

# SCO OpenServer Product Family Release 5

## subsystem(M)

---

### subsystem -- security subsystem component description

## Description

The operating system includes extensions to the UNIX system that segregate commands and data which are used to implement system services. Many of these commands have been grouped into subsystems. A group of commands and data performing similar security relevant tasks or together protecting a set of resources is termed a *protected subsystem*.

The operating system has the following protected subsystems:

- Memory
- Terminal
- Line Printer
- Backup
- Authentication
- Cron
- Audit

The description of each subsystem includes the following information:

#### Group and subsystem authorization name

Each subsystem is associated with a subsystem authorization. The commands and files associated with the subsystem take the subsystem authorization name as their group name. Users wishing to use the subsystem must have the appropriate subsystem authorization.

#### Commands

Each subsystem has a set of commands.

#### Helper programs

Some subsystems use helper programs. These are programs which call other programs.

#### Data files

A subsystem's programs use permanent and temporary data files.

The administrative functions associated with each subsystem can be selected from the SCOadmin manager. Help information is available with each option.

## The memory subsystem

The **mem** subsystem authorization is defined to grant users the ability to use the memory subsystem commands to view total system activity. Users without this authorization may only view their own processes. Traditional UNIX allowed any user to view total system activity. This authorization was introduced to allow the administrator to isolate users, and restrict their ability to sense the activity of other users.

### Mem authorization and group name

In order to look at information in the **mem** subsystem, an administrator must have the **mem** authorization. The administrator responsible for maintaining users' processes should be the only person with this authorization. This administrator may need to list users' processes in order to select one or more of them for removal (using the [kill\(C\)](#) command). The following is a table of command modifications managed by the **mem** authorization:

Command	With mem	Without mem
ps	lists all processes (standard behavior)	list processes owned by login user ID, or owned by real user ID of current process on current terminal
whodo	lists all processes (standard behavior)	list processes on terminals owned by user
ipcs	lists all objects (standard behavior)	list objects for which user is creator or owner or for which user has read access

### SCOadmin manager

The Memory subsystem does not have a SCOadmin manager selection as the Printer subsystem does. The Memory subsystem includes the system tables that contain information about memory and processes, which is accessed by several commonly-used UNIX utilities.

### Commands

**ps** An administrator with **mem** authorization can use the [ps\(C\)](#) command to list all users' processes. Using the command without the **mem** authorization shows only those processes associated with the user invoking it.

#### **whodo**

An administrator with **mem** authorization can use the [whodo\(C\)](#) command to list processes by terminal. Someone using the command without **mem** authorization sees only the processes associated with their terminal.

**ipcs** An administrator with **mem** authorization can use this command to view active semaphores, shared memory segments and message queues (known collectively as IPC entities). Without **mem**

authorization, a user is restricted to viewing IPC entities that they own or created and those which have read permission. Even entities that are writable, but not readable, cannot be displayed.

**crash** An administrator with **mem** authorization can run the **crash** program to report information on kernel data structures. The report includes security information.

An administrator can search for information by running **crash** and specifying an identifier name.

## Helper programs

**timex** Because **timex** uses internal kernel data structures, it must be run from an account in the **mem** group.

## Accounting programs

Accounting programs such as [sar\(ADM\)](#), [acctcom\(ADM\)](#), and [sar\(ADM\)](#) also use information in the **mem** subsystem. These programs must be run from an account in the **mem** group.

## Data files

All files through which programs may access kernel memory are protected with owner *root*, group **mem**, and mode *-r--r-----*. As for all files, the *root* account bypasses the discretionary check on these files, and *root* programs may violate the System Architecture requirement. All *root* programs (those running with effective ID equal to *root*) must take care when running other programs, because those programs inherit the right to modify the running copy of the TCB. The following files are protected by the **mem** subsystem according to the above owner, group, and mode:

*/etc/ps.data*

Cache relevant parts of the kernel symbol table to avoid lengthy lookups for each run of **ps**.

*/dev/mem*

Special device allowing access to physical memory including the operating system and all resident processes.

*/dev/kmem*

Special device allowing access to the operating system image.

*/dev/swap*

Special file for the disk partition used as the system swap device, storing memory images of non-resident processes.

*/unix* Executable file containing the binary copy of the operating system. Writing this file modifies the executing copy of the TCB when the system is rebooted.

## The terminal subsystem

The **terminal** subsystem protects the use of terminals by restricting the use of the [write\(C\)](#) and [mesg\(C\)](#) commands.

## Terminal authorization and group name

In order to send information from one terminal to another, the user sending information must have the **terminal** authorization and the receiving terminal must be configured to accept information from other terminals.

All terminals belong to the **terminal** group. Each terminal is owned by and can only be used by a given user identity.

## SCOadmin manager

The **terminal** subsystem does not control the Terminal Manager functions.

## Commands

When an unauthorized user uses the **write** command, any special control codes or escape sequences he sends are trapped and converted to presentable ASCII characters. All control codes are output as `^<char>` where `<char>` is the character whose ASCII code is the character sent plus 0100. For instance, ASCII NUL (0), SOH (1), and ACK (6) are output as `^@` (@ is 0100), `^A` (A is 0101) and `^F` respectively on the recipient's terminal. The ASCII ESC (033) character writes as `^I` and the DEL (0177) character writes as `^?`.

As an example of using the trusted write command, assume there is a hypothetical terminal that silently stores any string between two ASCII DC4 (024) characters. This string is transmitted from the same hypothetical terminal to the computer when the terminal receives a DC2 (022) character. Assume that a devious user knows the recipient of a **write** command has this terminal and tries to corrupt the recipient's session by sending a damaging message. If this user did not have the terminal authorization, the recipient would see the message: `How are y^Trm *^Tou today^E?`. The recipient would be alerted to an attempt on his session.

In addition, the **terminal** subsystem audits this event so you can locate suspect activity. On the other hand, if the sending user has the **terminal** authorization, the recipient would see the message: `How are you today?`

The following commands are modified to support the terminal subsystem.

Command	With terminal	Without terminal
<code>write</code>	unrestricted (standard behavior)	control codes output as <code>^&lt;char&gt;</code>
<code>mesg</code>	changes sense of group write permission only	same

A person with **terminal** authorization can use the [write\(C\)](#) command to write to another terminal and send control codes and escape sequences. A malicious user might use the command to send malicious commands and breach system security.

Without the authorization, a user can use the [write\(C\)](#) command, but control codes and escape sequences are displayed on the receiving terminal in their ASCII form, thus warning the recipient of suspicious activity. Such activity is recorded by the audit facilities.

The **mesg y** form of the command allows messages, but sets write permission for the terminal device group that has been set to **terminal** by the login program. The new write command is SGID to **terminal**, which allows it to send characters to user terminals that have used **mesg y** of the file enough for the **terminal** group to write to the terminal. The new **write** command handles this change. Unlike the less trusted **mesg**, UNIX **mesg** never allows any permission to all users.

## Data files

The data files for the terminal subsystem are the terminals themselves. They belong to the terminal group at the start and end of each session, and all access is denied except to the user. The preferred way for a user to open and close access to a terminal is to use the **mesg** command. When a session is not in progress on a terminal, only the super user can access the device file. Some terminal files are presented below.

### */dev/console*

This is the system console. Use of this terminal as a user terminal is discouraged because:

- Messages from the kernel appear on */dev/console*. To avoid losing these messages or intermixing them with user messages, it is better to use the console solely for the message output.
- On some systems, physical access to the console is equivalent to having access to the entire system. Use another terminal unless the system configuration prevents this. In any event, allow physical access to */dev/console* only to the most trusted users of the system.

### */dev/tty\**

Most of the terminals on the system are named */dev/tty1*, */dev/tty2*, */dev/tty3*, ... These devices may at times be owned by a protected subsystem (such as **uucp** or **terminal**) and be unavailable for general use. You have the option of configuring the terminals for login sessions, protected subsystems, or for nothing.

## Line printer subsystem

The purpose of the **lp** subsystem is to provide an administrative role that has control over printing facilities. Unlike the less trusted version of the **lp** commands, the trusted version does not require a special printer account that owns and executes (with the SUID bit set) all the printer programs. Instead, there is an *lp* group with multiple users as its members.

### Authorization/Group name

The **lp** authorization allows the user to be a printer administrator. This allows multiple Printer administrators. They force the administrator to have a login userid (LUID) of 0 or a login name of *lp*, a scheme that does not allow you much flexibility in account setups or individual accountability.

All printer administrators are allowed to execute some commands that non-authorized users cannot, and can perform certain actions within commands that are restricted from other users. Only administrators may run **accept**, **lpadmin**, **lpmove**, **lpsched**, **lpshut**, and **reject**. For the other commands, enhancements due to **lp** authorization are detailed under each command heading.

## SCOadmin manager

The **lp** authorization allows access to the printing functions using the Printer Manager.

### Commands

To determine the invoker, the Printer subsystem command uses the immutable login user ID (LUID). Less trusted versions use various other schemes, all of which could be fooled. The commands listed here perform exactly like their traditional (less trusted) versions except where noted:

#### **accept**

The **accept** command may only be used by printer administrators.

#### **cancel**

The less trusted version of **cancel** allowed any user to cancel any job. The originating user is notified of the cancellation via mail. The trusted version of **cancel** gives this right to printer administrators only. Mail is still sent to the originator when a job is canceled by the printer administrator. Other users can only remove jobs they submitted.

#### **disable**

The **disable** command operates without change from the less trusted version.

#### **enable**

The **enable** command operates without change from the less trusted version.

**lp** The trusted version of the **lp** command, with the **-w** option enabled by you, never writes to the terminal directly as does the less trusted version of **lp**. The trusted version of **lp** knows that the system prohibits direct writing to another user's terminal. Instead, the **write(C)** program is used to send the message; refer to the previous discussion of **write** in the **terminal** subsystem.

The trusted version of the **lp** command creates an output label for each file submitted. The output label contains the system label (the same as seen on most terminals), the owner, group, and mode of the file. To accurately determine the output label, the **lp** command cannot accept input from pipes. This is because the discretionary attributes of a file are not available if the file was accessed on the other end of a pipe. Note that input redirection and temporary files may still be printed.

Printer files are always copied to the printer spool by assuming the **-c** (copy) option, even if the user did not explicitly request it. By doing this, the **lp** subsystem ensures that the file cannot be altered between the time the request was made and the time it is printed. (The less trusted version of **lp** does not guarantee that the file cannot be updated, even while the printer is running.) As added protection, the file being copied is locked during the formation of the output label and the copy operation, so that the file and label output accurately reflects the file being printed.

#### **lpadmin**

This command may only be used by printer administrators.

#### **lpforms**

The **lpforms** command operates without change from the less trusted version.

#### **lpmove**

This command may only be used by printer administrators.

### **lpsched**

The **lpsched** command may only be used by the super user and *lp*. When the **lpsched** command uses a printer device dedicated to the **lp** subsystem, the subsystem guarantees exclusive use of the printer device each time it is used. Any prior activity (outside the **lp** subsystem) on that device is forcibly stopped. In this way, the **lp** subsystem ensures that the file being output is not interspersed with other output, unlike less trusted versions.

### **lpshut**

The **lpshut** command may only be used by printer administrators.

**lpstat** The trusted version of **lpstat** does not display other users' jobs if the invoking user does not have the **lp** authorization. Knowing the jobs of other users is not necessary since unauthorized users cannot hold or cancel those jobs anyway. Printer administrators see all printer jobs, and they can hold or cancel any job that has been submitted.

**reject** This command may only be used by printer administrators.

### **Data file**

*/usr/spool/lp*

All the files in this file hierarchy have the same formats and purposes as their counterparts in less trusted versions of UNIX. In the trusted version, the files are accessible by any printer administrator, so that the group permissions are the only ones of true importance. In all cases, the spool, its directories, and all data files allow no access to the user population. Hence, a user can be assured that a private file that is spooled for printing cannot be accessed or changed by untrusted users.

## **Backup subsystem**

The purpose of the **backup** subsystem is to provide a full set of disk and tape management tools without requiring detailed knowledge of UNIX. The backup administrator assumes responsibility of file system maintenance. The backup administrator is responsible for all actions which do not modify the format of file systems, while the *root* account is still responsible for formatting, configuring, and maintaining the consistency of file system disk partitions.

### **Authorization/Group name**

The **backup** authorization allows the user to be Backup administrator and perform file backups (*root* always has **backup** authorization). Users with **backup** or **create\_backup** authorization can make backups. Users with **backup** or **restore\_backup** authorization can restore from backups. The following authorizations are defined for the backup subsystem:

Authorization	Type	Purpose
backup	primary	enables full use of system backup commands
create_backup	secondary	allows user to create backups

queryspace	secondary	allows use of <b>df</b> program
restore_backup	secondary	allows user to restore from backups

All disk partitions are protected with owner *root*, group *backup* and mode *-r--r----*. The mount table (*/etc/mnttab*) is publicly readable, modified only by the **mount** command. The **df** program is SGID to *backup*, which enforces the **queryspace** and **backup** authorizations.

## SCOadmin manager

The **backup** authorization allows access to the backup functions using the Backup Manager.

## Commands

**df** The **df** command may only be used by Backup administrators. Otherwise, the options and output format remain the same as the less trusted version.

**mkfs** The **mkfs** command may only be used by a member of the *backup* group (or by the super user, which is discouraged). As always, this command must be used to initialize a filesystem after the partitions are laid out. Immediately after **mkfs** is run, you should run **labelit** to complete the initialization.

### labelit

The **labelit** program, documented in [volcopy\(ADM\)](#), associates the filesystem with a directory mount point.

## Helper programs

### /etc/mount

This program is used by **backupif** to display and modify the mounted file systems.

### /etc/fsck

This is used by **backup** to check and repair filesystems.

### /usr/bin/backup

This program is used to copy entire UNIX and XENIX filesystems to either magtape or cartridge tape.

### /bin/xbackup

This program is used to copy entire XENIX disk filesystems to either magtape or cartridge tape.

### /bin/xrestore

This program is used to replace entire XENIX filesystem images on magtape or cartridge tape to a clean (newly formatted with **mkfs**)

### /usr/bin/restore

This program is used to replace entire XENIX or UNIX filesystem images on magtape or cartridge tape.

### /usr/bin/cpio

This is the default backup program. **cpio** makes non-filesystem specific copies of filesystem data.

## Data files

### */etc/default/filesys*

This file contains the relationship between mounted filesystem devices and the directories on which they are mounted (mount points). It is used to display that relationship in both **df** and the **backup** selection. Because altering this file would display erroneous information to backup administrators and reading this file defaults the access protection created for the **backup** subsystem, this file must be accessible to the *backup* group only.

### */dev/[r]d[s]k\**

These block and character special files are the buffered interfaces to the disk partitions you have set up. They are used for mounting the filesystem they contain onto a directory. The *backup* group must be able to read and write these files. It is a severe security breach if others can access these files in any way.

## Authentication subsystem

The Authentication subsystem provides you with an exhaustive set of account management services. These services are:

- self-checking to prevent dangerous actions, and
- monitored extensively by the auditing system.

## Authorization/Group name

The **auth** authorization allows an Authentication administrator to perform sensitive actions on the Authentication database. This database contains all information on account ownership, types, authorizations, locked status, login times, password change times, and various other parameters.

With the **auth** authorization, an Authentication administrator may alter Authentication parameters for other users. Because this database directly controls the attributes of any account on the system, this subsystem controls user access to your system. The trust you place in the system can be no greater than that placed in the Authentication administrators. Not only must they be trustworthy people, but they must also not leave any uncorrected mistakes when assigning authorizations to the accounts they manage.

## SCOadmin manager

The **auth** authorization allows access to the user account management functions using the Account Manager.

## Commands

### **passwd**

The **passwd** command in UNIX has been greatly enhanced for both security and flexibility. The trusted system checks on system-wide password parameters as well as user-specific ones and, depending on the results found, the user has a choice of choosing their own password or having

one chosen for them. You can set each account to do either one of these, or do both. A closely related change is that, regardless of the method for getting the password, you can have the system screen passwords that are probable guesses by intruders. The password selection method, as well as the optional restriction screening, are set by Authentication administrators in SCOadmin for a single account or for system-wide use.

**login** The **login** command is no longer available as a command used in a session to start a new session. Instead, a user must first log out before logging in as another user.

Sublogins are forbidden since the LUID of a session may not change once it is set. This is to guarantee to you that the owner of a session is known at all times. If the **login** program were allowed to be run from a session, the login USERID would have to change and the guarantee would be broken.

The **login** program is still invoked from **getty** to start a user session. The procedure for logging in is almost the same. The user supplies a login name and the system requests a password. Once the password is entered, the system either lets the user log in or rejects the login attempt. A user may be rejected for a number of reasons:

1. The account does not exist.
2. The password was entered incorrectly.
3. The password lifetime has been passed.
4. The number of unsuccessful attempts made to the account has surpassed a system or account threshold.
5. The number of unsuccessful attempts made to the terminal has surpassed a system or terminal threshold.
6. An Authentication administrator has unconditionally locked the account.

Reasons 3 through 6 notify the user that the Authentication administrator has locked the account.

If the user enters the correct login name/password combination, the last successful and unsuccessful login times are displayed on the terminal. The user should view the dates and times of each to determine if someone else has used the account. These dates may also be used to determine whether a Trojan horse program is simulating the **login** procedure to obtain a password. A user with doubts about the authenticity of the login dates and times should report it to you. The earlier you take action on this, the better you can use fresh audit trails and people's recollections to find the source of the problem.

**su** The **su** program has been strengthened a great deal for security. It now uses information from the Authentication database in determining whether or not to allow a user to ``switch" to another user. The following rules apply:

- A user cannot use **su** to enter an account that has been locked.
- The **su** command cannot be used as a means to bypass the lock-checking done by **login**, **at**, and **cron**.

## newgrp

The **newgrp** command operates without change from the less trusted version.

**auths** The **auths** command is especially tailored for UNIX to allow all users to adjust their authorizations. No user can increase authorizations, but one can temporarily decrease authorizations in order to run an untrusted program or to prevent mistakes. More details on the authorizations and syntax are given in the man page for [auths\(C\)](#).

## Data files

### */usr/adm/sulog*

This file keeps track of the history of use of the **su** program. Each line represents an attempt to run the **su** program. The date and time are first recorded on the line. Then, a ``-" means the attempt failed; a ``+" means the attempt succeeded. After the ``-" or ``+" code, the terminal of the attempt is provided. Last, the login name (using the login UID) of the invoker of **su**, together with the login name of the (attempted) changed real UID is presented. As an example, the following log excerpt presents some interesting situations:

```
SU 02/29 19:19 + tty?? root-lp
SU 03/01 20:22 + tty2 blf-root
SU 03/04 04:13 + tty2 fred-proj1
SU 03/07 20:30 - tty2 reese-star
SU 03/07 20:30 + tty2 reese-star
SU 03/07 21:38 + modem auth-root
SU 03/07 21:39 + tty2 blf-root
SU 03/07 21:39 - tty7 daa-root
SU 03/07 21:40 - tty7 daa-root
SU 03/07 21:41 - tty7 daa-root
SU 03/07 21:47 + tty2 fred-proj1
```

Foremost, it appears as though the user *daa* is attempting to break into the *root* account, for there are many unsuccessful attempts (denoted with the ``-" attribute) in rapid succession. That should be investigated further.

The **su** program does not require one to become the *root* user. In the log above, users *root*, *fred* and *reese* chose to assume the identities of other users. In the effort by *reese* to become the *star* user, the first attempt failed and the next immediately succeeded. This occurs frequently and is quite natural when users mistype the password of the other account. You should get suspicious, however, when the number of unsuccessful attempts becomes large. Such attempts, like the case with *daa* above, probably means a breach of security.

The **su** program was used by *root* to enter the **lp** account. This occurrence was detached from any terminal, because of the special terminal designation of *tty??*. This particular case occurred from */etc/rc* where the **lpsched** daemon is run.

The */usr/adm/sulog* file needs attention periodically. It should be examined and then pruned, saving the most recent entries. The entries removed from the file should be archived if possible rather than completely deleted.

### */tcb/files/auth*

This directory consists of subdirectories that contain private account data for all the accounts in the system. There is a file for each account. Because of the sensitive nature of the data here, all

these files are completely protected from the users.

### */etc/auth/system*

This directory contains the system-wide authorization data for the machine. The */etc/auth/system* directory contains the Terminal Control database, the File Control database, the Command Control database and the System Defaults database. This information is accessible to the users but not writable. The */etc/auth/subsystems* directory contains one file per protected subsystem, each containing the user permissions for that protected subsystem. This permissions file may only be read by the programs that are part of that protected subsystem, and is written by the *auth* user.

## **cron subsystem**

The purpose of the **cron** subsystem is to allow **cron**, **at**, and **batch** services that are audited as closely as normal login sessions. The **cron** subsystem provides a useful interface for controlling these facilities.

### **Authorization/Group name**

The authorization for the **cron** subsystem is given to cron administrators who are allowed to view or alter the authority for users to run the services associated with the **cron** subsystem. A user may run the programs of the **cron** subsystem (excluding the use of the Cron Manager selections) without the authorization, provided that a **cron** administrator has granted the authority.

### **SCOadmin manager**

The **cron** authorization allows access to the process management functions using the Cron Manager.

### **Commands**

#### **at, batch, crontab**

These **at** commands operate in the same way as the less trusted version, except that the LUID (login UID), rather than the real UID, is used by **at** in determining the user. Because the LUID cannot be altered during a session, it promotes better accountability. **at** and **batch** jobs run with all of the login, real, and effective UIDs set to that of the login user.

### **Helper programs**

#### **/tcb/lib/cron**

This is the **cron** daemon that actually runs all **at**, **batch**, and **crontab** jobs. The **at**, **batch**, and **crontab** commands merely queue the jobs for the **cron** daemon to run. This daemon validates the account (ensures the account is not locked) before running the job.

### **Data files**

Although enumerated here, these data files are not manipulated directly by the **cron** administrator because of the arcane rules historically applied to them by the **cron** subsystem programs. Instead, the SCOadmin manager provides a more coherent interface, reducing the possibility that users or permissions are set up incorrectly.

#### */usr/lib/cron*

This is the directory containing all the **cron** administrative files.

*/usr/lib/cron/at.allow*

This file lists the users allowed to execute the **at** or **batch** programs. If this file exists, it is used to determine the user's authority.

*/usr/lib/cron/at.deny*

This file lists the users denied access to the **at** or **batch** programs. If */usr/lib/cron/at.allow* does not exist, */usr/lib/cron/at.deny* is used to determine the user's authority. You should be aware that an empty *at.deny* file permits access for all users.

*/usr/lib/cron/cron.allow*

This file lists the users allowed to execute the **crontab** program. If this file exists, it is used to determine the user's authority.

*/usr/lib/cron/cron.deny*

This file lists the users denied access to the **crontab** program. If */usr/lib/cron/cron.allow* does not exist, */usr/lib/cron/cron.deny* is used to determine the user's authority. You should be aware that an empty *cron.deny* file permits access for all users.

*/usr/lib/cron/.proto*

This file contains a list of commands that are executed before every **at** job. It contains commands primarily used to fix and restrict the environment of the user before running the job submitted.

*/usr/lib/cron/.proto.b*

This file contains a list of commands that are executed before every **batch** job. It contains commands primarily used to fix and restrict the environment of the user before running the job submitted.

*/usr/lib/cron/log*

This is a log of all **at**, **batch**, and **crontab** activity reported by the **cron** daemon since the system was rebooted. It provides an accurate ASCII log of all user initiated non-terminal activity. If the system is up for a very long time and there are many job submissions or **crontab** activity, this file should be periodically examined, pruned, and archived.

*/usr/lib/cron/OLDlog*

This is the log associated with the last time the system was up. Upon startup, the **cron** daemon moves any */usr/lib/cron/log* file here.

*/usr/spool/cron*

This is the root of the subtree where all **at**, **crontab**, and **batch** jobs are stored. **at** and **batch** jobs are automatically cleared when they have finished executing. The **-r** option of **crontab** removes a **crontab** job.

## Audit subsystem

The purpose of the **audit** subsystem is to provide an administrative role that has control over auditing facilities.

## Authorization/Group name

The **audit** authorization allows the user to be the audit administrator. The audit administrator can enable and disable auditing, examine audit records, generate reports and alter audit parameters.

## SCOadmin manager

The **audit** authorization allows access to the audit functions using the Audit Manager.

## Commands

### **auditcmd**

The command interface for audit subsystem activation, termination, statistic retrieval, and subsystem notification.

### **auditd**

The **auditd** utility is the daemon that runs when auditing is enabled.

### **reduce**

This program performs audit data analysis and reduction.

## Data files

*/tcb/files/audit/audit\_parms*  
audit parameters file

*/tcb/files/audit/\**  
audit log file directory

*/tcb/audittmp*  
audit compaction file directory

## Creating a new subsystem

The system administrator can create additional subsystems as desired.

To create a new subsystem, do the following:

1. Add a line to */etc/auth/system/authorize* of the following format:

*subsystem:class1,class2,...,classn*

where:

*subsystem* the name of your new subsystem  
*class1...n* optional name(s) of the authorizations

For example:

```
backup : dump, freespace
```

This defines the ``backup" subsystem (used to control read access to filesystems), which has two special cases: ``dump", actually make a backup of the filesystem, and ``freespace", ability to read the filesystem to determine how full it is (but for no other reason).

2. Create a group with the same name as the subsystem. Make the (empty) file `/etc/auth/subsystems/subsystem`, owner `auth` or `bin`, and the group owner is the new group `subsystem` with a mode of at least 440 (the mode must not grant any write permission to ``other").

You are finished creating the new subsystem. It should be automatically recognized and understood by the system and the SCOadmin manager. There can be at most 32 subsystems and all names must be unique.

## See also

[audit\(HW\)](#), [auditcmd\(ADM\)](#), [auditd\(ADM\)](#), [authck\(ADM\)](#), [authorize\(F\)](#), [auths\(C\)](#), [authcap\(F\)](#), [chg\\_audit\(ADM\)](#), [integrity\(ADM\)](#), [reduce\(ADM\)](#)

Chapter 5, ``Maintaining system security" in the *System Administration Guide*

## Standards conformance

**subsystem** is not part of any currently supported standard; it is an extension of AT&T System V provided by The Santa Cruz Operation, Inc.

*1 May 1995*