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.

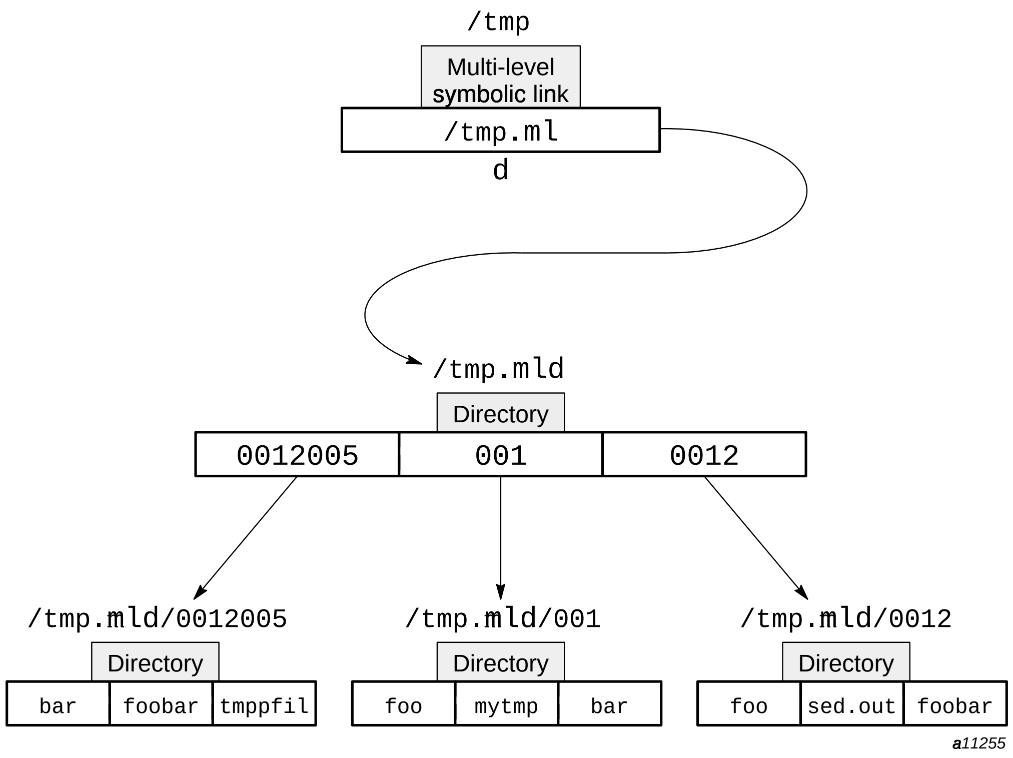

Figure 12. Structure of a multilevel directory (MLD)

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`.

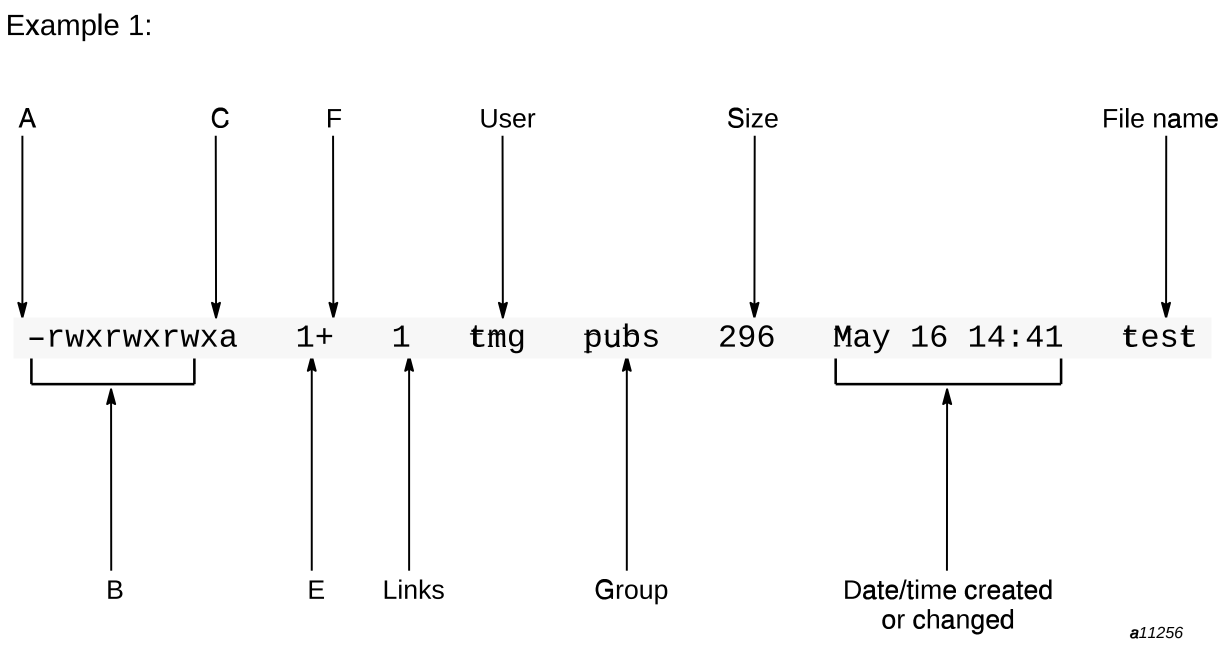
Example 1:
A
C
F
User
Size
File name
-rwxrwxrwxa 1+ 1 tmg pubs 296 May 16 14:41 test
B
E
Links
Group
Date/time created or changed
a11256

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.