ML-Safe configurations. Because procedures for using the card can be different for each site, you should contact your security administrator for this information.

2.5 Passwords

The password is the basic tool used by many systems to regulate access to a system. On a secure system, the proper use, protection, and auditing of passwords is the first line of defense in protecting system integrity.

When choosing and using passwords on the UNICOS system, you should observe the guidelines shown in Figure 7, page 31.

There are several system features that can be set by your security administrator that can affect how you use passwords. They are listed in Figure 7, page 31 and explained in the following sections.

2.5.1 Last login notification

At the time of each login, this feature allows the system to display the last login date, the last login time, the number of intervening login failures, and the ID of the terminal at which you last logged in. If you recognize any discrepancy, report it to your security administrator immediately.

2.5.2 Generic login message

This feature allows the system to display a generic `Login incorrect` message when an unsuccessful login attempt is detected. Because it does not explicitly identify the incorrect portion of the login entry, this form of reply makes it harder to guess user names and passwords.

2.5.3 Minimum password size

This feature allows your security administrator to specify a minimum number of characters for a password. To avoid the use of simple passwords, the system default is 6 characters.

Password Guidelines

- Change your passwords often
- Do not let others use your password
- Do not record your password in written form
- Do not choose easy passwords
- Do not use old passwords
- Do not use the same passwords on different systems
- Report any suspicious activity on your login to your security administrator
- Do not use words found in the dictionary
- Use upper/lower case letter and/or alpha/numeric combinations

Password Features

- Last login notification
- Generic login message
- Minimum password size
- Password locking
- Machine-generated passwords
- Login limits/login disable time-out
- Password aging

a11251

Figure 7. Password guidelines and features

2.5.4 Password locking

This feature allows your security administrator to lock your password to prohibit access by your account to the system. This feature is useful if your account is going to be inactive for an extended length of time (for example, while you are on vacation or medical leave) to prevent unauthorized attempts at using your login. If you try to log in after an extended absence and are unable to do so, check with your security administrator to see if your password has been locked.

2.5.5 Machine-generated passwords

If the machine-generated password feature is enabled on your UNICOS system, the `passwd`(1) command executes differently than on other UNICOS systems. This is shown in Example 4.

In this example, you are prompted to enter your old password and then, instead of allowing you to select a new one, the system generates a new password for you. You can select the first password generated or continue to request new passwords by pressing `CR` until a suitable password is generated.

> **Note:** Although the chances are extremely small, the password-generating algorithm can produce a password that may seem offensive to you. The appearance of such a password is random and is not intended to be offensive. You have the choice of rejecting any generated password and picking a subsequent password.

It is important that you do not select a new password in the presence of another person (or, at the very least, shield your screen from the other person's view) when using this feature. Also, when you are done selecting a password, remove or erase the screen, so that no one else can obtain your new password.

Example 4: Machine-generated password example

```
$ passwd
Changing password for jane
Old password:
Your new password is: kudniqui
Re-enter password or (CR) to get another:
Your new password is: keltifok
Re-enter password or (CR) to get another:
Your new password is: onyorja
Re-enter password or (CR) to get another:
Your new password is: rhecirou
Re-enter password or (CR) to get another:
Your new password is: osniyuib
Re-enter password or (CR) to get another:
$
```

2.5.6 Login limit and login disable time-out

Your UNICOS system can use the following site-configurable login features: login limit feature, login disable time-out feature, a delay between failed login attempts feature, and a feature that multiplies the delay between failed login attempts. Your security administrator is responsible for activating and assigning values to these parameters.

The login limit feature defines the number of successive failed login attempts you are allowed before disabling logins from your account.

Use of this feature prevents an unauthorized person from making an unlimited number of attempts at guessing your password. For example, if your security administrator sets the maximum number of failed login attempts feature to 3, you (or a malicious user) are allowed only three consecutive attempts at selecting the correct password. Even if the correct password is selected on the fourth try, the login attempt would not be successful (the generic message, `Login incorrect`, would appear on the screen).

The login disable time-out feature allows your security administrator to define the number of seconds a user is disabled after exceeding the maximum number of failed login attempts. So, if the disable time-out feature is set to 20, and the maximum number of failed login attempts feature is set to 3, after three failed login attempts, you (or a malicious user) would be unable to login until 20-seconds had expired. Then, you are allowed one attempt at logging in. If this attempt fails, the `Login incorrect` message appears for this failed

attempt and any subsequent login attempts that occur prior to the next 20 second interval. This delay continues until a successful login attempt is completed or your security administrator intervenes.

In order to make password guessing more difficult for malicious users, your security administrator can use a feature that defines the number of seconds that must pass between failed login attempts. For example, if the number of seconds is set to 10, then the login prompt would not appear for 10 seconds after each failed login attempt.

Your security administrator can further restrict such attempts by setting a feature that multiplies the number of seconds between failed login attempts by the number of successive failed attempts. This lengthens the delay between each incorrect attempt. For example, using the time set in the previous example, the login prompt would not appear for 20 seconds after the second failed login attempt, 30 seconds after the third failed login attempt, and so on.

The values of these features are site-dependent. It is up to your security administrator to decide how many attempts are allowed and what values are assigned to the features. Regardless of how they are set, it is important for you to watch for and report any suspicious login/password activity to your security administrator as soon as possible.

2.5.7 Password aging

Your security administrator can assign a maximum and minimum number of weeks that your password is valid. The maximum number specifies the maximum number of weeks that you can use your password. When this limit is reached, a message tells you that your password has expired and that you must pick a new one, as shown in Example 5. After picking the new password, the system prompts you to log in again. If your system is using the machine-generated password feature, you are prompted to enter your old password. The system then generates a new password for you.

Your security administrator can also assign a minimum number of weeks that you must use your new password. This feature prevents you from changing to a new password and then immediately changing back to your old one.

Your security administrator can also force you to change your password before the next login attempt.

Example 5: Password aging messages

```
login:jack
Password:
Active label set to: level2,none

Your password has expired.  Choose a new one.
Old password:
New password:
Re-enter new password:
login:
```

Using Access Control Lists (ACLs) [3]

This chapter describes the discretionary access controls used by a UNICOS system. The following topics are described:

- Overview of ACLs on a UNICOS system
- How you can create, display, remove, and duplicate access control lists (ACLs) for your files
- The order in which access control lists (ACLs) are checked to determine the allowed access

3.1 Overview of ACLs

On a traditional UNIX system, discretionary access to a named object is controlled by the object's permission bits. These permission bits define the read (r), write (w), and execute (x) access granted to the owning user, owning group, and all others. Using only this mechanism is too restrictive because it is impossible to define the allowed access for multiple users and/or groups of users.

The UNICOS MLS discretionary access control mechanism uses the access control list (ACL), which provides a method to extend the traditional discretionary access mechanisms. ACLs can be applied to all named objects.

An ACL contains one or more ACL entries; each ACL entry has the following information (shown here in pseudo code for the sake of convenience):

user:*group*:*permissions*:

The *user* field defines the user's login name. The *group* field defines the group name, while the *permissions* field is used to define any combination of r, w, x, or no (n) access. A user's *current groups* are defined as a combination of the effective group and all of the user's group list entries; current groups is also referred to as *member of a specific group* and *belonging to a group* in the following paragraphs.

The ACL entries are intersected with the file's group (mask) bits to determine the type of access allowed; this is called the *effective permissions*. The term *absolute permissions* refers to the permissions defined in an ACL entry. The ACL entry types are shown in the following list:

Entry type	Description
User-only	The absolute permissions defined for a specific user, regardless of the user's current groups. The format for this type is `uid:*:`.
User-group	The absolute permissions defined for a specific user when that user is a member of a specific group. The format for this type is `uid:gid:`.
Group-only	The absolute permissions defined for any user that is a member of the specified group. The format for this type is `*:gid:`.
Owning-group	The absolute permissions defined for the group that owns the file. The format for this type is `*::`.

The owning-group entry type defines the file's group permission bits as the ACL mask. This mask defines the maximum permissions allowed for the object group (an object group being all the ACL entries for an object).

Because no group is specified in the owning-group entry, this entry always pertains to the group that owns the object at the time the discretionary access check is made. If the owning group is changed (by using the `chgrp`(1) command), this owning-group entry then pertains to the new owning group.

> **Note:** When removing an ACL from a file, the owning group's permission is set to the permission granted the owning group with the ACL set. The owning group's permission does not change in the process of removing the ACL.

If there is a group-only entry for the group that owns the file, it is also used to determine the owning-group access. If neither an owning-group or group-only entry from the group that owns the file are found in the ACL, the owning-group permissions default to the ACL mask bits. See Section 3.3, page 55, for more information on the masking operation and the order in which entries are checked.

When you want file access to be controlled by an ACL, you must do the following:

1. Create an intermediate ACL file (by using the `spacl`(1) command).
2. Apply the intermediate ACL file to the file (by using the `spset`(1) command).

An intermediate ACL file is different from the system block where the information is stored. The intermediate file is a formatted version of the ACL that is created and displayed with the `spacl` command. The intermediate file information is copied to the system block when you use the `spset` command. Any changes to an intermediate ACL file are not copied into the system block unless you execute the `spset` command again. See Section 3.2.5, page 48, for more information on these differences.

ACLs can also be applied to Cray/REELlibrarian files and volumes. See the *Cray/REELlibrarian (CRL) User's Guide*, Cray Research publication SG–2126, and the `spset`(1) and `spacl`(1) man pages, for more information.

ACLs can also be applied to IPC objects (shared memory segments, message queues, and semaphores). See the `spset`(1) and `spclr`(1) man pages for more information.

3.2 Maintaining ACLs

You can use the `spacl` command to create, display, remove, or duplicate entries for a specified ACL. The `spset` command applies the ACL to a file.

The following sections explain how to use these commands. Each example in the following sections is built on knowledge defined in the previous section. Examining the examples out of sequence is not recommended.

3.2.1 Creating entries in an intermediate ACL file

The `spacl` command uses the following options to create entries in an intermediate ACL file:

- The `-a` *aclfile* option interactively adds entries to an intermediate ACL file. The *aclfile* argument is the name of the resultant intermediate ACL file.
- The `-i` *modfile aclfile* option inputs ACL text file changes from *modfile* into *aclfile*. The *modfile* argument is a file with add and/or remove statements. The *aclfile* argument is the name of the resultant intermediate ACL file.
- The `-t` *tmodfile aclfile* option inputs ACL text file changes from *tmodfile* into *aclfile*. The *tmodfile* argument is a file of an ACL display format from a previous execution of a `spget -a` or `spacl -l` command. The *aclfile* argument is the name of the resultant intermediate ACL file. The `-t` option is explained in Section 3.2.7, page 52.

3.2.1.1 Interactive creation of intermediate ACL files (`spacl -a`)

The `spacl -a` command interactively adds new entries to the intermediate ACL file (*aclfile*) by prompting you for the user name, group name, and permissions for each ACL entry. You can use the wildcard (*) character to specify all users in a group or a specific user in any group; specifying a wildcard character for both the user and group (*:*) in a single entry is not allowed. Specifying *:: defines the owning-group entry.

Example 6 shows how to create an intermediate ACL file called `myacl` with four entries. At the `enter user's name OR an *` prompt in the first entry, the user called `tnn` is specified. The `spacl` command checks the specified user name against the valid choices listed in the user database (UDB). If you specify an invalid choice, the system prompts you for another choice.

At the `enter group name OR an *` prompt in the first entry, the wildcard (*) character specifies that `tnn` can belong to any group. The group entry is checked against the entries in `/etc/groups` to ensure it is a valid choice. If you specify an invalid choice, the system prompts you for another choice.

Finally, at the `enter access mode (n = none)` prompt, you must specify the access modes for the entry. You can specify any combination of `r`, `w`, and `x`, or `n`. The `rw` definition in the example results in absolute read and write access being allowed for `tnn` to the file protected by `myacl`.

The second entry allows a user called `jog`, who must belong to the `trng` group, to be allowed absolute permissions of read and write.

The third entry allows any user (specified by use of the wildcard character), who belongs to the `trng` group, to be allowed absolute permission of read.

The fourth entry defines the owning-group entry. This entry allows the owning group absolute permission of read and write.

As shown in the fifth entry, you cannot specify a wildcard character for both the user name and group. If you do, the `NO user or group name entered` message appears. Also, as shown in the sixth entry, duplicate user name and group entries are not allowed. If you try to enter duplicate entries, the `duplicate entry NOT accepted` message appears.

Example 6: Creating ACL entries (`spacl -a`)

```
$ spacl -a myacl

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

enter user's name OR an *......tnn
enter group name OR an *......*
enter access mode (n = none)......rw


ADD MODE

enter user's name OR an *......jog
enter group name OR an *......trng
enter access mode (n = none)......rw

ADD MODE

enter user's name OR an *......*
enter group name OR an *......trng
enter access mode (n = none)......r

ADD MODE

enter user's name OR an *......*
enter group name OR an *.....
enter access mode (n = none)......rw
```

Entering `quit` at any point saves the changes and exits the interactive session (as shown in the last entry). Executing a `CONTROL-C` discards all changes and exits the session.

The absolute permissions defined in ACL entries are always intersected with the file's mask bits to determine the requester's effective access permissions. See Section 3.3, page 55, for more information on the masking operation and the order in which entries are checked. All of the following examples in this section are showing the absolute permissions.

3.2.1.2 Creation of intermediate ACL file entries using an input file (`spacl -i`)

The `-i` *modfile aclfile* option inputs ACL edit statements through *modfile* into *aclfile*. The following is the format for adding a record (line) in *modfile*:

`a:`*user_name*`:`*group_name*`:`*access_mode*`:`

Example 7, page 42 shows how to use the `spacl -i` command. When using the `-i` option, the user name, group name, and permission guidelines presented in the previous section apply.

In Example 7, page 42 the intermediate ACL file called `myacl2` is created and contains the same entries as the intermediate ACL file called `myacl` in the previous example, except for the last entry, which denies `root` access. See Section 3.3.2, page 59, for more information.

Example 7: Creating ACL entries (`spacl -i`)

```
$ cat modfile
a:tnn:*:rw:
a:jog:trng:rw:
a:*:trng:r:
a:*::rw:
a:root:*:n:
$ spacl -i modfile myacl2
spacl: end of file on modfile
```

3.2.2 Displaying intermediate ACL files (`spacl -l`)

You can display the contents of an intermediate ACL file by using one of the following methods:

- The `spacl -l` *aclfile* command, which lists a long version of the ACL entry.
- The `spacl -s` *aclfile* command, which lists a condensed version of the output shown when using the `-l` option.

The display of these two commands is shown in Example 8. The main difference between the two displays is that the uid and gid numbers (that correspond to the user name and group name, respectively) are shown in the `-l` display, but not in the `-s` display. The uid and gid numbers are unique identification numbers assigned to users and groups by your system administrator.

Example 8: Displaying an intermediate ACL file (`spacl -l` and `-s`)

```
$ spacl -l myacl
# ACL owner's uid: 10505
# ACL owner's name: ben

uid:1822     gid: -   user = tnn    group = *    mode = rw-
uid:927      gid: 28  user = jog    group = trng mode = rw-
uid:-        gid: 28  user = *      group = trng mode = r--
uid:-        gid: -   user = *      group =      mode = rw-

$ spacl -s myacl
# ACL owner's name: ben

user = tnn     group = *      mode = rw-
user = jog     group = trng   mode = rw-
user = *       group = trng   mode = r--
user = *       group =        mode = rw-
```

3.2.3 Removing entries in intermediate ACL file

You can use the following `spacl` options to remove entries from a specified intermediate ACL file:

- The `-r` *aclfile* option interactively removes entries from an intermediate ACL file. The *aclfile* argument is the name of the resultant intermediate ACL file.
- The `-i` *modfile1 aclfile* option inputs ACL text file changes from *modfile1* into *aclfile*. The *modfile1* argument is a file with add or remove statements. The *aclfile* argument is the name of the resultant intermediate ACL file.

3.2.3.1 Interactive removal of intermediate ACL files (`spacl -r`)

The `-r` option removes entries from the *aclfile*. As with the `-a` option, you are prompted for a user name/group name pair that exists in the specified ACL. The permission prompt does not appear when using the `-r` option.

When removing entries, a question mark (`?`) is used as a wildcard character; specifying `?:?` is not allowed. The `*` is used to match user names and/or group names that were specified that way in the original entry.

Use of the `-r` option is shown in Example 9, page 45. First, the contents of `myacl2` (created in Section 3.2.1.2, page 42) are displayed by using the `spacl -l` command.

Then, by using the `spacl -r` command, the first entry removes the user called `tnn` in any group (specified by the `*`) from `myacl2`. The second entry removes all users (specified by the `?`) in the `trng` group from `myacl2`, which results in both the `jog:trng` and `*:trng` entries being removed. The third entry removes the owning-group entry.

These modifications produce an intermediate ACL file with only one entry, as shown in Example 9. This is the entry that denies `root` access.

Executing a `CONTROL-C` discards all changes and exits the interactive session. Entering `quit` saves the changes and exits the interactive session.

Example 9: Removing an intermediate ACL file entry (`spacl -r`)

```
$ spacl -l myacl2
# ACL owner's uid: 10505
# ACL owner's name: ben

uid:1822     gid: -   user = tnn    group = *    mode = rw-
uid:927      gid: 28  user = jog    group = trng mode = rw-
uid:0        gid: -   user = root   group = *    mode = n
uid:-        gid: 28  user = *      group = trng mode = r--
uid:-        gid: -   user = *      group =      mode = rw-
$ spacl -r myacl2

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

REMOVE MODE

enter user's name OR a ? OR an *......tnn
enter group name OR a ? OR an *.......*

REMOVE MODE

enter user's name OR a ? OR an *......?
enter group name OR a ? OR an *.......trng

REMOVE MODE

enter user's name OR a ? OR an *......*
enter group name OR a ? OR an *.......

REMOVE MODE

enter user's name OR a ? OR an *......quit
entry session terminated

$ spacl -l myacl2
# ACL owner's uid: 10505
# ACL owner's name: ben

uid:0        gid: -   user = root   group = *    mode = n
```

3.2.3.2 Removing entries in intermediate ACLs files using an input file (`spacl -i`)

The `-i` *modfile aclfile* option inputs ACL edit statements through *modfile* into *aclfile*. Using this option to remove entries is similar to the add example shown in Section 3.2.1.2, page 42. The following is the format for removing a record (line) in *modfile*:

`r:`*user_name*`:`*group_name*`:`

You can use the question mark (`?`) when removing all instances of a user or group, as shown in the `spacl -r` example; specifying `?:?` is not allowed.

3.2.4 Modifying intermediate ACL files

If you want to modify an existing intermediate ACL file, you can do so by in one of the following ways:

- If you are adding a new entry, use the `spacl -a` command as described in Section 3.2.1.1, page 40.
- If you want to delete an entry, use the `spacl -r` command as described in Section 3.2.3.1, page 43.
- If you want to change the access mode of an existing ACL entry, you must first remove the entry by using the `spacl -r` command and then recreate the entry with the new access mode by using the `spacl -a` command.

Example 10 shows how to modify the ACL entry for `jog`. First, the entry for `jog` is created to allow read access by using the `spacl - a` command. Second, the entry for `jog` is removed by using the `spacl -r` command. Last, the entry for `jog` is recreated (again, by using the `spacl -a` command), but this time both read and write access are allowed.

Example 10: Modifying an existing ACL entry

```
$ spacl -a myacl2

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

ADD MODE

enter user’s name OR an *......jog
enter gruop name OR an *......trng
enter access mode (n = none)...r

enter user’s name OR an *......quit
entry session terminated

& spacl -r myacl2

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

REMOVE MODE

enter user’s name OR a ? OR an *......jog
enter gruop name OR an *......trng

REMOVE MODE

enter user’s name OR a ? OR an *......quit
entry session terminated

& spacl -a myacl2

ENTER "quit" to END SESSION, "ctrol-c" to ABORT

ADD MODE

enter user’s name or an *......jog
enter group name OR an *......trng
enter access mode (n=none).....rw

ADD MODE

enter user’s name OR a ? OR an *.....quit
entry session terminated
```

3.2.5 Applying ACLs to files (spset -a)

Once an intermediate ACL file has been created or modified, you must apply it to the file by using the `spset -a` *aclfile files* command. You must be the owner of the file (or the security administrator) to apply an ACL to a file. The `spset -a` command copies the contents of an intermediate ACL file to an ACL system block. The system block is defined by a pointer in the inode of the ACL-protected file.

An ACL system block is not shared by multiple files in the file system, even though the same intermediate ACL file can be applied to multiple files by using the `spset -a` command.

As shown in Figure 8, `myacl` is the *aclfile* (intermediate ACL file). It has an inode associated with it, which identifies the data block(s) from which the intermediate ACL file information is copied when the `spset -a` command is executed.

Figure 8 also shows that a system block is allocated for `file1` and `file2`. The address of the system block is stored in the inode of each file; this is the pointer to the ACL. The system block ACL information is transparent to you as a user and can be accessed only by the system kernel.

The contents of `myacl` are converted into binary and copied into the system block for each file each time the `spset -a` command is executed. If any of the entries in `myacl` are modified, the new information is not automatically copied into the system block unless you use the `spset -a` command again.

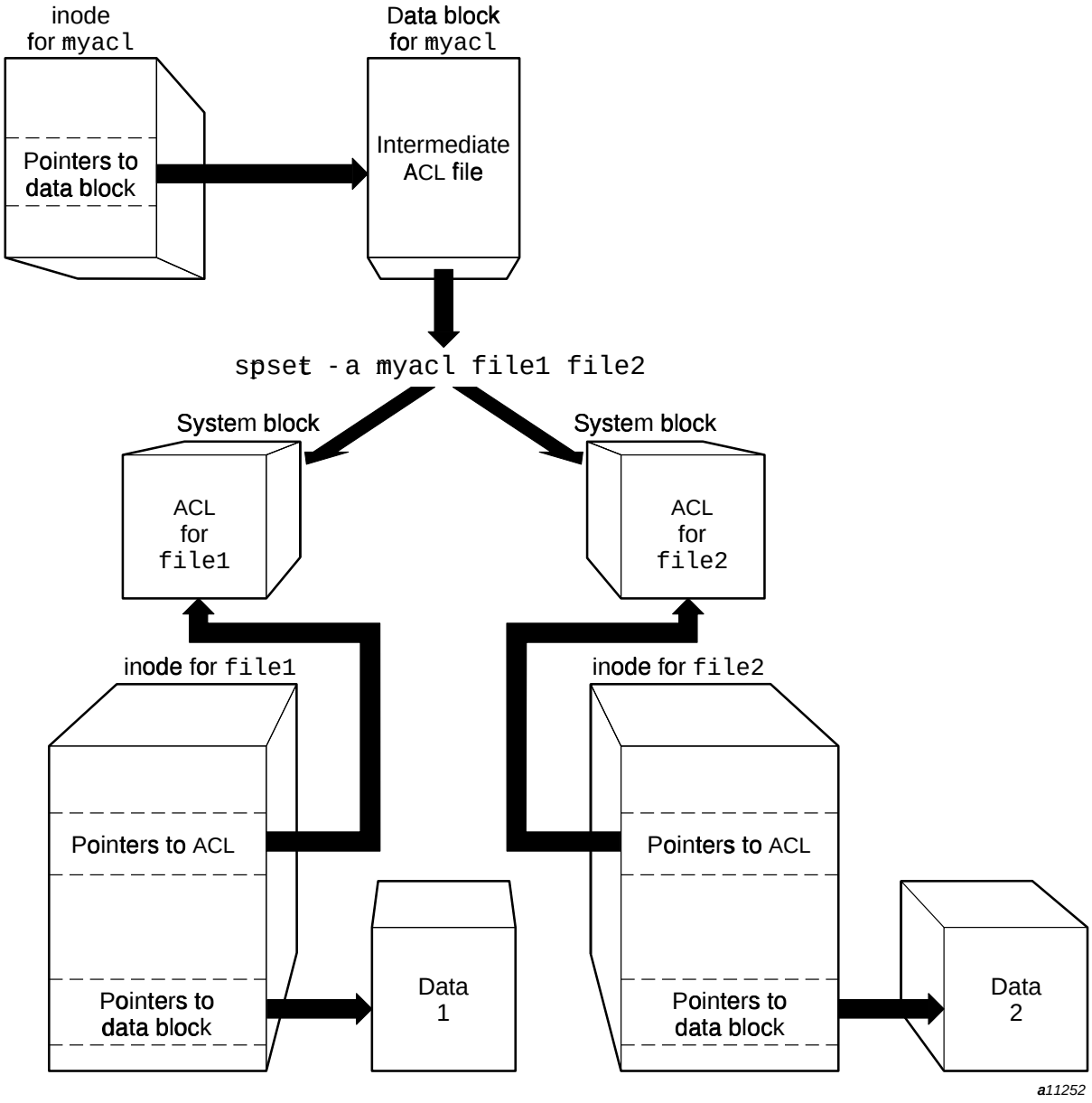

Figure 8. Connections between ACLs and files

Example 11 shows how to use the `spset -a` command. In this example, `myacl` is first displayed by using the `spacl -l` command and then applied to the file called `newfile` by using the `spset -a` command.

As stated previously, any changes made to an intermediate ACL file are not automatically copied into the system block. In Example 12, changes are made to `myacl` by using the `spacl -a` command to add the `jack:*:r` entry.

To update the system block with this new information, the `spset -a` command must be executed again. Because this file is already protected by an ACL, the system will ask you if you want to replace it. Notice that the last step in Example 12 removes `myacl`. This does not affect `newfile` because it has its own copy of the ACL entries.

Example 11: Applying ACLs (`spset -l`)

```
$  spacl -l myacl

#ACL owner's uid: 10505
#ACL owner's name: ben

uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
$ spset -a myacl newfile
```

Example 12: Applying ACLs (`spset -a`)

```
$ spacl -a myacl

ENTER "quit" to END SESSION, "ctrl-c" to ABORT

ADD MODE

enter user's name OR an *......jack
enter group name OR an *.......*
enter access mode (n=none).....r

ADD MODE

enter user's name OR an *......quit
entry session terminated
$ spset -a myacl newfile
spset: replace existing acl for file? y
$ rm myacl
```

3.2.6 Displaying ACLs applied to files (`spget -a`)

If you are unsure which files have ACLs applied to them, use the `ls -e` command as shown in Example 13. An `a` appears prior to the file name if an ACL is applied. If you use the `ls -le` command, the `a` appears immediately after the UNICOS mode permissions. See Section 5.2, page 78, for more information on the `ls` command.

You can display the ACL entries by using the `spget -a` or `spget -ar` commands, as shown Example 14. The `spget -ar` command displays a reduced format, which is useful when duplicating ACL entries (an example of this is shown in the next section).

Example 13: Displaying ACL entries (`spget -e`)

```
$ ls -e
  0 file
  0 modfile
  0 myacl2
a 0 newfile
  0 testfile
$ ls -le newfile
-rw-------a  0   1 ben   trng     60  May 16 15:10  newfile
```

Example 14: Displaying ACL entries (`spget -a`)

```
$ spget -a newfile
# ACL Information for: newfile
uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-

$ spget -ar newfile
#ACL information for: newfile
user = tnn     group = *     mode = rw-
user = jack    group = *     mode = r--
user = jog     group = trng  mode = rw-
user = *       group = trng  mode = r--
user = *       group =       mode = rw-
```

3.2.7 Duplicating ACLs (`spset -d` or `spacl -t`)

It may be necessary to duplicate the contents of an ACL. An example of this is if you want to apply an ACL to another of your files, but you cannot locate the intermediate ACL file.

There are two ways you can duplicate an ACL from one file and apply it to another file. The first way is to use the `spset -d` command; the second way is a three-step process using the `spget -ar`, `spacl -t`, and the `spset -a` commands. The second method generates an intermediate ACL file.

To duplicate the ACL permissions found in the ACL applied to one file and apply them to another file, use the `spset -d` *file1 file2* command, as shown in Example 15. In this example, *file1* is the file that has the ACL that you want to duplicate for *file2*.

Example 15: Duplicating ACLs (`spset -d`)

```
$ spget -a newfile
# ACL Information for: newfile
uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
$ spset -d newfile testfile
$ spget -a testfile
# ACL Information for: testfile
uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
$ ls -le testfile
-rw-------a  0   1 jack  trng     77  May 16 15:15  testfile
```

To duplicate an ACL from one file to another, even though the intermediate ACL file has been removed from the directory, you can use the following three-step process (shown in Example 16):

1. Create a file that contains the ACL information in a reduced format. You do this by using the `spget -ar` command to redirect the output of a file's ACL into a text file. The `-r` option (which can only be used with the `-a` option) gets a reduced version of the ACL information.

2. Create an intermediate ACL file. You do this by using the `spacl -t` command to convert the reduced format from step 1 into an intermediate ACL file.

3. Apply the ACL file created in step 2 to the new file. You do this by using the `spset -a` command.

When using this method, you can edit the reduced format `text` before issuing the `spacl -t` command. Use a text editor to add or remove entries or modify

existing entries. Thus, a similar, but not identical ACL is applied to `file` when the `spset -a` is executed.

Example 16: Duplicating ACLs (`spacl -t`)

```
$ spget -ar testfile > text
$ spacl -t text myacl
spacl:  end of file on text
$ spacl -l myacl
#ACL owner's uid: 10505
#ACL owner's name: ben

uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
$ spset -a myacl file
$ spget -a file
# ACL Information for: file
uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
```

3.2.8 Removing ACLs (`spclr -a`)

You can use the `-a` option of the `spclr`(1) command to remove an ACL from one or more files, as shown in Example 17.

The `spclr` command can fail for the following reasons:

- You are not the owner of the file or you are not the security administrator.
- An ACL is not applied to the file.
- An invalid ACL was detected.

If you do not have the correct permission to access the directory in which the file is located, you will get a `permission denied` message. If you are not the owner of the file (or are not an active security administrator), and you try to remove an ACL, you will get a `spclr: acl remove error for` *filename* message.

Example 17: Removing ACLs from files (`spclr -a`)

```
$ ls -le newfile
-rw-------a  0   1 ben  trng     77  May 16 15:15  newfile
$ spclr -a newfile
$ ls -le newfile
-rw-------   0   1 ben  trng     77  May 16 15:15  newfile
```

3.3 How ACLs are checked

The order in which ACL entries are searched and the masking operation is shown in Figure 9. The following explanation follows the numbered sequence shown in the figure.

When you attempt to gain access to a file, the ACL of the file is searched. The first check made in the ACL is to see if you are `root` (this step is not shown in Figure 9). Depending on your system configuration, one of the following occurs:

- On systems using the super-user mechanism, `root` is always granted DAC access to a file, even if there is an entry for `root` in the ACL.
- On systems using the PAL-based privilege mechanism, `root` is not given any special consideration for DAC access. It is treated the same as an entry for any user and the ACL can be used to determine (or deny) access for `root`. A process is allowed to override the DAC restrictions only if the process belonging to `root` has the `PRIV_DAC_OVERRIDE` privilege in its effective privilege set.

If you are not `root` or your system is using the PAL-based privilege mechanism, then a check is made to determine if you are the owner of the file (shown in step 1 of Figure 9). If you are, the access specified by the file's owner permission bits are granted. The ACL is not checked in this case; regardless of the ACL entry, the type of access that a file's owner is granted can be determined by using the `ls -l` command.

If you are not the owner of the file, then the ACL is searched for the set of `uid:gid:` and/or `uid:*:` entry or entries that relates to you and your current groups (shown in step 2 of Figure 9). The current groups are your effective gid and your group list. If there is a `uid:gid:` entry or entries for you and any of your current groups, the entry or a union of these entries is intersected with the file's mask bits to determine the type of access allowed.

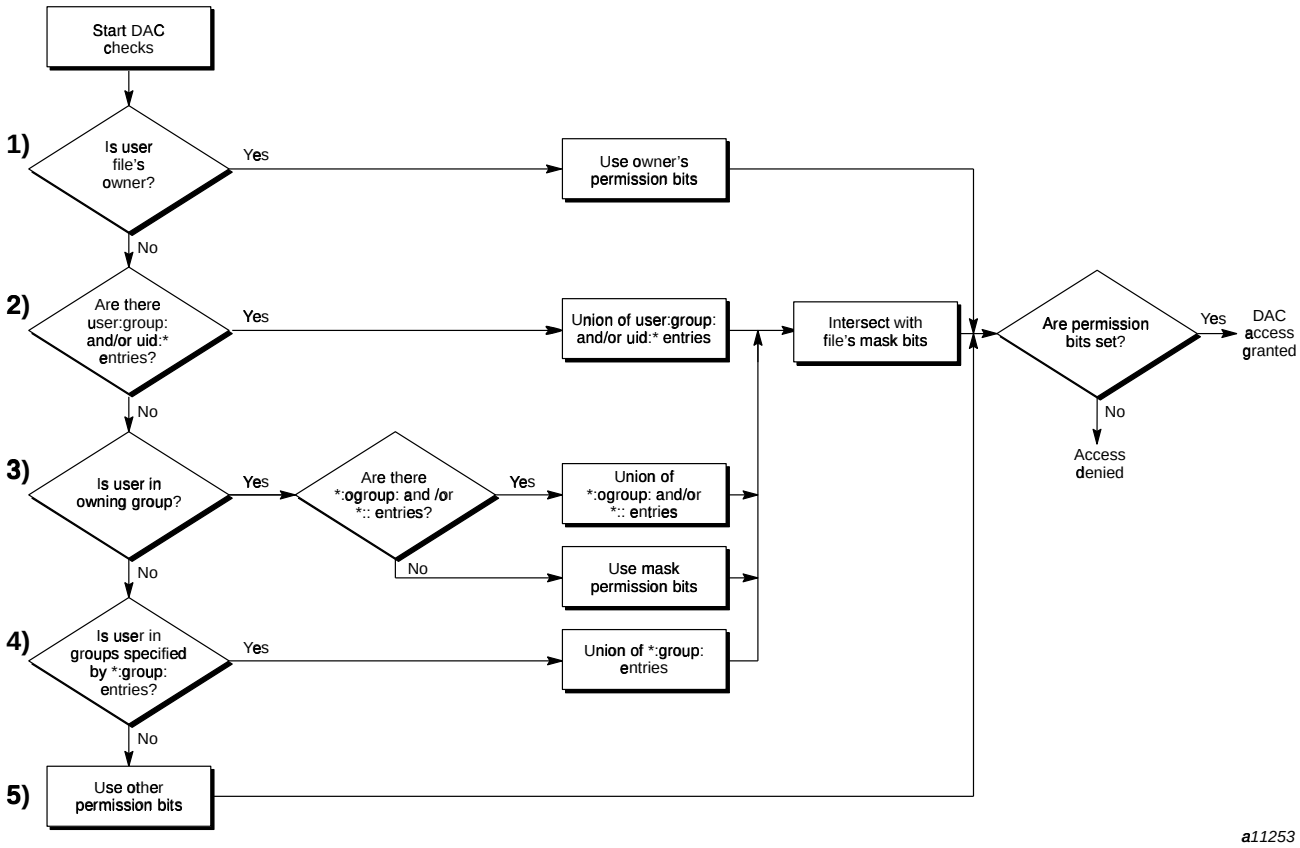

Figure 9. Flowchart of how UNICOS ACLs are checked

An example of this check is shown with the following ACL entries for a user named Jack with groups training and testing wants to access a file that has its mask bits set to `r-x`. Assume for this example that Jack has only the following entries defined for him:

```
user: jack   group: training   mode: r--
user: jack   group: testing    mode: -w-
user: jack   group: *          mode: --x
```

The union of these entries (which would be `rwx`) is intersected with the file's mask bits, resulting in Jack having `r-x` access to that file. If Jack requests write access to this file, it is denied.

The union of multiple entries is used because of the fact that the UNICOS operating system supports multiple groups. Thus, users can simultaneously belong to all of their groups and no single group takes precedence over any other group.

If no match was found in step 2, then a check is made to see if you belong to the owning group of the file (shown in step 3 of Figure 9). If yes, a search is made for a `*:ogroup:` (where `ogroup` is the group-only entry for the owning group) and/or a `*::` entry (where `*::` is the owning-group entry). If either of these entries is found, the entry or a union of these entries is intersected with the file's mask bits to determine the type of access allowed.

If neither a `*:ogroup:` or `*::` entry is found, then the file's mask permissions are used to determine the type of access allowed to the member of the owning group.

If you do not belong to the owning group, then the ACL is checked for a `*:group:` entry that matches your current groups (shown in step 4 of Figure 9). If one or more matching entries are found, then the entry or a union of all `*:group:` entries are intersected with the file's mask bits to determine the type of access allowed.

If no matching `*:group` entry is found, then access is granted according to the file's other permission bits (shown in step 5 of Figure 9). There is no intersection with the file's mask bits in this check.

A user or group can be denied permission by entering `n` in the ACL, as shown in the following example:

```
jack : * : n
```

As explained previously, multiple entries for the same user or group are combined for the masking operation. When specifying `n` (no) access, be certain that multiple matching entries are not specified in the ACL. See Section 3.2.1, page 39, for more information on adding entries to an ACL.

3.3.1 Displaying masked ACL permissions (`spget -ae`)

You can use the `spget -ae` command to show the masked ACL entries. Execution of this command masks each ACL entry's mode bits against the file's mask permissions and displays the resultant mode bits. Example 18, page 58 shows how the `-ae` option works (the `-e` option can be used only with the `-a` option).

In Example 18, page 58, execution of the `ls -le` command reveals the standard permission bits for `testfile` allow read and write access for the user only (`-rw-------a`). Executing the `spget -a` command displays the absolute permissions defined in the ACL entries for `testfile`.

Because the mask permission bits for `testfile` do not allow access for any group, the masking operation results in no access for any user listed in the ACL. This result is shown in the display of the `spget -ae` command.

In Example 19, page 59, the file's permission bits are changed by executing `chmod g+r`, which results in the mask permissions for `testfile` allowing read access. The masking operation results in read access for all the users. This result is shown in the display of the `spget -ae` command.

Example 18: Displaying the masked ACL mode bits (`spget -ae`)

```
$ ls -le testfile
-rw-------a  0   1 ben  trng     77  May 16 15:15  testfile
$
$ spget -a testfile
# ACL Information for: testfile
uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
$
$ spget -ae testfile
# ACL Information for: testfile
uid:1822     gid:-     user = tnn    group = *     mode = n
uid:196      gid:-     user = jack   group = *     mode = n
uid:927      gid:28    user = jog    group = trng  mode = n
uid:-        gid:28    user = *      group = trng  mode = n
uid:-        gid:28    user = *      group =       mode = n
```

Example 19: Displaying the masked ACL mode bits (`spget -ae`)

```
$ chmod g+r testfile
$ ls -le testfile
-rw-r-----a  0   1 ben  trng     77  May 16 15:15  testfile
$
$ spget -ae testfile
# ACL Information for: testfile
uid:1822     gid:-     user = tnn    group = *     mode = r--
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = r--
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = r--
```

In Example 20, page 59, the file's permission bits are changed by executing `chmod g+w`, which results in the mask permissions for `testfile` allowing read and write access. The masking operation results in read and write access for the `tnn:*`, `jog:trng`, and owning-group entries. The `jack:*` and `*:trng` entries are not granted write access because the ACL entries defined an absolute access of read only.

Example 20: Displaying the masked ACL mode bits (`spget -ae`)

```
$ chmod g+w testfile
$ ls -le testfile
-rw-rw----a  0   1 ben  trng     77  May 16 15:15  testfile
$
$ spget -ae testfile
# ACL Information for: testfile
uid:1822     gid:-     user = tnn    group = *     mode = rw-
uid:196      gid:-     user = jack   group = *     mode = r--
uid:927      gid:28    user = jog    group = trng  mode = rw-
uid:-        gid:28    user = *      group = trng  mode = r--
uid:-        gid:-     user = *      group =       mode = rw-
```

3.3.2 ACLs and `root` access

On a system using the PAL-based privilege mechanism, `root` is subject to all ACL rules. An ACL entry for `root` is treated the same as for any other user.

On a system using the super-user mechanism, `root` can override the ACL, even if it contains an entry for `root`.

If the file's ACL does contain an entry for `root`, then the entry is masked against the file's mask permissions as explained in Section 3.3, page 55. This means that you can deny `root` access by entering `n` in the ACL, as shown in Example 7, page 42, and in the following example:

```
root:*:n
```

Using this type of entry does not completely prevent `root` access for the following reasons:

- `root` has the ability to bypass a file's ACL protection by changing the permission mode, owner, and/or owning group of a file.
- `root` is allowed to access file system data through device special files, thus allowing `root` the ability to grant itself discretionary access to an object by changing the contents of the object's inode.

On a system using the super-user mechanism, if a file does not have an ACL, then `root` is automatically granted rwx access if all the mandatory access control checks are passed. For systems using the PAL-based privilege mechanism, if the file does not have an ACL, `root` is granted access through the permission mode bits.

3.3.3 `umask(1)` default access permissions

On a UNICOS system, the default setting for `umask` is 077 (instead of 027 as on previous non-MLS systems). This default setting is necessary to meet the TCSEC requirements. This default must be used on a Cray ML-Safe configuration of the UNICOS system.

Use of this default affects you in the following ways:

- If you want new files to automatically have group and/or world access, you must manually set the file access permission bits to enable this type of access.
- If you want to use a default other than 077, place `umask` in your `$HOME/.profile` or `$HOME/.login` file.

Using Security Labels [4]

This chapter describes the mandatory access controls used by a UNICOS system. Mandatory access controls are rules that control access based directly on a comparison of the subject's clearance and the object's classification. On a UNICOS system with the multilevel security (MLS) feature, these rules are incorporated into the UNICOS security policy. This chapter describes the following:

- The UNICOS security policy
- The commands needed to display and change security labels

4.1 UNICOS security policy

The UNICOS security policy is defined as the set of rules and practices by which a system regulates the disclosure of information. The UNICOS security policy uses security labels to enforce the following rules (see Figure 10 for a summary of this policy):

- A subject may read or execute an object only if the subject's current security label dominates the security label of the object.
- A subject may write or append to an object only if the subject's security label is equal to the security label of the object.

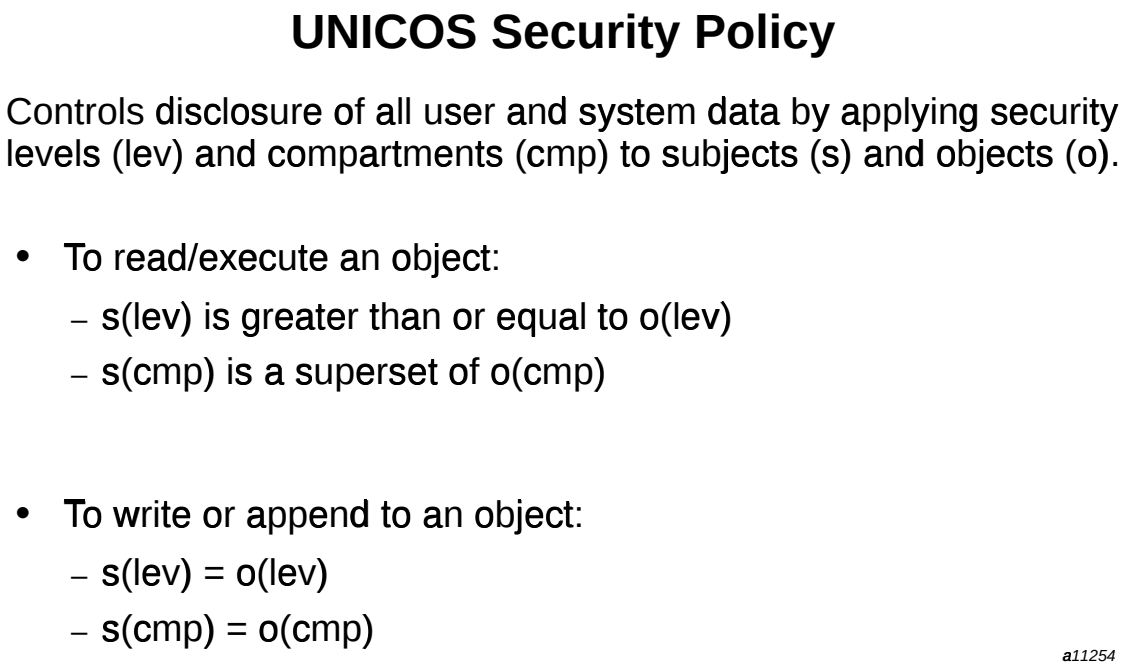

Figure 10. Mandatory access controls; UNICOS security policy

4.1.1 Changing security labels (`setulvl` and `setucmp`)

When you log into a UNICOS system, you are assigned a label consisting of a security level and a set of compartments. You can choose to operate at the assigned label throughout the entire session or you can execute the `setulvl`(1) and/or `setucmp`(1) commands to upgrade your label.

If you upgrade your label using either or both of these commands, the resulting label must dominate your current label and be dominated by your maximum security label (your maximum label is defined by your maximum security level and authorized compartments assigned to you in the UDB).

You cannot upgrade your label on a Cray ML-Safe configuration of the UNICOS system, because your active security label and your maximum security label are always equal on a Cray ML-Safe configuration.

The `setulvl`(1) command can set the security level component of a security label to a specified number. The number can range form 0 to 16, where 0 is the lowest clearance and 16 is the highest clearance. Optionally, your security

administrator may assign names to these values; if this has been done at your site, you can specify a named value instead of a number.

Note: You cannot successfully use the `setulvl` command on a Cray ML-Safe configuration or a network connection configured with the IP security option on a UNICOS system. You are allowed to operate only at the security label established when the network login socket connection was created.

Example 21 shows the `setulvl`(1) command used with the number 2 as an argument. This example also shows how to use the `spget`(1) command without options to display your security parameters, including your maximum, minimum, and default security levels.

Example 22 shows the `setulvl`(1) command used with the `level2` as an argument. In both cases, the message returned by `setulvl`(1) confirms that the level has changed to 2.

Example 21: Example of `setulvl` command

```
$ setulvl 2
setulvl:  New security label is
Level[2:level2] Compartments [none]
$  spget
permits equal 00000
               none
security level is 2
               level2
maximum level is 16
               level16
minimum level is 0
               level0
authorized compartments are 010
               test
active compartments are 00
               none
integrity class is 0
               class0
maximum class is 0
               class0
active categories are 00
               none
authorized categories are 00000000000
               none
```

Example 22: Example of `setulvl level2` command

```
$ setulvl level2
setulvl:  New security label is
Level[2:level2] Compartments [none]
```

The `setucmp`(1) command is similar in operation to the `setulvl`(1) command, except that it adds compartments to your current compartment set rather than setting an absolute value. Compartments can be set in one of the following ways:

- By using an octal mask; the mask is a bit value corresponding to one or more compartments to be activated. This argument must be expressed in octal.
- By specifying `ALL`, which activates all authorized compartments
- By specifying a comma-separated list of names

Example 23 shows how the `setucmp` command is used to change your security compartments. This example also shows how to use the `spget`(1) command without options to display your security parameters, including your active and authorized compartments.

Note: You cannot successfully use the `setucmp` command on a Cray ML-Safe configuration or a network connection configured with the IP security option on a UNICOS system to add to your security compartments. You are allowed to operate only at the security label established when the network login socket connection was created.

Example 23: Example of `setucmp` command

```
$ setucmp test
setucmp:  New security label is
Level [2:level2] Compartments [test]
$ spget
permits equal 00000
               none
security level is 2
               level2
maximum level is 16
               level16
minimum level is 0
               level0
authorized compartments are 010
               test
active compartments are 010
               test
integrity class is 0
               class0
maximum class is 0
               class0
active categories are 00
               none
authorized categories are 00000000000
               none
```

The `setulvl`(1) and `setucmp`(1) commands can be issued successfully only when the shell is the only process in your session. For example, if you log in to the C shell (`csh`(1)) and execute the Korn shell (`ksh`(1)) as a sub-shell, or execute a command in the background, you are not allowed to change your label from the sub-shell or while the background process is still running. If you replace your C shell with a Korn shell by using the `exec`(1) shell built-in command, you are allowed to change your label from the Korn shell, provided no other process was currently in your session.

Changing your label can fail for the following reasons:

- You tried to change to a label that does not dominate your current label.
- You tried to change to a label that is not dominated by your maximum label.
- You have open files other than the controlling `tty` within your process.

- The shell from which you issued the request is not the only process in the session.
- You have an open socket connection.
- The label you requested is not within the range of labels allowed on your system.
- Your system is configured to enforce strict B1 compliance and the security label is not equal to the current label.

4.1.2 MLS permissions

On UNICOS systems, you can be assigned MLS permissions (shown in Figure 11) at login time. These permissions are specified with your other login account attributes in the user database (UDB).

MLS permissions grant a user special capabilities on a UNICOS system. The MLS permissions are useful only on a UNICOS system using the super-user mechanism; they have no affect on UNICOS systems that use only the PAL-based privilege mechanism.

You can use the `spget`(1) command to display your MLS permissions. You may see the `suidgid` permission assigned to your account.

MLS Permissions

Set setuid/setgid (`suidgid`): Gives the user explicit permission to set the set-user-ID (setuid) and/or set-group-ID (setgid) bits for a file. Restricted management of setuid and setgid files is enforced only on UNICOS systems configured to do so. This permission is used only on UNICOS systems using the super-user mechanism.

MLS Permissions

- Set setuid/setgid (`suidgid`): Gives the user explicit permission to set the set-user-ID (setuid) and/or set-group-ID (setgid) bits for a file. Restricted management of setuid and setgid files is enforced only on UNICOS systems configured to do so. This permission is used only on UNICOS systems using the super-user mechanism.

***a**11303*

Figure 11. Permissions

Creating Directories and Files [5]

This chapter describes the mandatory access controls used by the UNICOS multilevel security (MLS) feature to control access to objects. The following topics are described:

- Assigning security labels to directories
- Wildcard directories
- Multilevel directories (MLDs)
- Assigning security labels to files
- Displaying a directory's or file's security attributes
- Creating directories and files
- Removing files and directories (including `setuid` and `setgid` files)

5.1 Assigning security labels to objects

The following sections describe the rules that must be observed when creating and using directories and files on a UNICOS system.

5.1.1 Assigning labels to directories

Creating a directory with the `mkdir`(1) command on a UNICOS system can be influenced by your system configuration.

If secure `mkdir` behavior is enforced, your active label must be equal to the parent directory's label. The new directory is created at your active security label. A Cray ML-Safe configuration of the UNICOS system must enforce the secure behavior of `mkdir`.

If secure `mkdir` behavior is not enforced, your active label must dominate the label of the parent directory. The directory is created with your active security label.

When secure `mkdir` behavior is enforced, you can use the `mkdir -L` command to create a directory with a security label that dominates your active security label (assuming the requested label is within your authorized range). If the relabeling fails, the directory's label remains at your active security label. If the

`mkdir -L -p` command is used, only the last directory in the path is relabeled. All intermediate directories are created at your active label. Example 24, page 70 is an example of using the `mkdir -L` command. You cannot view this directory until you log out and reestablish a connection at security label of the directory.

Example 24: Example of `mkdir -L` command

```
$ mkdir -L 2 /tmp/testdir
```

Any user can upgrade the label of an empty directory (for example, when using the `spdev -K` or `spdev -L` commands) if all of the following conditions are met:

- You have MAC write access to the target directory.
- You are the owner of the target directory.
- The target directory is empty.

If you are properly authorized, you can override these restrictions and change the label of any directory; the definition of properly authorized depends on which TFMgmt mechanism your system is using. See the `mkdir`(1) man page for more information.

The `spdev` command should be used to change security labels, because it is the only command that does so "atomically" (that is, in one step). If you use the `spset -l` or `spset -c` commands to change the label, you lose write access before successfully completing the change. For example, if you use the `spset -l` command to change the level of a file, you lose write access to the directory, making it impossible to change the compartment. This is shown in the following example:

```
$ mkdir bar
$spset -l 5 -c comp24 bar
spset: setflvl failed for emptydir : not owner
```

Regardless of your system configuration, to create a directory, the security label of the directory must always fall within the security label range of the file system on which the directory resides.

5.1.2 Wildcard directories

Warning: Wildcard directories are not supported on a Cray ML-Safe configuration. Multilevel directories must be used on Cray ML-Safe configuration. See Section 5.1.3, page 71 for more information.

A wildcard directory is a directory that is labeled with security level 63, which enables it to contain files at any security label within the boundaries of the file system. Use of wildcard directories avoids replication of special directories for every use.

Wildcard directories are established primarily for trusted daemons, such as NQS, that must service many requests and output queues; only your security administrator can set the wildcard level on a directory. Access to files in the wildcard directory is always subject to the UNICOS MLS discretionary and mandatory access controls.

The `/usr/tmp` and `/tmp` directories, which are accessible to many system utilities, are assigned the wildcard security level to allow them to contain files with varying security labels.

5.1.3 Multilevel directories (MLDs)

The use of wildcard directories does not meet the TCSEC criteria. Multilevel directories (MLDs) provide a method of sharing a common directory name while partitioning the actual directory contents according to security labels. MLDs allow the trusted environment to continue to use shared directories without wildcard labels, which create the potential for write-down security policy violations.

Warning: Wildcard directories are not supported on a Cray ML-Safe configuration. Therefore, only the use of MLDs is allowed on a Cray ML-Safe configuration.

The MLD mechanism creates a new type of symbolic link called a *multilevel symbolic link*. A normal symbolic link redirects path name lookups by substituting new information in the path name at the point where the symbolic link is encountered.

A multilevel symbolic link operates in much the same manner, but when the multilevel symbolic link is expanded in a path name, a file name composed of a kernel-generated representation of the process label is appended to the contents

of the symbolic link before the substitution. The resulting symbolic link expansion changes according to the label of the process doing the expansion.

This multilevel expansion allows a directory tree to exist in which users at different labels are transparently redirected to different directories, based on their security label. This allows users at different labels to work in different directories, and the information they store in these different directories is controlled by the normal MAC rules for directory searching and changing.

Figure 12 shows the structure of a MLD. This figure shows the `/tmp` directory replaced by a multilevel symbolic link of the same name. The actual directory has moved to a new name (`/tmp.mld`), which is the target of the symbolic link. Beneath the `/tmp.mld` are a set of labeled subdirectories containing files put there by users or application programs.

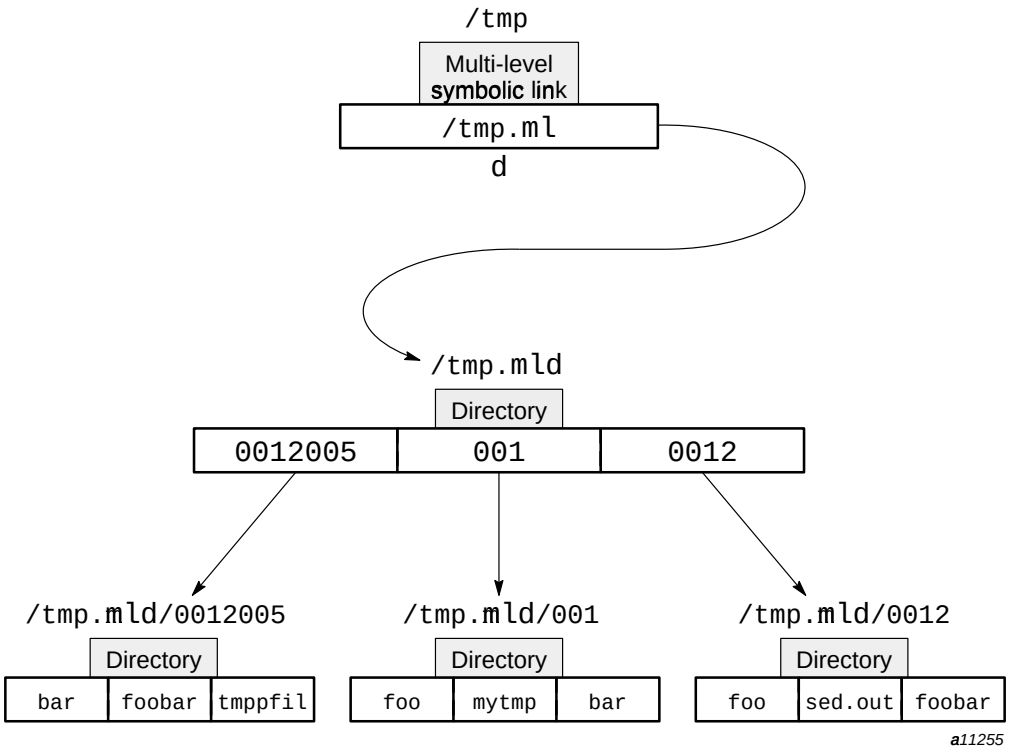

Figure 12. Structure of a multilevel directory (MLD)

The naming convention of MLDs is as follows:

- Subdirectory names all have a leading 0.
- Following the leading 0 is a two-digit octal representation of the security level of the subdirectory.
- Following the two-digit security level is the octal representation of the compartment set of the directory (if there is a compartment set). If no compartment set is used, only the level is represented.

In Figure 12, the directory named `0012005` breaks down to a security level of 1 and a compartment set of 2005(octal). The directory named `001` breaks down to a security level of 1 with no compartments. The directory named `0012` breaks down into a security level of 1, with a compartment set of 2(octal).

The following conversion list will help you in converting this representation:

Label	Octal representation
0	000
1	001
2	002
3	003
4	004
5	005
6	006
7	007
8	010
9	011
10	012
11	013
12	014
13	015
14	016
15	017
16	020
51	063 (`syslow`)
54	066 (`syshigh`)

63 077 (wildcard)

Note: The naming convention for MLDs outlined previously may change in future UNICOS releases.

Although a new type of symbolic link is used, the MLD structure is the same as a "regular" directory, so no new access rules are needed. Directories that comprise a MLD are searched, read, and changed just like any other directory, subject to the UNICOS MLS MAC and DAC policies. Because MLDs are implemented with symbolic links, the behavior of the shell is different from that of the kernel in regards to symbolic links and relative path name resolution.

The MLD structure should be fairly transparent to you. If you use relative path names instead of full path names while in a MLD, you need to understand the MLD path naming conventions outlined previously.

Example 25 shows the use of a MLD `/tmp` structure. Assume in example 1 that the user has a security label of level 1 and no compartments. In example 2, assume the user has reconnected with a security label of level 2 and no compartments.

Example 25: Example of MLD `/tmp` structure

```
Example 1 (assume a security label of level 1 and
no compartments):

$ touch /tmp/foo
$ ls /tmp
foo
$ ls /tmp.mld/001
foo
$ cd /tmp
$ /bin/pwd
/tmp.mld/001
$ cd /

Example 2 (assume a security label of level 2 and
no compartments):
$ ls /tmp
$ ls /tmp.mld/001
foo
$ touch /tmp/bar
$ ls /tmp
bar
$ ls /tmp.mld/002
bar
$ ls /tmp.mld/003
/tmp.mld/003: Permission denied.
$ rm /tmp.mld/001/foo
rm: /tmp.mld/001/foo: 644 mode? y
rm: /tmp.mld/001/foo not removed
Permission denied.
$ rm /tmp.mld/002/bar
```

The following list contains the Cray Research products or commands that use MLDs when running a Cray ML-Safe configuration. The mail directories require the use of MLDs on either a UNICOS configuration or a Cray ML-Safe configuration:

- `jtmp` directories
- `/tmp` directory
- `/usr/tmp` directory

- `cron`(8) and `at`(1) spool directories
- `lpr`(1B) and `lpd`(8) spool directories
- Mail directories
- NQS directories
- CRL debug log directory as defined by `${RLLOGDIR}`

Only a properly authorized user can create a MLD. See your security administrator if you need a MLD created for your use.

5.1.4 Assigning security labels to files

When you create a regular file, directory, named pipe, or socket, it is assigned your active security label. This information is recorded in both the memory and disk versions of the inode (except for sockets, which do not use inodes) describing the object.

When you create a block or character special file, it is assigned a security label of level 0, null compartments, and it is created in the `OFF` state.

Only a properly authorized user can change the security label of any file. The definition of properly authorized depends on which system management mechanism your system uses.

You, as a nonadministrative user, can create files, subdirectories, or links within a directory if your security label equals that of the directory. Because of this and the upgrade restrictions (see Section 5.1.1, page 69), any nonadministrative management of a directory tree can only increase the security label as you move deeper into the directory structure. For administrative users, these restrictions do not apply.

5.1.5 Terminal drivers

In the UNICOS system, the terminal driver devices (for example, `/dev/ttyp*`) are assigned security labels when opened by a user. The labels for these files are set equal to the user's active security label.

Note: On a Cray ML-Safe configuration, the label range of a network connection is always constricted to a single label. Therefore, any attempt to change the label by using the `setulvl`(1) command, `setulvl`(2) system call, `setucmp`(1) command, or `setucmp`(2) system call, as outlined in the following text, fails.

When a nonprivileged process uses the `setulvl`(1) command, `setulvl`(2) system call, `setucmp`(1) command, or `setucmp`(2) system call to change its active security label, the following occurs:

1. The current process is checked to ensure it is not a child of an existing process in its session (with the exception of being a child of `init`, which is allowed) and that it has no child processes. If either of these cases is true, the request fails.
2. The requested label is checked to ensure that it dominates the current active label of the process; if not, the request fails.
3. The requested process label is checked against the label range of the process; if not, the request fails.
4. The active label of the process is changed to the requested label.
5. The active label of the process controlling tty changes to the requested label.

When a nonprivileged process uses the `setusrv`(1) command or `setusrv`(2) system call, the following occurs:

1. The requested `usrv` structure is checked to ensure it does not attempt to change the active label, active class, or active categories. If any of these attributes are changed, the request fails.
2. The active label in the requested `usrv` structure is checked to ensure that it is still within the range defined in the `usrv` structure. If the requested label is out of range, the request fails.
3. The label range in the requested `usrv` structure is checked to ensure that it is within the label range of the current process. If the label range is not completely contained in the range of the current process, the request fails.
4. The class and category in the request are checked to ensure that they are still in the requested range. If either class or categories exceeds the requested range, the request fails.
5. The requested integrity range is checked to ensure that it is within the current integrity range of the process. If the requested integrity range exceeds the integrity range of the current process, the request fails.
6. The permissions are checked to ensure they are a subset of the current permissions. If the requested permissions are not a subset of the current permissions, the request fails.

7. The new `usrv` structure is applied to the process. The controlling `tty` label does not change as part of this procedure.

5.2 Displaying the security attributes of directories and files

To display the security attributes of files and directories, use the `ls -le` command; you can use the `ls -lde` command to display only a directory's attributes. In addition to the information already defined in Figure 13, the `ls` display also displays the following information:

- Field A indicates the file type; a `-` indicates a regular file. A `d` indicates the file is a directory.
- Field B indicates the file's `user/group/world` permission bits.
- An `a` in field C indicates that the file has an access control list (ACL).
- An `i` indicates that the file has one integrity class or category assigned to it. If it is displayed, ask an appropriately authorized user to set these values to 0.
- Field E can contain a number, `T`, `*`, or a `?`. A number indicates the security level of the file; an `*` indicates that the file has a wildcard level (63) assigned to it; and a `?` indicates that the file's security level is outside your minimum/maximum security level range. The `?` also appears in this field if the file's active compartments are not part of your authorized set of compartments. A `T` indicates that the file has a TFMgmt-executable level (60) assigned to it. Ask an appropriately authorized user to relabel the file with an appropriate security label.
- A `+` sign in field F indicates that file has compartments that are part of your authorized set of compartments. This sign is shown only if the compartments are also part of your authorized set; otherwise a `?` appears in field E.

Example 26 shows the information displayed for a user's home directory and for `/etc/udb`.

Example 1:

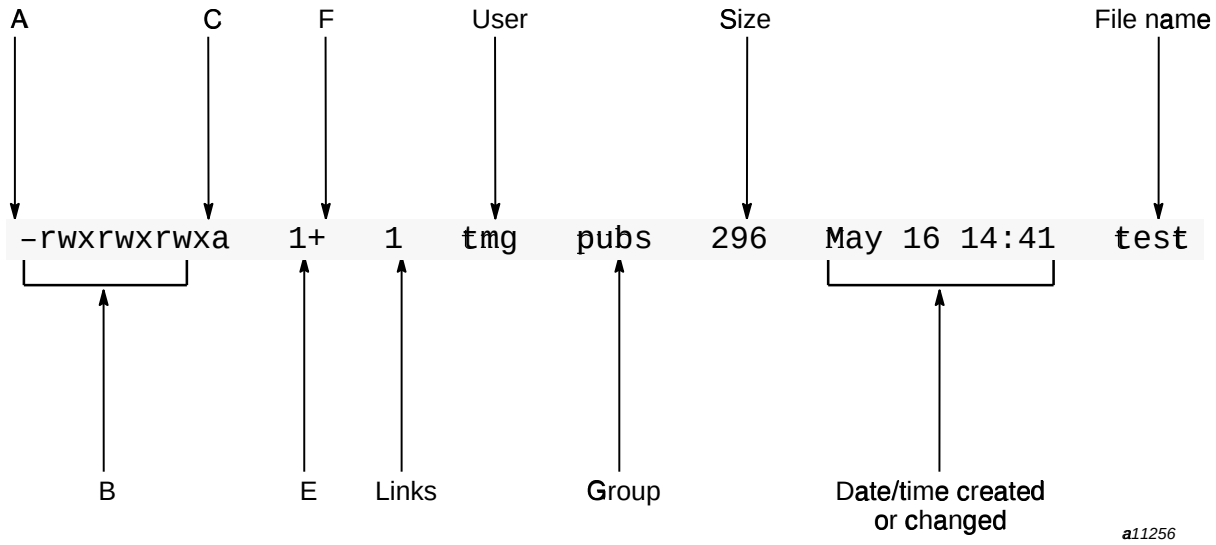

Figure 13. Displaying a file's security attributes (`ls -le` command)

Example 26: Displaying a file's security attributes (`ls -le` command)

```
$ ls -le
-rw-------a  0  1 ben trng    329 May 9 14:53  file
-rw-------a  0  1 ben trng     51 May 9 14:45  modfile
-rw-------a  0  1 ben trng   2072 May 9 15:01  myacl
-rw-------a  0  1 ben trng   2072 May 9 15:01  myacl2
-rw-------a  0  1 ben trng   2072 May 9 14:48  newfile
-rw-------a  0  1 ben trng     60 May 9 14:49  testfile
-rw-------a  0  1 ben trng    293 May 9 14:07  text

$ ls -le /etc/udb
```

To obtain specific information about a directory or a file, use the `spget -f` command, as shown in Example 27.

Example 27: Displaying a file's security attributes (`spget -f` command)

```
$ ls -le /usr/lib/nqs/nqsdaemon
---x------   0  1 root  bin 1350312 May 23 08:04  nqsdaemon
$ spget -f /usr/lib/nqs/nqsdaemon
Security Values for: nqsdaemon
         level:   0
                  level0
  compartments:   0
                  none
         class:   0
                  class0
    categories: 01000000000
                  daemon
         flags:   0
                  none
```

5.3 Creating files

Note: The examples used in this section apply only to non-network sessions or for UNICOS systems not configured as Cray ML-Safe. On a Cray ML-Safe configuration, when using a network login connection, you are allowed to work only at the security label established when the socket connection is made. You cannot use the `setulvl` and `setucmp` commands to change your security label.

As stated previously in this chapter, a file is assigned your active security label when you create it. However, a nondirectory file within its parent directory must have a security label equal to that of the parent directory. If you change your security label, you might have to create a new directory in order to create the file (unless the directory has a wildcard security label or is a MLD). Example 28, Example 29, page 82, and Example 30, page 83 show how this is done.

In Example 28, Jack has a security label that consists of an active security level of 0, no active compartments, and authorized compartments of A and B. The `spget -f` command in Example 28 shows Jack's current directory is at security level 0 with no compartments, and displays the security attributes of `month` and `/tmp`.

The example in Example 28 fails on a Cray ML-Safe configuration if Jack's active label does not dominate the label of the target file.

Example 28: Example of creating files (part 1)

```
$ spget -f . month /tmp
Security Values for: .
         level:   0
                  level0
  compartments:   0
                  none
         class:   0
                  class0
    categories:   0
                  none
         flags:   0
                  none

Security Values for: month
         level:   1
                  level1
  compartments:   0200
                  A
         class:   0
                  class0
    categories:   0
                  none
         flags:   0
                  none

Security Values for: /tmp
         level:   63
                  wildcard
  compartments:    0
                  none
         class:   0
                  class0
    categories:   0
                  none
         flags:   0
                  none
```

In Example 29, page 82, Jack creates `file1` in his current directory. The `ls -le` display of this file shows that the file has a security level of 0 and no active

compartments. Then Jack decides to raise his security level to 1 by executing the `setulvl` command; he next attempts to create `file2`.

Permission to create this file is denied because Jack raised his security level; the new security level does not fall within the range of his current directory. He can, however, create `file2` in `/tmp`, because `/tmp` has a wildcard security level.

Example 29: Example of creating files (part 2)

```
$ cat > file1
test file
$ ls -le file1
-rw-------  0  1 jack  trng   9  May 9 14:41  file1
$ setulvl 1
$ setulvl:  New security label is
Level[1:level1] Compartments[none]
$ cat >file2
file2:Permission denied
$ cat >/tmp/file2
test file
$ ls -le /tmp/file2
-rw-------  1  1 jack  trng   9  May 9 14:43  /tmp/file2
```

In Example 30, page 83, Jack creates a new directory called `private`. After changing to the `private` directory, he again attempts to create `file2`. This attempt is successful because both the file and the directory are at security level 1.

Jack then adds compartment A to his active set and tries to create `file3`. This attempt fails because `file3` would have had both a security level and an active comprtment, while the directory has only a security level. However, when Jack tries to create `file3` in the directory `month`, he is successful, because the directory and the file have the same security level and compartments.

Example 30: Example of creating files (part 3)

```
$ mkdir private
$ ls -le
drwx------  1  2 jack  trng   9  May 9 14:45  private
$ cd private
$ cat >file2
test file
$ ls -le file2
-rw-------  1  1 jack  trng   9  May 9 14:46  file2
$ setucmp A
setucmp:  New security label is
Level[1:level1] Compartments[A]
$ cat >file3
file3:Permission denied
$ cd ../month
$ cat >file3
test file
$ ls -le file3
-rw-------  1+  1 jack  trng   9  May 9 14:49  file3
$ spget -f file3

Security Values for: file3
         level:   1
                 level1
  compartments:   200
                 A
         class:   0
                 class0
    categories:   0
                 none
         flags:   0
                 none
```

5.4 Removing files and directories (`spclr`, `rm`, and `rmdir`)

To remove an ordinary file, you must have write permission on the parent directory. In addition, your active security label must equal the security label of the targeted file and the parent directory of the file.

Removing directories on a UNICOS system depends on your system configuration. If secure behavior is enforced, your active label must be equal to

the label of the parent directory and dominated by the directory label to remove a directory. If secure behavior is not enforced, your active security label must equal the label of the directory being removed and dominate the parent directory's label.

See your security administrator to determine which behavior is used on your system. A Cray ML-Safe configuration must enforce secure behavior.

When secure behavior is enforced, any user can remove an upgraded directory if all of the following conditions are met:

- You have write access to the parent directory.
- The label of the target directory dominates your active security label.
- The target directory is empty.

If you are properly authorized, you can override these restrictions and remove any directory; properly authorized depends on which TFMgmt mechanism your system is using. See the `rmdir`(1) man page for more information.

Your security administrator can set configuration options that determine how a file's data blocks are overwritten. Depending on how these options are configured, when a file is overwritten (cleared) by using the `spclr`(1) command, the data blocks associated with the file are usually overwritten with zeros (the pattern can be site-specified, although the recommended pattern is zeros).

The `spclr -s` and `splcr -d` commands remove files and overwrite (clear) associated disk space with a pattern. The `-s` option clears the disk space associated with the file(s). The disk space is overwritten once with a pattern that is usually set to zeros.

The `-d` option declassifies the disk space associated with the file(s). The disk space is overwritten with a declassify pattern (set by your security administrator), then overwritten with the negated pattern, and finally overwritten with the original pattern. If your system is not properly configured, the `-d` option can fail; see your security administrator.

Example 31 shows how to use the `spclr -s` command (this example is based on the files created in Example 29, page 82). Jack's current security level is 1 and his active compartment is A. His attempt to remove `file1` fails because the file's security level is 0 and has no active compartment. If Jack changes to the `month` directory (which has security level of 1 and a compartment of A), he can remove `file3`.

Example 31: Example of `spclr -s` command

```
$ cd
$ spclr -s file1
file: 600 mode? y
spclr: file not cleared
Permission denied.
spclr: file not removed
Permission denied
$ cd month
$ spclr -s file3
$
```

Example 32 shows how to use the `rm`(1) and `rmdir`(1) commands. In this example, the user has an active security level of 0 and no compartments active (as shown in the `spget` command display). This example assumes that the secure directory behavior is not enforced.

The execution of `spget -f` shows that a file called `file2` exists in the directory called `level2` and that `level2/file2` has an active security level of 2 and no compartments active. When the user tries to remove `level2/file2`, permission is denied and the file is not removed. This attempt fails because the user's active security level and compartments do not match that of the file or its parent directory. When the user raises his active security level to 2 (by using the `setulvl` command), he can then remove both the file and the directory.

Example 32: Example of `rm` and `rmdir` commands

```
$ spget

permits equal 00
               none
security level is 0
               level0
maximum level is  3
               level3
minimum level is  0
               level0
authorized compartments are 040
               train
active compartments are 00
               none
integrity class is 0
               class0
maximum class is   0
               class0
active categories are 00
               none
authorized categories are 00
               none
$ spget -f level2/file2
Security Values for: level2/file2
         level:   2
                  level2
  compartments:   0
                  none
         class:   0
                  class0
    categories:   0
                  none
         flags:   0
                  none
$ rm level2/file2
rm: level2/file2 not removed.
Permission denied
$ setulvl 2
Level[2:level2] Compartments[none]
$ rm level2/file2
$ rmdir level2
```

5.4.1 `Setuid` and `setgid` files

Your security administrator can configure your UNICOS system to restrict the management of set-user-ID (`setuid`) and set-group-ID (`setgid`) files. If this has been done, attempts by unauthorized users to create, copy, link, move (to the same file system), or remove `setuid` and `setgid` files are denied. Moving (to another file system) or copying `setuid` and `setgid` files is allowed, but the `setuid` and `setgid` mode bits are cleared for unauthorized users.

Example 33 shows the messages that are displayed when a user (Mary), without the appropriate authorization, tries to link `/tmp/file` to `/tmp/myfile` (the `s` bits in the `user/group/world` permission bit field of the `ls -l` command display indicates `/tmp/file` has the `setuid` and `setgid` bits set).

Example 33: Copying and linking setuid and setgid files

```
$ spget

permits equal 00
               none
security level is 0
               level0
maximum level is  3
               level3
minimum level is  0
               level0
authorized compartments are 040
               none
active compartments are 00
               none
integrity class is 0
               class0
maximum class is   0
               class0
active categories are 00
               none
authorized categories are 00
               none
$ ls -le /tmp/file
-rwsr-sr-x   0   1  joe  adm  232504 Aug 23 08:04  /tmp/file
$ ln /tmp/file /tmp/myfile
cmd-2631 ln: Failed to create link for target '/tmp/myfile'
Security mandatory access violation
$
```

5.4.2 Archive commands

When using the `cpio`(1) command to archive files on a UNICOS system, only the files that your active security label dominate are archived. The security label protecting the archive becomes your active label. There are several options to the `cpio` command that control which file security attributes are archived. Refer to the `cpio`(1) man page for more information on using this command.

You must have read access to the archive to restore the files that are contained in the archive. Nonadministrative users can restore secure archives, but the security attributes in the archive are not restored. Only properly authorized

users can restore the security attributes of an archive. When you restore an archive, all files are restored at your active label.

Network Access [6]

This chapter provides cross-references to the manuals that contain user information for using the UNICOS multilevel security (MLS) feature on TCP/IP and NQE/NQS. In addition, the following topics are addressed:

- RQS controls
- Station controls

For more information on using the UNICOS MLS feature on TCP/IP, see the UNICOS MLS section in the *TCP/IP Network User's Guide*, Cray Research publication SG–2009.

For more information on using the UNICOS MLS feature on NQS, see the UNICOS MLS section in the *NQE Administration*, Cray Research publication SG–2150.

6.1 RQS controls

RQS is the Remote Queuing System that runs on systems other than Cray Research systems. The RQS VAX/VMS, VM, and MVS versions of the `qsub`(1) command do not currently support the use of options to specify security levels or compartments.

Note: RQS is not part of the set of Cray ML-Safe components.

The RQS UNIX version of `qsub` has options to allow the security level and compartments to be specified. When RQS UNIX is configured with a remote queue to a UNICOS MLS system, any jobs you submit run at the security level and compartment specified by the `qsub -L` and `-C` options.

If the request is submitted by using the `qsub` command (without specifying the `-L` or `-C` options), the request is assigned your default security label as assigned in the UDB.

When using the `qsub -L` option and/or the `qsub -C` option, the requested label must dominate the job submission's label and be within the valid range as defined by the user's user database entry and the host's network access list (NAL) entry. If these requirements are not met, the job is deleted.

6.2 Station listable output

If a station supports listable output, it must obtain the file's security label from the dataset header in order to print the following:

- A header, banner page
- A trailer page
- A header line per page
- A footer line per page

Each of these labels should include the file's security level and compartments and should be tailored to the site's printer labeling formats. This is the only non UNICOS implementation necessary; however, it is essential to provide marked listable station output.

Miscellaneous Information [7]

The following sections describe the following miscellaneous information for the UNICOS MLS feature:

- The `cron`(8) command
- The `su`(1) command
- Tape security
- Data migration security
- Cray/REELlibrarian security

7.1 The `cron` command

The UNICOS system allows the daemon to initiate jobs on your behalf with the resultant processes executing with your current security attributes. Also, you do not have to use the `qsub`(1) command to have your job run with your current security attributes.

In Example 34, Jack wants to run a program called `util`. To have `util` execute with the `train` compartment, Jack changes his `crontab` file to have NQS execute `util`.

Example 34: Example of `cron` command using `qsub`

Before:

```
$ crontab -l
0 3 * * 1-5   /usr/jack/util /f/data
```

After:

```
$ crontab -l
0 3 * * 1-5   qsub -C train /usr/jack/script.util
$ cat script.util
/usr/jack/util /f/data
```

Example 35, page 94, shows how to accomplish what was shown in Example 34, page 93, without using NQS.

Example 35: Example of `cron` command without using `qsub`

```
$ setucmp train

$ crontab
0 3 * * 1-5   /usr/jack/util /f/data
$ [CONTROL-D]
```

7.2 Using the su command

The `su`(1) command allows you to change your user ID to that of another user, but allows your security attributes to remain the same.

All attempts to use the `su` command are logged in the `su` log. You are limited to two failed `su` attempts per minute. A caution is issued after the second failure, and you are logged off and your login is disabled after the third failure in any minute.

In addition, all successful `su` attempts result in a `setuid`(2) system call entry in the `su` log. Your security administrator can audit this log to determine who has been executing `setuid` and which users have excessive `su` failures.

7.3 Tape security

See the *Tape Subsystem User's Guide*, Cray Research publication SG–2051, for information on how to use tapes in connection with the UNICOS MLS feature.

7.4 Data migration security

See the *Cray Data Migration Facility (DMF) Administrator's Guide*, Cray Research publication SG–2135, for more information on how to use DMF in connection with the UNICOS MLS feature.

7.5 Cray/REELlibrarian security

See the *Cray/REELlibrarian (CRL) User's Guide*, Cray Research publication SG–2126, for more information on how to use CRL in connection with the UNICOS MLS feature.

Overview of TCSEC Trusted System Divisions [A]

The *Trusted Computer System Evaluation Criteria* (TCSEC) divides trusted systems into four divisions called A, B, C, and D, with A being the most trusted and D the least trusted. These divisions are hierarchical (as shown in Figure 14); that is, protection provided in division C must be incorporated into a division B system's additional security mechanisms, and so on. The following sections describe the TCSEC security policy and accountability requirements for each division.

A.1 Division D criteria

Division D systems provide minimal protection. This division is reserved for systems that have been evaluated, but failed the requirements for a higher division rating.

A.2 Division C criteria

Division C systems provide discretionary protection and are divided into two classes: C1 and C2. C1 systems provide discretionary security protection by using the following security mechanisms:

- Security policy
  - Discretionary access controls (for example, `owner/group/world` permissions)
- Accountability
  - User identification (for example, a login procedure) and user authentication (for example, passwords)

C2 systems provide controlled access protection by adding the following security mechanisms (shown in boldface type) to the C1 mechanisms:

- Security policy
  - Discretionary access controls (for example, owner/group/world permissions)

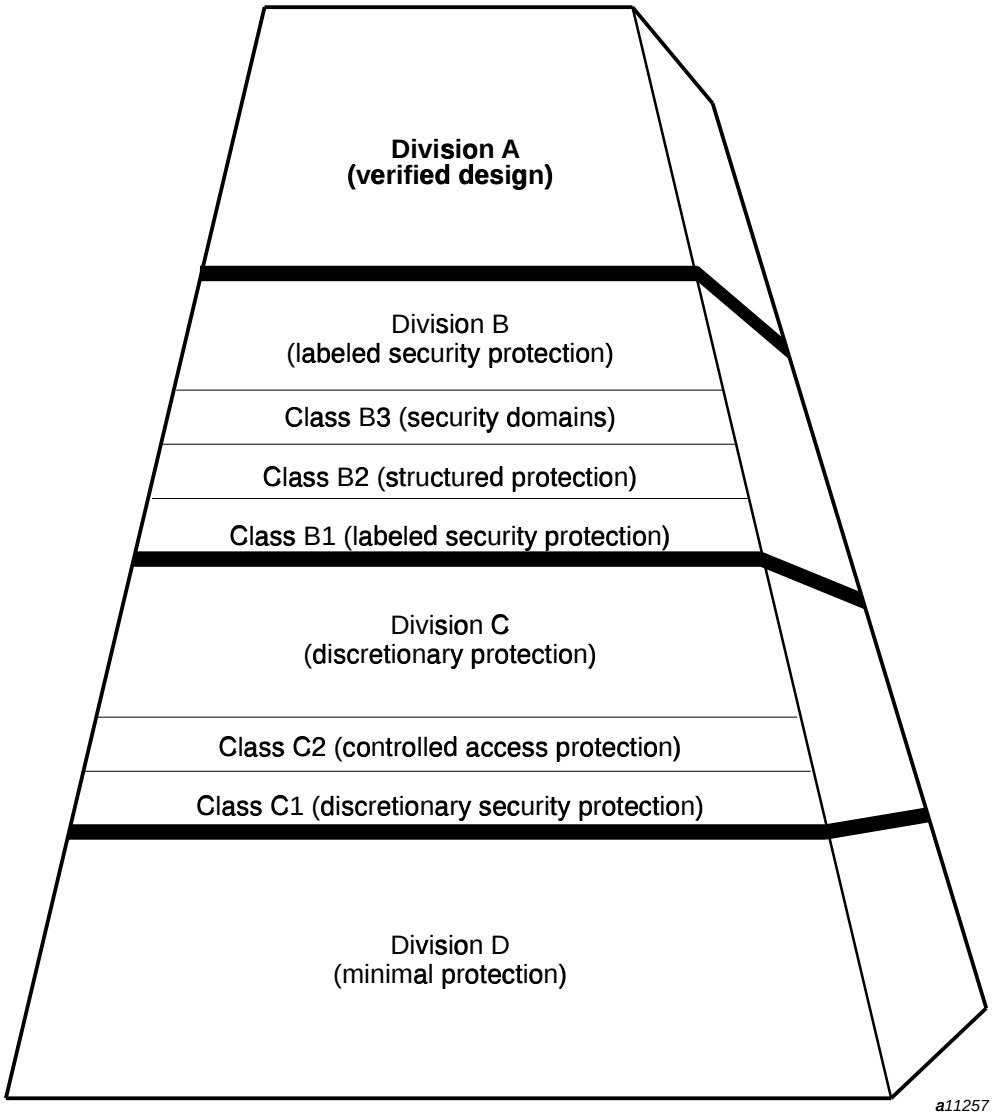

Figure 14. Divisions

- Accountability
  - User identification (for example, a login procedure) and user authentication (for example, passwords)

  - **Individual accountability**
  - **Audit trail (all subject access to an object is logged in the security log)**

A.3 Division B criteria

Division B systems provide mandatory access controls and are divided into three classes: B1, B2, and B3.

B1 systems provide labeled security protection by adding the following security mechanisms (shown in boldface type) to the division C mechanisms:

- Security policy
  - Discretionary access controls (for example, owner/group/world permissions)
  - Finer discretionary access controls (for example, to the granularity of a single user)
  - Object reuse capability (objects are scrubbed before accessed)
  - **Security labels for each subject and object**
  - **Mandatory access controls, which are enforced for every subject's access to an object**
- Accountability
  - User identification (for example, a login procedure) and user authentication (for example, passwords)
  - Individual accountability
  - Audit trail (all subject access to an object is logged in the security log)

B2 systems provide structured protection by adding the following security mechanisms (shown in boldface type) to division C and B1 mechanisms:

- Security policy
  - Discretionary access controls (for example, owner/group/world permissions)
  - Finer discretionary access controls (for example, to the granularity of a single user)
  - Object reuse capability (objects are scrubbed before accessed)

  - Security labels for each subject and object **and use of labels extended to include devices (for example, terminals, consoles, printers, and disks)**
  - Mandatory access controls, which are enforced for **all resources**
- Accountability
  - User identification (for example, a login procedure) and user authentication (for example, passwords)
  - Individual accountability
  - Audit trail (all subject access to an object is logged in the security log)
- Assurance
  - **Trusted facility management provided to divide system administrator functions between system operators and system administrators**

B3 systems provide security domains by adding the following security mechanisms (shown in boldface type) to division C, B1, and B2 mechanisms:

- Security policy
  - Discretionary access controls (for example, owner/group/world permissions)
  - Finer discretionary access controls (for example, to the granularity of a single user)
  - Object reuse capability (objects are scrubbed before accessed)
  - Security labels for each subject and object
  - Mandatory access controls, which are enforced for all resources
  - Use of security labels extended to include devices (for example, terminals, consoles, printers, and disks)
- Accountability
  - User identification (for example, a login procedure) and user authentication (for example, passwords)
  - Individual accountability
  - Audit trail (all subject access to an object is logged in the security log)

- Assurance
  - Trusted facility management provided to divide system administrator functions between system operators, system administrators, **and security administrators**

A.4 Division A criteria

Division A systems provide verified protection and are functionally equivalent to a B3 system. The following is needed to acquire a division A rating:

- Formal design specifications and techniques
- Formal proof of the security policy
- Stringent configuration management techniques