man pages section 1: User Commands

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 835-8001
December 2000

Please
Recycle

Contents

Preface

Overview

A man page is provided for both the naive user and the sophisticated user who is familiar with the Trusted Solaris operating environment and is in need of online information. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Trusted Solaris Reference Manual

In the AnswerBook2™ and online man command forms of the man pages, all man pages are available:

- Trusted Solaris man pages that are unique for the Trusted Solaris environment
- SunOS 5.8 man pages that have been changed in the Trusted Solaris environment
- SunOS 5.8 man pages that remain unchanged.

The printed manual, the *Trusted Solaris 8 Reference Manual* contains:

- Man pages that have been added to the SunOS operating system by the Trusted Solaris environment
- Man pages that originated in SunOS 5.8, but have been modified in the Trusted Solaris environment to handle security requirements.

Users of printed manuals need both manuals in order to have a full set of man pages, since the *SunOS 5.8 Reference Manual* contains the common man pages that are not modified in the Trusted Solaris environment.

Man Page Sections

The following contains a brief description of each section in the man pages and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2 of this volume.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals, and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.
- Section 9 provides reference information needed to write device drivers in the kernel operating systems environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer may include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man`(1) for more information about man pages in general.

NAME

This section gives the names of the commands or functions documented, followed by a brief description of what they do.

SYNOPSIS

This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full pathname is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.

The following special characters are used in this section:

[ ] The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.

. . . Ellipses. Several values may be provided for the previous argument, or the previous argument can be specified multiple times, for example, ‘ "`filename . . .`" .

| Separator. Only one of the arguments separated by this character can be specified at a time.

{ } Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

PROTOCOL

This section occurs only in subsection 3R to indicate the protocol description file.

DESCRIPTION

This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, functions and such, are described under USAGE.

IOCTL

This section appears on pages in Section 7 only. Only the device class which supplies appropriate parameters to the ioctl (2) system call is called `ioctl` and generates its own heading. `ioctl` calls for a specific device are listed alphabetically (on the man page for that specific device). `ioctl` calls are used for a particular class of devices all of which have an `io` ending, such as `mtio`(7I)

OPTIONS

This secton lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.

OPERANDS

This section lists the command operands and describes how they affect the actions of the command.

OUTPUT

This section describes the output – standard output, standard error, or output files – generated by the command.

RETURN VALUES

If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or –1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.

ERRORS

On failure, most functions place an error code in the global variable `errno` indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.

USAGE

This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality:

- Commands
- Modifiers
- Variables
- Expressions
- Input Grammar

EXAMPLES

This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as `example%`, or if the user must be root, `example#`. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.

ENVIRONMENT VARIABLES

This section lists any environment variables that the command or function affects, followed by a brief description of the effect.

EXIT STATUS

This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.

FILES

This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.

ATTRIBUTES

This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See `attributes`(5) for more information.

SUMMARY OF TRUSTED SOLARIS CHANGES

This section describes changes to a Solaris item by Trusted Solaris software. It is present in man pages that have been modified from Solaris software.

SEE ALSO

This section lists references to other man pages, in-house documentation and outside publications. The references are divided into two sections, so that users of printed manuals can easily locate a man page in its appropriate printed manual.

DIAGNOSTICS

This section lists diagnostic messages with a brief explanation of the condition causing the error.

WARNINGS

This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.

NOTES

This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.

BUGS

This section describes known bugs and, wherever possible, suggests workarounds.

CHAPTER

User Commands

NAME

Intro – Introduction to commands and application programs

DESCRIPTION

This section describes Solaris and Trusted Solaris™ commands. These commands can be:

- Commands that are unique to and originate in the Trusted Solaris environment, such as `getlabel`(1), which allows users to see the label of a file.
- `SunOS` 5.8 (Solaris 8) commands that have been modified to work within the Trusted Solaris security policy, such as `tar`(1), which has a new `−S` option that maintains security attributes, such as labels, on archives. Man pages for modified commands have been rewritten to remove information that is not accurate for how the command behaves within the Trusted Solaris operating environment. Modified man pages also have added descriptions for new features, options, and arguments.
- `SunOS` 5.8 commands that remain unchanged from the Solaris 8 release, such as `who`(1).

In the Trusted Solaris environment, even if a particular command is installed, not all users may be configured to use that command. Your site's security administrator may restrict the use of any command and may change any command's *security attributes* using *execution profiles*. (Security attributes, execution profiles, and other Trusted Solaris terms are defined in the DEFINITIONS section below.) Users who do not have a particular command in any of their execution profiles cannot use that command. Even if a command is in one of a user's execution profiles, that command still may not work as expected because the *label range* or another of the command's *security attributes* specified in the execution profile may limit how the command can be used. If any of the commands described in this section does not work at all or does not work as expected, check with your security administrator.

Trusted Solaris Information Label Changes

Information labels (ILs) are not supported in Trusted Solaris 7 and later releases. Trusted Solaris software interprets any ILs on communications and files from systems running earlier releases as `ADMIN_LOW`.

Objects still have CMW labels, and CMW labels still include the IL component: `IL[SL]`; however, the IL component is fixed at `ADMIN_LOW`.

As a result, Trusted Solaris 7 and later releases have the following characteristics:

- ILs do not display in window labels; SLs (Sensitivity Labels) display alone within brackets.
- ILs do not float.
- Setting an IL on an object has no effect.
- Getting an object's IL will always return `ADMIN_LOW`.

- Although certain utilities, library functions, and system calls can manipulate IL strings, the resulting ILs cannot be set on any objects.
- Sensitivity labels, not information labels, display on printer banners.
- IL–related privileges are no longer used.
- In auditing, the `ilabel` token is recorded as `ADMIN_LOW`, when it is recorded. The audit event numbers 519 (AUE_OFLOAT), 520 (AUE_SFLOAT), and 9036 (AUE_iil_change) continue to be reserved, but those events are no longer recorded.

SPECIALIZED PAGES

Section 1 specialized pages are categorized as follows:

1B Commands found only in the *SunOS/BSD Compatibility Package.* Refer to the *Source Compatibility Guide* for more information.

Printer commands in this section are modified in the Trusted Solaris environment.

1C Commands for communicating with other systems.

No commands in this section are modified in the Trusted Solaris environment.

1F Commands associated with *Form and Menu Language Interpreter* (FMLI).

No commands in this section are modified in the Trusted Solaris environment.

1S Commands specific to the SunOS system.

No commands in this section are modified in the Trusted Solaris environment.

OTHER SECTIONS

See these sections of the *man pages section 1M: Trusted Solaris System Administration Commands* and the *man pages section 1M: System Administration Commands* for more information.

- Section 1M for system maintenance commands.

  Some commands in this section have been modified in the Trusted Solaris environment, and there are added commands.
- Section 4 for information on file formats.

  Some file formats in this section have been modified in the Trusted Solaris environment, and there are added entries.
- Section 5 for descriptions of publicly available files and miscellaneous information pages.

  The Trusted Solaris environment adds privilege macros and PAM module authentication information to this section.

- Section 6 in this manual for computer demonstrations.

  No entries in this section are modified in the Trusted Solaris environment.

For tutorial information about commands and proceduresthat are unchanged from the Solaris 8 release, see:

- *OpenWindows Advanced User's Guide*

For tutorial information about commands and procedures particular to the Trusted Solaris environment, see the Trusted Solaris administrator's document set.

Trusted Solaris Manual Page Display

The manual pages are available in three formats: online, AnswerBook2™ collections, and in printed form.

Online man pages — Include all man pages in the Solaris and Trusted Solaris environments. To view, enter the man page name, such as **`man ppriv`** or **`man cp`** in a terminal window in the Trusted Solaris environment.

AnswerBook2™ collections — Include all man pages in the Trusted Solaris environment in the *Trusted Solaris Reference Manual Collection*, and all man pages in the Solaris operating environment in the *Solaris Reference Manual Collection*. Hyperlinks connect Trusted Solaris man pages to Solaris man pages where necessary. To view, go to `http://docs.sun.com`, or use the collections on your AnswerBook2 server.

Printed *Trusted Solaris 8 Reference Manual* — Includes only those man pages that have been modified from their Solaris counterparts, or that originate in the Trusted Solaris environment. Printed versions of SunOS 5.8 man pages are found in the *SunOS 5.8 Reference Manual*.

Manual Page Command Syntax

Unless otherwise noted, commands described in the SYNOPSIS section of a manual page accept options and other arguments according to the following syntax and should be interpreted as explained below.

name [-*option*...] [*cmdarg*...] where:

[ ] — Surround an *option* or *cmdarg* that is not required.

... — Indicates multiple occurrences of the *option* or *cmdarg*.

name — The name of an executable file.

{ } — The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.

option — (Always preceded by a “–”.) *noargletter...* or, *argletter optarg*[`,` ...]

noargletter — A single letter representing an option without an option-argument. Note that more than one *noargletter* option can be grouped after one “–” (Rule 5, below).

argletter — A single letter representing an option requiring an option-argument.

optarg — An option-argument (character string) satisfying a preceding *argletter*. Note that groups of *optargs* following an *argletter* must be separated by commas, or separated by a tab or space character and quoted (Rule 8, below).

cmdarg — Path name (or other command argument) *not* beginning with “–”, or “–” by itself indicating the standard input.

Command Syntax Standard: Rules

These command syntax rules are not followed by all current commands, but all new commands will obey them. `getopts`(1) should be used by all shell procedures to parse positional parameters and to check for legal options. It supports Rules 3-10 below. The enforcement of the other rules must be done by the command itself.

1. Command names (*name* above) must be between two and nine characters long.
2. Command names must include only lower-case letters and digits.
3. Option names (*option* above) must be one character long.
4. All options must be preceded by “–”.
5. Options with no arguments may be grouped after a single “–”.
6. The first option-argument (*optarg* above) following an option must be preceded by a tab or space character.
7. Option-arguments cannot be optional.
8. Groups of option-arguments following an option must either be separated by commas or separated by tab or space character and quoted (–`o xxx,z,yy` or – `o "xxx z yy"`).
9. All options must precede operands (*cmdarg* above) on the command line.
10. “– –” may be used to indicate the end of the options.

11.The order of the options relative to one another should not matter.

12.The relative order of the operands (*cmdarg* above) may affect their significance in ways determined by the command with which they appear.

13.“–” preceded and followed by a space character should only be used to mean standard input.

Rules for the Display and Entering of Labels

The Trusted Solaris environment always displays *label*s in uppercase. Users may enter labels in any combination of uppercase and lowercase. Depending on how the system is configured and how the user is set up, a user may see *sensitivity labels* or no labels at all in the top frame of each window and in the *trusted stripe*, among other places in the user’s workspace. Sensitivity labels display within brackets, in the long form (within the window system).

If you need to enter labels on the command line, see the expanded Rules for the Display and Entering of Labels in `Intro`(1M).

DEFINITIONS

ACL

See *access control list*

Access Control List

A type of *discretionary access control* based on a list of entries that the owner can specify for a file or directory. An access control list (ACL) can restrict or permit access to any number of individuals and groups, allowing finer-grained control than provided by the standard UNIX *permission bits.*

Accreditation Range

Actually not a range, but a set made up of labels. See *user accreditation range* and *system accreditation range* for more about the two types of accreditation ranges in the Trusted Solaris environment.

Allocatable Device

A device to which access is controlled in the Trusted Solaris environment by making the device allocatable to a single user at a time. Not all devices are allocatable. Allocatable devices include tape drives, floppy drives, audio devices, and CD-ROM devices. (See *device allocation*.)

Authorization

A right granted to a user to perform an action that would otherwise not be allowed by the Trusted Solaris *security policy.* Certain commands require the user to have certain authorizations to succeed. Similar to the use of *privilege* on programs.

CDE action

A bundling mechanism used in Trusted Solaris to allow one or more commands to be specified for a particular task that in turn may be assigned to one or more users. A CDE action can have a set of options and arguments specified along with each of the command(s) and can use a dialog box to prompt the user for additional arguments. Each CDE action usually has its own icon, is assigned its own set of *security attributes*, and may be specified in an *execution profile.*

CMW Label

Consists of an *information label* followed by a *sensitivity label* in brackets, in the form: INFORMATION LABEL [SENSITIVITY LABEL]. See Trusted Solaris Information Label Changes under DESCRIPTION.

Classification

The hierarchical portion of a *sensitivity label* or *clearance*, each of which has only one classification. In a sensitivity label assigned to a file or directory, a classification indicates a relative level of protection based on the sensitivity of the information contained in the file or directory. In a clearance assigned to a user and to *process*es that execute applications and commands on behalf of the user, a classification indicates a level of trust.

Clearance

The upper bound of the set of labels at which a user may work, whose lower bound is the *minimum label* assigned by the security administrator as the *initial label*. There are three types of clearance: *user clearance*, *process clearance*, and *session clearance*.

Compartments

A set of words in a *sensitivity label* or *clearance*. The compartment represents areas of interest or work groups associated with the labels that contain them and with the files that are assigned the labels and the individuals that work with them.

DAC

See *discretionary access control*.

Discretionary Access Control

The type of access granted or denied by the owner of a file or directory at the discretion of the owner. The Trusted Solaris environment provides two kinds of discretionary access controls (DAC): *permission bits* and *access control lists*.

Device Allocation

A mechanism for protecting the information on an *allocatable device* from access by anybody except the user who allocates the device. Until a device is deallocated, no one but the user who allocated a device can access any information associated with the device. Device clean scripts may be run when the device is deallocated to clean information from the device before the device may be accessed again by another user. For a user to allocate a device, that user must have been granted the device allocation *authorization* by the *security administrator*, and the user process' sensitivity label must be within the device’s*label range*. Upon deallocation of a storage device, such as a tape or floppy drive, the system prompts the user to remove the storage media and supplies a label that the user is prompted to write on the physical label, for guidance on how the media should be handled, if sensitivity labels are configured for display.

Dominate

When any type of label (*sensitivity label* or *clearance*) has a security level equal to or greater than the security level of another label to which it is being compared, the first label is said to dominate the second. The *classification* of the dominant label must equal or be higher than the classification of the second label, and the dominant label must include all the words (*compartments* and *markings*, if present) in the other label. Two equal labels dominate each other. Sensitivity labels are compared for dominance when MAC decisions are being made. See *strictly dominate.* See also Trusted Solaris Information Label Changes under DESCRIPTION.

Execution Profile

See *rights profile*.

File Access

Because in UNIX systems just about everything (including a spreadsheet, a printer, a letter, a chapter of a book, or a mail message) is handled as a file, which is stored in a directory—to do just about anything the user must access files and directories. The conditions for access are described here. (Even though devices are treated as files in the UNIX system, devices have slightly different mandatory access rules than files or directories, and these rules are separately described in this section.) A file, directory, or device may be accessed in three ways:

- The *name* of the file, directory, or device may be *viewed*,
- The *contents* or the *attributes* of the file, directory, or device may be *viewed*, or
- The *contents* or the *attributes* of the file, directory, or device may be *modified*.

In the Trusted Solaris environment, each of these types of access is granted or denied based not only on whether the basic UNIX *discretionary access control* checks have been passed but also on whether the *mandatory access control* checks have been passed.

All types of access require that the *sensitivity label* of the *process dominate*s the sensitivity label of all directories in the pathname and that the owner of the process (the person who executed the command) has discretionary search access for each directory in the pathname. View access to the name of the file, directory or device requires only that this part of the check is passed.

For view access (read access) to the contents or attributes of a file or a directory, the process’ sensitivity label must dominate the sensitivity label of the file or directory. For view access to the contents of a device (for example, so you can read information stored on a tape in a tape drive), the process’ sensitivity label must be equal to the sensitivity label of the device. The owner of the process also must have discretionary read access to the file, directory, or device.

For a process to write into a file or to modify the file’s attributes, the sensitivity label of the file must dominate the sensitivity label of the process and must be within the process’ clearance, which is set to be the *session clearance*. For a process to write into a directory (create a file), the sensitivity label of the process must equal the sensitivity label of the directory. For a process to write to a device (for example, store information on a tape in a tape drive), the sensitivity label of the process must also equal the sensitivity label of the device. The security policy for device files can differ from the policy for regular files based on how the policy is defined in the `device_policy`(4) file, which can be changed by the security administrator. The owner of the process must have discretionary write access to the file, directory, or device.

For each type of failure of a MAC or DAC check, a specific override *privilege* may be applied to the command, depending on the type of access being denied. A privilege can be made available to a command only by the action of a security administrator, because the security administrator must ensure that the user who executes the command is cleared to, or that the command may be trusted to, use the privilege in a trustworthy manner.

These conditions and the listed override privileges apply to any type of access:

- If the sensitivity label of the process does not dominate the sensitivity label of a directory in the pathname, then the process must have the privilege to search up (search a directory whose sensitivity label dominates the sensitivity label of the process), which is `file_mac_search`.
- If the user executing the command does not have discretionary search permission for a directory in the pathname, then the process must have the privilege to override search restrictions when accessing a directory, which is `file_dac_search`.

These conditions and the listed override privileges apply to view (read) access:

- If the sensitivity label of the process does not dominate the sensitivity label of a file or equal the sensitivity label of a directory or device, then the process must have the privilege to override MAC read restrictions, which is `file_mac_read`.
- If the user executing the command does not have discretionary read permission for the file or directory, then the process must have the privilege to override DAC read restrictions, which is `file_dac_read`.

These conditions and the listed override privileges apply to modify (write) access:

- If the sensitivity label of file does not dominate or if the sensitivity label of a directory or device does not equal the sensitivity label of the process, the process must have the privilege that overrides MAC write restrictions, allowing the user to write up and to write above the user's clearance, which is `file_mac_write`.
- If the user executing the command does not have discretionary write permission for the file or directory, then the process must have the privilege to override DAC write restrictions, which is `file_dac_write`.

Initial Label

The user's *minimum label* set by the security administrator when specifying a user's security attributes, this is the *sensitivity label* of the first workspace that comes up after the user's first login.

Label

A security identifier assigned to a file or directory based on the level at which the information being stored in that file or directory should be protected. Depending on how the *security administrator* has configured the environment, users may see the complete *CMW label*, only the *sensitivity label* portion, or no labels at all.

Label Range

A set of sensitivity *label*s assigned to file systems, hosts, networks, sockets, printers, workstations, and *allocatable device*s, specified by designating a maximum label and a minimum label. In general, restricted label ranges can be used to restrict access to a device such as a workstation or a printer. For hosts and networks, label ranges are used to limit the labels at which communications are allowed. For file systems, the minimum and maximum labels limit the sensitivity labels at which information may be stored on each file system. Trusted Solaris environments have multilabel file systems configured with a label range from the lowest sensitivity label to the highest sensitivity label. Remote hosts that do not recognize labels are assigned a single sensitivity label, along with any other hosts that the security administrator wishes to restrict to a single label; the label range on a file system mounted from such a host is configured to be restricted to the same sensitivity label as the remote host's sensitivity label. For allocatable devices, the minimum and maximum labels limit the sensitivity labels at which devices may be allocated and restrict the sensitivity labels at which information can be stored or processed using the device.

MAC

See *mandatory access control.*

MLD

See *multilevel directory*.

Mandatory Access Control

A type of control based on comparing the *sensitivity label* of a file, directory, or device to the sensitivity label of the *process* that is trying to access it. Even though directories and devices are managed like files in the UNIX system, different MAC rules apply to directories and devices than the rules that apply to files. Before a file may be accessed for writing, MAC checks ensure that the sensitivity label of the file dominates the sensitivity label of the process—a policy called *write up.* A process cannot write to a file whose sensitivity label is higher than the process' clearance, which is set to be equal to the *session clearance.* (The write up policy also includes *write equal.*) Before a directory or a device may be accessed for writing,MAC checks ensure that the sensitivity label of the directory or device is equal to the sensitivity label of the process—a policy called *write equal.* Before a file or directory may be accessed for viewing (reading or searching), MAC checks ensure that the sensitivity label of the process dominates the sensitivity label of the file or directory—a policy called *read down.* Before a device may be accessed for viewing, MAC checks ensure that the sensitivity label of the process equals the sensitivity label of the device—a policy called *read equal.* (The read down policy also includes *read equal.*)

The rule that applies when a process at one sensitivity label attempts to read or write a file at another sensitivity label is *write up, read down* (WURD). The rule that applies when a process at one sensitivity label attempts to write a directory at another sensitivity label is *write equal, read down.* The rule that applies when a process at one sensitivity label attempts to write a device at another sensitivity label is *read equal, write equal*.

Minimum Label

For users, the lower bound of the *sensitivity label*s at which a particular user can work, which is specified as the *initial label* by the security administrator while setting the user's account. For the system, the sensitivity label specified in the minimum label field by the security administrator in the `label_encodings` file sets the lower bound for all users.

Multilevel Directory

A directory in which information at differing *sensitivity labels* is maintained in separate subdirectories called *single-level directories* (SLDs), while appearing to most interfaces to be a single directory under a single name. In the Trusted Solaris environment, directories that are used by multiple standard applications to store files at varying labels, such as the `/tmp` directory, `/var/spool/mail`, and users' `$HOME` directories, are set up to be MLDs. A user working in an MLD sees only files at the sensitivity label of the user's process.

Permission Bits

A type of *discretionary access control* in which the owner specifies a set of bits to signify who can read, write, or execute a file or directory. Three different sets of permissions are assigned to each file or directory: one set for the owner; one set for all members of the group specified for the file or directory; and one set for all others. See also *access control lists.*

Privilege

A right granted to a process executing a command that allows the command or one or more of its options to bypass some aspect of *security policy.* A privilege is only granted by a site's *security administrator* after the command itself or the person using it has been judged to be able to use that privilege in a trustworthy manner.

Process

An action executing a command on behalf of the user who invokes the command, a process receives a number of security attributes from the user, including the user ID (UID), the group ID (GID), the supplementary group list, and the user's audit ID (AUID). *Security attribute*s received by a process include any privileges available to the command being executed, the process clearance (which is set to be the same as the *session clearance*), and the sensitivity label of the current workspace. In a *rights profile*, a *process label* and *clearance* can be assigned to a command so that when the command runs, its process gets the *clearance* and *label* specified in the *rights profile*.

Process Clearance

Clearance assigned to a command in a *rights profile*, which becomes the clearance of the *process* executing the command.

Process Label

Label assigned to a command in a *rights profile*, which becomes the label of the *process* executing the command.

Profile Mechanism

A mechanism that allows site security administrators to bundle commands, CDE actions, and the *security attributes* associated with those commands and actions into an *execution profile,* which may then be assigned to one or more users depending on the tasks that they need to perform.

Rights Profile

A bundling mechanism for commands and *CDE actions* and for optional security attributes that may be assigned to the commands and CDE actions. Rights profiles allow Trusted Solaris administrators to control who can execute which commands and to control the attributes these commands have when they are executed. When a user logs in, all execution profiles assigned to that user are in effect, and the user has access to all the commands and CDE actions assigned in all of that user's profiles. Also called a *right* or *profile.*

Routing

When a Trusted Solaris host boots, it loads routing information so it can transmit data. If the file `/etc/tsolgateways` (which is maintained manually by the administrator) exists, then the gateways in the file serve as the host's defaults. If `/etc/tsolgateways` does not exist, then the host uses the default routes from the file `/etc/defaultrouter`, which is also maintained manually by the administrator. If either file exists, then the host is said to use static routing.

If neither the `/etc/tsolgateways` nor the `/etc/defaultrouter` file exists, then the host uses dynamic routing and must start a special daemon, either `in.rdisc`(1M) (the network router discovery daemon) if it is available, or `in.routed`(1M) (the network routing daemon) if `in.rdisc` is not available. If the host also serves as a gateway (that is, a host that connects to two or more networks), then both `in.rdisc` and `in.routed` are started.

At boot time, the `tnrhdb` and `tnrhtp` files (which reside in the `/etc/security/tsol/boot` directory) are loaded into the kernel to enable hosts to communicate with the NIS+ master. By default, `/etc/security/tsol/boot/tnrhdb` contains the entry `0.0.0.0:tsol`, indicating that the network is a Trusted Solaris network.

SLD

See *single-level directory.*

Security Administrator

In an organization where sensitive information must be protected, the person or persons who define and enforce the site's *security policy* and who are cleared to access all information being processed at the site. In the Trusted Solaris software environment, an administrative role that is assigned to one or more individuals who have the proper clearance and whose task is to configure the security attributes of all users and machines so that the software enforces the site's security policy.

Security Attribute

An attribute used in enforcing the Trusted Solaris *security policy*. Various sets of security attributes, both from the base Solaris and the Trusted Solaris systems, are assigned to *process*es, users, files, directories, file systems, hosts on the trusted network, allocatable devices, and other entities. Security attributes for users from the base Solaris system include the user ID (UID), audit ID (AUID), group ID

(GID), supplementary group IDs (SGIDs). Security attributes for users from the Trusted Solaris environment include the *clearance, minimum label*, (*initial label*), and any *authorization*s. An important Trusted Solaris security attribute for files is the CMW label, the sensitivity label portion of which is used in access decisions. A *label range* security attribute is assigned to file systems, to allocatable devices and to printers. A UID, GID, a label range, and one or more *privilege*s may be associated with commands and *CDE actions* by security administrators in *execution profiles.* The mentioned security attributes and others are assigned to hosts in Trusted Network databases, which are used to control the security of communications in a Trusted Solaris distributed environment.

Security Policy

In the Trusted Solaris environment, the set of DAC and MAC rules that define how information may be accessed. At a customer site, the set of rules that define the sensitivity of the information being processed at that site and the measures that are used to protect the information from unauthorized access.

Sensitivity Label

A security *label* assigned to a file or directory or process, which is used to limit access based on the security level of the information contained therein.

Session Clearance

A *clearance* that is in effect only during a particular login session, this type of clearance is set by the user when starting a session. Each process started during a session has a *process clearance* equal to the session clearance. The session clearance may be set either to be the same as or lower than the *user clearance.*

Single-level Directory

A directory within an MLD containing files at only a single *sensitivity label.* When a user working at a particular sensitivity label changes into an MLD, the user’s working directory actually changes to a single-label directory within the MLD, whose sensitivity label is the same as the sensitivity label at which the user is working.

System Accreditation Range

The set of all valid (well-formed) labels created according to the rules defined by each site’s security administrator in the `label_encodings` file, plus the two administrative labels that are used in every Trusted Solaris environment, `ADMIN_LOW` and `ADMIN_HIGH`.

Strictly Dominate

When any type of label (*sensitivity label* or *clearance*) has a security level greater than the security level of another label to which it is being compared, the first label strictly *dominates* the second label. Strict dominance is dominance without equality, which occurs either when the classification of the first label is higher than that of the second label and the first label contains all the compartments in the second label or when the classifications of both labels are the same while the first label contains all the compartments in the second label plus one or more additional compartments.

Trusted Stripe

A region that cannot be spoofed along the bottom of the screen, which by default provides the following as visual feedback about the state of the window system:

a trusted path indicator and the window sensitivity label. When *sensitivity label*s are configured to not be viewable for a user, then the type of label that is viewable is displayed and the other is not. When *sensitivity label*s are not configured to be displayed for a user, the trusted stripe is reduced to an icon that displays only the trusted path indicator.

Tunneling

It is possible to route secure data through clusters containing non-Trusted Solaris gateways. This procedure is called tunneling. A cluster is a contiguous set of either Trusted Solaris hosts and gateways only, or non-Trusted Solaris hosts and gateways only. An edge gateway is a gateway (Trusted Solaris or non-Trusted Solaris) that connects a cluster to a cluster of the opposite type.

To transmit data by a route through a non-Trusted Solaris cluster and a Trusted Solaris cluster, two conditions must be met:

- All the gateways in the non-Trusted Solaris cluster must have the same security attributes.
- If there is more than one possible route and the routes enter the non-Trusted Solaris cluster through the same edge gateway and can exit from the cluster through different edge gateways, then the emetric for these routes must be equal.

User Accreditation Range

The set of all possible labels at which any normal user may work on the system, as defined by each site's security administrator. The rules for well-formed labels that define the *system accreditation range* are additionally restricted by the values specified in the ACCREDITATION RANGE section of the site's `label_encodings`(4) file: the upper bound, the lower bound, the combination constraints and other restrictions.

User Clearance

The *clearance* assigned by the *security administrator* that sets the upper bound of the set of labels at which one particular user may work at any time. The user may decide to accept or further restrict that clearance during any particular login session, when setting the *session clearance* after login.

TRUSTED SOLARIS DIFFERENCES

The responsibilities and privileges of the super-user have been divided among several administrative roles. When a man page that has not been modified for the Trusted Solaris system states that super-user is required to execute a certain command or option, remember that one or more privileges are required instead.

The ability of the UNIX super-user to bypass access restrictions, to execute restricted commands, and to use some command options not available to other users has been replaced with the *profile mechanism*, which allows the security administrator to assign to various users different sets of commands and to assign different privileges to the commands using *execution profiles*. When a command or one of its options needs a privilege in order to succeed, that privilege is a *required* privilege; if the required privilege is not given to the command in a user's execution profile by the security administrator, the command won't work. Required privileges are indicated on the man page with the words "must have," as shown in this sentence: "The `ifconfig`(1M) command must have the `sys_net_config` privilege to modify network interfaces."

In other cases, when the command is designed to work within security policy and it fails when certain DAC or MAC checks are not passed, an *override* privilege may be assigned at the security administrator's discretion. On man pages, the names of privileges that may be used to override access restrictions are given in the `ERRORS` section. The override privileges that may be given to bypass DAC or MAC restrictions on files or directories are given below:

The DAC override privileges are `file_dac_read` and `file_dac_write`. If a user does not have DAC access to a file, the security administrator may assign one or both of these privileges to the command, depending on whether read or write access or both are desired. The MAC override privileges are `file_mac_read` and `file_mac_write`. If a user does not have MAC access to a file, the security administrator may assign one or both of these privileges to the command, depending on whether read or write access or both are desired.

Besides being able to assign an override privilege, the security administrator has other options. For example, to avoid the use of privilege the security administrator may specify that the command will execute with another user's ID (usually the root ID 0) or group ID, one that allows access to the file or directory based on its permissions or its ACL.

SUMMARY OF TRUSTED SOLARIS CHANGES

Commands may not work as expected in the Trusted Solaris environment because Trusted Solaris administrators may limit the conditions under which commands may be accessed by each user or restrict commands from being accessed by certain users.

ILs are no longer supported. See Trusted Solaris Information Label Changes under DESCRIPTION.

The printed *Trusted Solaris 8 Reference Manual* contains only the Trusted Solaris original and modified (from the Solaris environment) man pages. The online set of man pages viewed by the `man` command accesses all man pages; AnswerBook2™ can access all man pages in the AnswerBook2 collections. The `SEE ALSO` man page heading has been subdivided to help users of the printed manual locate a referenced man page.

Besides the usual UNIX DAC checks performed when a process acting on behalf of a user attempts to access a file or directory, *mandatory access* checks also must be passed. For each possible type of access failure, a specific override *privilege* may be assigned to the command at the security administrator's discretion.

When a SUMMARY OF TRUSTED SOLARIS CHANGES is provided on a modified man page, it is intended as a convenience to summarize for you the major changes all in one place. Do not rely on the `SUMMARY OF TRUSTED SOLARIS CHANGES` alone, but also read the entire man page.

ATTRIBUTES

See `attributes`(5) in the *SunOS 5.8 Reference Manual* for a discussion of the attributes listed in this section.

SEE ALSO

Commands that are listed under the Trusted Solaris 8 Reference Manual heading in the `SEE ALSO` section are commands that have been changed or added in the Trusted Solaris environment. Commands that are listed under the SunOS 5.8 Reference Manual heading in the `SEE ALSO` section are Solaris commands that are unchanged in the Trusted Solaris environment. If you are using printed manuals, refer to the *SunOS 5.8 Reference Manual* for Solaris commands that are unchanged in the Trusted Solaris environment.

Trusted Solaris 8 Reference Manual

Trusted Solaris references are listed under this heading.

Trusted Solaris user's document set, *Trusted Solaris Administration Overview*, and the *Trusted Solaris Administrator's Procedures* manuals.

SunOS 5.8 Reference Manual

`SunOS` 5.8 and Solaris 8 references that are unchanged in the Trusted Solaris environment are listed under this heading.

`getopts`(1), `wait`(1), `exit`(2), `getopt`(3C), `wait`(3UCB), `attributes`(5)

Source Compatibility Guide

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of "normal" termination) one supplied by the program [see `wait`(3UCB) and `exit`(2)]. The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and non-zero to indicate troubles such as erroneous parameters, or bad or inaccessible data. It is called variously "exit code", "exit status", or "return code", and is described only where special conventions are involved.

WARNINGS

Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings and therefore become confused upon encountering a null character (the string terminator) within a line.

Name	Description
`adornfc`(1)	Display the pathname with the final component adorned
`at`(1)	execute commands at a later time
`atq`(1)	Display the jobs queued to run at specified times
`atrm`(1)	Remove jobs spooled by at or batch

`auths`(1)	print authorizations granted to a user
`batch`(1)	See `at`(1)
`cancel`(1)	Cancel print request
`chgrp`(1)	Change file group ownership
`chmod`(1)	Change the permissions mode of a file
`chown`(1)	Change file ownership
`crle`(1)	configure runtime linking environment
`crontab`(1)	User crontab file
`date`(1)	write the date and time
`disable`(1)	See `enable`(1)
`dtappsession`(1)	Start a new Application Manager session
`enable`(1)	Enable / disable LP printers
`find`(1)	Find files
`getfattrflag`(1)	Gets the file’s security attributes flag
`getfpriv`(1)	Gets the privileges assigned to files
`getlabel`(1)	Get the CMW label for files
`getmldadorn`(1)	Display the multilevel directory adornment of the file system
`getsldname`(1)	Display file-system single-level directory name
`ipcrm`(1)	Remove a message queue, semaphore set, or shared memory ID
`ipcs`(1)	Report inter-process communication facilities status
`kbd`(1)	Manipulate the state of keyboard or display the type of keyboard or change the default keyboard abort sequence effect
`ld`(1)	link-editor for object files
`login`(1)	Sign on to the system
`lp`(1)	Submit print request

`lpc`(1B)	Line printer control program
`lpq`(1B)	Display the content of a print queue
`lpr`(1B)	Submit print requests
`lprm`(1B)	Remove print requests from the print queue
`lpstat`(1)	Print information about the status of the print service
`mkdir`(1)	Make directories
`mldpwd`(1)	Display the pathname of the current working directory, including any MLD adornments and SLD names
`mldrealpath`(1)	Display the canonicalized absolute pathname, including any MLD adornments and SLD names
`nca`(1)	the Solaris Network Cache and Accelerator (NCA)
`ncakmod`(1)	start or stop the NCA kernel module
`nispasswd`(1)	change NIS+ password information
`passwd`(1)	Change login password and password attributes
`pattr`(1)	Get the viewable process attribute flags
`pclear`(1)	Get process clearance
`pcred`(1)	See `proc`(1)
`pfiles`(1)	See `proc`(1)
`pflags`(1)	See `proc`(1)
`plabel`(1)	Get the label of a process
`pldd`(1)	See `proc`(1)
`pmap`(1)	See `proc`(1)
`ppriv`(1)	Get the effective privileges of a process
`pprivtest`(1)	Test effective privilege set of the process
`proc`(1)	Proc tools
`profiles`(1)	print rights profiles for a user
`prun`(1)	See `proc`(1)
`psig`(1)	See `proc`(1)
`pstack`(1)	See `proc`(1)

pstop(1)	See proc(1)
ptime(1)	See proc(1)
ptree(1)	See proc(1)
pwait(1)	See proc(1)
pwdx(1)	See proc(1)
rm(1)	Remove directory entries
rmdir(1)	See rm(1)
roles(1)	print roles granted to a user
setfattrflag(1)	Sets the file’s security attribute flags
setfpriv(1)	Change the privilege sets associated with a file
setlabel(1)	Sets the CMW label for files
snca(1)	See nca(1)
tar(1)	Create tape archives and add or extract files
testfpriv(1)	Check or test the privilege sets associated with a file
tfind(1)	See find(1)
uname(1)	Print name of current system
vacation(1)	Reply to mail automatically

NAME

adornfc – Display the pathname with the final component adorned

SYNOPSIS

adornfc *pathname*

DESCRIPTION

`adornfc` adorns the final component of *pathname* unless it is already adorned. *pathname* is a pathname to a filesystem object.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

RETURN VALUES

`adornfc` exits with one of the following values:

`0` Success

`1` Usage error

`2` Failure, error message is the system error number from `adornfc`(3TSOL).

SEE ALSO

Trusted Solaris 8 Reference Manual

`adornfc`(3TSOL)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

at, batch – execute commands at a later time

SYNOPSIS

at [–c | –k | –s][–m] [–f *file*] [–p *project*] [–q *queuename*] –t *time*

at [–c | –k | –s][–P] [–m] [–f *file*] [–p *project*] [–q *queuename*] *timespec…*

at –l [–p *project*] [–q *queuename*] [*at_job_id. ..*]

at –r *at_job_id. ..*

batch [–p *project*]

DESCRIPTION

at

The at utility reads commands from standard input and groups them together as an *at-job* , to be executed at a later time.

The at-job will be executed in a separate invocation of the shell, running in a separate process group with no controlling terminal, except that the environment variables, current working directory, file creation mask (see umask(1)), and system resource limits (for sh and ksh only, see ulimit(1)) in effect when the at utility is executed will be retained and used when the *at-job* is executed.

When the at-job is submitted, the *at_job_id* and scheduled time are written to standard error. The *at_job_id* is an identifier that will be a string consisting solely of alphanumeric characters and the period character. The *at_job_id* is assigned by the system when the job is scheduled such that it uniquely identifies a particular job.

User notification and the processing of the job's standard output and standard error are described under the –m option.

Users are permitted to use at and batch (see below) if their name appears in the file /usr/lib/cron/at.allow . If that file does not exist, the file /usr/lib/cron/at.deny is checked to determine if the user should be denied access to at . If neither file exists, only a user with the solaris.jobs.user authorization is allowed to submit a job. If only at.deny exists and is empty, global usage is permitted. The at.allow and at.deny files consist of one user name per line.

batch

The batch utility reads commands to be executed at a later time. It is the equivalent of the command:

```
at –q b –m now
```

where queue b is a special at queue, specifically for batch jobs. Batch jobs will be submitted to the batch queue for immediate execution.

In the Trusted Solaris environment, the at and batch commands allow a user to create an *at-job* file that is installed in the appropriate SLD that matches the invoking process' sensitivity label. The at command also allows a user to list or remove the at-jobs owned by the current user at the invoking process' sensitivity label. A user can list or remove an at-job belonging to another user if the invoking user has the `solaris.jobs.admin` authorization.

OPTIONS

The following options are supported. Note that if a user's login shell is a profile shell, the login shell is used to run the at-job. If the shell is specified with `–c` , `–k` , `–s` , or `–P` , the specified shell is used. Otherwise, the `$SHELL` environment variable determines which shell to use. If `$SHELL` is null, `sh` is used by default.

`–c` C shell. `csh`(1) is used to execute the at-job.

`–k` Korn shell. `ksh`(1) is used to execute the at-job.

`–s` Bourne shell. `sh`(1) is used to execute the at-job.

`–P` Profile shell. Either `pfsh`(1M) is used to execute the at-job; or `pfksh` or `pfcsh` is used, depending on whether the `–s` , `–k` , or `–c` option is specified.

`–f` *file* Specifies the path of a file to be used as the source of the at-job, instead of standard input.

`–l` (The letter ell.) Reports all jobs scheduled for the current user (or if the current user has the appropriate authorizations, report jobs for other users) at the invoking process's sensitivity label, if no *at_job_id* operands are specified. If *at_job_id* s are specified, reports only information for these jobs. If the at-job is not owned by the current user, its job information will be displayed if the invoking user has the `solaris.jobs.admin` authorization.

`–m` Sends mail to the invoking user after the at-job has run, announcing its completion. Standard output and standard error produced by the at-job will be mailed to the user as well, unless redirected elsewhere. Mail will be sent even if the job produces no output.

If `–m` is not used, the job's standard output and standard error will be provided to the user by means of mail, unless they are redirected elsewhere; if there is no such output to provide, the user is not notified of the job's completion.

`–p` *project* Specifies under which project the `at` or `batch` job will be run. When used with the `–l` option, limits the search to that particular project. Values for *project* will be interpreted first as a project name, and then as a possible project ID , if entirely numeric. By default, the user's current project is used.

`–q` *queuename*
: Specifies in which queue to schedule a job for submission. When used with the `–l` option, limits the search to that particular queue. Values for *queuename* are limited to the lower case letters `a` through `z` . By default, at-jobs will be scheduled in queue `a` . In contrast, queue `b` is reserved for batch jobs. Since queue `c` is reserved for cron jobs, it can not be used with the `–q` option.

`–r` *at_job_id*
: Removes the jobs with the specified *at_job_id* operands that were previously scheduled by the `at` utility. If the specified *at_job_id* is not owned by the current user, it is removed if the invoking user has the `solaris.jobs.admin` authorization.

`–t` *time*
: Submits the job to be run at the time specified by the *time* option-argument, which must have the format as specified by the `touch`(1) utility.

OPERANDS

The following operands are supported:

at_job_id
The name reported by a previous invocation of the `at` utility at the time the job was scheduled.

timespec
Submit the job to be run at the date and time specified. All of the *timespec* operands are interpreted as if they were separated by space characters and concatenated. The date and time are interpreted as being in the timezone of the user (as determined by the `TZ` variable), unless a timezone name appears as part of *time* below.

In the "C" locale, the following describes the three parts of the time specification string. All of the values from the `LC_TIME` categories in the "C" locale are recognized in a case-insensitive manner.

time

The *time* can be specified as one, two or four digits. One- and two-digit numbers are taken to be hours, four-digit numbers to be hours and minutes. The time can alternatively be specified as two numbers separated by a colon, meaning *hour* `:` *minute* . An AM / PM indication (one of the values from the `am_pm` keywords in the `LC_TIME` locale category) can follow the time; otherwise, a 24-hour clock time is understood. A timezone name of GMT , UCT , or ZULU (case insensitive) can follow to specify that the time is in Coordinated Universal Time. Other timezones can be specified using the `TZ` environment variable. The *time* field can also be one of the following tokens in the "C" locale:

`midnight` Indicates the time 12:00 am (00:00).

`noon` Indicates the time 12:00 pm.

`now` Indicate the current day and time. Invoking `at now` will submit an at-job for potentially immediate execution (that is, subject only to unspecified scheduling delays).

date

An optional *date* can be specified as either a month name (one of the values from the `mon` or `abmon` keywords in the `LC_TIME` locale category) followed by a day number (and possibly year number preceded by a comma) or a day of the week (one of the values from the `day` or `abday` keywords in the `LC_TIME` locale category). Two special days are recognized in the "C" locale:

`today` Indicates the current day.

`tomorrow` Indicates the day following the current day.

If no *date* is given, `today` is assumed if the given time is greater than the current time, and `tomorrow` is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.

increment

The optional *increment* is a number preceded by a plus sign (`+`) and suffixed by one of the following: `minutes` , `hours` , `days` , `weeks` , `months` , or `years` . (The singular forms will be also accepted.) The keyword `next` is equivalent to an increment number of `+ 1` . For example, the following are equivalent commands:

```
at 2pm + 1 week
at 2pm next week
```

USAGE

The format of the `at` command line shown here is guaranteed only for the "C" locale. Other locales are not supported for `midnight` , `noon` , `now` , `mon` , `abmon` , `day` , `abday` , `today` , `tomorrow` , `minutes` , `hours` , `days` , `weeks` , `months` , `years` , and `next` .

Since the commands run in a separate shell invocation, running in a separate process group with no controlling terminal, open file descriptors, traps and priority inherited from the invoking environment are lost.

EXAMPLES

at

EXAMPLE 1 Typical sequence at a terminal

This sequence can be used at a terminal:

```
$ at –m 0730 tomorrow
sort < file >outfile
<EOT>
```

EXAMPLE 2 Redirecting output

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
$ at now + 1 hour <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
```

EXAMPLE 3 Self-rescheduling a job

To have a job reschedule itself, `at` can be invoked from within the at-job. For example, this "daily-processing" script named `my.daily` will run every day (although `crontab` is a more appropriate vehicle for such work):

```
# my.daily runs every day
at now tomorrow < my.daily
daily-processing
```

EXAMPLE 4 Various time and operand presentations

The spacing of the three portions of the "C" locale *timespec* is quite flexible as long as there are no ambiguities. Examples of various times and operand presentations include:

```
at 0815am Jan 24
at 8 :15amjan24
at now "+ 1day"
at 5 pm FRIday
at ’17
 utc+
 30minutes’
```

EXAMPLE 5 Using the pfcsh shell for an at-job

An example of using the pfcsh shell for an at-job includes:

```
at -c -P 0815am Jan 24 date
```

batch

EXAMPLE 6 Typical sequence at a terminal

This sequence can be used at a terminal:

```
$ batch
sort <file >outfile
<EOT>
```

EXAMPLE 7 Redirecting output

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
  $ batch <<!
  diff file1 file2 2>&1 >outfile | mailx mygroup
  !
```

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `at` and `batch` : `LC_CTYPE` , `LC_MESSAGES` , `NLSPATH` , and `LC_TIME` .

`SHELL` Determine a name of a command interpreter to use to invoke the at-job, when the user's login shell is not `pfsh` . If the variable is unset or `NULL` , `sh` will be used. If it is set to a value other than `sh` , the implementation will use that shell; a warning diagnostic will be printed telling which shell will be used.

`TZ` Determine the timezone. The job will be submitted for execution at the time specified by *timespec* or `−t` *time* relative to the timezone specified by the `TZ` variable. If *timespec* specifies a timezone, it will override `TZ` . If *timespec* does not specify a timezone and `TZ` is unset or `NULL` , an unspecified default timezone will be used.

`DATEMSK` If the environment variable `DATEMSK` is set, `at` will use its value as the full path name of a template file containing format strings. The strings consist of format specifiers and text characters that are used to provide a richer set of allowable date formats in different languages by appropriate settings of the environment variable `LANG` or `LC_TIME` . The list of allowable format specifiers is located in the `getdate`(3C) manual page. The formats described in the `OPERANDS` section for the *time* and *date* arguments, the special names `noon` , `midnight` , `now` , `next` , `today` , `tomorrow` , and the *increment* argument are not recognized when `DATEMSK` is set.

EXIT STATUS

The following exit values are returned:

`0` The `at` utility successfully submitted, removed or listed a job or jobs.

`>0` An error occurred, and the job will not be scheduled.

FILES

`/usr/lib/cron/at.allow`	Names of users, one per line, who are authorized access to the `at` and `batch` utilities
`/usr/lib/cron/at.deny`	Names of users, one per line, who are denied access to the `at` and `batch` utilities.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

at

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Not enabled

batch

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
CSI	Enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

To succeed, the `at` command requires the following forced privileges: `proc_audit_tcb` , `file_chown` , and `file_dac_read` .

An ancillary file is created in the `/var/spool/cron/atjobs` directory for each `at-job` file. By convention, the file is named `at_job_id.ad` ; and it is used by the `clock` daemon to set up the at-job to run.

The at-jobs are run with the profile shell if the user's login shell is the profile shell. Otherwise, the user's specified shell (by the –`c` , –`s` , –`k` , or –`P` options), or the `$SHELL` environment variable (default `sh` if `$SHELL` is `NULL`) is used to run the at-jobs.

SEE ALSO

Trusted Solaris 8 Reference Manual

`auths`(1) , `crontab`(1) , `cron`(1M) , `pfsh`(1M)

SunOS 5.8 Reference Manual

`csh`(1) , `date`(1) , `ksh`(1) , `sh`(1) , `touch`(1) , `ulimit`(1) , `umask`(1) , `getdate`(3C) , `auth_attr`(4) , `attributes`(5) , `environ`(5)

NOTES

Regardless of queue used, `cron`(1M) has a limit of 100 jobs in execution at any time.

There can be delays in `cron` at job execution. In some cases, these delays can compound to the point that `cron` job processing appears to be hung. All jobs will be executed eventually. When the delays are excessive, the only workaround is to kill and restart `cron` .

NAME

atq – Display the jobs queued to run at specified times

SYNOPSIS

atq [–c] [–n] [*username*…]

DESCRIPTION

atq displays the at-jobs queued up for the user at the invoking process's sensitivity label. at(1) is a utility that allows users to execute commands at a later date.

If no options are given, the jobs are displayed in chronological order of execution.

When a user invokes atq without specifying *username*, the user's at the invoking process's sensitivity label are displayed. If the invoking user's name is neither in the /etc/cron.d/at.admin file nor a role user *and* the user has the modify at users authorization, other users' at-jobs are also displayed.

When a username other than the invoking user's is specified, the named user's at-jobs are displayed under either of two conditions. The first condition is when the specified username is in the /etc/cron.d/at.admin file (which contains a list of administratives users for at) or is a role user; *and* the invoking user has the modify at admin authorization. The second condition is when the specified username is neither in the /etc/cron.d/at.admin file, nor a role user; *and* the invoking user has the modify at users authorization.

OPTIONS

The following options are supported:

–c Display the queued jobs in the order they were created (that is, the time that the at command was given).

–n Displays only the total number of jobs currently in the queue.

SUMMARY OF TRUSTED SOLARIS CHANGES

To succeed, the atq command must have the file_dac_read privilege in its set of forced privileges. The current user's at-jobs are displayed at the SL of the invoking process. The modify at users authorization is required to view others' at-jobs.

FILES

/var/spool/cron/atjobs	Spool area for at-jobs.
/etc/cron.d/at.admin	Names of administrative users for at; one per line. Do not put roles in this file.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

Trusted Solaris 8 Reference Manual	`at`(1), `atrm`(1), `cron`(1M)
SunOS 5.8 Reference Manual	`auths`(1), `auth_attr`(4), `attributes`(5)

NAME

atrm – Remove jobs spooled by at or batch

SYNOPSIS

atrm [–afi] [[*job #*] [*user*…]]

DESCRIPTION

atrm removes delayed-execution jobs specified by job number that were created with the at(1) command, but have not yet executed—if the jobs are owned by the invoking account at the invoking processes' sensitivity label. The list of these jobs and associated job numbers can be displayed by using atq(1).

When a username other than the invoking user's is specified, atrm removes the named user's at-jobs under either of two conditions. The first condition is when the specified username is in the /etc/cron.d/at.admin file (which contains a list of administratives users for at) or is a role user; *and* the invoking user has the modify at admin authorization. The second condition is when the specified username is neither in the /etc/cron.d/at.admin file, nor a role user; *and* the invoking user has the modify at users authorization.

atrm needs the proc_audit_tcb privilege to succeed.

OPTIONS

The following options are supported:

–a All. Remove all unexecuted jobs at the invoking processes' sensitivity label that were created by the invoking user. The at-jobs owned by another user are removed only when one of the two conditions described in the DESCRIPTION section is met.

–f Force. All information regarding the removal of the specified jobs is suppressed.

–i Interactive. atrm asks if a job should be removed. If you respond with a **y**, the job will be removed.

SUMMARY OF TRUSTED SOLARIS CHANGES

atrm needs the proc_audit_tcb privilege to succeed. atrm removes jobs only at the sensitivity label of the current process. atrm removes jobs belonging to another user only if both the account invoking atrm has needed authorizations and the specified *user* name meets additional requirements described in the conditions in the DESCRIPTION section.

FILES

/var/spool/cron/atjobs	Spool area for at-jobs
/etc/cron.d/at.admin	List of default system account names, one per line. Seldom needs to be updated. Never add the names of role accounts to this file.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SEE ALSO

Trusted Solaris 8 Reference Manual

at(1), atq(1), cron(1M)

Trusted Solaris Administrator's Procedures

SunOS 5.8 Reference Manual

auths(1), auth_attr(4), attributes(5)

System Administration Guide, Volume 2

NAME

auths – print authorizations granted to a user

SYNOPSIS

auths [*user* …]

DESCRIPTION

The auths command prints on standard output the authorizations that you or the optionally-specified user or role have been granted. Authorizations are rights that are checked by certain privileged programs to determine whether a user may execute restricted functionality.

Each user may have zero or more authorizations. Authorizations are represented by fully-qualified names, which identify the organization that created the authorization and the functionality that it controls. Following the Java convention, the hierarchical components of an authorization are separated by dots (.), starting with the reverse order Internet domain name of the creating organization, and ending with the specific function within a class of authorizations, for example, “com.acme.files.write”. The exceptions to this convention are authorizations from Sun Microsystems, Inc. These use the prefix “solaris.”, as in the example, “solaris.files.write”.

A trailing asterisk (*) to the right of a dot indicates all matching authorizations and can be used when assigning all authorizations within a class.

A user’s authorizations are looked up in user_attr(4) and in the /etc/security/policy.conf file (see policy.conf(4)). Authorizations may be specified directly in user_attr(4) or indirectly through prof_attr(4). Authorizations may also be assigned to every user in the system directly as default authorizations or indirectly through default profiles in the /etc/security/policy.conf file.

EXAMPLES

EXAMPLE 1 Sample output

The auths output has the following form:

```
example% auths tester01 tester02
tester01 : solaris.system.date, solaris.jobs.admin
tester02 : solaris.system.*
example%
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

FILES

/etc/user_attr Local source of extended attributes associated with users and roles.

`/etc/security/auth_attr`	Local source for authorization names and descriptions.
`/etc/security/policy.conf`	Provides the security policy configuration for user-level attributes.
`/etc/security/prof_attr`	Local source for rights profile names, descriptions, and other attributes of profiles.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

Classes of authorizations can be assigned to accounts using a wildcard (asterisk) to the right of a dot in an authorization.

SEE ALSO

Trusted Solaris 8 Reference Manual

`profiles`(1), `roles`(1), `policy.conf`(4), `prof_attr`(4)

SunOS 5.8 Reference Manual

`getauthattr`(3SECDB), `auth_attr`(4), `user_attr`(4), `attributes`(5)

NAME

at, batch – execute commands at a later time

SYNOPSIS

at [–c | –k | –s][–m] [–f *file*] [–p *project*] [–q *queuename*] –t *time*

at [–c | –k | –s][–P] [–m] [–f *file*] [–p *project*] [–q *queuename*] *timespec…*

at –l [–p *project*] [–q *queuename*] [*at_job_id. ..*]

at –r *at_job_id. ..*

batch [–p *project*]

DESCRIPTION

at

The at utility reads commands from standard input and groups them together as an *at-job* , to be executed at a later time.

The at-job will be executed in a separate invocation of the shell, running in a separate process group with no controlling terminal, except that the environment variables, current working directory, file creation mask (see umask(1)), and system resource limits (for sh and ksh only, see ulimit(1)) in effect when the at utility is executed will be retained and used when the *at-job* is executed.

When the at-job is submitted, the *at_job_id* and scheduled time are written to standard error. The *at_job_id* is an identifier that will be a string consisting solely of alphanumeric characters and the period character. The *at_job_id* is assigned by the system when the job is scheduled such that it uniquely identifies a particular job.

User notification and the processing of the job's standard output and standard error are described under the –m option.

Users are permitted to use at and batch (see below) if their name appears in the file /usr/lib/cron/at.allow . If that file does not exist, the file /usr/lib/cron/at.deny is checked to determine if the user should be denied access to at . If neither file exists, only a user with the solaris.jobs.user authorization is allowed to submit a job. If only at.deny exists and is empty, global usage is permitted. The at.allow and at.deny files consist of one user name per line.

batch

The batch utility reads commands to be executed at a later time. It is the equivalent of the command:

```
at –q b –m now
```

where queue b is a special at queue, specifically for batch jobs. Batch jobs will be submitted to the batch queue for immediate execution.

In the Trusted Solaris environment, the at and batch commands allow a user to create an *at-job* file that is installed in the appropriate SLD that matches the invoking process' sensitivity label. The at command also allows a user to list or remove the at-jobs owned by the current user at the invoking process' sensitivity label. A user can list or remove an at-job belonging to another user if the invoking user has the `solaris.jobs.admin` authorization.

OPTIONS

The following options are supported. Note that if a user's login shell is a profile shell, the login shell is used to run the at-job. If the shell is specified with `−c` , `−k` , `−s` , or `−P` , the specified shell is used. Otherwise, the `$SHELL` environment variable determines which shell to use. If `$SHELL` is null, `sh` is used by default.

`−c` C shell. `csh`(1) is used to execute the at-job.

`−k` Korn shell. `ksh`(1) is used to execute the at-job.

`−s` Bourne shell. `sh`(1) is used to execute the at-job.

`−P` Profile shell. Either `pfsh`(1M) is used to execute the at-job; or `pfksh` or `pfcsh` is used, depending on whether the `−s` , `−k` , or `−c` option is specified.

`−f` *file* Specifies the path of a file to be used as the source of the at-job, instead of standard input.

`−l` (The letter ell.) Reports all jobs scheduled for the current user (or if the current user has the appropriate authorizations, report jobs for other users) at the invoking process's sensitivity label, if no *at_job_id* operands are specified. If *at_job_id* s are specified, reports only information for these jobs. If the at-job is not owned by the current user, its job information will be displayed if the invoking user has the `solaris.jobs.admin` authorization.

`−m` Sends mail to the invoking user after the at-job has run, announcing its completion. Standard output and standard error produced by the at-job will be mailed to the user as well, unless redirected elsewhere. Mail will be sent even if the job produces no output.

If `−m` is not used, the job's standard output and standard error will be provided to the user by means of mail, unless they are redirected elsewhere; if there is no such output to provide, the user is not notified of the job's completion.

`−p` *project* Specifies under which project the `at` or `batch` job will be run. When used with the `−l` option, limits the search to that particular project. Values for *project* will be interpreted first as a project name, and then as a possible project ID , if entirely numeric. By default, the user's current project is used.

`−q` *queuename*
: Specifies in which queue to schedule a job for submission. When used with the `−l` option, limits the search to that particular queue. Values for *queuename* are limited to the lower case letters `a` through `z` . By default, at-jobs will be scheduled in queue `a` . In contrast, queue `b` is reserved for batch jobs. Since queue `c` is reserved for cron jobs, it can not be used with the `−q` option.

`−r` *at_job_id*
: Removes the jobs with the specified *at_job_id* operands that were previously scheduled by the `at` utility. If the specified *at_job_id* is not owned by the current user, it is removed if the invoking user has the `solaris.jobs.admin` authorization.

`−t` *time*
: Submits the job to be run at the time specified by the *time* option-argument, which must have the format as specified by the `touch`(1) utility.

OPERANDS

The following operands are supported:

at_job_id
: The name reported by a previous invocation of the `at` utility at the time the job was scheduled.

timespec
: Submit the job to be run at the date and time specified. All of the *timespec* operands are interpreted as if they were separated by space characters and concatenated. The date and time are interpreted as being in the timezone of the user (as determined by the `TZ` variable), unless a timezone name appears as part of *time* below.

In the "C" locale, the following describes the three parts of the time specification string. All of the values from the `LC_TIME` categories in the "C" locale are recognized in a case-insensitive manner.

time
: The *time* can be specified as one, two or four digits. One- and two-digit numbers are taken to be hours, four-digit numbers to be hours and minutes. The time can alternatively be specified as two numbers separated by a colon, meaning *hour* `:` *minute* . An AM / PM indication (one of the values from the `am_pm` keywords in the `LC_TIME` locale category) can follow the time; otherwise, a 24-hour clock time is understood. A timezone name of GMT , UCT , or ZULU (case insensitive) can follow to specify that the time is in Coordinated Universal Time. Other timezones can be specified using the `TZ` environment variable. The *time* field can also be one of the following tokens in the "C" locale:

`midnight`	Indicates the time 12:00 am (00:00).
`noon`	Indicates the time 12:00 pm.
`now`	Indicate the current day and time. Invoking `at now` will submit an at-job for potentially immediate execution (that is, subject only to unspecified scheduling delays).

date

An optional *date* can be specified as either a month name (one of the values from the `mon` or `abmon` keywords in the `LC_TIME` locale category) followed by a day number (and possibly year number preceded by a comma) or a day of the week (one of the values from the `day` or `abday` keywords in the `LC_TIME` locale category). Two special days are recognized in the "C" locale:

`today`	Indicates the current day.
`tomorrow`	Indicates the day following the current day.

If no *date* is given, `today` is assumed if the given time is greater than the current time, and `tomorrow` is assumed if it is less. If the given month is less than the current month (and no year is given), next year is assumed.

increment

The optional *increment* is a number preceded by a plus sign (`+`) and suffixed by one of the following: `minutes` , `hours` , `days` , `weeks` , `months` , or `years` . (The singular forms will be also accepted.) The keyword `next` is equivalent to an increment number of `+ 1` . For example, the following are equivalent commands:

```
at 2pm + 1 week
at 2pm next week
```

USAGE

The format of the `at` command line shown here is guaranteed only for the "C" locale. Other locales are not supported for `midnight` , `noon` , `now` , `mon` , `abmon` , `day` , `abday` , `today` , `tomorrow` , `minutes` , `hours` , `days` , `weeks` , `months` , `years` , and `next` .

Since the commands run in a separate shell invocation, running in a separate process group with no controlling terminal, open file descriptors, traps and priority inherited from the invoking environment are lost.

EXAMPLES

at

EXAMPLE 1 Typical sequence at a terminal

This sequence can be used at a terminal:

```
$ at –m 0730 tomorrow
sort < file >outfile
<EOT>
```

EXAMPLE 2 Redirecting output

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
$ at now + 1 hour <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
```

EXAMPLE 3 Self-rescheduling a job

To have a job reschedule itself, `at` can be invoked from within the at-job. For example, this "daily-processing" script named `my.daily` will run every day (although `crontab` is a more appropriate vehicle for such work):

```
# my.daily runs every day
at now tomorrow < my.daily
daily-processing
```

EXAMPLE 4 Various time and operand presentations

The spacing of the three portions of the "C" locale *timespec* is quite flexible as long as there are no ambiguities. Examples of various times and operand presentations include:

```
at 0815am Jan 24
at 8 :15amjan24
at now "+ 1day"
at 5 pm FRIday
at ’17
 utc+
 30minutes’
```

EXAMPLE 5 Using the pfcsh shell for an at-job

An example of using the pfcsh shell for an at-job includes:

```
at -c -P 0815am Jan 24 date
```

batch

EXAMPLE 6 Typical sequence at a terminal

This sequence can be used at a terminal:

```
$ batch
sort <file >outfile
<EOT>
```

EXAMPLE 7 Redirecting output

This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
  $ batch <<!
  diff file1 file2 2>&1 >outfile | mailx mygroup
  !
```

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `at` and `batch` : `LC_CTYPE` , `LC_MESSAGES` , `NLSPATH` , and `LC_TIME` .

`SHELL` Determine a name of a command interpreter to use to invoke the at-job, when the user's login shell is not `pfsh` . If the variable is unset or `NULL` , `sh` will be used. If it is set to a value other than `sh` , the implementation will use that shell; a warning diagnostic will be printed telling which shell will be used.

`TZ` Determine the timezone. The job will be submitted for execution at the time specified by *timespec* or `−t` *time* relative to the timezone specified by the `TZ` variable. If *timespec* specifies a timezone, it will override `TZ` . If *timespec* does not specify a timezone and `TZ` is unset or `NULL` , an unspecified default timezone will be used.

`DATEMSK` If the environment variable `DATEMSK` is set, `at` will use its value as the full path name of a template file containing format strings. The strings consist of format specifiers and text characters that are used to provide a richer set of allowable date formats in different languages by appropriate settings of the environment variable `LANG` or `LC_TIME` . The list of allowable format specifiers is located in the `getdate`(3C) manual page. The formats described in the `OPERANDS` section for the *time* and *date* arguments, the special names `noon` , `midnight` , `now` , `next` , `today` , `tomorrow` , and the *increment* argument are not recognized when `DATEMSK` is set.

EXIT STATUS

The following exit values are returned:

`0` The `at` utility successfully submitted, removed or listed a job or jobs.

`>0` An error occurred, and the job will not be scheduled.

FILES

`/usr/lib/cron/at.allow` Names of users, one per line, who are authorized access to the `at` and `batch` utilities

`/usr/lib/cron/at.deny` Names of users, one per line, who are denied access to the `at` and `batch` utilities.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

at

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Not enabled

batch

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWesu
CSI	Enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

To succeed, the `at` command requires the following forced privileges: `proc_audit_tcb` , `file_chown` , and `file_dac_read` .

An ancillary file is created in the `/var/spool/cron/atjobs` directory for each `at-job` file. By convention, the file is named `at_job_id.ad` ; and it is used by the `clock` daemon to set up the at-job to run.

The at-jobs are run with the profile shell if the user's login shell is the profile shell. Otherwise, the user's specified shell (by the –`c` , –`s` , –`k` , or –`P` options), or the `$SHELL` environment variable (default `sh` if `$SHELL` is `NULL`) is used to run the at-jobs.

SEE ALSO

Trusted Solaris 8 Reference Manual

`auths`(1) , `crontab`(1) , `cron`(1M) , `pfsh`(1M)

SunOS 5.8 Reference Manual

`csh`(1) , `date`(1) , `ksh`(1) , `sh`(1) , `touch`(1) , `ulimit`(1) , `umask`(1) , `getdate`(3C) , `auth_attr`(4) , `attributes`(5) , `environ`(5)

NOTES

Regardless of queue used, `cron`(1M) has a limit of 100 jobs in execution at any time.

There can be delays in `cron` at job execution. In some cases, these delays can compound to the point that `cron` job processing appears to be hung. All jobs will be executed eventually. When the delays are excessive, the only workaround is to kill and restart `cron` .

NAME

cancel – Cancel print request

SYNOPSIS

cancel [*request-ID*…] [*destination*…]

cancel –`u` *user*… [*destination*…]

DESCRIPTION

The `cancel` utility cancels print requests. There are two forms of the `cancel` command.

The first form of `cancel` has two optional arguments: print requests (*request-ID*) and destinations (*destination*). Specifying *request-ID* with *destination* cancels *request-ID* on *destination*. Specifying only the destination cancels the current print request on *destination*. If *destination* is not specified, `cancel` cancels the requested print request on all destinations.

The second form of `cancel` cancels a user's print requests on specific destinations.

Users can only cancel print requests associated with their username. By default, users can only cancel print requests on the host from which the print request was submitted. If an administrator has set `user-equivalence=true` in `/etc/printers.conf` on the print server, users can cancel print requests associated with their username on any host. Users with `cancel any print job` authorization can cancel print requests on the host from which the print request was submitted. Users with `cancel any print job` authorization can also cancel print requests from the print server.

The print client commands locate destination information in a very specific order. See `printers.conf`(4) and `printers`(4) for details.

OPTIONS

The following options are supported:

–`u` *user* The name of the user for which print requests are to be canceled. Specify *user* as a username.

OPERANDS

The following operands are supported:

destination The destination on which the print requests are to be canceled. *destination* is the name of a printer or class of printers (see `lpadmin`(1M)). If *destination* is not specified, `cancel` cancels the requested print request on all destinations. Specify *destination* using atomic, POSIX-style (*server*`:`*destination*), or Federated Naming Service (FNS) (`. . ./service/printer/. . .`) names. See `NOTES` for information regarding using POSIX-style destination names with `cancel`. See `printers.conf`(4) for information regarding the naming conventions for atomic and FNS names, and `standards`(5) for information regarding POSIX.

request-ID	The print request to be canceled. Specify *request-ID* using LP-style request IDs (*destination-number*).
user	The name of the user for which the print requests are to be canceled. Specify *user* as a username.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
non-zero	An error occurred.

FILES

`/var/spool/print/*`	LP print queue.
`$HOME/.printers`	User-configurable printer database.
`/etc/printers.conf`	System printer configuration database.
`printers.conf.byname`	NIS version of `/etc/printers.conf`.
`fns.ctx_dir.`*domain*	NIS+ version of `/etc/printers.conf`.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SUMMARY OF TRUSTED SOLARIS CHANGES

Cancelling other users' requests requires the `cancel any print job` authorization. Cancel requests at other sensitivity labels require the `bypass system mac check` authorization.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lp`(1), `lpq`(1B), `lpr`(1B), `lprm`(1B), `lpstat`(1), `lpadmin`(1M)

SunOS 5.8 Reference Manual

`printers`(4), `printers.conf`(4), `attributes`(5), `standards`(5)

NOTES

POSIX-style destination names (*server*:*destination*) are treated as print requests if *destination* has the same format as an LP-style *request-ID*. See `standards`(5).

NAME

chgrp – Change file group ownership

SYNOPSIS

chgrp [–`fhRM`] *group file…*

DESCRIPTION

The `chgrp` utility sets the group ID of the file named by each *file* operand to the group ID specified by the *group* operand.

For each `file` operand, it performs actions equivalent to the `chown`(2) function, called with the following arguments:

- The *file* operand is used as the *path* argument.
- The user ID of the file is used as the *owner* argument.
- The specified group ID is used as the *group* argument.

Unless `chgrp` is invoked by a process with appropriate privileges, the set-user-ID and set-group-ID bits of a regular file will be cleared upon successful completion; the set-user-ID and set-group-ID bits of other file types may be cleared.

The operating system has a configuration option `_POSIX_CHOWN_RESTRICTED`, to restrict ownership changes. When this option is in effect, the owner of the file may change the group of the file only to a group to which the owner belongs. To arbitrarily change owner IDs, this command needs the `file_chown` privilege, whether or not this option is in effect.

OPTIONS

`–f` Force. Do not report errors.

`–h` If the file is a symbolic link, change the group of the symbolic link. Without this option, the group of the file referenced by the symbolic link is changed.

`–R` Recursive. `chgrp` descends through the directory, and any subdirectories, setting the specified group ID as it proceeds. When a symbolic link is encountered, the group of the target file is changed (unless the `–h` option is specified), but no recursion takes place.

`–M` `chgrp` processes all accessible SLDs in multilevel directories as it descends through the directory tree.

OPERANDS

The following operands are supported:

group A group name from the group database or a numeric group ID. Either specifies a group ID to be given to each file named by one of the *file* operands. If a numeric *group* operand exists in the group database as a group name, the group ID number associated with that group name is used as the group ID.

file A path name of a file whose group ID is to be modified.

USAGE

See `largefile`(5) for the description of the behavior of `chgrp` when encountering files greater than or equal to 2 GB (2^{31} bytes).

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `chgrp`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

EXIT STATUS

The following exit values are returned:

`0` The utility executed successfully and all requested changes were made.

`>0` An error occurred.

SUMMARY OF TRUSTED SOLARIS CHANGES

The `–M` option processes all accessible single-level directories in multilevel directories. To arbitrarily change owner IDs, `chgrp` requires the `file_chown` privilege.

FILES

`/etc/group` Local group file

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Enabled (see NOTES)

SEE ALSO

Trusted Solaris 8 Reference Manual

`chmod`(1), `chown`(1), `chown`(2), `fpathconf`(2)

SunOS 5.8 Reference Manual

`id`(1M), `group`(4), `passwd`(4), `system`(4), `attributes`(5), `environ`(5), `largefile`(5)

NOTES

`chgrp` is CSI-enabled except for the *group* name.

NAME

chmod – Change the permissions mode of a file

SYNOPSIS

chmod [–`fRM`] *absolute-mode file…*

chmod [–`fRM`] *symbolic-mode-list file…*

DESCRIPTION

The `chmod` command changes or assigns the mode of a file. The mode of a file specifies its permissions and other attributes. The mode may be absolute or symbolic.

Absolute mode

An absolute *mode* is specified using octal numbers:

`chmod` *nnnn file…*

where:

n a number from `0` to `7`. An absolute mode is constructed from the OR of any of the following modes:

`4000` Set user ID on execution.

`20#0` Set group ID on execution if # is `7`, `5`, `3`, or `1`.

Enable mandatory locking if # is `6`, `4`, `2`, or `0`.

For directories, files are created with BSD semantics for propagation of the group ID. With this option, files and subdirectories created in the directory inherit the group ID of the directory, rather than of the current process. It may be cleared only by using symbolic mode.

`1000` Turn on sticky bit. See `chmod`(2).

`0400` Allow read by owner.

`0200` Allow write by owner.

`0100` Allow execute (search in directory) by owner.

`0700` Allow read, write, and execute (search) by owner.

`0040` Allow read by group.

`0020` Allow write by group.

`0010` Allow execute (search in directory) by group.

`0070` Allow read, write, and execute (search) by group.

`0004` Allow read by others.

`0002` Allow write by others.

`0001`	Allow execute (search in directory) by others.
`0007`	Allow read, write, and execute (search) by others.

Note that the `setgid` bit cannot be set (or cleared) in absolute mode; it must be set (or cleared) in symbolic mode using `g+s` (or `g-s`).

Symbolic mode

A symbolic *mode* specification has the following format:

`chmod` *symbolic-mode-list* `file`...

where: *symbolic-mode-list* is a comma-separated list (with no intervening whitespace) of symbolic mode expressions of the form:

[`who`] *operator* [*permissions*]

Operations are performed in the order given. Multiple *permissions* letters following a single operator cause the corresponding operations to be performed simultaneously.

`who` zero or more of the characters `u`, `g`, `o`, and `a` specifying whose permissions are to be changed or assigned:

`u`	user's permissions
`g`	group's permissions
`o`	others' permissions
`a`	all permissions (user, group, and other)

If `who` is omitted, it defaults to `a`, but the setting of the file mode creation mask (see `umask` in `sh`(1) or `csh`(1) for more information) is taken into account. When `who` is omitted, `chmod` will not override the restrictions of your user mask.

operator either `+`, –, or `=`, signifying how permissions are to be changed:

`+` Add permissions.

If *permissions* is omitted, nothing is added.

If `who` is omitted, add the file mode bits represented by *permissions*, *except* for the those with corresponding bits in the file mode creation mask.

If `who` is present, add the file mode bits represented by the *permissions*.

– Take away permissions.

If *permissions* is omitted, do nothing.

If `who` is omitted, clear the file mode bits represented by *permissions*, *except* for those with corresponding bits in the file mode creation mask.

If `who` is present, clear the file mode bits represented by *permissions*.

= Assign permissions absolutely.

If `who` is omitted, clear all file mode bits; if `who` is present, clear the file mode bits represented by `who`.

If *permissions* is omitted, do nothing else.

If `who` is omitted, add the file mode bits represented by *permissions*, *except* for the those with corresponding bits in the file mode creation mask.

If `who` is present, add the file mode bits represented by *permissions*.

Unlike other symbolic operations, = has an absolute effect in that it resets all other bits represented by `who`. Omitting *permissions* is useful only with = to take away all permissions.

permission any compatible combination of the following letters:

`r`	read permission
`w`	write permission
`x`	execute permission
`l`	mandatory locking
`s`	user or group set-ID
`t`	sticky bit
`u,g,o`	indicate that *permission* is to be taken from the current user, group or other mode respectively.

Permissions to a file may vary depending on your user identification number (UID) or group identification number

(GID). Permissions are described in three sequences each having three characters:

`User`	`rwx`
`Group`	`rwx`
`Other`	`rwx`

This example (user, group, and others all have permission to read, write, and execute a given file) demonstrates two categories for granting permissions: the access class and the permissions themselves.

The letter `s` is only meaningful with `u` or `g`, and `t` only works with `u`.

Mandatory file and record locking (`l`) refers to a file's ability to have its reading or writing permissions locked while a program is accessing that file.

In a directory which has the set-group-ID bit set (reflected as either `-----s---` or `-----l---` in the output of `'ls -ld'`), files and subdirectories are created with the group-ID of the parent directory rather than that of current process.

It is not possible to permit group execution and enable a file to be locked on execution at the same time. In addition, it is not possible to turn on the set-group-ID bit and enable a file to be locked on execution at the same time. The following examples, therefore, are invalid and elicit error messages:

```
chmod g+x,+l file
chmod g+s,+l file
```

Only the owner of a file or directory (or a user running the command with the `file_setdac` privilege) may change that file's or directory's mode. Only a user invoking the command with the `sys_config` privilege may set the sticky bit on a non-directory file. If the command is invoked without the `sys_config` privilege, `chmod` will mask the sticky-bit but will not return an error. In order to turn on a file's set-group-ID bit, your own group ID must correspond to the file's and group execution must be set.

OPTIONS

The following options are supported:

`–f` Force. `chmod` will not complain if it fails to change the mode of a file.

`–R` Recursively descend through directory arguments, setting the mode for each file as described above. When symbolic links are encountered, the mode of the target file is changed, but no recursion takes place.

`–M` `chmod` processes all single-level directories as it descends multilevel directories.

OPERANDS

The following operands are supported:

mode Represents the change to be made to the file mode bits of each file named by one of the *file* operands; see the DESCRIPTION section for more information.

file A path name of a file whose file mode bits are to be modified.

USAGE

See `largefile`(5) for the description of the behavior of `chmod` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

EXAMPLE 1 Deny execute permission to everyone:

```
example% chmod a–x file
```

EXAMPLE 2 Allow only read permission to everyone:

```
example% chmod 444 file
```

EXAMPLE 3 Make a file readable and writable by the group and others:

```
example% chmod go+rw file
example% chmod 066 file
```

EXAMPLE 4 Cause a file to be locked during access:

```
example% chmod +l file
```

EXAMPLE 5 Allow everyone to read, write, and execute the file and turn on the set group-ID.

```
example% chmod a=rwx,g+s file
example% chmod 2777 file
```

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `chmod`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

The –`M` option processes all accessible single-level directories in multilevel directories. Running the command by a user other than the owner of a file or directory requires the `file_setdac` privilege. Setting the sticky bit on a non-directory file requires the `sys_config` privilege.

SEE ALSO

Trusted Solaris 8 Reference Manual

`chmod`(2)

SunOS 5.8 Reference Manual

`ls`(1), `attributes`(5), `environ`(5), `largefile`(5), `getfacl`(1), `setfacl`(1)

NOTES

Absolute changes don't work for the set-group-ID bit of a directory. You must use `g+s` or `g-s`.

`chmod` permits you to produce useless modes so long as they are not illegal (for instance, making a text file executable). `chmod` does not check the file type to see if mandatory locking is meaningful.

If the file system is mounted with the *nosuid* option, *setuid* execution is not allowed.

If you use `chmod` to change the file group owner permissions on a file with ACL entries, both the file group owner permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions may change the effective permissions for additional users and groups who have ACL entries on the file. Use the `getfacl`(1) command to make sure the appropriate permissions are set for all ACL entries.

NAME

chown – Change file ownership

SYNOPSIS

chown [–`fhRM`] *owner* [: *group*] *file…*

DESCRIPTION

The `chown` utility will set the user ID of the file named by each *file* to the user ID specified by *owner*, and, optionally, will set the group ID to that specified by *group*.

If `chown` is invoked without the `file_setid` privilege to change the ownership of a file, the set-user-ID bit is cleared.

Only the owner of a file (or a user invoking the command with the `file_chown` privilege) may change the owner of that file.

The operating system has a configuration option, `_POSIX_CHOWN_RESTRICTED`, to restrict ownership changes. When this option is in effect, the owner of the file is prevented from changing the owner ID of the file. The command requires the `file_chown` privilege to arbitrarily change owner IDs, whether or not this option is in effect.

OPTIONS

The following options are supported:

–`f` Do not report errors.

–`h` If the file is a symbolic link, change the owner of the symbolic link. Without this option, the owner of the file referenced by the symbolic link is changed.

–`R` Recursive. `chown` descends through the directory, and any subdirectories, setting the ownership ID as it proceeds. When a symbolic link is encountered, the owner of the target file is changed (unless the –`h` option is specified), but no recursion takes place.

–`M` `chown` processes all accessible single-level directories as it descends multilevel directories.

OPERANDS

The following operands are supported:

owner[`:` *group*] A user ID and optional group ID to be assigned to *file*. The *owner* portion of this operand must be a user name from the user database or a numeric user ID. Either specifies a user ID to be given to each file named by *file*. If a numeric *owner* exists in the user database as a user name, the user ID number associated with that user name will be used as the user ID. Similarly, if the *group* portion of this operand is present, it must be a group name from the group database or a numeric group ID. Either specifies a group ID to be given to each file. If a numeric group operand exists in the group database as a group name, the group ID number associated with that group name will be used as the group ID.

file A pathname of a file whose user ID is to be modified.

USAGE

See largefile(5) for the description of the behavior of chown when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

EXAMPLE 1 Changing ownership of all files in the hierarchy

To change ownership of all files in the hierarchy, including symbolic links, but not the targets of the links:

```
  example% chown −R −h owner[:group] file…
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of chown: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0 The utility executed successfully and all requested changes were made.

>0 An error occurred.

SUMMARY OF TRUSTED SOLARIS CHANGES

The −M option processes all accessible single-level directories in multilevel directories. If chown is invoked without the file_setid privilege to change the ownership of a file, chown clears the file's set-user-ID bit. To arbitrarily change owner IDs, chown requires the file_chown privilege.

FILES

/etc/passwd System password file

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Enabled (see NOTES)

SEE ALSO

Trusted Solaris 8 Reference Manual

chgrp(1), chmod(1), chown(2), fpathconf(2)

SunOS 5.8 Reference Manual

passwd(4), system(4), attributes(5), environ(5), largefile(5)

NOTES

chown is CSI-enabled except for the *owner* and *group* names.

NAME

crle – configure runtime linking environment

SYNOPSIS

crle [`–64`] [`–a` *name*] [`–c` *conf*] [`–f` *flags*] [`–i` *name*] [`–I` *name*] [`–g` *name*] [`–G` *name*]
[`–l` *dir*] [`–o` *dir*] [`–s` *dir*] [`–t`[ELF | AOUT]] [`–u`] [`–v`]

DESCRIPTION

The `crle` utility provides for the creation and display of a runtime linking configuration file. Without any arguments, or with just the `–c` option, `crle` displays the contents of a configuration file, any system defaults and the command-line required to regenerate the configuration file. When used with any other options, a new configuration file is created or updated. The configuration file is read and interpreted by the runtime linker, `ld.so.1`(1), during process start-up.

The default configuration file is `/var/ld/ld.config` for 32-bit objects and `/var/ld/64/ld.config` for 64-bit objects. *Note*: It is recommended that any new configuration file is first created in a temporary location. The environment variable `LD_CONFIG` can be set to this new configuration file to cause its use by the runtime linker instead of any default. After verification, the new configuration file can be moved to the default location if desired. Setting the environment variable `LD_NOCONFIG` to any value results in the runtime linker ignoring any configuration files, and may prove useful during experimentation.

The configuration file may contain the following information:

Default Search Paths
: The runtime linker uses a prescribed search path for locating the dynamic dependencies of an object. This search path starts with the components of any `LD_LIBRARY_PATH` definition, followed by the components of an object's `runpath` and finally any defaults specific to the object's type. This last component of the search path can be expressed within the configuration file. *Note*: Typical use of this facility should augment any system defaults; see the `–l` option.

Trusted Directories
: When processing a secure application the runtime linker restricts the use of `LD_LIBRARY_PATH` and the directories from which preload and audit libraries may be used to known trusted directories. These trusted directories can be specified within the configuration file. *Note*: Typical use of this facility should augment any system defaults; see the `–s` option.

Directory Cache
: The location of shared objects within defined directories can be maintained as a cache within

the configuration file. This directory cache can reduce the overhead of searching for application dependencies.

Alternative Objects
: In conjunction with the directory cache, shared objects may have alternative objects specified for use at runtime. These alternate objects may be supplied by the user, or can be created by `crle` as copies of shared objects fixed to known memory locations. These fixed alternative objects can require less processing at runtime than their original shared object counterpart.

Defining alternative default search paths, or additional trusted directories can be useful for administrators who wish to install third party software in a central location, or otherwise alter the search path of applications that may not have been coded with suitable runpaths.

Defining user supplied alternative objects provides a means of replacing dependencies other than via symbolic links or requiring `LD_LIBRARY_PATH` settings.

The directory cache and `crle` generated alternate objects can provide a means of reducing the runtime start-up overhead of applications that require many dependencies, or whose dependencies are expensive to relocate (this may be the case when shared objects contain position-*dependent* code).

When `crle` generated alternate objects are specified within a configuration file, `ld.so.1`(1) performs some minimal consistency verification of the alternative objects against their originating objects. This verification is intended to avert application failure should an applications configuration information become out-of-sync with the underlying system components. When this situation arises the flexibility offered by dynamic linking system components may be compromised, and diagnosing the application failure may be difficult. *Note*: No verification of directory cache information is performed. Any changes to the directory structure will not be seen by a process until the cache is rebuilt.

System shared objects are often well tuned and may have no benefit being cached. The directory cache and alternative object features are typically applicable to user applications and shared objects.

`crle` creates alternate objects for the shared objects discovered when using the −`I` and −`G` options by calls to `dldump`(3DL). The alternate object is created in the directory specified by the preceding −`o` option, or defaults to the directory in which the configuration file is created. The flags used for the `dldump( )` are specified using the −`f` option, or default to `RTLD_REL_RELATIVE`.

OPTIONS

The following options are supported:

–64

Specifies to process 64-bit objects, the default is 32-bit.

–a *name*

This option adds an alternative to *name* to the configuration file. The actual alternative file must be supplied by the user. Multiple occurrences of this option are permitted. If *name* is a directory each shared object within the directory is added to the cache.

–c *conf*

Specifies to use the configuration file name *conf*. If this option is not supplied the default configuration file is used.

–f *flags*

This option provides the symbolic *flags* argument to the dldump(3DL) calls used to generate alternate objects. Any of the RTLD_REL flags defined in /usr/include/dlfcn.h can be used. Multiple flags can be or'ed together using the "|" character, and in this case the string should be quoted to avoid expansion by the shell. If no *flags* values are provided the default flag is RTLD_REL_RELATIVE.

–i *name*

This option adds an individual *name* to the configuration cache. Multiple occurrences of this option are permitted. *name* may be a shared object or a directory. If *name* is a directory each shared object within the directory is added to the cache. *Note*: If *name* does not exist, it is marked in the cache as a nonexistent directory.

–I *name*

This option is the same as –i and in addition any shared objects have alternatives created via dldump(3DL). If the –f flag contains RTLD_REL_EXEC then *name* may be a dynamic executable, for which an alternative is created. Only one dynamic executable can be specified in this manner as the cache created is specific to this application.

–g *name*

This option adds the group *name* to the configuration cache. Each object is expanded to determine its dependencies. Multiple occurrences of this option are permitted. *name* may be a dynamic executable, shared object or a directory.

The *name* itself, if it is a shared object, and its dependencies are added to the cache. If *name* is a directory each shared object within the directory, and its dependencies, are added to the cache.

`−G` *name*

This option is the same as `−g` and in addition any shared objects have alternatives created via `dldump`(3DL). If *name* is a dynamic executable, and the `−f` flag contains `RTLD_REL_EXEC`, then an alternative for the dynamic executable is also created. Only one dynamic executable can be specified in this manner as the cache created is specific to this application.

`−l` *dir*

This option specifies a new default search directory *dir* for `ELF` or `AOUT` objects. Multiple occurrences of this option are permitted. The type of object applicable to the search is specified by the preceding `−t` option, or defaults to `ELF`.

The system default search path for `ELF` objects is `/usr/lib` for 32-bit objects, and `/usr/lib/64` for 64-bit objects. The system default search paths for `AOUT` objects is `/usr/4lib`, `/usr/lib` and `/usr/local/lib`.

Use of this option *replaces* the system default search path, and thus it is normally required that a `−l` option be used to specify the original system default in relation to any new paths being applied. However, if the `−u` option is in effect, and a configuration file does *not* exist, the system defaults are added to the new configuration file before the new paths specified with the `−l` option.

`−o` *dir*

This option specifies the directory *dir* in which any alternate objects will be created. Without this option alternate objects are created in the directory in which the configuration file is created. Multiple occurrences of this option are permitted, the directory *dir* being used to locate alternatives for any following command-line options. Alternative objects are not permitted to override their associated originals.

`–s` *dir*

This option specifies a new trusted directory *dir* for *secure* `ELF` or `AOUT` objects. See `SECURITY` in `ld.so.1`(1) for a definition of secure objects.

Multiple occurrences of this option are permitted. The type of object applicable to the search is specified by the preceding `–t` option, or defaults to `ELF`.

The system default trusted directory for secure `ELF` objects is `/usr/lib/secure` for 32-bit objects and `/usr/lib/secure/64` for 64-bit objects. The system default trusted directories for secure `AOUT` objects are `/usr/4lib`, `/usr/lib`, `/usr/ucblib` and `/usr/local/lib`.

Use of this option *replaces* the system default trusted directories, and thus it is normally required that a `–s` option be used to specify the original system default in relation to any new directories being applied. However, if the `–u` option is in effect, and a configuration file does *not* exist, the system defaults are added to the new configuration file before the new directories specified with the `–s` option.

`–t ELF | AOUT`

This option toggles the object type applicable to any `–l` or `–s` options that follow. The default object type is `ELF`.

`–u`

This option requests that a configuration file be updated, possibly with the addition of new information. Without other options any existing configuration file is inspected and its contents recomputed. Additional arguments allow information to be appended to the recomputed contents. See NOTES.

If a configuration file does not exist it will be created as directed by the other arguments. In the case of the `–l` and `–s` options any system defaults will first be applied to the configuration file before the directories specified with these options.

`–v`

Verbose mode. When creating a configuration file, a trace of the files being processed is written to the standard out. When printing the contents of a configuration file, more extensive directory and file information is provided.

By default the runtime linker attempts to read the configuration file `/var/ld/ld.config` for each 32-bit application it processes or `/var/ld/64/ld.config` for each 64-bit application. When processing an alternative application the runtime linker will use a `$ORIGIN/ld.config.`*app-name* configuration file if present (see NOTES). Applications may reference an alternative configuration file either by setting the `LD_CONFIG` environment variable (see `ld.so.1`(1)), or by recording a configuration file name in the application at the time it is built using the link-editors –`c` option (see `ld`(1)).

EXAMPLES

EXAMPLE 1 Update (and display) of a new default search path for ELF objects

```
example% crle  -u  -l /local/lib
example% crle

Configuration file [2]: /var/ld/ld.config
  Default Library Path (ELF):  /usr/lib:/local/lib
  Trusted Directories (ELF):   /usr/lib/secure  (system default)

Command line:
  crle  -l /usr/lib:/local/lib


example% crle  -u  -l /usr/local/lib
example% crle

Configuration file [2]: /var/ld/ld.config
  Default Library Path (ELF):  /usr/lib:/local/lib:/usr/local/lib
  Trusted Directories (ELF):   /usr/lib/secure  (system default)

Command line:
  crle  -l /usr/lib:/local/lib:/usr/local/lib
```

In this example, the default configuration file initially did not exist, and thus the new search path `/local/lib` is appended to the system default. The next update appends the search path `/usr/local/lib` to those already established in the configuration file.

EXAMPLE 2 Creation (and display) of a new default search path and new trusted directory for ELF objects

```
example% crle  -l /local/lib  -l /usr/lib  -s /local/lib
example% crle

Configuration file [2]: /var/ld/ld.config
  Default Library Path (ELF):  /local/lib:/usr/lib
  Trusted Directories (ELF):   /local/lib
```

```
Command line:
  crle  -l /local/lib:/usr/lib  -s /local/lib
```

With this configuration, third party applications may be installed in `/local/bin` and their associated dependencies in `/local/lib`. The default search path allows the applications to locate their dependencies without the need to set `LD_LIBRARY_PATH`. *Note*: The system default trusted directory has been replaced with this example.

EXAMPLE 3 Creation of a directory cache for ELF objects

```
example% crle -i  /usr/dt/lib -i /usr/openwin/lib -i /usr/lib \
-c config
example% ldd -s ./main
....
   find library=libc.so.1; required by ./main
    search path=/usr/dt/lib:/usr/openwin/lib  (RPATH ./main)
    trying path=/usr/dt/lib/libc.so.1
    trying path=/usr/openwin/lib/libc.so.1
    search path=/usr/lib  (default)
    trying path=/usr/lib/libc.so.1
        libc.so.1 =>     /usr/lib/libc.so.1

example% LD_CONFIG=config  ldd  -s  ./main
....
   find library=libc.so.1; required by ./main
    search path=/usr/dt/lib:/usr/openwin/lib  (RPATH ./main)
    search path=/usr/lib  (default)
    trying path=/usr/lib/libc.so.1
        libc.so.1 =>     /usr/lib/libc.so.1
```

With this configuration, the cache reflects that the system library `libc.so.1` does not exist in the directories `/usr/dt/lib` or `/usr/openwin/lib`. Therefore, the search for this system file ignores these directories even though the application's runpath indicates they should be searched.

EXAMPLE 4 Creation of an alternative object cache for an ELF executable

```
example% crle  -c /local/$HOST/.xterm/ld.config  -f RTLD_REL_ALL  \
-G /usr/openwin/bin/xterm
example% ln  -s /local/$HOST/.xterm/xterm  /local/$HOST/xterm
example% ldd  /usr/local/$HOST/xterm
    libXaw.so.5 =>  /local/$HOST/.xterm/libWaw.so.5  (alternate)
    libXmu.so.4 =>  /local/$HOST/.xterm/libXmu.so.4  (alternate)
    ....
    libc.so.1 =>    /local/$HOST/.xterm/libc.so.1  (alternate)
    ....
```

With this configuration, a new `xterm` and its dependencies are created. These new objects are fully relocated to themselves and result in faster start-up than the originating objects. *Note*: The execution of this application uses its own specific configuration file. This model is generally more flexible than using the environment variable `LD_CONFIG`, as the configuration file will not be erroneously used by other applications such as `ldd`(1) or `truss`(1).

EXAMPLE 5 Creating an alternative object cache to replace an ELF shared object

```
example% ldd  /usr/sbin/vold
    libthread.so.1 =>  /usr/lib/libthread.so.1
    ....

example% crle  -a /usr/lib/libthread.so.1  -o  /usr/lib/lwp
example% crle

Configuration file [2]: /var/ld/ld.config
  Default Library Path (ELF):   /usr/lib  (system default)
  Trusted Directories (ELF):   /usr/lib/secure  (system default)

Directory: /usr/lib
  libthread.so.1  (alternate: /usr/lib/lwp/libthread.so.1)

example% ldd  /usr/sbin/vold
    libthread.so.1 =>  /usr/lib/lwp/libthread.so.1  (alternate)
    ....
```

With this configuration, any dependency that would normally resolve to `/usr/lib/libthread.so.1` will instead resolve to `/usr/lib/lwp/libthread.so.1`. See `threads`(3THR).

EXIT STATUS

The creation or display of a configuration file results in a `0` being returned; otherwise any error condition is accompanied with a diagnostic message and a non-zero value being returned.

NOTES

Tagging an alternative application to use an application specific configuration file can only be achieved if the original application contains one of the *.dynamic* tags `DT_FLAGS_1` or `DT_FEATURE_1`. Without these entries any application specific configuration file must be specified using the `LD_CONFIG` environment variable. Care should be exercised with this latter method as this environment variable will be visible to any forked applications.

The use of the –`u` option requires version 2 of `crle`. This version level is evident from displaying the contents of a configuration file:

```
example% crle

Configuration file [2]: /var/ld/ld.config
  ......
```

With a version 2 configuration file, `crle` is capable of constructing the command-line arguments required to regenerate the configuration file and to provide full update capabilities. Although the update of a version 1 configuration file is possible, the contents of the configuration file may be insufficient for `crle` to compute the entire update requirements.

SUMMARY OF TRUSTED SOLARIS CHANGES

See the `ld`(1) man page, under `ENVIRONMENT VARIABLES`, `LD_LIBRARY_PATH`, for information on trusted directories.

FILES

`/var/ld/ld.config`	Default configuration file for 32-bit applications.
`/var/ld/64/ld.config`	Default configuration file for 64-bit applications.
`/var/tmp`	Default location for temporary configuration file (see `tempnam`(3C)).
`/usr/lib/lddstub`	Stub application employed to `dldump`(3DL) 32-bit objects.
`/usr/lib/64/lddstub`	Stub application employed to `dldump`(3DL) 64-bit objects.
`/usr/lib/libcrle.so.1`	Audit library employed to `dldump`(3DL) 32-bit objects.
`/usr/lib/64/libcrle.so.1`	Audit library employed to `dldump`(3DL) 64-bit objects.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtoo

SEE ALSO

Trusted Solaris 8 Reference Manual

`ld`(1)

SunOS 5.8 Reference Manual

`ld.so.1`(1), `tempnam`(3C), `dldump`(3DL), `threads`(3THR), `attributes`(5)

NAME

crontab – User crontab file

SYNOPSIS

crontab [*filename*]

crontab [–elr] *username*

DESCRIPTION

The crontab utility manages a user's access with cron (see cron(1M)) by copying, creating, listing, and removing crontab files. If invoked without options, crontab copies the specified file, or the standard input if no file is specified, into a directory that holds all users' crontabs.

crontab Access Control

User access to crontab is allowed:

- If the user's name appears in /etc/cron.d/cron.allow.
- If /etc/cron.d/cron.allow does not exist and the user's name is not in /etc/cron.d/cron.deny.

User access to crontab is denied:

- If /etc/cron.d/cron.allow exists and the user's name is not in it.
- If /etc/cron.d/cron.allow does not exist and user's name is in /etc/cron.d/cron.deny.

Note that the rules for allow and deny apply to root only if the allow/deny files exist.

The allow/deny files consist of one user name per line.

crontab Entry Format

A crontab file consists of lines of six fields each. The fields are separated by spaces or tabs. The first five are integer patterns that specify the following:

```
minute (0–59),
hour (0–23),
day of the month (1–31),
month of the year (1–12),
day of the week (0–6 with 0=Sunday).
```

Each of these patterns may be either an asterisk (meaning all legal values) or a list of elements separated by commas. An element is either a number or two numbers separated by a minus sign (meaning an inclusive range). Note that the specification of days may be made by two fields (day of the month and day of the week). Both are adhered to if specified as a list of elements. See EXAMPLES.

The sixth field of a line in a crontab file is a string that is executed by the shell at the specified times. A percent character in this field (unless escaped by \) is translated to a NEWLINE character.

Only the first line (up to a '%' or end of line) of the command field is executed by the shell. Other lines are made available to the command as standard input.

Any line beginning with a '#' is a comment and will be ignored. The file should not contain blank lines.

The shell is invoked from your $HOME directory with an arg0 of sh. Users who desire to have their .profile executed must explicitly do so in the crontab file. cron supplies a default environment for every shell, defining HOME, LOGNAME, SHELL(=/bin/sh), TZ, and PATH. The default PATH for user cron jobs is /usr/bin; while root cron jobs default to /usr/sbin:/usr/bin. The default PATH can be set in /etc/default/cron; see cron(1M).

If you do not redirect the standard output and standard error of your commands, any generated output or errors will be mailed to you.

OPTIONS

The following options are supported:

–e Edits a copy of the current user's crontab file, or creates an empty file to edit if crontab does not exist at the sensitivity label of the invoking process. When editing is complete, the file is installed as the user's crontab file.

If a *username* is specified, then then specified user's crontab file, rather than the current user's crontab file, is edited. A user can edit another user's crontab file under either of the following conditions:

- If the user has modify cron admin authorization *and* the specified *username* is a role user or is in the /etc/cron.d/cron.admin file (which contains a list of administrative users for the cron).
- If the user has modify cron users authorization and the specified *username* is *not* a role user and is *not* in the /etc/cron.d/cron.admin file.

The environment variable EDITOR or VISUAL determines which editor is invoked with the –e option when the user is not assigned the profile shell. The default editor is ed(1). If the user is assigned the profile shell to run in a restricted environment, the –e option determines the editor as follows: if the environment variable is set to be vi, the adminvi editor is used; if it is set to dtpad, the TSOLdtpad editor is used; and if neither variable is set, the adminvi editor is used. Note that all crontab jobs should be submitted using crontab; you should not add jobs by just editing the crontab file because cron will not be aware of changes made this way.

–l Lists the crontab file for the current user at the sensitivity label of the invoking process. A user can list another user's crontab file under either of two conditions. The first condition is when the specified username is in the /etc/cron.d/cron.admin file or is a role user; *and* the user has the modify cron admin authorization. The second condition is when the specified username is neither in the

/etc/cron.d/cron.admin file, nor a role user; *and* the user has the modify cron users authorization.

–r Removes a user's crontab (at the invoking process's sensitivity label) from the crontabs directory. A user can remove another user's crontab file under the following conditions:

- When the user has modify cron admin authorization, *user* must either be the name of a role account or be one of the special system account names listed in the /etc/cron.d/cron.admin file.
- When the user has modify cron users authorization, the specified *user*'s name must *not* be the name of a role account and *not* be in the /etc/cron.d/cron.admin file.

EXAMPLES

EXAMPLE 1 Clean up core files every weekday morning at 3:15 am

```
  15 3 * * 1-5 find $HOME –name core 2>/dev/null | xargs rm –f
```

EXAMPLE 2 Mail a birthday greeting

```
  0 12 14 2 * mailx john%Happy Birthday!%Time for lunch.
```

EXAMPLE 3 Specify days of the month and week

This example

```
  0 0 1,15 * 1
```

would run a command on the first and fifteenth of each month, as well as on every Monday.

To specify days by only one field, the other field should be set to *. For example:

```
  0 0 * * 1
```

would run a command only on Mondays.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of crontab: LC_TYPE, LC_MESSAGES, and NLSPATH.

EDITOR Determine the editor to be invoked when the –e option is specified. The default editor is ed(1). If both the EDITOR and VISUAL environment variables are set, the value of the VISUAL variable is selected as the editor.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

FILES

/etc/cron.d	Main cron directory
/etc/cron.d/cron.allow	List of allowed users
/etc/default/cron	Contains cron default settings.
/etc/cron.d/cron.deny	List of denied users
/var/cron/log	Accounting information

/var/spool/cron/crontabs	Spool area for crontab

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

The crontab command requires the following forced privileges: proc_audit_tcb, file_chown, file_owner, and proc_setid.

Authorization is required to view, modify, or remove another user's crontab file.

An ancillary file is created in the /var/spool/cron/crontabs directory for each crontab file. By convention, the file is named username.ad; and it is used by the clock daemon to set up the cron job to run.

cron jobs are run with the profile shell if the user's login shell (in the passwd entry) or $SHELL is the profile shell. Otherwise, sh is used.

The default Trusted Solaris environment has an /etc/cron.d/cron.deny file, and an /etc/cron.d/cron.admin file.

SEE ALSO

Trusted Solaris 8 Reference Manual

atq(1), atrm(1), cron(1M)

SunOS 5.8 Reference Manual

auths(1), ed(1), sh(1), su(1M), auth_attr(4), attributes(5), environ(5)

NOTES

If you inadvertently enter the crontab command with no argument(s), do not attempt to get out with CTRL-D. This removes all entries in your crontab file. Instead, exit with CTRL-C.

If an authorized user modifies another user's crontab file, resulting behavior may be unpredictable. Instead, the authorized user should first su(1M) to the other user's login before making any changes to the crontab file.

When updating a user's crontab file with the crontab command, the cron process sees this update immediately when no cron jobs are running. However, if cron is running any cron job(s) at the time of updating, it could take a maximum of 60 seconds before cron is aware of this update. Therefore, to be safe, a new job should be started at least 60 seconds after the current date and time.

NAME

date – write the date and time

SYNOPSIS

/usr/bin/date [–u] [+ *format*]

/usr/bin/date [–a [–] *sss.fff*]

/usr/bin/date [–u] [[*mmdd*] *HHMM mmddHHMM* [*cc*] *yy*][.*SS*]

/usr/xpg4/bin/date [–u] [+ *format*]

/usr/xpg4/bin/date [–a [–] *sss.fff*]

/usr/xpg4/bin/date [–u] [[*mmdd*] *HHMM mmddHHMM* [*cc*] *yy*][.*SS*]

DESCRIPTION

The `date` utility writes the date and time to standard output or attempts to set the system date and time. By default, the current date and time will be written.

Specifications of native language translations of month and weekday names are supported. The month and weekday names used for a language are based on the locale specified by the environment variable `LC_TIME`; see `environ`(5).

The following is the default form for the "C" locale:

%a %b %e %T %Z %Y

For example,

```
Fri Dec 23 10:10:42 EST 1988
```

OPTIONS

The following options are supported:

–a [–] *sss.fff* Slowly adjust the time by *sss.fff* seconds (*fff* represents fractions of a second). This adjustment can be positive or negative. The system's clock will be sped up or slowed down until it has drifted by the number of seconds specified. Only a user with the `solaris.system.date` authorization may adjust the time.

–u Display (or set) the date in Greenwich Mean Time (GMT—universal time), bypassing the normal conversion to (or from) local time.

OPERANDS

The following operands are supported:

+*format* If the argument begins with `+`, the output of `date` is the result of passing *format* and the current time to `strftime( )`. `date` uses the conversion specifications listed on the `strftime`(3C) manual page, with the conversion specification for `%C` determined by whether `/usr/bin/date` or `/usr/xpg4/bin/date` is used:

`/usr/bin/date` Locale’s date and time representation. This is the default output for `date`.

`/usr/xpg4/bin/date` Century (a year divided by 100 and truncated to an integer) as a decimal number [00-99].

The string is always terminated with a `NEWLINE`. An argument containing blanks must be quoted; see the `EXAMPLES` section.

mm Month number

dd Day number in the month

HH Hour number (24 hour system)

MM Minute number

SS Second number

cc Century minus one (for example, *cc* is `20` for a date in the 21st century)

yy Last 2 digits of the year number

The month, day, year, and century may be omitted; the current values are applied as defaults. For example, the following entry:

```
example% date 10080045
```

sets the date to Oct 8, 12:45 a.m. The current year is the default because no year is supplied. The system operates in GMT. `date` takes care of the conversion to and from local standard and daylight time. Only a user with the `solaris.system.date` authorization may change the date. After successfully setting the date and time, `date` displays the new date according to the default format. The `date` command uses `TZ` to determine the correct time zone information; see `environ`(5).

EXAMPLES

EXAMPLE 1 Generating output

The command

```
example% date ’+DATE: %m/%d/%y%nTIME:%H:%M:%S’
```

generates as output

```
DATE: 08/01/76

TIME: 14:45:05
```

EXAMPLE 2 Setting the current time

The command

```
example# date 1234.56
```

sets the current time to 12:34:56.

EXAMPLE 3 Setting another time and date in Greenwich Mean Time

The command

```
example# date -u 010100302000
```

sets the date to January 1st, 12:30 am, 2000, which will be displayed as

```
Thu Jan 01 00:30:00 GMT 2000
```

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of date: LC_CTYPE, LC_TIME, LC_MESSAGES, and NLSPATH.

TZ Determine the timezone in which the time and date are written, unless the –u option is specified. If the TZ variable is not set and the –u is not specified, the system default timezone is used.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

/usr/bin/date

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

/usr/xpg4/bin/date

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

Changing the date and time requires the solaris.system.date authorization.

SEE ALSO

SunOS 5.8 Reference Manual

strftime(3C), attributes(5), environ(5), XPG4(5)

DIAGNOSTICS

no permission — You do not have the solaris.system.date authorization and you tried to change the date or time.

bad conversion — The date or time set is syntactically incorrect.

NOTES

If you attempt to set the current date to one of the dates that the standard and alternate time zones change (for example, the date that daylight time is starting or ending), and you attempt to set the time to a time in the interval between the end of standard time and the beginning of the alternate time (or the end of the alternate time and the beginning of standard time), the results are unpredictable.

Using the date command from within windowing environments to change the date can lead to unpredictable results and is unsafe. It may also be unsafe in the multi-user mode, that is, outside of a windowing system, if the date is changed rapidly back and forth. The recommended method of changing the date is 'date –a'.

NAME

enable, disable – Enable / disable LP printers

SYNOPSIS

/usr/bin/enable *printer…*

/usr/bin/disable [–c | –W][–r [*reason*]] *printer…*

DESCRIPTION

The `enable` command activates the named *printer* s, enabling them to print requests submitted by the `lp` command. If the printer is remote, the command enables only the transfer of requests to the remote system. Run the `enable` command on the remote system to activate the printer.

(Run `lpstat` **–p** to get the status of *printer* s.)

The `disable` command deactivates the named *printer,* disabling it from printing requests submitted by `lp` . By default, any requests that are currently printing on the designated printer(s) are reprinted in their entirety either on the same printer or on another member of the same class of printers. If the printer is remote, this command stops only the transmission of jobs to the remote system. Run the `disable` command on the remote system to disable the printer.

(Run `lpstat` **–p** to get the status of *printer* s.)

OPTIONS

The following options are supported for use with `disable` :

–c Cancel any requests that are currently printing on *printer* . This option cannot be used with the –W option. If the printer is remote, the –c option will be silently ignored.

–W Wait until the request currently being printed is finished before disabling *printer* . This option cannot be used with the –c option. If the printer is remote, the –W option will be silently ignored.

–r [*reason*] Assign a *reason* for the disabling of the printer(s). This *reason* applies to all printers specified. This *reason* is reported by `lpstat` –p . Enclose *reason* in quotes if it contains blanks. The default reason is `unknown reason` for the existing printer, and `new printer` for a printer added to the system but not yet enabled.

OPERANDS

The following operands are supported for both `enable` and `disable` :

printer The name of the printer to be enabled or disabled. Specify *printer* using atomic name. See `printers.conf`(4) for information regarding the naming conventions for atomic names.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

non-zero An error occurred.

FILES

`/var/spool/lp/*` LP print queue.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlpu
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

Use of the `enable` and `disable` commands requires the `administer printing authorization`.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lp`(1) , `lpstat`(1)

SunOS 5.8 Reference Manual

`printers.conf`(4) , `attributes`(5)

NAME

dtappsession – Start a new Application Manager session

SYNOPSIS

/usr/dt/bin/dtappsession [*hostname*]

DESCRIPTION

`dtappsession` is a specialized version of the `Xsession` shell script. It is an alternative to using the CDE remote login that allows you to access a remote host without logging out of your current CDE session. `dtappsession` starts a new instance of the CDE Application Manager in its own ToolTalk™ session. It can be used to remotely display the Application Manager back to your local display after logging in to a remote host via `rlogin`(1)

A new, independent instance of `ttsession`(1) starts a simple session management window. This window displays the title

```
remote_hostname: Remote Administration
```

where *remote_hostname* is the system that is being accessed. The window also displays an `Exit` button. Clicking `Exit` terminates the ToolTalk session and all windows that are part of the session.

The Application Manager that is displayed can be used to start remote CDE actions to run in this session. Exiting the Application Manager does not terminate the session, and it is not recommended. Clicking `Exit` is the recommended way to end the session. To avoid confusing the remote CDE applications with local ones, it is recommended that a new CDE workspace be created for clients in the remote session.

The *hostname* is not needed when the `DISPLAY` environment variable is set to the local hostname on the remote host.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWdtdte

EXAMPLES

EXAMPLE 1 Remote Login and dtappsession

After creating a new CDE workspace, type the following in a terminal window:

```
  # rlogin remote_hostname
  password: /*enter the remote password*/

  # dtappsession local_hostname /* on the remote host */
```

SUMMARY OF TRUSTED SOLARIS CHANGES

In the Trusted Solaris environment, `dtappsession` can be used for remote administration by administrative roles that have the ability to login into the remote host. (Remote login requires the `remote login` authorization if an entry does not exist in `/etc/hosts.equiv` or an `rhosts` file on the remote host and if the entry allows logins without a password. See the rlogin(1) man page.) Remote login to role accounts requires that the Trusted Path process attribute be asserted, so both the local and remote hosts must be running the Trusted Solaris environment, and the role must have been already assumed on the local host. For the administrative role to remotely log in, the line that sets CONSOLE=/dev/console must be commented out in the /etc/default/login file. In addition, the dtappsession command must be listed in one of the role's execution profiles.

dtappsession does not require any privilege, and it does not need to run on a Trusted Solaris system. When installed in /usr/dt/bin on a standard Solaris environment, along with the startApp.ds file, dtappsession can be used to administer the remote Solaris system from a local Trusted Solaris system. However, in this case, the CDE workspace used for remote display must be a normal workspace, rather than a role workspace.

If the root role is used to do remote administration using Solstice™ tools, an entry for the remote host must be made in the NIS+ admin group if the remote host is not a NIS+ master. See nisgrpadm(1).

FILES

/usr/dt/bin/startApp.ds	Dt Korn shell script for session manager window

BUGS

X11/CDE applications that do not register with the ToolTalk session manager will not exit automatically when the session is terminated. Such applications must be explicitly terminated.

SEE ALSO

dtfile(1), nisgrpadm(1), rlogin(1), ttsession(1), attributes(5)

Trusted Solaris Administrator's Procedures

NAME

enable, disable – Enable / disable LP printers

SYNOPSIS

/usr/bin/enable *printer…*

/usr/bin/disable [`–c` | `–W`][`–r` [*reason*]] *printer…*

DESCRIPTION

The `enable` command activates the named *printer* s, enabling them to print requests submitted by the `lp` command. If the printer is remote, the command enables only the transfer of requests to the remote system. Run the `enable` command on the remote system to activate the printer.

(Run `lpstat` **–p** to get the status of *printer* s.)

The `disable` command deactivates the named *printer,* disabling it from printing requests submitted by `lp` . By default, any requests that are currently printing on the designated printer(s) are reprinted in their entirety either on the same printer or on another member of the same class of printers. If the printer is remote, this command stops only the transmission of jobs to the remote system. Run the `disable` command on the remote system to disable the printer.

(Run `lpstat` **–p** to get the status of *printer* s.)

OPTIONS

The following options are supported for use with `disable` :

`–c` Cancel any requests that are currently printing on *printer* . This option cannot be used with the `–W` option. If the printer is remote, the `–c` option will be silently ignored.

`–W` Wait until the request currently being printed is finished before disabling *printer* . This option cannot be used with the `–c` option. If the printer is remote, the `–W` option will be silently ignored.

`–r` [*reason*] Assign a *reason* for the disabling of the printer(s). This *reason* applies to all printers specified. This *reason* is reported by `lpstat` `–p` . Enclose *reason* in quotes if it contains blanks. The default reason is `unknown reason` for the existing printer, and `new printer` for a printer added to the system but not yet enabled.

OPERANDS

The following operands are supported for both `enable` and `disable` :

printer The name of the printer to be enabled or disabled. Specify *printer* using atomic name. See `printers.conf`(4) for information regarding the naming conventions for atomic names.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

non-zero	An error occurred.

FILES

`/var/spool/lp/*`	LP print queue.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWlpu
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

Use of the `enable` and `disable` commands requires the `administer printing authorization`.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lp`(1) , `lpstat`(1)

SunOS 5.8 Reference Manual

`printers.conf`(4) , `attributes`(5)

NAME

find, tfind – Find files

SYNOPSIS

find *path*… *expression*

tfind *path*… *expression*

DESCRIPTION

The `find` utility recursively descends the directory hierarchy for each *path* , seeking files that match a Boolean *expression* written in the primaries given below.

`find` can descend to arbitrary depths in a file hierarchy and will not fail due to path length limitations (unless a *path* operand specified by the application exceeds `PATH_MAX` requirements).

The `tfind` command supports execution of commands in restricted environments defined by the profile-shell mechanism. `tfind` contains all the functionality of `find` , except for the expressions `–exec` *command* and `–ok` *command* . For these expressions `tfind` invokes *command* through the profile shell (`pfexec`(1)).

OPERANDS

The following operands are supported:

path A path name of a starting point in the directory hierarchy.

expression The first argument that starts with a – , or is a `!` or a `(` , and all subsequent arguments will be interpreted as an *expression* made up of the following primaries and operators. In the descriptions, wherever *n* is used as a primary argument, it will be interpreted as a decimal integer optionally preceded by a plus (`+`) or minus (`-`) sign, as follows:

+*n*	more than *n*
n	exactly *n*
`–n`	less than *n*

Expressions

Valid expressions are:

`–atime` True if the file was accessed *n* days ago. The access time of directories in *path* is changed by `find` itself.

n

`–cpio` Always true; write the current file on *device* in `cpio` format (5120-byte records).

device

`–ctime` True if the file’s status was changed *n* days ago.

n

`–depth` Always true; causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when `find` is used with `cpio`(1) to transfer files that are contained in directories without write permission.

`–exec` *command*
True if the executed *command* returns a zero value as exit status. The end of *command* must be punctuated by an escaped semicolon. A command argument `{}` is replaced by the current path name. If issued from `tfind` , the command is invoked through a profile shell (`pfsh`).

`–follow`
Always true; causes symbolic links to be followed. When following symbolic links, `find` keeps track of the directories visited so that it can detect infinite loops; for example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the `–type l` expression.

`–fstype` *type*
True if the filesystem to which the file belongs is of type *type* .

`–group` *gname*
True if the file belongs to the group *gname* . If *gname* is numeric and does not appear in the `/etc/group` file, or in the NIS / NIS+ tables, it is taken as a group ID .

`–inum` *n*
True if the file has inode number *n* .

`–links` *n*
True if the file has *n* links.

`–local`
True if the file system type is not a remote file system type as defined in the `/etc/dfs/fstypes` file. `nfs` is used as the default remote filesystem type if the `/etc/dfs/fstypes` file is not present. Note that `–local` will descend the hierarchy of non-local directories. See EXAMPLES for an example of how to search for local files without descending.

`–ls`
Always true; prints current path name together with its associated statistics. These include (respectively):

- inode number
- size in kilobytes (1024 bytes)
- protection mode
- number of hard links

- user
- group
- size in bytes
- modification time.

If the file is a special file the size field will instead contain the major and minor device numbers.

If the file is a symbolic link the pathname of the linked-to file is printed preceded by ‘ → ’. The format is identical to that of **ls –gilds** (see ls(1B)). Note: Formatting is done internally, without executing the ls program.

–M

In all multilevel directories (MLD) encountered, search single-level directories (SLD s) that are dominated by the sensitivity label of the process. However, if the effective privilege set of the process contains the file_mac_read and file_mac_search privileges, search all SLD s. The file system enforces all underlying DAC policies and privilege interpretations.

If –M is *not* specified and *path* points to an adorned MLD, traverse only this MLD ’s SLD s. For all other MLD s encountered, automatically translate to the SLD at the sensitivity label of the process even if find is run with all privileges.

If –M is *not* specified and *path* points to an unadorned MLD , for this and all other MLD s encountered, automatically translate to the SLD at the sensitivity label of the process even if find is run with all privileges.

If –M is *not* specified and *path* does not point to an MLD , for all MLD s encountered, automatically translate to the SLD at the sensitivity label of the process even if find is run with all privileges.

–mount

Always true; restricts the search to the file system containing the directory specified. Does not list mount points to other file systems.

–mtime *n*

True if the file’s data was modified *n* days ago.

`–name` *pattern*
True if *pattern* matches the current file name. Normal shell file name generation characters (see `sh`(1)) may be used. A backslash (`\\`) is used as an escape character within the pattern. The pattern should be escaped or quoted when `find` is invoked from the shell.

`–ncpio` *device*
Always true; write the current file on *device* in `cpio –c` format (5120 byte records).

`–newer` *file*
True if the current file has been modified more recently than the argument *file* .

`–nogroup`
True if the file belongs to a group not in the `/etc/group` file, or in the NIS / NIS+ tables.

`–nouser`
True if the file belongs to a user not in the `/etc/passwd` file, or in the NIS / NIS+ tables.

`–ok` *command*
Like `–exec` except that the generated command line is printed with a question mark first and is executed only if the user responds by typing **`y`** . If issued from `tfind` , `command` is invoked through a profile shell (`pfsh`).

`–perm` [–] *mode*
The *mode* argument is used to represent file mode bits. It will be identical in format to the < *symbolic* mode > operand described in `chmod`(1) , and will be interpreted as follows. To start, a template will be assumed with all file mode bits cleared. An *op* symbol of:

`+` Will set the appropriate mode bits in the template.

– Will clear the appropriate bits.

`=` Will set the appropriate mode bits, without regard to the contents of the file mode creation mask of a process.

The *op* symbol of – cannot be the first character of *mode* ; this restriction avoids ambiguity with the optional leading hyphen. Because the initial mode is all bits off, there are no symbolic modes that need to use – as the first character.

If the hyphen is omitted, the primary will evaluate as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if *mode* is prefixed by a hyphen, the primary will evaluate as true if at least all the bits in the resulting template are set in the file permission bits.

`–perm` [–] *onum*
True if the file permission flags exactly match the octal number *onum* . [See `chmod`(1) .] If *onum* is prefixed by a minus sign (`–`), only the bits that are set in *onum* are compared with the file-permission flags, and the expression evaluates true if they match.

`–print`
Always true; causes the current path name to be printed.

`–prune`
Always yields true. Do not examine any directories or files in the directory structure below the *pattern* just matched. (See `EXAMPLES`). Specifying `–depth` overrides the `–prune` option, which will have no effect.

`–size` *n* [`c`]
True if the file is *n* blocks long (512 bytes per block). If *n* is followed by a `c` , the size is in bytes.

`–type` *c*
True if the type of the file is *c* , where *c* is `b` , `c` , `d` , `D` , `f` , `l` , `m` , `p` , or `s` for block special file, character special file, directory, door, plain file, symbolic link, MLD , FIFO(named pipe), or socket, respectively.

`–user` *uname*
True if the file belongs to the user *uname* . If *uname* is numeric and does not appear as a login name in the `/etc/passwd` file, it is taken as a user ID .

True if the file belongs to the user *uname* . If *uname* is numeric and does not appear as a login name in the `/etc/passwd` file, or in the NIS / NIS+ tables, it is taken as a user ID .

`–xdev`
Same as the `–mount` primary.

Complex Expressions

The primaries may be combined using the following operators (in order of decreasing precedence):

1) (*expression*)
True if the parenthesized expression is true. (Parentheses are special to the shell and must be escaped.)

2)
`!`
expression

The negation of a primary (`!` is the unary *not* operator).

3)
expression
`[`
`–a`
`]`
expression

Concatenation of primaries (the `AND` operation is implied by the juxtaposition of two primaries).

4)
expression
`–o`

expression

Alternation of primaries (`–o` is the `OR` operator).

Note: When you use `find` in conjunction with `cpio` , if you use the `–L` option with `cpio` , then you must use the `–follow` expression with `find` and vice versa, otherwise there will be undesirable results.

If no *expression* is present, `–print` is used as the expression. Otherwise, if the given expression does not contain any of the primaries `–exec` , `–ok` , or `–print` , the given expression will be effectively replaced by

(*given_expression*) `–print`

The `–user` , `–group` , and `–newer` primaries each will evaluate their respective arguments only once. Invocation of *command* specified by `–exec` or `–ok` does not affect subsequent primaries on the same file.

USAGE

See `largefile`(5) for the description of the behavior of `find` when encountering files greater than or equal to 2 Gbyte (2 ^{31} bytes).

EXAMPLES

EXAMPLE 1 Writing out the hierarchy directory

The following commands are equivalent:

```
example% find .  example% find . –print
```

They both write out the entire directory hierarchy from the current directory.

EXAMPLE 2 Removing files

Remove all files in your home directory named `a.out` or `*.o` that have not been accessed for a week:

```
example% find $HOME \\( -name a.out -o -name *.ó \\) \\
-atime +7 -exec rm {} \\;
```

EXAMPLE 3 Printing all file names but skipping SCCS directories

Recursively print all file names in the current directory and below, but skipping SCCS directories:

```
example% find . –name SCCS –prune –o –print
```

EXAMPLE 4 Printing all file names and the SCCS directory name

Recursively print all file names in the current directory and below, skipping the contents of SCCS directories, but printing out the SCCS directory name:

```
example% find . –print –name SCCS –prune
```

EXAMPLE 5 Testing for the newer file

The following command is roughly equivalent to the –nt extension to test(1) :

```
example$ if [ -n "$(find file1 -prune -newer file2)"
]; then   printf %s\\\
 "file1 is newer than file2"
```

EXAMPLE 6 Selecting a file using 24–hour mode

The descriptions of –atime , –ctime , and –mtime use the terminology *n* “24-hour periods”. For example, a file accessed at 23:59 will be selected by:

```
example% find . –atime –1 –print
```

at 00:01 the next day (less than 24 hours later, not more than one day ago). The midnight boundary between days has no effect on the 24-hour calculation.

EXAMPLE 7 Finding files by a literal in their names

Find files with “abc” in their names; search all SLD s dominated by the sensitivity label as the find process:

```
example% find begin_path –M –type f –name ’*abc*’
```

EXAMPLE 8 Traversing directories by sensitivity label

Find MLD s with “xyz” in their names; search all SLD s dominated by the sensitivity label as the find process:

```
example% find begin_path –M –type m –name ’*xyz*’
```

EXAMPLE 9 Removing files with “abc” in their names

Remove files with “abc” in their names; begin at the current directory and perform the removal through a profile shell (pfsh).

```
example% tfind . –type f –name ’*abc*’ –exec rm {} \\;
```

EXAMPLE 10 Printing files matching a user’s permission mode

Recursively print all file names whose permission mode exactly matches read, write, and execute access for user, and read and execute access for group and other:

```
example% find . –perm u=rwx,g=rx,o=rx
```

The above could alternatively be specified as follows:

```
example% find . –perm a=rwx,g-w,o-w
```

EXAMPLE 11 Printing files with write access for other

Recursively print all file names whose permission includes, but is not limited to, write access for other:

```
example% find . -perm -o+w
```

EXAMPLE 12 Printing local files without descending non-local directories

```
example% find . ! -local -prune -o -print
```

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `find` : `LC_COLLATE` , `LC_CTYPE` , `LC_MESSAGES` , `LC_TIME` , and `LC_ALL` , and `NLSPATH` .

EXIT STATUS

The following exit values are returned:

`0` All *path* operands were traversed successfully.

`>0` An error occurred.

SUMMARY OF TRUSTED SOLARIS CHANGES

Modifications to the `find` command deal with multilevel directories. A new `–M` option enables traversing MLD s. A new argument (`m`) for the `–type` option enables selecting the MLD type.

FILES

`/etc/passwd`	Password file
`/etc/group`	Group file
`/etc/dfs/fstypes`	File that registers distributed file system packages

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

SEE ALSO

Trusted Solaris 8 Reference Manual

`chmod`(1) , `stat`(2)

SunOS 5.8 Reference Manual

`cpio`(1) , `ls`(1) , `pfexec`(1) , `sh`(1) , `test`(1) , `umask`(2) , `attributes`(5) , `environ`(5) , `largefile`(5)

WARNINGS

The following options are obsolete and will not be supported in future releases:

–`cpio` *device* Always true; write the current file on *device* in `cpio` format (5120-byte records).

–`ncpio` *device* Always true; write the current file on *device* in `cpio` –`c` format (5120 byte records).

NOTES

When using `find` to determine files modified within a range of time, one must use the –`time` argument *before* the –`print` argument; otherwise, `find` will give all files.

NAME

getfattrflag – Gets the file's security attributes flag

SYNOPSIS

/usr/bin/getfattrflag *filename*…

/usr/bin/getfattrflag [–t] [–m] [–p] *filename*…

/usr/bin/getfattrflag [–t] [–q –m] [–q –p] [–q –s] *filename*…

DESCRIPTION

getfattrflag displays the security attributes flags of *filename.* To display a file's attributes flag information, you must have DAC read and execute permission to all directories in the path name leading to the file, and MAC read access to the file. If no option is specified, the –m, –p, and –s options are applied by default.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

OPTIONS

–m Determine if *filename* is a multilevel directory.

–p Determine if *filename* is a public object. To display the true value of the flag, the process must have the file_audit privilege.

–q Quiet mode. This option must be used with one (and only one) of the other options. No verbose output is supplied.

–s Determine if *filename* is a single-level directory.

–t If *filename* is a multilevel directory, this option causes getfattrflag to return the flag values for the underlying SLD. Without this option, the flag values for the MLD are returned.

EXAMPLES

EXAMPLE 1 Use of getfattrflag

getfattrflag does not distinguish between directories and regular files. If no option is specified, getfattrflag returns the current value of all flags.

```
example% getfattrflag fl1
fl1: is not a multilevel directory, is not a single-level
directory, is a public object

example% getfattrflag -p fl1
fl1: is a public object

example% getfattrflag -m fl1
fl1: is not a multilevel directory
```

RETURN VALUES

getfattrflag exits with one of the following values:

`0`	True value returned for requested flag.
`1`	False value return for requested flag.
`2`	Error occurred.

NOTES

Using the `–m` and `–t` options together returns false unless *filename* is a fully adorned pathname to a multilevel directory.

SEE ALSO

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

getfpriv – Gets the privileges assigned to files

SYNOPSIS

getfpriv *filename*…

getfpriv [–s] –a *filename*…

getfpriv [–s] –f *filename*…

DESCRIPTION

getfpriv gets the privileges associated with each *filename.* With no options, both the forced and allowed sets are displayed. The forced privileges are displayed first followed by the allowed set. The default output is as follows:

filename FORCED: *p1,p2,p3...* ALLOWED: *p1,p2,p3...*

The –s option is used when getfpriv is invoked within the command line of setfpriv(1). The output of the command with the –s option is as follows:

p1,p2,p3...

For example, if the allowed privileges need to be set on *file1*, exactly as they were set on *filename*, the command line of setfpriv would look like the following:

setfpriv –s –a ‘getfpriv –s –a *filename*‘ *file1*

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

OPTIONS

–a Display the privileges in the allowed set only.

–s Print the list of privileges in a format suitable for use by setfpriv(1). This option is a modifier and must be used with either the –a or –f option.

–f Display privileges in the forced set only.

RETURN VALUES

getfpriv exits with one of the following values:

0 Successful completion.

1 Unsuccessful completion.

SEE ALSO

Trusted Solaris 8 Reference Manual

setfpriv(1)

SunOS 5.8 Reference Manual

attributes(5)

NAME

getlabel – Get the CMW label for files

SYNOPSIS

/usr/bin/getlabel [–hiIlLsSx] *filename*…

DESCRIPTION

`getlabel` gets the CMW label associated with each *filename.* When options are not specified, the output format of the CMW label is displayed in default format. When the specified options conflict, `getlabel` terminates with an error. Conflicting options include `–i` and `–I`, `–s` and `–S`, and `–l` and `–L`.

OPTIONS

`–h` Get the label of the symbolic link instead of the file it points to.

`–i` Get the information label (IL) portion from the CMW label associated with the specified file, and display it. ILs display as `ADMIN_LOW`. See `NOTES` below.

`–I` Get the information label portion from the CMW label associated with the specified file, and display it. ILs display as `ADMIN_LOW`. See `NOTES` below.

`–l` Get the CMW label associated with the specified file, and display the CMW label in short form; equivalent to `–i –s`.

`–L` Get the CMW label associated with the specified file, and display the CMW label in long form; equivalent to `–I –S`.

`–s` Get the sensitivity label portion from the CMW label associated with the specified file, and display the sensitivity label in short form.

`–S` Get the sensitivity label portion from the CMW label associated with the specified file, and display the sensitivity label in long form.

`–x` Get the CMW label associated with the specified file, and display the label in hexadecimal form.

RETURN VALUES

`getlabel` exits with one of the following values:

`0` Successful completion

`1` Unsuccessful completion due to usage error

`2` Unable to translate label

`3` Unable to allocate memory

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

SEE ALSO

Trusted Solaris 8 Reference Manual

`setlabel`(1)

SunOS 5.8 Reference Manual

`attributes`(5)

NOTES

Information labels (ILs) are not supported in Trusted Solaris 7 and later releases. Trusted Solaris software interprets any ILs on communications and files from systems running earlier releases as `ADMIN_LOW`.

- Objects still have CMW labels, and CMW labels still include the IL component: `IL[SL]`; however, the IL component is fixed at `ADMIN_LOW`.

  As a result, Trusted Solaris 7 and later releases have the following characteristics:

  - ILs do not display in window labels; SLs (Sensitivity Labels) display alone within brackets.
  - ILs do not float.
  - Setting an IL on an object has no effect.
  - Getting an object's IL will always return `ADMIN_LOW`.
  - Although certain utilities, library functions, and system calls can manipulate IL strings, the resulting ILs cannot be set on any objects.
  - Sensitivity labels, not information labels, display on printer banners.
  - IL–related privileges are no longer used.
  - In auditing, the `ilabel` token is recorded as `ADMIN_LOW`, when it is recorded. The audit event numbers 519 (AUE_OFLOAT), 520 (AUE_SFLOAT), and 9036 (AUE_iil_change) continue to be reserved, but those events are no longer recorded.

NAME

getmldadorn – Display the multilevel directory adornment of the file system

SYNOPSIS

getmldadorn *pathname*

DESCRIPTION

`getmldadorn` displays the MLD adornment of the file system on which *pathname* resides.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

RETURN VALUES

`getmldadorn` exits with one of these values:

`0`	Success
`1`	Usage error
`2`	Failure; error message is the system error number from `getmldadorn`(2).

SEE ALSO

Trusted Solaris 8 Reference Manual

`getmldadorn`(2)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

getsldname – Display file-system single-level directory name

SYNOPSIS

/usr/bin/getsldname [–s *sensitivity_label*] *pathname*

DESCRIPTION

`getsldname` displays the SLD name associated with the sensitivity label of the current process within the multilevel directory (MLD) referred to by the specified full *pathname*. The final component of *pathname* must be a MLD.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

OPTIONS

`–s` Get the SLD name associated with the sensitivity label provided.

DIAGNOSTICS

`getsldname` exits with one of the following values:

`0` Success

`1` Usage error

`2` Failure; error message is the system error number from `getcmwplabel`(2)

`3` Failure; error message is the system error number from `getsldname`(2)

SEE ALSO

Trusted Solaris 8 Reference Manual

`getcmwplabel`(2), `getsldname`(2)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

ipcrm – Remove a message queue, semaphore set, or shared memory ID

SYNOPSIS

ipcrm [–`l` *slabel*] [–`m` *shmid*] [–`q` *msqid*] [–`s` *semid*] [–`M` *shmkey*] [–`Q` *msgkey*]
[–`S` *semkey*…]

DESCRIPTION

`ipcrm` removes one or more messages, semaphores, or shared memory identifiers.

The invoking process must have both mandatory and discretionary access to the IPC or must be suitably privileged.

OPTIONS

The identifiers are specified by the following options:

–`l` *slabel* Use the specified sensitivity *slabel* (instead of the current sensitivity label) of the process in conjunction with subsequent –`M`, –`Q`, and –`S` options.

–`m` *shmid* Remove the shared memory identifier *shmid* from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.

–`q` *msqid* Remove the message queue identifier *msqid* from the system and destroy the message queue and data structure associated with it.

–`s` *semid* Remove the semaphore identifier *semid* from the system and destroy the set of semaphores and data structure associated with it.

–`M` *shmkey* Removes the shared memory identifier, created with key *shmkey*, from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.

–`Q` *msgkey* Remove the message queue identifier, created with key *msgkey*, from the system and destroy the message queue and data structure associated with it.

–`S` *semkey* Remove the semaphore identifier, created with key *semkey*, from the system and destroy the set of semaphores and data structure associated with it.

The details of the removes are described in `msgctl`(2), `shmctl`(2), and `semctl`(2). Use the `ipcs` command to find the identifiers and keys.

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `ipcrm`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipc

SUMMARY OF TRUSTED SOLARIS CHANGES

There is a new option, `–l`, for operating on keys at a specific sensitivity label. Appropriate privilege is required to override failed access checks. For more information on required privileges, see the `IPC_RMID` option in `msgctl`(2), `semctl`(2), and `shmctl`(2).

SEE ALSO

Trusted Solaris 8 Reference Manual

`ipcs`(1), `msgctl`(2), `msgget`(2), `msgrcv`(2), `msgsnd`(2), `semctl`(2), `semget`(2), `semop`(2), `shmctl`(2), `shmget`(2), `shmop`(2)

SunOS 5.8 Reference Manual

`attributes`(5), `environ`(5)

NAME

ipcs – Report inter-process communication facilities status

SYNOPSIS

/usr/bin/ipcs [–aAbcilmopqst] [–C *corefile*] [–N *namelist*]

/usr/xpg4/bin/ipcs [–aAbcimopqst] [–C *corefile*] [–N *namelist*]

DESCRIPTION

The utility `ipcs` prints information about active inter-process communication facilities. The information that is displayed is controlled by the options supplied. Without options, information is printed in short format for message queues, shared memory, and semaphores that are currently active in the system.

/usr/xpg4/bin/ipcs See NOTES.

OPTIONS

The following options are supported:

–m Prints information about active shared memory segments.

–q Prints information about active message queues.

–s Prints information about active semaphores.

If –m, –q, or –s are specified, information about only those indicated is printed. If none of these three is specified, information about all three is printed subject to these options:

–a Uses all XCU5 print options. (This is a shorthand notation for –b, –c, –o, –p, and –t.)

–A Uses all print options. (This is a shorthand notation for –b, –c, –i, –l, –o, –p, and –t.)

–b Prints information on biggest allowable size: maximum number of bytes in messages on queue for message queues, size of segments for shared memory, and number of semaphores in each set for semaphores. See below for meaning of columns in a listing.

–c Prints creator’s login name and group name. See below.

–C *corefile* Uses the file *corefile* in place of `/dev/mem` and `/dev/kmem`. Use a core dump obtained from `savecore`(1M) in place of `/dev/mem` and `/dev/kmem`. Without the –C option (default), the running system image is used.

–i Prints number of ISM attaches to shared memory segments.

–l Prints the sensitivity label associated with the object.

–N *namelist* Uses the file *namelist* in place of `/dev/ksyms`.

–o Prints information on outstanding usage: number of messages on queue and total number of bytes in messages on queue for message queues and number of processes attached to shared memory segments.

–p Prints process number information: process ID of last process to send a message, process ID of last process to receive a message on message queues, process ID of creating process, and process ID of last process to attach or detach on shared memory segments. See below.

–t Prints time information: time of the last control operation that changed the access permissions for all facilities, time of last msgsnd(2) and last msgrcv(2) on message queues, time of last shmat(2) and last shmdt(2) on shared memory (see shmop(2)), time of last semop(2) on semaphores. See below.

–t Prints time information: time of the last control operation that changed the access permissions for all facilities, time of last msgsnd(2) and last msgrcv(2) on message queues, time of last shmat(2) and last shmdt(2) on shared memory (see shmop(2)), time of last semop(2) on semaphores. See below.

The column headings and the meaning of the columns in an ipcs listing are given below; the letters in parentheses indicate the options that cause the corresponding heading to appear; “all” means that the heading always appears. Note: These options only determine what information is provided for each facility; they do not determine which facilities are listed.

T (all) Type of the facility:

q	message queue
m	shared memory segment
s	semaphore

ID (all) The identifier for the facility entry.

KEY (all) The key used as an argument to msgget( ), semget( ), or shmget ( ) to create the facility entry. (Note:The key of a shared memory segment that has been removed is changed to IPC_PRIVATE until all processes attached to the segment detach it.)

MODE (all) The facility access modes and flags: The mode consists of 11 characters that are interpreted as follows. The first two characters are:

R A process is waiting on a msgrcv(2).

S A process is waiting on a msgsnd(2).

D The associated shared memory segment has been removed. It will disappear when the last process attached to the segment detaches it. (Note: If the shared memory segment identifier is removed via an IPC_RMID call to shmctl(2) before the process has detached from the segment with shmdt(2), the segment is no longer visible to ipcs and it will not appear in the ipcs output.)

C The associated shared memory segment is to be cleared when the first attach is executed.

- The corresponding special flag is not set.

The next nine characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

The permissions are indicated as follows:

r Read permission is granted.

w Write permission is granted.

a Alter permission is granted.

– The indicated permission is not granted.

OWNER (all) The login name of the owner of the facility entry.

GROUP (all) The group name of the group of the owner of the facility entry.

CREATOR (a,A,c) The login name of the creator of the facility entry.

CGROUP (a,A,c) The group name of the group of the creator of the facility entry.

`CBYTES` (a,A,o)
The number of bytes in messages currently outstanding on the associated message queue.

`QNUM` (a,A,o)
The number of messages currently outstanding on the associated message queue.

`QBYTES` (a,A,b)
The maximum number of bytes allowed in messages outstanding on the associated message queue.

`LSPID` (a,A,p)
The process ID of the last process to send a message to the associated queue.

`LRPID` (a,A,p)
The process ID of the last process to receive a message from the associated queue.

`STIME` (a,A,t)
The time the last message was sent to the associated queue.

`RTIME` (a,A,t)
The time the last message was received from the associated queue.

`CTIME` (a,A,t)
The time when the associated entry was created or changed.

`ISMATTCH` (a,i)
The number of ISM attaches to the associated shared memory segments.

`NATTCH` (a,A,o)
The number of processes attached to the associated shared memory segment.

`SEGSZ` (a,A,b)
The size of the associated shared memory segment.

`CPID` (a,A,p)
The process ID of the creator of the shared memory entry.

`LPID` (a,A,p)
The process ID of the last process to attach or detach the shared memory segment.

`ATIME` (a,A,t)
The time the last attach was completed to the associated shared memory segment.

`DTIME` (a,A,t)
The time the last detach was completed on the associated shared memory segment.

`NSEMS` (a,A,b)
The number of semaphores in the set associated with the semaphore entry.

`NSEMS` (a,A,b,t)	(For `/usr/xpg4/bin/ipcs`) The number of semaphores in the set associated with the semaphore entry.
`LABEL` (l)	The sensitivity label of the object.
`OTIME` (a,A,t)	The time the last semaphore operation was completed on the set associated with the semaphore entry.

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `ipcs`: `LANG`, `LC_ALL`, `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

`TZ` Determine the timezone for the time strings written by `ipcs`.

FILES

`/etc/group`	group names
`/etc/passwd`	user names
`/dev/mem`	memory
`/dev/ksyms`	system namelist

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWipc (32-bit) SUNWipcx (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

There is a new option, `–l`, for printing labels attached to an IPC object. Appropriate privilege is required to override failed access checks. For more information on required privileges, see the `IPC_STAT` option of `msgctl`(2), `semctl`(2), and `shmctl`(2).

SEE ALSO

Trusted Solaris 8 Reference Manual

`ipcrm`(1), `msgctl`(2), `msgget`(2), `msgrcv`(2), `msgsnd`(2), `semget`(2), `semop`(2), `semctl`(2), `shmctl`(2), `shmget`(2), `shmop`(2)

SunOS 5.8 Reference Manual

`savecore`(1M), `attributes`(5), `environ`(5)

NOTES

If the user specifies either the –`C` or –`N` flag, the real and effective UID / GID is set to the real UID / GID of the user invoking `ipcs`.

Things can change while `ipcs` is running; the information it gives is guaranteed to be accurate only when it was retrieved.

When the corresponding facility is not installed or has not been used since the last reboot, `/usr/xpg4/bin/ipcs` will report

```
"%s facility not in system.\n", facility
```

while `/usr/bin/ipcs` will report

```
"%s facility is inactive.\n", facility
```

where *facility* is “`Message Queue`”, “`Shared Memory`”, or “`Semaphore`”, as appropriate.

NAME

kbd – Manipulate the state of keyboard or display the type of keyboard or change the default keyboard abort sequence effect

SYNOPSIS

kbd [–r] [–t] [–c on | off][–a enable | disable][–d *keyboard device*]

kbd –i [–d *keyboard device*]

DESCRIPTION

The kbd utility manipulates the state of the keyboard, or displays the keyboard type, or allows the default keyboard abort sequence effect to be changed. The abort sequence also applies to serial console devices. The kbd utility sets the /dev/kbd default keyboard device.

EXTENDED DESCRIPTION

The –i option reads and processes default values for the keyclick and keyboard abort settings from the /etc/default/kbd keyboard default file. Only keyboards that support a clicker respond to the –c option. To turn clicking on by default, add or change the value of the KEYCLICK variable in the /etc/default/kbd file to:

```
KEYCLICK=on
```

Next, run the command kbd –i to change the setting. Valid settings for the KEYCLICK variable are on and off; all other values are ignored. If the KEYCLICK variable is not specified in the default file, the setting is unchanged.

The keyboard abort sequence effect (L1-A or STOP-A on the keyboard and BREAK on the serial console input device on most systems) can only be changed with the –a option. In the Trusted Solaris environment, this requires a process with the sys_devices privilege.. The system can be configured to ignore the keyboard abort sequence or trigger on the standard or alternate sequence.

A BREAK condition that originates from an erroneous electrical signal cannot be distinguished from one deliberately sent by remote DCE. As a remedy, use the –a option with Alternate Break to switch break interpretation. Due to the risk of incorrect sequence interpretation, binary protocols such as PPP, SLIP, and others should not be run over the serial console port when Alternate Break sequence is in effect. The Alternate Break sequence has no effect on the keyboard abort. For more information on the Alternate Break sequence, se zs(7D) , se(7D), and asy(7D).

On many systems, the default effect of the keyboard abort sequence is to suspend the operating system and enter the debugger or the monitor. Some systems feature key switches with a secure position. On these systems, setting the key switch to the secure position overrides any software default set with this command.

To permanently change the software default effect of the keyboard abort sequence, first add or change the value of the KEYBOARD_ABORT variable in the /etc/default/kbd file to:

```
KEYBOARD_ABORT=disable
```

Next, run the command kbd –i to change the setting. Valid settings are enable, disable, and alternate; all other values are ignored. If the variable is not specified in the default file, the setting is unchanged.

To set the abort sequence to the hardware BREAK, set the value of the KEYBOARD_ABORT variable in the /etc/default/kbd file to:

```
KEYBOARD_ABORT=enable
```

To change the current setting, run the command kbd –i. To set the abort sequence to the Alternate Break character sequence, first set the current value of the KEYBOARD_ABORT variable in the /etc/default/kbd file to:

```
KEYBOARD_ABORT=alternate
```

Next, run the command kbd –i to change the setting. When the Alternate Break sequence is in effect, only serial console devices are affected.

OPTIONS

The kbd utility supports the following options:

–i
Set keyboard defaults from the keyboard default file. This option is mutually exclusive with all other options except for the –d *keyboard device* option. The –i option instructs the keyboard command to read and process keyclick and keyboard abort default values from the /etc/default/kbd file. The –i option requires the sys_devices privilege.

–r
Reset the keyboard as if power-up.

–t
Return the type of the keyboard being used.

–c *on/off state*
Turn the clicking of the keyboard on or off.

on	Enable clicking.
off	Disable clicking.

–a *enable/disable state*
Enable or disable the keyboard abort sequence effect. By default, a keyboard abort sequence (typically, Stop-A or L1-A on the keyboard and BREAK on the serial console device) suspends the operating system on most systems. This default behavior can be changed using this option. The –a option requires the sys_devices privilege.

`enable` — Enable the default effect of the keyboard abort sequence, which is to suspend the operating system and enter the debugger or the monitor.

`disable` — Disable the default effect and ignore keyboard abort sequences.

`alternate` — Enable the alternate effect of the keyboard abort sequences (suspend the operating system and enter the debugger or the monitor) upon receiving the Alternate Break character sequence on the console. The Alternate Break sequence is defined by the drivers `zs`(7D), `se`(7D), `asy`(7D). Due to a risk of incorrect sequence interpretation, binary protocols cannot be run over the serial console port when this value is used.

–`d` *keyboard device* — Specify the keyboard device being set. The default is `/dev/kbd`.

EXAMPLES

EXAMPLE 1 Displaying the keyboard type

To display the keyboard type:

```
example% kbd –t
type 4 Sun keyboard
example%
```

EXAMPLE 2 Setting keyboard defaults

To set keyboard defaults as specified in the keyboard default file.

```
example$ kbd –i
example#
```

FILES

`/etc/rcS` — Shell script containing commands necessary to get the system to single-user mode.

`/dev/kbd` — Keyboard device file.

/etc/default/kbd Keyboard default file containing software defaults for keyboard configurations.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Architecture	SPARC
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

The kbd utility must have DAC access to /dev/kbd, or may use the privileges file_dac_read and file_dac_write to override access restrictions. In addition, the –a and –i options require the sys_devices privilege.

SEE ALSO

Trusted Solaris 8 Reference Manual

kb(7M)

SunOS 5.8 Reference Manual

loadkeys(1), kadb(1M), keytables(4), attributes(5), zs(7D), se(7D), asy(7D)

NOTES

Some server systems have key switches with a secure key position that can be read by system software. This key position overrides the normal default of the keyboard abort sequence effect and changes the default so the effect is disabled. When the key switch is in the secure position on these systems, the keyboard abort sequence effect cannot be overridden by the software default, which is settable with the kbd utility.

Currently, there is no way to determine the state of the keyboard click setting.

NAME

ld – link-editor for object files

SYNOPSIS

/usr/ccs/bin/ld [–64] [–a | –r][–b] [–c *name*] [–G] [–i] [–m] [–s] [–t] [–V] [–B direct] [–B dynamic | static] [–B group] [–B local] [–B eliminate] [–B reduce] [–B symbolic] [–d y | n] [–D *token*] [–e *epsym*] [–F *name* | –f *name*][–h *name*] [–I *name*] [–L *path*] [–l *x*] [–M *mapfile*] [–N *string*] [–o *outfile*] [–p *auditlib*] [–P *auditlib*] [–Q y | n] [–R *path*] [–S *supportlib*] [–u *symname*] [–Y P,*dirlist*] [–z allextract | defaultextract | weakextract] [–z combreloc] [–z defs | nodefs] [–z endfiltee] [–z ignore | record] [–z lazyload | nolazyload] [–z initfirst] [–z interpose] [–z loadfltr] [–z muldefs] [–z nodelete] [–z nodefaultlib] [–z nodlopen] [–z nodump] [–z nopartial] [–z noversion] [–z now] [–z origin] [–z redlocsym] [–z text | textwarn | textoff] *filename*…

DESCRIPTION

The ld command combines relocatable object files, performs relocation, and resolves external symbols. ld operates in two modes, static or dynamic, as governed by the –d option. In static mode, –dn, relocatable object files given as arguments are combined to produce an executable object file. If the –r option is specified, relocatable object files are combined to produce one relocatable object file. In dynamic mode, –dy, the default, relocatable object files given as arguments are combined to produce an executable object file that will be linked at execution with any shared object files given as arguments. If the –G option is specified, relocatable object files are combined to produce a shared object. In all cases, the output of ld is left in a.out by default.

If any argument is a library, ld searches exactly once at the point it encounters the library in the argument list. The library may be either a relocatable archive or a shared object. For an archive library, ld loads only those routines that define an unresolved external reference. ld searches the archive library symbol table sequentially with as many passes as are necessary to resolve external references that can be satisfied by library members. See ar(3HEAD). Thus, the order of members in the library is functionally unimportant, unless multiple library members exist that define the same external symbol.

A shared object consists of an indivisible, whole unit, that has been generated by a previous link-edit of one or more input files. When the link-editor processes a shared object, the entire contents of the shared object become a logical part of the resulting output file image. The shared object is not physically copied during the link-edit as its actual inclusion is deferred until process execution. This logical inclusion means that all symbol entries defined in the shared object are made available to the link-editing process.

There is no specific option that tells `ld` to link 64-bit objects; the class of the first object that gets processed by `ld` determines whether it is to perform a 32-bit or a 64-bit link edit.

OPTIONS

The following options are supported:

`–64`
Creates a 64-bit object. By default the class of the object being generated is determined from the first ELF object processed from the command line. This option is useful when creating an object directly with `ld` whose input is solely from a `mapfile` (see the `–M` option).

`–a`
In static mode only, produces an executable object file; gives errors for undefined references. This is the default behavior for static mode. `–a` may not be used with the `–r` option.

`–b`
In dynamic mode only, when creating an executable, does not do special processing for relocations that reference symbols in shared objects. Without the `–b` option, the link-editor creates special position-independent relocations for references to functions defined in shared objects and arranges for data objects defined in shared objects to be copied into the memory image of the executable by the runtime linker.

The `–b` option is intended for specialized applications and is not recommended for general use. Its use suppresses all specialized processing required to insure an application's shareability, and may even prevent the relocation of 64–bit applications. An application's shareability can always be guaranteed by building it with the use of position-independent code.

`–B direct`
Establishes direct binding information by recording the relationship between each symbol reference and the dependency that provides the definition. The runtime linker uses this information to search directly for the symbol in the associated object rather than to carry out its default symbol search. Direct binding information can only be established to dependencies specified with the link-edit. Thus, you should use the `–z defs` option. Objects that wish to interpose on symbols in a direct binding environment should identify themselves as interposers with the `–z interpose` option. The use of `–B direct` enables `–z lazyload` for all dependencies.

`–B dynamic | static`
Options governing library inclusion. `–B dynamic` is valid in dynamic mode only. These options may be specified any number of times on the command line as toggles: if the `–B static` option is given, no shared objects will be accepted until `–B dynamic` is seen. See also the `–l` option.

`–B eliminate`
Causes any global symbols not assigned to a version definition to be eliminated from the symbol table. This option achieves the same symbol elimination as the *auto-elimination* directive available as part of a *mapfile* version definition.

`–B group`
Establishes a shared object and its dependencies as a group. Objects within the group will be bound to other members of the group at runtime. The runtime processing of an object containing this flag mimics that which occurs if the object is added to a process using `dlopen`(3DL) with the `RTLD_GROUP` mode. As the group must be self contained, use of the `–B group` option also asserts the `–z defs` option.

`–B local`
Causes any global symbols, not assigned to a version definition, to be reduced to local. Version definitions can be supplied via a *mapfile* and indicate the global symbols that should remain visible in the generated object. This option achieves the same symbol reduction as the *auto-reduction* directive available as part of a *mapfile* version definition and may be useful when combining versioned and non-versioned relocatable objects.

`–B reduce`
When generating a relocatable object, causes the reduction of symbolic information defined by any version definitions. Version definitions can be supplied via a *mapfile* to indicate the global symbols that should remain visible in the generated object. When a relocatable object is generated, by default version definitions are only recorded in the output image. The actual reduction of symbolic information will be carried out when the object itself is used in the construction of a dynamic executable or shared object. This option is applied automatically when dynamic executable or shared object is created.

`–B symbolic`
In dynamic mode only. When building a shared object, binds references to global symbols to their definitions, if available, within the object. Normally, references to global symbols within shared objects are not bound until runtime, even if definitions are available, so that definitions of the same symbol in an executable or other shared object can override the object's own definition. `ld` will issue warnings for undefined symbols unless `–z defs` overrides.

The `–B symbolic` option is intended for specialized applications and is not recommended for general use. To reduce the runtime relocation overhead of an object, the creation of a version definition is recommended.

–c *name*
Records the configuration file *name* for use at runtime. Configuration files may be employed to alter default search paths, provide a directory cache and provide alternative object dependencies. See crle(1).

–d y | n
When –d y, the default, is specified, ld uses dynamic linking; when –d n is specified, ld uses static linking. See also –B dynamic|static.

–D *token,token,. .*
Prints debugging information, as specified by each *token*, to the standard error. The special token help indicates the full list of tokens available.

–e *epsym*
Sets the entry point address for the output file to be that of the symbol *epsym*.

–f *name*
Useful only when building a shared object. Specifies that the symbol table of the shared object is used as an auxiliary filter on the symbol table of the shared object specified by *name*. Multiple instances of this option are allowed. This option may not be combined with the –F option.

–F *name*
Useful only when building a shared object. Specifies that the symbol table of the shared object is used as a filter on the symbol table of the shared object specified by *name*. Multiple instances of this option are allowed. This option may not be combined with the –f option.

–G
In dynamic mode only, produces a shared object. Undefined symbols are allowed.

–h *name*
In dynamic mode only, when building a shared object, records *name* in the object's dynamic section. *name* will be recorded in executables that are linked with this object rather than the object's UNIX System file name. Accordingly, *name* will be used by the runtime linker as the name of the shared object to search for at runtime.

–i
Ignores LD_LIBRARY_PATH. This option is useful when an LD_LIBRARY_PATH setting is in effect to influence the runtime library search, which would interfere with the link-editing being performed.

–I *name*
When building an executable, uses *name* as the path name of the interpreter to be written into the program header. The default in static mode is no

interpreter; in dynamic mode, the default is the name of the runtime linker, `ld.so.1`(1). Either case may be overridden by `–I` *name*. `exec`(2) will load this interpreter when it loads `a.out` and will pass control to the interpreter rather than to `a.out` directly.

`–l` *x*

Searches a library `lib`*x*`.so` or `lib`*x*`.a`, the conventional names for shared object and archive libraries, respectively. In dynamic mode, unless the `–B static` option is in effect, `ld` searches each directory specified in the library search path for a `lib`*x*`.so` or `lib`*x*`.a` file. The directory search stops at the first directory containing either. `ld` chooses the file ending in `.so` if `–l`*x* expands to two files with names of the form `lib`*x*`.so` and `lib`*x*`.a.` If no `lib`*x*`.so` is found, then `ld` accepts `lib`*x*`.a.` In static mode, or when the `–B static` option is in effect, `ld` selects only the file ending in `.a`. `ld` searches a library when it encounters its name, so the placement of `–l` is significant.

`–L` *path*

Adds *path* to the library search directories. `ld` searches for libraries first in any directories specified by the `–L` options and then in the standard directories. This option is useful only if it precedes the `–l` options to which it applies on the command line. The environment variable `LD_LIBRARY_PATH` may be used to supplement the library search path (see `LD_LIBRARY_PATH` below).

`–m`

Produces a memory map or listing of the input/output sections, together with any non-fatal multiply-defined symbols, on the standard output.

`–M` *mapfile*

Reads *mapfile* as a text file of directives to `ld`. This option may be specified multiple times. If *mapfile* is a directory, then all regular files, as defined by `stat`(2), within the directory will be processed. See *Linker and Libraries Guide* for a description of mapfiles. There are mapfiles in `/usr/lib/ld` that show the default layout of programs as well as mapfiles for linking 64–bit programs above or below 4 gigabytes. See the FILES section below.

`–N` *string*

This option causes a `DT_NEEDED` entry to be added to the *.dynamic* section of the object being built. The value of the `DT_NEEDED` string will be the *string* specified on the command line. This option is position dependent, and the `DT_NEEDED` *.dynamic* entry will be relative to the other dynamic dependencies discovered on the link-edit line.

`–o` *outfile*

Produces an output object file named *outfile*. The name of the default object file is `a.out`.

–p *auditlib*

Identifies an audit library, *auditlib*, that is used to audit this object at runtime. Any shared object identified as requiring auditing of itself has this requirement inherited by any object specifying this shared object as a dependency (see –P option).

–P *auditlib*

Identifies an audit library, *auditlib*, that is used to audit this object's dependencies at runtime. Dependency auditing can also be inherited from dependencies identified as requiring auditing (see –p option).

–Q y | n

Under –Q y, an ident string is added to the *.comment* section of the output file to identify the version of the link-editor used to create the file. This results in multiple ld idents when there have been multiple linking steps, such as when using ld –r. This is identical with the default action of the cc command. –Q n suppresses version identification.

–r

Combines relocatable object files to produce one relocatable object file. ld will not complain about unresolved references. This option cannot be used in dynamic mode or with –a.

–R *path*

A colon-separated list of directories used to specify library search directories to the runtime linker. If present and not NULL, it is recorded in the output object file and passed to the runtime linker. Multiple instances of this option are concatenated together with each *path* separated by a colon.

–s

Strips symbolic information from the output file. Any debugging information, that is *.debug*, *.line*, and *.stab* sections, and their associated relocation entries will be removed. Except for relocatable files or shared objects, the symbol table and string table sections will also be removed from the output object file.

–S *supportlib*

The shared object *supportlib* is loaded with the link-editor and given information regarding the linking process. Support shared objects may also be supplied using the SGS_SUPPORT environment variable. See *Linker and Libraries Guide* for more details.

–t

Turns off the warning for multiply-defined symbols that have different sizes or alignments.

–u *symname*

Enters *symname* as an undefined symbol in the symbol table. This is useful for loading entirely from an archive library, since initially the symbol table is empty, and an unresolved reference is needed to force the loading of the first routine. The placement of this option on the command line is significant; it must be placed before the library that will define the symbol.

–V

Outputs a message giving information about the version of `ld` being used.

–Y `P,`*dirlist*

Changes the default directories used for finding libraries. *dirlist* is a colon-separated path list.

–z `allextract` | `defaultextract` | `weakextract`

Alters the extraction criteria of objects from any archives that follow. By default, archive members are extracted to satisfy undefined references and to promote tentative definitions with data definitions. Weak symbol references do not trigger extraction. Under –z `allextract`, all archive members are extracted from the archive. Under –z `weakextract`, weak references trigger archive extraction. –z `defaultextract` provides a means of returning to the default following use of the former extract options.

–z `combreloc`

Combines multiple relocation sections. Reduces overhead when objects are loaded into memory.

–z `defs`

Forces a fatal error if any undefined symbols remain at the end of the link. This is the default when an executable is built. It is also useful when building a shared object to assure that the object is self-contained, that is, that all its symbolic references are resolved internally.

–z `endfiltee`

Marks a filtee so that when processed by a filter it terminates any further filtee searches by the filter.

–z `ignore` | `record`

Ignores, or records, dynamic dependencies that are not referenced as part of the link-edit. By default, –z `record` is in effect.

–z `initfirst`

Marks the object so that its runtime initialization occurs before the runtime initialization of any other objects brought into the process at the same time. In addition, the object runtime finalization will occur after the runtime finalization of any other objects removed from the process at the same time. This option is only meaningful when building a shared object.

–z `interpose`

Marks the object as an interposer. When direct bindings are in effect (see –B direct), the runtime linker will search for symbols in any interposers before the object associated to the direct binding.

–z lazyload | nolazyload
Enables or disables the marking of dynamic dependencies to be lazily loaded. Dynamic dependencies which are marked lazyload will not be loaded at initial process start-up, but instead will be delayed until the first binding to the object is made.

–z loadfltr
Marks the object to require that when building a filter, its filtees be processed immediately at runtime. Normally, filter processing is delayed until a symbol reference is bound to the filter. The runtime processing of an object that contains this flag mimics that which occurs if the LD_LOADFLTR environment variable is in effect. See ld.so.1(1).

–z muldefs
Allows multiple symbol definitions. By default, multiple symbol definitions that occur between relocatable objects will result in a fatal error condition. This option suppresses the error condition and allows the first symbol definition to be taken.

–z nodefs
Allows undefined symbols. This is the default when a shared object is built. When used with executables, the behavior of references to such undefined symbols is unspecified.

–z nodelete
Marks the object as non-deletable at runtime. The runtime processing of an object that contains this flag mimics that which occurs if the object is added to a process using dlopen(3DL) with the RTLD_NODELETE mode.

–z nodefaultlib
Marks the object so that the runtime default library search path (used after any LD_LIBRARY_PATH or *runpaths*) is ignored. This option implies that all dependencies of the object can be satisfied from its *runpath.*

–z nodlopen
Marks the object as not available to dlopen(3DL), either as the object specified by the dlopen( ), or as any form of dependency required by the object specified by the dlopen( ). This option is only meaningful when building a shared object.

–z nodump
Marks the object as not available to dldump(3DL).

–z nopartial

If there are any partially initialized symbols in the input relocatable object files, the partially initialized symbols are expanded when the output file is generated.

−z noversion
Does not record any versioning sections. Any version sections or associated .*dynamic* section entries will not be generated in the output image.

−z now
Marks the object to override the runtime linker's default mode and require non-lazy runtime binding. This is similar to adding the object to the process by using dlopen(3DL) with the RTLD_NOW mode, or setting the LD_BIND_NOW environment variable in effect. See ld.so.1(1).

−z origin
Marks the object as requiring immediate $ORIGIN processing at runtime.

−z redlocsym
Eliminates all local symbols except for the SECT symbols from the symbol table SHT_SYMTAB. All relocations that refer to local symbols will be updated to refer to the corresponding SECT symbol.

−z text
In dynamic mode only, forces a fatal error if any relocations against non-writable, allocatable sections remain.

−z textoff
In dynamic mode only, allows relocations against all allocatable sections, including non-writable ones. This is the default when building a shared object.

−z textwarn
In dynamic mode only, lists a warning if any relocations against non-writable, allocatable sections remain. This is the default when building an executable.

ENVIRONMENT VARIABLES

LD_LIBRARY_PATH
A list of directories in which to search for libraries specified with the −l option. Multiple directories are separated by a colon. In the most general case, it will contain two directory lists separated by a semicolon:

dirlist1;*dirlist2*

If ld is called with any number of occurrences of −L, as in:

`ld` ... −L*path1* ... −L*pathn* ...

then the search path ordering is:

dirlist1 path1 ... *pathn dirlist2* `LIBPATH`

When the list of directories does not contain a semicolon, it is interpreted as *dirlist2*.

The `LD_LIBRARY_PATH` environment variable also affects the runtime linkers searching for dynamic dependencies.

Note: When running a privileged program, set-user-ID or set-group-ID program, the runtime linker will only search for libraries in any full pathname specified within the executable as a result of a runpath being specified when the executable was constructed, or in a trusted directory [see `crle`(1)].

`LD_LIBRARY_PATH_64`
Similar to the `LD_LIBRARY_PATH` environment variable. Overrides `LD_LIBRARY_PATH` when searching for 64-bit dependencies.

`LD_OPTIONS`
A default set of options to `ld`. `LD_OPTIONS` is interpreted by `ld` just as though its value had been placed on the command line, immediately following the name used to invoke `ld`, as in:

`ld $LD_OPTIONS` ... *other-arguments* ...

`LD_RUN_PATH`
An alternative mechanism for specifying a runpath to the link-editor (see −`R` option). If both `LD_RUN_PATH` and the −`R` option are specified, −`R` supersedes.

`SGS_SUPPORT`
Provides a colon separated list of shared objects that are loaded with the link-editor and given information regarding the linking process. See also the −`S` option.

Notice that environment variable-names beginning with the characters '`LD_`' are reserved for possible future enhancements to `ld` and `ld.so.1`(1).

SUMMARY OF TRUSTED SOLARIS CHANGES

For a privileged program, all runtime dependencies must resolve to trusted directories [see `crle`(1)].

FILES

`lib`*x*`.so`	libraries
`lib`*x*`.a`	libraries
`a.out`	output file
LIBPATH	usually `/usr/lib` or `/usr/lib/sparcv9` for 64-bit SPARCV9 libraries.

`/usr/lib/ld/map.default`
mapfile showing default layout of 32-bit programs

`/usr/lib/ld/sparcv9/map.default`
mapfile showing default layout of 64-bit SPARCV9 programs

`/usr/lib/ld/sparcv9/map.above4G`
mapfile showing suggested layout above 4 gigabytes of 64-bit SPARCV9 programs

`/usr/lib/ld/sparcv9/map.below4G`
mapfile showing suggested layout below 4 gigabytes of 64-bit SPARCV9 programs

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtoo

SEE ALSO

Trusted Solaris 8 Reference Manual

`crle`(1), `exec`(2), `stat`(2)

SunOS 5.8 Reference Manual

`as`(1), `gprof`(1), `ld.so.1`(1), `pvs`(1), `dlopen`(3DL), `dldump`(3DL), `elf`(3ELF), `a.out`(4), `ar`(3HEAD), `attributes`(5)

Linker and Libraries Guide, *Binary Compatibility Guide*

NAME

login – Sign on to the system

SYNOPSIS

login [–p] [–d *device*] [–h *hostname* | [*terminal*] | –r *hostname* | [–T] | [–U *uid*]][*name* [*environ*] …]

DESCRIPTION

The login command is used at the beginning of each terminal session to identify oneself to the system. login is invoked by the system when a connection is first established, after the previous user has terminated the login shell by issuing the exit command.

If login is invoked as a command, it must replace the initial command interpreter. To invoke login in this fashion, type:

```
exec login
```

from the initial shell. The C shell and Korn shell have their own builtins of login. See ksh(1) and csh(1) for descriptions of login builtins and usage.

login asks for your user name if it is not supplied as an argument, and your password, if appropriate. Where possible, echoing is turned off while you type your password, so it will not appear on the written record of the session.

If you make any mistake in the login procedure, the message:

```
Login incorrect
```

is printed and a new login prompt will appear. If you make five incorrect login attempts, all five may be logged in /var/adm/loginlog, if it exists. The TTY line will be dropped.

If password aging is turned on and the password has "aged" (see passwd(1) for more information), login is denied with a message to use the desktop to log in and change the password.

After a successful login, accounting files are updated. Device owner, group, and permissions are set according to the contents of the /etc/logindevperm file, and the time you last logged in is printed (see logindevperm(4)).

Except for remote logins, login asks you to select the sensitivity label (SL) at which you will operate for this terminal session. You must enter a label that you are authorized to use and that is valid for the device.

The user-ID, group-ID, supplementary group list, and working directory are initialized, and the command interpreter (usually ksh) is started.

The basic *environment* is initialized to:

```
HOME=your-login-directory
LOGNAME=your-login-name
PATH=/usr/bin:
SHELL=last-field-of-passwd-entry
MAIL=/var/mail/TZ=timezone-specification
```

For Bourne shell and Korn shell logins, the shell executes `/etc/profile` and `$HOME/.profile`, if it exists. For C shell logins, the shell executes `/etc/.login`, `$HOME/.cshrc`, and `$HOME/.login`. The default `/etc/profile` and `/etc/.login` files check quotas (see `quota`(1M)), print `/etc/motd`, and check for mail. None of the messages are printed if the file `$HOME/.hushlogin` exists. The name of the command interpreter is set to – (dash), followed by the last component of the interpreter's path name, for example, `–sh`.

If the *login-shell* field in the password file (see `passwd`(4)) is empty, then the default command interpreter, `/usr/bin/sh`, is used. If this field is * (asterisk), then the named directory becomes the root directory. At that point, `login` is re-executed at the new level, which must have its own root structure.

The environment may be expanded or modified by supplying additional arguments to `login`, either at execution time or when `login` requests your login name. The arguments may take either the form *xxx* or *xxx=yyy*. Arguments without an = (equal sign) are placed in the environment as:

`L`*n=xxx*

where *n* is a number starting at `0` and is incremented each time a new variable name is required. Variables containing an = (equal sign) are placed in the environment without modification. If they already appear in the environment, then they replace the older values.

There are two exceptions: The variables `PATH` and `SHELL` cannot be changed. This prevents people logged into restricted shell environments from spawning secondary shells that are not restricted. `login` understands simple single-character quoting conventions. Typing a `\` (backslash) in front of a character quotes it and allows the inclusion of such characters as spaces and tabs.

Alternatively, you can pass the current environment by supplying the `–p` flag to `login`. This flag indicates that all currently defined environment variables should be passed, if possible, to the new environment. This option does not bypass any environment variable restrictions mentioned above. Environment variables specified on the login line take precedence, if a variable is passed by both methods.

To enable remote logins by administrative users (that is, administrative roles), edit the /etc/default/login file by inserting a pound sign (#) before the CONSOLE=/dev/console entry. See FILES.

SECURITY

The login command uses pam(3PAM) for authentication, account management, session management, and password management. The PAM configuration policy, listed through /etc/pam.conf, specifies the modules to be used for login. Here is a partial pam.conf file with entries for the login command using the UNIX authentication, account management, session management, and password management module.

```
login  auth       required  /usr/lib/security/pam_unix.so.1
login  account    required  /usr/lib/security/pam_unix.so.1
login  session    required  /usr/lib/security/pam_unix.so.1
login  password   required  /usr/lib/security/pam_unix.so.1
```

When login is invoked through rlogind or telnetd, the service name used by PAM is rlogin or telnet respectively.

OPTIONS

The following options are supported:

–d *device*
login accepts a device option, *device*. *device* is taken as the path name of the TTY port on which login is to operate. The use of the device option can be expected to improve login performance because login will not need to call ttyname(3C).

–h *hostname* [*terminal*]
Used by in.telnetd(1M) to pass information about the remote host and terminal type.

–p
Used to pass environment variables to the login shell.

–r *hostname*
Used by in.rlogind(1M) to pass information about the remote host.

–T
in.rlogind(1M) uses this option to indicate that the trusted path process attribute is set on the remote host for the process invoking rlogin.

–U *uid*	`in.rlogind`(1M) uses this option to pass information about the UID of the invoker of `rlogin`. If uid and *name* are both passed by `in.rlogind`(1M), the UID of *name* must match the *uid* value or login is denied.

EXIT STATUS

Upon success, `login` returns `0`. Upon failure, `login` returns a nonzero value.

FILES

`$HOME/.cshrc`	initial commands for each csh
`$HOME/.hushlogin`	suppresses login messages
`$HOME/.login`	user’s login commands for csh
`$HOME/.profile`	user’s login commands for sh and ksh
`$HOME/.rhosts`	private list of trusted hostname/username combinations
`/etc/.login`	system-wide csh login commands
`/etc/logindevperm`	login-based device permissions
`/etc/motd`	message-of-the-day
`/etc/nologin`	message displayed to users attempting to login during machine shutdown
`/etc/passwd`	password file
`/etc/profile`	system-wide sh and ksh login commands
`/etc/shadow`	list of users’ encrypted passwords
`/usr/bin/sh`	user’s default command interpreter
`/var/adm/lastlog`	time of last login
`/var/adm/loginlog`	record of failed login attempts
`/var/adm/utmpx`	accounting
`/var/adm/wtmpx`	accounting
`/var/mail/`*your-name*	mailbox for user *your-name*

`/etc/default/login`

Default value can be set for the following flags in `/etc/default/login`. For example: `TIMEZONE=EST5EDT`

`TIMEZONE`	Sets the `TZ` environment variable of the shell (see `environ`(5)).

`HZ` Sets the `HZ` environment variable of the shell.

`ULIMIT` Sets the file size limit for the login. Units are disk blocks. Default is zero (no limit).

`CONSOLE` If this flag is set, administrative users can log in only on that device. This setting will not prevent execution of remote commands with `rsh`(1). Comment out this line to allow login by administrative users.

`PASSREQ` Determines if login requires a non-null password.

`ALTSHELL` Determines if login should set the `SHELL` environment variable.

`PATH` Sets the initial shell `PATH` variable.

`SUPATH` Sets the initial shell `PATH` variable for root.

`TIMEOUT` Sets the number of seconds (between `0` and `900`) to wait before abandoning a login session.

`UMASK` Sets the initial shell file creation mode mask. See `umask`(1).

`SYSLOG` Determines whether the `syslog`(3C) `LOG_AUTH` facility should be used to log all root logins at level `LOG_NOTICE` and multiple failed login attempts at`LOG_CRIT`.

`SLEEPTIME` If present, sets the number of seconds to wait before login failure is printed to the screen and another login attempt is allowed. Default is `4` seconds. Minimum is `0` seconds. Maximum is `5` seconds.

`RETRIES` Sets the number of retries for logging in (see `pam`(3PAM)). The default is 5.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

You are prompted to select the label for your session at login time (except for remote login). Restrictions on labels and UIDs apply. The `DESCRIPTION` section explains these restrictions. The Trusted Solaris environment adds two options: `–T` and `–U`. (See `OPTIONS`.)

SEE ALSO

Trusted Solaris 8 Reference Manual

passwd(1), in.rlogind(1M)

SunOS 5.8 Reference Manual

csh(1), exit(1), ksh(1), mail(1), mailx(1), newgrp(1), rlogin(1), rsh(1), sh(1), shell_builtins(1), telnet(1), umask(1), in.telnetd(1M), logins(1M), su(1M), syslogd(1M), useradd(1M), userdel(1M), pam(3PAM), rcmd(3SOCKET), syslog(3C), ttyname(3C), hosts.equiv(4), logindevperm(4), loginlog(4), nologin(4), pam.conf(4), passwd(4), profile(4), shadow(4), utmpx(4), wtmpx(4), attributes(5), environ(5), pam_unix(5), termio(7I)

DIAGNOSTICS

`Login incorrect`

The user name or the password cannot be matched.

`Not on system console`

Administrative user login denied. Check the `CONSOLE` setting in `/etc/default/login`.

`No directory!  Logging in with home=/`

The user’s home directory named in the passwd(4) database cannot be found or has the wrong permissions. Contact your system administrator.

`No shell`

Cannot execute the shell named in the passwd(4) database. Contact your system administrator.

`NO LOGINS: System going down in` *N* `minutes`

The machine is in the process of being shut down and logins have been disabled.

WARNINGS

Users with a UID greater than 76695844 are not subject to password aging, and the system does not record their last login time.

If you use the `CONSOLE` setting to disable administrative user logins, make sure that remote command execution by administrative users is also disabled. See rsh(1), rcmd(3SOCKET), and hosts.equiv(4) for further details.

NAME

lp – Submit print request

SYNOPSIS

lp [–c] [–m] [–p] [–s] [–w] [–d *destination*] [–f *form-name*] [–H *special-handling*] [–n *number*] [–o *option*] [–P *page-list*] [–q *priority-level*] [–S *character-set* | *print-wheel*][–t *title*] [–T *content-type* [–r]] [–y *mode-list*] [*file*…]

lp –i *request-ID*… [–c] [–m] [–p] [–s] [–w] [–d *destination*] [–f *form-name*] [–H *special-handling*] [–n *number*] [–o *option*] [–P *page-list*] [–q *priority-level*] [–S *character-set* | *print-wheel*][–t *title*] [–T *content-type* [–r]] [–y *mode-list*]

DESCRIPTION

lp submits print requests to a destination. There are two formats of the lp command.

The first form of lp prints files (*file*) and associated information (collectively called a *print request*). If *file* is not specified, lp assumes the standard input. Use a hyphen (‘–’) with *file* to specify the standard input. Files are printed in the order in which they appear on the command line.

The LP print service associates a unique *request-ID* (with the –i option) with each request and displays it on the standard output. This *request-ID* can be used later with the –i option when canceling or changing a request, or when determining its status. (See the section on cancel for details about canceling a request, and lpstat(1) for information about checking the status of a print request.)

The second form of lp changes print request options. The print request identified by *request-ID* is changed according to the printing options specified. The printing options available are the same as those with the first form of lp. If the request has finished printing when the lp command is executed, the change is rejected. If the request is in the process of printing, it is stopped and restarted from the beginning (unless the –P option has been given).

The print client commands locate destination information in a specific order. See printers(4) and printers.conf(4) for details.

OPTIONS

Printers that have a 4.*x* or BSD-based print server are not configured to handle BSD protocol extensions. lp handles print requests sent to such destinations differently (see NOTES).

The following options are supported:

–c
: Copies *file* before printing.

  This option has no effect in the Trusted Solaris environment.

–d *dest*
: Choose *dest* as the printer or class of printers that is to do the printing. If *dest* is a printer, then the request will be printed only on that specific printer. If *dest* is a class of printers, then the request will be printed on the

first available printer that is a member of the class. If *dest* is `any`, then the request will be printed on any printer which can handle it. Under certain conditions, (unavailability of printers, file space limitations, and so on) requests for specific destinations may not be accepted (see `lpstat`(1)). By default, *dest* is taken from the environment variable `LPDEST` (if it is set). Otherwise, a default destination (if one exists) for the computer system is used. Destination names vary between systems (see `lpstat`(1)).

–`f` *form-name*
Prints *file* on *form-name*. The LP print service ensures that the form is mounted on the printer. The print request is rejected if the printer does not support *form-name*, if *form-name* is not defined for the system, or if the user is not allowed to use *form-name* (see `lpforms`(1M)). When the –`d any` option is given, the request is printed on any printer that has the requested form mounted and can handle all other needs of the print request.

–`H` *special-handling*
Prints the print request according to the value of *special-handling*. The following *special-handling* values are acceptable:

`hold`	Do not print the print request until notified. If printing has already begun, stop it. Other print requests will go ahead of a request that has been put on hold (*held print request*) until the print request is resumed.
`resume`	Resume a held print request. If the print request had begun to print when held, it will be the next print request printed, unless it is superseded by an `immediate` print request.
`immediate`	Print the print request next. If more than one print request is assigned, the most recent print request is printed next. If a print request is currently printing on the desired printer, a `hold` request must be issued to allow the immediate request to print. The `immediate` request is only available to LP administrators.

–`m`
Sends mail after *file* has printed (see `mail`(1)). By default, no mail is sent upon normal completion of a print request.

–`n` *number*
Prints a specific number of copies of *file*. Specify *number* as a digit. The default for *number* is `1`.

–`o` *option*

Specify printer-dependent *options*. Specify several options by specifying `–o` *option* multiple times. (`–o` *option* `–o` *option* `–o` *option*). Printer-dependent options may also be specified using the `–o` keyletter once, followed by a list of options enclosed in double quotes (`–o` " *option option option*"). The following options are valid:

`nobanner`

Do not print a banner page or a trailer page with this request. This option can be disallowed by the LP administrator. Use of this option requires the `print without banner` authorization.

`nofilebreak`

Prints multiple files without inserting a form feed between them.

`nolabels`

Prints this request without page header and footer labels. Use of this option requires the `print without labels` authorization.

`length`=*scaled-decimal-number*

Print this request with pages *scaled-decimal-number* lines long. A *scaled-decimal-number* is an optionally scaled decimal number that gives a size in lines, columns, inches, or centimeters, as appropriate. The scale is indicated by appending the letter “i” for inches, or the letter “c” for centimeters. For length or width settings, an unscaled number indicates lines or columns; for line pitch or character pitch settings, an unscaled number indicates lines per inch or characters per inch (the same as a number scaled with “i”). For example, `length=66` indicates a page length of `66` lines, `length=11i` indicates a page length of `11` inches, and `length=27.94c` indicates a page length of `27.94` centimeters. This option may not be used with the `–f` option.

`width`=*scaled-decimal-number*

Print this request with page-width set to *scaled-decimal-number* columns wide. (See the explanation of *scaled-decimal-numbers* in the discussion of `length`, above.) This option may not be used with the `–f` option.

`lpi`=*scaled-decimal-number*

Print this request with the line pitch set to *scaled-decimal-number* lines per inch. This option may not be used with the `–f` option.

`cpi`=*scaled-decimal-number*

Print this request with the character pitch set to *scaled-decimal-number* characters per inch. Character pitch can also be set to `pica` (representing `10` characters per inch) or `elite` (representing `12` characters per inch), or it can be `compressed` (representing as many characters as a printer can handle). There is no standard number of characters per inch for all printers; see the Terminfo database (see `terminfo`(4)) for the default character pitch for your printer. This option may not be used with the –`f` option.

`stty=`*stty-option-list*

Prints the request using a list of options valid for the `stty` command (see `stty`(1). Enclose the list in single quotes (`''`) if it contains blanks.

–`P` *page-list*

Prints the pages specified in *page-list* in ascending order. Specify *page-list* as a of range of numbers, single page number, or a combination of both.

–`P` can only be used if there is a filter available to handle it; otherwise, the print request will be rejected.

–`p`

Enables notification on completion of the print request. Delivery of the notification is dependent on additional software.

–`q` *priority-level*

Assigns the print request a priority in the print queue. Specify *priority-level* as an integer between from `0` and `39`. Use `0` to indicate the highest priority; `39` to indicate the lowest priority. If no priority is specified, the default priority for a print service is assigned by the LP administrator. The LP administrator may also assign a default priority to individual users.

–`s`

Suppresses the display of messages sent from `lp`.

–`S` *character-set* **|** *print-wheel*

Prints the request using the *character-set* or *print-wheel.* If a form was requested and requires a character set or print wheel other than the one specified with the –`S` option, the request is rejected. Printers using mountable print wheels or font cartridges use the print wheel or font cartridge mounted at the time of the print request, unless the –`S` option is specified.

Printers Using Print Wheels: If `print` *wheel* is not one listed by the LP administrator as acceptable for the printer the request is rejected unless the print wheel is already mounted on the printer.

Printers Using Selectable or Programmable Character Sets: If the –`S` option is not specified, `lp` uses the standard character set. If *character-set* is not defined in the `terminfo` database for the printer (see `terminfo`(4)), or is not an alias defined by the LP administrator, the request is rejected.

`–t` *title*
Prints a title on the banner page of the output. Enclose *title* in quotes if it contains blanks. If *title* is not not specified, the name of the file is printed on the banner page.

`–T` *content-type* [`–r`]
Prints the request on a printer that can support the specified *content-type*. If no printer accepts this type directly, a filter will be used to convert the content into an acceptable type. If the `–r` option is specified, a filter will not be used. If `–r` is specified, and no printer accepts the *content-type* directly, the request is rejected. If the *content-type* is not acceptable to any printer, either directly or with a filter, the request is rejected.

Submitting a request with the "postscript" type requires the `print a Postscript file` authorization, whether or not –`T` is used.

`–w`
Writes a message on the user's terminal after the *files* have been printed. If the user is not logged in, then mail will be sent instead.

`–y` *mode-list*
Prints the request according to the printing modes listed in *mode-list*. The allowed values for *mode-list* are locally defined.

This option may be used only if there is a filter available to handle it; otherwise, the print request will be rejected.

OPERANDS

The following operands are supported:

file The name of the file to be printed. Specify *file* as a pathname or as a hyphen (`'–'`) to indicate the standard input. If *file* is not specified, `lp` uses the standard input.

USAGE

See `largefile`(5) for the description of the behavior of `lp` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
`non-zero`	An error occurred.

FILES

`/var/spool/lp/*`	LP print queue.
`$HOME/.printers`	User-configurable printer database.
`/etc/printers.conf`	System printer configuration database.
`printers.conf.byname`	NIS version of `/etc/printers.conf`.

`fns.ctx_dir.`*domain* NIS+ version of `/etc/printers.conf`.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu
CSI	Enabled (see `NOTES`)

SUMMARY OF TRUSTED SOLARIS CHANGES

The `–c` option is accepted but is ignored; a copy of the file is always made before printing. The `–o nobanner` option requires the `print without banners` authorization. The `–o nolabels` option is added. Submitting a request with the "postscript" type requires the `print a PostScript file` authorization.

SEE ALSO

Trusted Solaris 8 Reference Manual

`cancel`(1), `enable`(1), `lpstat`(1), `lpq`(1B), `lprm`(1B), `accept`(1M), `lpadmin`(1M), `lpfilter`(1M), `lpforms`(1M), `lpsched`(1M), `lpshut`(1M), `lpsystem`(1M), `lpusers`(1M)

SunOS 5.8 Reference Manual

`mail`(1), `postprint`(1), `pr`(1), `stty`(1), `printers`(4), `printers.conf`(4), `terminfo`(4), `attributes`(5), `environ`(5), `largefile`(5), `standards`(5)

NOTES

CSI-capability assumes that printer names are composed of ASCII characters.

Printers that have a 4.*x* or BSD-based print server. are not configured to handle BSD protocol extensions. `lp` handles print requests sent to such printers in the following ways:

1. Print requests with more than 52 filenames will be truncated to 52 files. `lp` displays a warning message.
2. The `–f`, `–H`, `–o`, `–P`, `–p`, `–q`, `–S`, `–T`, and `–y` options may require a protocol extension to pass to a print server. If `lp` cannot handle the print request, it displays a warning message.

   LP administrators enable protocol extensions by setting a printer's `bsdaddr` entry in `/etc/printers.conf`. Changing the `bsdaddr` entry in `/etc/printers.conf` to:

   destination`:bsdaddr=`*server*`,`*destination*`,Solaris`

   generates a set of BSD print protocol extensions that can be processed by a Solaris print server. `lp` supports only `Solaris` protocol extensions at this time.

NAME

lpc – Line printer control program

SYNOPSIS

/usr/ucb/lpc [*command* [*parameter*…]]

DESCRIPTION

The `lpc` utility controls the operation of printers.

Use `lpc` to perform the following functions:

- start or stop a printer
- disable or enable a printer’s spooling queue
- rearrange the order of jobs in a print queue
- display the status of a printer’s print queue and printer daemon

`lpc` can be run from the command line or interactively. Specifying `lpc` with the optional *command* and *parameter* arguments causes `lpc` to interpret the first argument as an `lpc` command, and all other arguments as parameters to that command. Specifying `lpc` without arguments causes it to run interactively, prompting the user for `lpc` commands with `lpc>`. By redirecting the standard input, `lpc` can read commands from a file.

USAGE

`lpc` commands may be typed in their entirety or abbreviated to an unambiguous substring. Specify the *printer* parameter by the name of the printer (for example, as `lw`), not as you would specify it to `lpr`(1B) or `lpq`(1B) (not as –`Plw`).

Some `lpc` commands are available to all users; others are available only to users who have the `administer printing` authorization.

All users may execute the following commands.

`?` [*command* …] | `help` [*command* …]
Displays a short description of *command*. *command* is an `lpc` command. If *command* is not specified, displays a list of `lpc` commands.

`exit` | `quit` Exits from `lpc`.

`restart` [`all` | *printer* …]
Attempts to start a new printer daemon. `restart` is useful when a print daemon dies unexpectedly and leaves jobs in the print queue. `all` specifies to perform this command on all locally attached printers. *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.

`status` [`all` | *printer* …]
Displays the status of print daemons and print queues. `all` specifies perform this command on all locally attached printers. *printer* indicates perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.

Users who have the `administer printing` authorization may execute the following `lpc`commands

`abort` [`all` | *printer …*]

Terminates an active spooling daemon. Disables printing (by preventing new daemons from being started by `lpr`(1B)) for *printer*. `all` specifies perform this command on all locally attached printers. *printer* indicates perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.Use of this command requires the `administer printing` authorization.

`clean` [`all` | *printer …*]

Removes files created in the print spool directory by the print daemon from *printer 's* print queue. `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names. Use of this command requires the `administer printing` authorization.

`disable` [`all` | *printer …*]

Turns off the print queue for *printer*. Prevents new printer jobs from being entered into the print queue for *printer*by `lpr`(1B). `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names. Use of this command requires the `administer printing` authorization.

`down` [`all` | *printer …*] [*message*]

Turns the queue for *printer* off and disables printing on *printer*. Inserts *message* in the printer status file. *message* does not need to be quoted; multiple arguments to *message* are treated as arguments are to `echo`(1). Use `down` to take a printer down and inform users. `lpq`(1B) indicates that the printer is down, as does the `status` command. `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.

`enable` [`all` | *printer …*]

Enables `lpr`(1B) to add new jobs in the spool queue. `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.Use of this command requires the `administer printing` authorization.

`start` [`all` | *printer …*]
Enables printing. Starts a spooling daemon for the *printer*. `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.

`stop` [`all` | *printer …*]
Stops a spooling daemon after the current job is complete. Disables printing at that time. `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names. Use of this command requires the `administer printing` authorization.

`topq` *printer* [*request-ID …*] [*user …*]
Moves *request-ID* or print jobs belonging to *user* on *printer* to the beginning of the print queue. Specify *user* as a user's login name. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.Use of this command requires the `administer printing` authorization.

`up` [`all` | *printer…*]
Turns the queue for *printer* on and enables printing on *printer*. Deletes the message in the printer status file (inserted by `down`). Use `up` to undo the effects of `down`. `all` specifies to perform this command on all locally attached printers *printer* indicates to perform this command on specific printers. Specify *printer* as an atomic name. See `printers.conf`(4) for information regarding naming conventions for atomic names.

EXIT STATUS

The following exit values are returned:

`0`	Successful completion.
non-zero	An error occurred.

FILES

`/var/spool/lp/*`	LP print queue.
`/var/spool/lp/system/pstatus`	Printer status information file.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscplp

SUMMARY OF TRUSTED SOLARIS CHANGES

Functions of this command that are restricted to the super-user in Solaris require the `administer printing` authorization in the Trusted Solaris environment.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lpstat`(1), `lpq`(1B), `lpr`(1B), `lprm`(1B), `lpsched`(1M), `lpshut`(1M)

SunOS 5.8 Reference Manual

`echo`(1), `printers.conf`(4), `attributes`(5)

DIAGNOSTICS

`Ambiguous command`
Indicates that the `lpc` command or abbreviation matches more than one command.

`?Invalid command`
Indicates that the `lpc` command or abbreviation is not recognized.

`?Privileged command`
Indicates that the `lpc` command or abbreviation can be executed only by users who have the `administer printing` authorization.

`lpc:` *printer* `:  unknown printer to the print service`
Indicates that *printer* does not exist in the LP database. Check that *printer* was correctly specified. Use `lpstat –p` or the `status` command (see `lpstat`(1) or `USAGE`) to check the status of printers.

`lpc:  error on opening queue to spooler`
Indicates that the connection to `lpsched` failed. Usually means that the printer server has died or is hung. Use `/usr/lib/lp/lpsched` to check if the printer spooler daemon is running.

`lpc:  Can’t send message to LP print service`
`lpc:  Can’t receive message from LP print service`
Indicates that the LP print service stopped. Contact the LP administrator.

`lpc:  Received unexpected message from LP print service`
Indicates a problem with the software. Contact the LP administrator.

NAME

lpq – Display the content of a print queue

SYNOPSIS

/usr/ucb/lpq [–P *destination*] [–l] [+ [*interval*]] [*request-ID…*] [*user…*]

DESCRIPTION

The `lpq` utility displays the information about the contents of a print queue. A print queue is comprised of print requests that are waiting in the process of being printed.

`lpq` displays the following information to the standard output:

- the username of the person associated with a print request
- the position of a print request in the print queue
- the name of file or files comprising a print request
- the job number of a print request
- the size of the file requested by a print request. File size is reported in bytes

Normally, only as much information as will fit on one line is displayed. If the name of the input file associated with a print request is not available, the input file field indicates the standard input. Jobs are normally queued on a first-in-first-out basis. Filenames comprising a job may be unavailable, such as when `lpr` is used at the end of a pipeline; in such cases the filename field indicates the standard input.

Normally, `lpq` displays only the user's own print jobs. If the user has the `list all print jobs` authorization, `lpq` displays other users' print jobs as well.

The print client commands locate destination information in a specific order. See `printers.conf`(4) and `printers`(4) for details.

If `lpq` warns that there is no daemon present (that is, due to some malfunction), the `lpc`(1B) command can be used to restart a printer daemon.

OPTIONS

The following options are supported:

–P *destination* Displays information about printer or class of printers (see `lpadmin`(1M)). Specify *destination* using atomic, POSIX-style (*server*:*destination*), or Federated Naming Service (FNS) (…`/service/printer/`…) names. See `printers.conf` (4) for information regarding the naming conventions for atomic and FNS names, and `standards`(5) for information regarding POSIX.

–l Displays information in long format. Long format includes the name of the host from which a print request originated in the display.

–M Display multilabel queue information. Without this option, only jobs at the user's sensitivity label are displayed. If the –M option is used, all jobs at sensitivity labels dominated by the user's sensitivity label are displayed. If the –`M` option is used and the user has the `bypass system mac check` authorization, jobs at all sensitivity labels are displayed.

`+` [*interval*]
: Displays information at specific time intervals. Stops displaying information when the print queue is empty. Clears the screen before reporting displaying the print queue. Specify *interval* as the number of seconds between displays. If *interval* is not specified, only executes once.

OPERANDS

The following operands are supported:

request-ID
: The job number associated with a print request.

user
: The name of the user about whose jobs `lpq` reports information. Specify *user* as a valid username.

EXIT STATUS

The following exit values are returned:

`0`
: Successful completion.

non-zero
: An error occurred.

FILES

`/var/spool/print/[cd]f*`
: Spooling directory and request files for jobs awaiting transfer.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscplp

SUMMARY OF TRUSTED SOLARIS CHANGES

The –`M` option is added. To display other users' print jobs requires that the user has the `list all print jobs` authorization, unless the `PRINT_LIST` is set in `/etc/default/print`.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lp`(1), `lpstat`(1), `lpc`(1B), `lpr`(1B), `lprm`(1B), `lpsched`(1M)

SunOS 5.8 Reference Manual

`echo`(1), `printers.conf`(4), `attributes`(5)

DIAGNOSTICS

printer `is printing`
: The `lpq` program queries the spooler `LPSCHED` about the status of the printer. If the printer is disabled, the administrator can restart the spooler using `lpc`(1B).

printer `waiting for auto-retry (offline ?)`

The daemon could not open the printer device. The printer may be turned off-line. This message can also occur if a printer is out of paper, the paper is jammed, and so on. Another possible cause is that a process, such as an output filter, has exclusive use of the device. The only recourse in this case is to kill the offending process and restart the printer with `lpc`.

`waiting for` *host* `to come up`

A daemon is trying to connect to the remote machine named *host*, in order to send the files in the local queue. If the remote machine is up, `lpd` on the remote machine is probably dead or hung and should be restarted using `lpc`.

`sending to` *host*

The files are being transferred to the remote *host*, or else the local daemon has hung while trying to transfer the files.

`printer disabled reason:`

The printer has been marked as being unavailable with `lpc`.

`lpq:  The LP print service isn't running or can't be reached.`

The `lpsched` process overseeing the spooling queue does not exist. This normally occurs only when the daemon has unexpectedly died. You can restart the printer daemon with `lpc`.

`lpr:` *printer*`:  unknown printer`

The `printer` was not found in the System V LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use `lpstat –p` (see `lpstat`(1)) or `lpc status` (see `lpc`(1B)) to discover the reason.

`lpr:  error on opening queue to spooler`

The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon `/usr/lib/lpsched` is running.

`lpr:  Can't send message to LP print service`

These indicate that the LP print service has been stopped. Get help from the system administrator.

`lpr:  Can't receive message from LP print service`

These indicate that the LP print service has been stopped. Get help from the system administrator.

`lpr:  Received unexpected message from LP print service`

It is likely there is an error in this software. Get help from system administrator.

NOTES

Output formatting is sensitive to the line length of the terminal; this can result in widely-spaced columns.

NAME

lpr – Submit print requests

SYNOPSIS

/usr/ucb/lpr [–P *destination*] [–# *number*] [–C *class*] [–J *job*] [–T *title*] [–i [*indent*]]
[–1 | –2 | –3 | –4 *font*][–w *cols*] [–m] [–h] [–s] [–filter_option] [*file*…]

DESCRIPTION

The lpr utility submits print requests to a destination. lpr prints files (*file*) and associated information, collectively called a *print request*. If *file* is not specified, lpr assumes the standard input.

The print client commands locate destination information in a very specific order. See printers(4) and printers.conf(4) for details.

Print requests with more than 52 files specified will be truncated to 52 files. lpr displays a warning message.

OPTIONS

The following options are supported:

–P *destination* Prints *file* on a specific printer or class of printers (see lpadmin(1M)). Specify *destination* using atomic, POSIX-style (*server*:*destination*), or Federated Naming Service (FNS) (…/service/printer/…) names. See printers.conf(4) for information regarding the naming conventions for atomic and FNS names, and standards(5) for information regarding POSIX.

–# *number* Prints a specific number of copies. Specify *number* as a positive integer. The default for *number* is 1.

–C *class* Prints *class* as the job classification on the banner page of the output. Enclose *class* in double quotes if it contains blanks. If *class* is not specified, the name of the system (as returned by hostname) is printed as the job classification. See hostname(1).

–J *job* Prints *job* as the job name on the banner page of the output. Enclose *job* in double quotes if it contains blanks. If *job* is not specified, *file* (or in the case of multiple files, the first file specified on the command line) is printed as the job name on the banner page of the output.

–T *title* Prints a title on the banner page of the output. Enclose *title* in double quotes if it contains blanks. If *title* is not specified, *file* is printed on the banner page.

–i *indent* Indents the output a specific number of SPACE characters. Use *indent* to indicate the number of SPACE characters to be indented. Specify *indent* as a positive integer. Eight SPACE characters is the default.

−1|−2|−3|−4 *font* Mounts the specified font in the font position 1, 2, 3, or 4. Specify *font* as a valid font name.

−w *cols* Prints *file* with pages of a specific width. *cols* indicates the number of columns wide.

−m Sends mail after *file* has printed. See mail(1). By default, no mail is sent upon normal completion of a print request.

−h Suppresses printing of the banner page of the output. Use of this option requires the print without banners authorization.

−s Uses full pathnames (as opposed to symbolic links) to *file* rather than trying to copy them. This means *file* should not be modified or removed until it has completed printing. Option −s only prevents copies of local files from being made on the local machine. Option −s only works with specified *file*s. If the lpr command is at the end of a pipeline, *file* is copied to the spool. This option is not supported in the Trusted Solaris environment.

– *filter_option*

Notifies the print spooler that *file* is not a standard text file. Enables the spooling daemon to use the appropriate filters to print *file*.

*filter_option*s offer a standard user interface. All options may not be available for, or applicable to, all printers.

Specify *filter_option* as a single character.

If *filter_option* is not specified and the printer can interpret PostScript®, inserting '%!' as the first two characters of *file* causes *file* to be interpreted as PostScript. In the Trusted Solaris environment, printing a file containing PostScript commands requires the print a PostScript file authorization.

The following *filter_option*s are supported:

p Use pr to format the files. See pr(1).

l Print control characters and suppress page breaks.

t *file* contains troff (cat phototypesetter) binary data.

n *file* contains ditroff data from device independent troff.

d *file* contains T_EX® data in DVI format from Stanford.

g *file* contains standard plot data produced by plot(1B) routines.

`v` *file* contains a raster image. *printer* must support an appropriate imaging model such as PostScript in order to print the image.

`c` *file* contains data produced by *cifplot*.

`f` Interprets the first character of each line as a standard FORTRAN carriage control character.

OPERANDS

The following operands are supported:

file The name of the file to be printed. Specify *file* as a pathname. If *file* is not specified, `lpr` uses the standard input.

USAGE

See `largefile`(5) for the description of the behavior of `lpr` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

non-zero An error occurred.

FILES

`/var/spool/print/.seq` File containing the sequence numbers for job ID assignment.

`/var/spool/print/[cd]f*` Spooling directories and files.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscplp
CSI	Enabled (see NOTES)

SUMMARY OF TRUSTED SOLARIS CHANGES

Use of the `–h` option requires the `print without banners` authorization. Printing a file that contains PostScript commands requires the `print a PostScript file` authorization, unless the `PRINT_POSTSCRIPT` option is set in `/etc/default/print`. The `–s` option is accepted but ignored; a copy of the file is always made before printing.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lp`(1), `lpstat`(1), `lpc`(1B), `lpq`(1B), `lprm`(1B), `lpadmin`(1M), `lpsched`(1M)

SunOS 5.8 Reference Manual

`hostname`(1), `mail`(1), `plot`(1B), `pr`(1), `troff`(1), `printers`(4), `printers.conf`(4), `attributes`(5), `largefile`(5), `standards`(5)

DIAGNOSTICS

`lpr:` *destination* `|:  unknown destination`

destination was not found in the LP configuration database. Usually this is a typing mistake; however, it may indicate that the destination does not exist on the system. Use **`lpstat −p`** to display information about the status of the print service.

`lpr:` *printer* `: unknown printer`
The *printer* was not found in the LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use **`lpstat −p`** (see `lpstat`(1)) or **`lpc status`** (see `lpc`(1B)) to discover the reason.

`lpr: error on opening queue to spooler`
The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check to see whether the printer spooler daemon `/usr/lib/lpsched` is running.

`lpr:` *printer* `: printer queue is disabled`
This means the queue was turned off with `/usr/etc/lpc disable` *printer* to prevent `lpr` from putting files in the queue. This is normally done by the system manager when a printer is going to be down for a long time. The printer can be turned back on by an administrator with `lpc`.

`lpr: Can’t send message to the LP print service`
`lpr: Can’t receive message from the LP print service`
These indicate that the LP print service has been stopped. Get help from the system administrator.

`lpr: Received unexpected message from LP print service`
It is likely there is an error in this software. Get help from system administrator.

`lpr: There is no filter to convert the file content`
Use the ‘`lpstat −p −l`’ command to find a printer that can handle the file type directly, or consult with your system administrator.

`lpr: cannot access the file`
Make sure file names are valid.

NOTES

`lpr` is CSI-enabled except for the *printer* name.

`lp` is the preferred interface.

Command-line options cannot be combined into a single argument as with some other commands. The command: `lpr −p`

is not equivalent to `pr | lpr`.

`lpr −p` puts the current date at the top of each page, rather than the date last modified.

Fonts for troff(1) and T_EX® reside on the printer host. It is currently not possible to use local font libraries.

lpr objects to printing binary files.

NAME

lprm – Remove print requests from the print queue

SYNOPSIS

/usr/ucb/lprm [–P *destination*] [–] [*request-ID* …] [*user*…]

DESCRIPTION

The `lprm` utility removes print requests (*request-ID*) from the print queue.

If invoked without arguments, `lprm` deletes the user's current print request. `lprm` reports the name of the file associated with print requests that it removes, but is silent if there are no applicable print requests to remove.

To remove a job belonging to another user, the user must have the `cancel any print job` authorization. `lprm` then removes all jobs that belong to the specified user.

You can remove a specific job by supplying its job number (*request-ID*) as an argument. To find the job number, run `lpq`(1B). See EXAMPLES.

`lprm` can normally cancel only requests that are at its own sensitivity label. To cancel jobs at other SLs, the user must have the `bypass system mac check` authorization.

The print client commands locate destination information in a very specific order. See `printers`(4) and `printers.conf`(4) for details.

OPTIONS

The following options are supported.

–P *destination* The name of the printer or class of printers (see `lpadmin`(1M)) from which to remove print requests. Specify destination using atomic, POSIX-style (*server*`:`*destination*), or Federated Naming Service (FNS) (…`/service/printer/`…) names. See `printers.conf`(4) for information regarding the naming conventions for atomic and FNS names, and `standards`(5) for information regarding POSIX.

– If a user specifies this option, `lprm` removes all print requests owned by that user. If invoked by a user with the `administer printing` authorization, it removes all requests in the print queue. Job ownership is determined by the user's login name and host name on the machine from which `lprm` was executed. See `NOTES`.

OPERANDS

The following operands are supported.

user Removes print requests associated with a specific user. Specify *user* as a valid user name. Use of this operand requires the `administer printing` authorization.

request-ID Removes a specific print request. Specify *request-ID* as the job number (`Job`) associated with a print request and reported by `lpq`. See `lpq`(1B).

EXAMPLES

EXAMPLE 1 Find and Remove a Print Job

The following example finds the job number on the printer `killtree` using `lpq`, then removes the job:

```
admin$ lpq –P killtree
killtree is ready and printing
```

```
Rank      Owner     Job         Files          Total Size

active    wendy     385     standard input     35501 bytes
```

```
admin$ lprm –P killtree 385
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

FILES

`/var/spool/print/[cd]f*` Spooling directories and files.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWscplp

SUMMARY OF TRUSTED SOLARIS CHANGES

For `lprm` to cancel other users' requests, requires that the user have the `cancel any print job` authorization. For `lprm` to cancel requests at other sensitivity labels requires that the user have the `bypass system mac check` authorization.

SEE ALSO

Trusted Solaris 8 Reference Manual

`lp`(1), `lpstat`(1), `lpc`(1B), `lpq`(1B), `lpr`(1B), `lpadmin`(1M), `lpsched`(1M)

SunOS 5.8 Reference Manual

`printers`(4), `printers.conf`(4), `attributes`(5), `standards`(5)

DIAGNOSTICS

`lprm:` *printer* `: unknown printer`

The `printer` was not found in the System V LP database. Usually this is a typing mistake; however, it may indicate that the printer does not exist on the system. Use '`lpstat –p`' (see `lpstat`(1)) or '`lpc status`' (see `lpc`(1B)) to discover the reason.

`lprm: error on opening queue to spooler`

The connection to `lpsched` on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check if the printer spooler daemon `/usr/lib/lpsched` is running.

`lprm:  Can’t send message to the LP print service`
`lprm:  Can’t receive message from the LP print service`
These indicate that the LP print service has been stopped. Get help from the system administrator.

`lprm:  Received unexpected message from the LP print service`
It is likely there is an error in this software. Get help from system administrator.

`lprm:  Can’t cancel request`
You are not allowed to remove another’s request.

NOTES

An active job may be incorrectly identified for removal by an `lprm` command issued with no arguments. During the interval between an `lpq` command and the execution of `lprm`, the next job in queue may have become active; you can remove that job unintentionally if you own it. To avoid this, supply `lprm` with the job number as an argument.

NAME

lpstat – Print information about the status of the print service

SYNOPSIS

lpstat [–d] [–r] [–R] [–s] [–t] [–a [*list*]] [–c [*list*]] [–f [*list*] [–l]] [–o [*list*]] [–p [*list*] [–D] [–l]] [–P] [–S [*list*] [–l]] [–u [*login-ID-list*]] [–v [*list*]] [–M]

DESCRIPTION

`lpstat` displays information about the current status of the LP print service to standard output.

If no options are given, then `lpstat` prints the status of all the user’s print requests made by `lp` [see `lp`(1) . Any arguments that are not *options* are assumed to be *request-IDs* as returned by `lp`. The `lpstat` command prints the status of such requests. The *options* may appear in any order and may be repeated and intermixed with other arguments. Some of the keyletters below may be followed by an optional *list* that can be in one of two forms: a list of items separated from one another by a comma, or a list of items separated from one another by spaces enclosed in quotes. For example:

```
example% lpstat –u "user1 user2 user3"
```

Specifying `all` after any key letter that takes *list* as an argument causes all information relevant to the key letter to be printed. For example, the command:

```
example% lpstat –o all
```

prints the status of all output requests.

The omission of a *list* following such key letters causes all information relevant to the key letter to be printed. For example, the command:

```
example% lpstat –o
```

prints the status of all output requests.

The print client commands locate printer information in a very specific order. See `printers.conf`(4) and `printers`(4) for details.

OPTIONS

The following options are supported:

–a [*list*] Reports whether print destinations are accepting requests. *list* is a list of intermixed printer names and class names.

–c [*list*] Print name of all classes and their members. *list* is a list of class names.

–d Print the system default destination for output requests.

–f[*list*] [–l]
Print a verification that the forms in *list* are recognized by the LP print service. *list* is a list of forms; the default is `all`. The –l option will list the form descriptions.

–M
Include multilabel queue information in the output for the –o option. If the –M option is not used, only jobs at the user's current sensitivity label are displayed. If the –M option is used, all jobs at sensitivity labels dominated by the the user's sensitivity label are displayed. If the –M option is used and the user has the `bypass system mac check` authorization, jobs at all sensitivity labels are displayed.

–o [*list*]
Print the status of output requests: *list* is a list of intermixed printer names, class names, and *request-IDs*. The keyletter –o may be omitted. Normally, `lpstat` displays only the invoking user's output requests. If the user has the `list all print jobs` authorization, `lpstat` displays other users' print jobs as well.

–p [*list*] [–D] [–l]
Print the status of printers. *list* is a list of printer names. If the –D option is given, a brief description is printed for each printer in *list*. If the –l option is given, and the printer is on the local machine, a full description of each printer's configuration is given, including the form mounted, the acceptable content and printer types, a printer description, the interface used, and so on.

–P
Print the paper types.

–r
Print the status of the LP request scheduler.

–R
Print a number showing the position of each job in the print queue.

–s
Print a status summary, including the status of the LP scheduler, the system default destination, a list of class names and their members, a list of printers and their associated devices, a list of the machines sharing print services, a list of all forms currently mounted, and a list of all recognized character sets and print wheels.

-S [*list*] [–l]
Print a verification that the character sets or the print wheels specified in *list* are recognized by the LP print service. Items in *list* can be character sets or print wheels; the default for the list is `all`. If the –l option is given, each line is appended by a list of printers that can handle the print wheel or character set. The list also shows whether the print

wheel or character set is mounted, or specifies the built-in character set into which it maps.

–t — Print all status information. This includes all the information obtained with the –s option, plus the acceptance and idle/busy status of all printers.

–u [*login-ID-list*] — Print the status of output requests for users. The *login-ID-list* argument may include any or all of the following constructs:

login-ID	a user on any system
system_name!*login-ID*	a user on system *system_name*
system_name!all	all users on system *system_name*
all!*login-ID*	a user on all systems
all	all users on all systems

–v [*list*] — Print the names of printers and the path names of the devices associated with them. For network printers, print the remote system names for the printers. *list* is a list of printer names.

EXIT STATUS

The following exit values are returned:

0	Successful completion.
non-zero	An error occurred.

FILES

/var/spool/print/*	LP print queue
$HOME/.printers	User-configurable printer database
/etc/printers.conf	System configuration database

SUMMARY OF TRUSTED SOLARIS CHANGES

The –M option is now included. The list all jobs authorization is required for display of other users' print jobs, unless the PRINT_LIST option is set in /etc/default/print. The bypass system mac check authorization is required for display of print jobs at sensitivity labels not dominated by the user's sensitivity label.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWpcu

SEE ALSO

Trusted Solaris 8 Reference Manual

cancel(1), enable(1), lp(1), lpq(1B), lpr(1B), lprm(1B)

SunOS 5.8 Reference Manual

printers(4), printers.conf(4), attributes(5), standards(5)

NAME

mkdir – Make directories

SYNOPSIS

mkdir [–m *mode*] [–p] [–M] *dir*…

DESCRIPTION

The mkdir command creates the named directories in mode 777 (possibly altered by the file mode creation mask umask(1)).

Standard entries in a directory (for instance, the files “.”, for the directory itself, and “. .”, for its parent) are made automatically. mkdir cannot create these entries by name. Creation of a directory requires write permission in the parent directory.

The owner-ID and group-ID of the new directories are set to the process’s effective user-ID and group-ID, respectively. mkdir calls the mkdir(2) system call.

setgid and mkdir

To change the setgid bit on a newly created directory, you must use chmod g+s or chmod g-s after executing mkdir.

The setgid bit setting is inherited from the parent directory.

OPTIONS

The following options are supported:

–m *mode* This option allows users to specify the mode to be used for new directories. Choices for modes can be found in chmod(1).

–p With this option, mkdir creates *dir* by creating all the non-existing parent directories first. The mode given to intermediate directories will be the difference between 777 and the bits set in the file mode creation mask. The difference, however, must be at least 300 (write and execute permission for the user).

–M With this option, mkdir creates *dir* as a multilevel directory.

OPERANDS

The following operand is supported:

dir A path name of a directory to be created.

USAGE

See largefile(5) for the description of the behavior of mkdir when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

EXAMPLE 1 Using mkdir

The following example:

```
  example% mkdir -p ltr/jd/jan
```

creates the subdirectory structure ltr/jd/jan.

EXAMPLE 2 Using filesystem attributes to create a multilevel directory

The following example specifies a new adornment or prefix for `filesystem1` and creates a multilevel directory with the specified MLD prefix [see `setfsattr`(1M)]:

```
example% setfsattr -m .MULTI. /dev/filesystem1
example% mount /dev/filesystem1 /mnt
example% mkdir /mnt/.MULTI.directory1
```

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `mkdir`: `LC_CTYPE`, `LC_MESSAGES`, and `NLSPATH`.

EXIT STATUS

The following exit values are returned:

`0` All the specified directories were created successfully or the `−p` option was specified and all the specified directories now exist.

`>0` An error occurred.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

The `−M` option creates a multilevel directory. Multilevel directories can also be created without the `−M` option by specifying a directory name with the `MLD` prefix for that filesystem. See the example above using `setfsattr`(1M).

SEE ALSO

Trusted Solaris 8 Reference Manual

`setfsattr`(1M), `intro`(2), `mkdir`(2)

SunOS 5.8 Reference Manual

`rm`(1), `sh`(1), `umask`(1), `attributes`(5), `environ`(5), `largefile`(5)

NAME

mldpwd – Display the pathname of the current working directory, including any MLD adornments and SLD names

SYNOPSIS

mldpwd

DESCRIPTION

`mldpwd` prints the canonicalized pathname of the (current) working directory. MLD adornments and SLD names are displayed as encountered. The example below illustrates the differences between `mldpwd` and `pwd`.

```
example% cd /usr/wendy/january/reports
example% mldpwd
/usr/wendy/january/.MLD.reports/.SLD.1
example% pwd
/usr/wendy/january/reports
example%
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

SEE ALSO

SunOS 5.8 Reference Manual

`pwd`(1), `attributes`(5)

NAME

mldrealpath – Display the canonicalized absolute pathname, including any MLD adornments and SLD names

SYNOPSIS

mldrealpath *pathname*

DESCRIPTION

`mldrealpath` expands all symbolic links and resolves references to '`/./`', '`/../`', extra '`/`' characters, and MLD translations in *pathname*. The resulting path will have no symbolic link components, nor any '`/./`', '`/../`', nor any unadorned MLDs, nor any hidden SLD names.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

RETURN VALUES

`mldrealpath` exits with one of the following values:

`0` Success

`1` Usage error

`2` Failure; error message is the system error number from `mldrealpath`(3TSOL).

SEE ALSO

Trusted Solaris 8 Reference Manual

`mldrealpath`(3TSOL)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

nca, snca – the Solaris Network Cache and Accelerator (NCA)

DESCRIPTION

The Solaris Network Cache and Accelerator (“ NCA ”) is a kernel module designed to provide improved web server performance. The kernel module, `ncakmod` , services HTTP requests. To improve the performance of servicing HTTP requests, the NCA kernel module maintains an in-kernel cache of web pages. If the NCA kernel module cannot service the request itself, it passes the request to the `http` daemon (`httpd`) by means of a private interface. The logging facility, `ncalogd` , logs all requests. This private interface uses the Solaris Doors RPC mechanism. See, for example, `door_create`(3DOOR) , `door_call`(3DOOR) , and `door_bind`(3DOOR) .

The NCA cache consistency is maintained by honoring HTTP headers dealing with a given content type and expiration date, much the same way as a proxy cache.

The NCA is disabled in the Trusted Solaris environment.

For configuration information, see *System Administration Guide, Volume 3* .

NCA is intended to be run on a dedicated webserver. Running other large processes while running NCA may cause undesirable behavior.

FILES

`/etc/nca/ncakmod.conf`	Lists configuration parameters for NCA .
`/etc/nca/ncalogd.conf`	Lists configuration parameters for NCA logging.
`/etc/nca/nca.if`	Lists the physical interfaces on which NCA will run.
`/etc/hostname.{}{0-9}`	Lists all physical interfaces configured on the server.
`/etc/hosts`	Lists all host names associated with the server. Entries in this file must match with entries in `/etc/hostname.{}{0–9}` for NCA to function.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWncar
Interface Stability	Evolving

SUMMARY OF TRUSTED SOLARIS CHANGES

The Network Cache and Accelerator kernel module is disabled in the Trusted Solaris environment.

SEE ALSO

Trusted Solaris 8 Reference Manual

`door_create`(3DOOR) , `nca.if`(4)

SunOS 5.8 Reference Manual

`door_bind`(3DOOR) , `door_call`(3DOOR) , `ncakmod.conf`(4) , `attributes`(5)

System Administration Guide, Volume 3

NAME

ncakmod – start or stop the NCA kernel module

SYNOPSIS

/etc/init.d/ncakmod `start | stop`

DESCRIPTION

`ncakmod` is used to start or stop the Solaris Network Cache and Accelerator ("NCA") kernel module.

The NCA kernel module is disabled in the Trusted Solaris environment.

When the `start` option is specified at the command-line, the NCA kernel module will be activated for all physical interfaces listed in the `nca.if` file. When the `ncakmod` command is invoked with the `stop` option, the NCA kernel module will print the following message:

```
To stop NCA, please set the status configuration parameter to disable
in ncakmod.conf and then reboot your system. See the ncakmod.conf(4)
manual page for more information.
```

Note that in order to properly stop NCA on your system, you must first edit the `ncakmod.conf`(4) file and set the status field to "disable," then reboot your system.

OPTIONS

`start` Starts the NCA kernel module.

`stop` Describes the current method for stopping the NCA feature.

EXAMPLES

EXAMPLE 1 Starting and Stopping the NCA Feature

The following command is used to start the NCA feature:

```
example% /etc/init.d/ncakmod start
```

FILES

`/etc/init.d/ncakmod` The NCA kernel module startup script.

`/etc/nca/ncakmod.conf` Specifies configuration options for the NCA kernel module.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWncar
Interface Stability	Evolving

SUMMARY OF TRUSTED SOLARIS CHANGES

The Network Cache and Accelerator kernel module is disabled in the Trusted Solaris environment.

SEE ALSO

Trusted Solaris 8 Reference Manual

`nca`(1), `nca.if`(4)

SunOS 5.8 Reference Manual

`ncakmod.conf`(4), `attributes`(5)

NAME

nispasswd – change NIS+ password information

SYNOPSIS

nispasswd [–ghs] [–D *domainname*] [*username*]

nispasswd –a

nispasswd [–D *domainname*] [–d [*username*]]

nispasswd [–l] [–f] [–n *min*] [–x *max*] [–w *warn*] [–D *domainname*] *username*

DESCRIPTION

The nispasswd utility changes a password, gecos (finger) field (–g option), home directory (–h option), or login shell (–s option) associated with the *username* (invoker by default) in the NIS+ passwd table.

Additionally, the command can be used to view or modify aging information associated with the user specified if the invoker has the right NIS+ privileges.

nispasswd uses secure RPC to communicate with the NIS+ server, and therefore, never sends unencrypted passwords over the communication medium.

nispasswd does not read or modify the local password information stored in the /etc/passwd and /etc/shadow files.

When used to change a password, nispasswd prompts non-privileged users for their old password. It then prompts for the new password twice to forestall typing mistakes. When the old password is entered, nispasswd checks to see if it has “aged” sufficiently. If “aging” is insufficient, nispasswd terminates; see getspnam(3C).

The old password is used to decrypt the username’s secret key. If the password does not decrypt the secret key, nispasswd prompts for the old secure-RPC password. It uses this password to decrypt the secret key. If this fails, it gives the user one more chance. The old password is also used to ensure that the new password differs from the old by at least three characters. Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements described below. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated twice. The new password is used to re-encrypt the user’s secret key. Hence, it also becomes their secure-RPC password. Therefore, the secure-RPC password is no longer a different password from the user’s password.

Passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant.
- Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" refers to all upper or lower case letters.

- Each password must differ from the user's login *username* and any reverse or circular shift of that login *username.* For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.
- New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

Network administrators, who own the NIS+ password table, may change any password attributes if they establish their credentials (see `keylogin`(1)) before invoking `nispasswd`. Hence, `nispasswd` does not prompt these privileged-users for the old password and they are not forced to comply with password aging and password construction requirements.

Any user may use the `–d` option to display password attributes for his or her own login name. The format of the display will be:

username status mm/dd/yy min max warn

or, if password aging information is not present,

username status

where

username The login ID of the user.

status The password status of *username*: "PS" stands for password exists or locked, "LK" stands for locked, and "NP" stands for no password.

mm/dd/yy The date password was last changed for *username.* (Note that all password aging dates are determined using Greenwich Mean Time (Universal Time) and, therefore, may differ by as much as a day in other time zones.)

min The minimum number of days required between password changes for *username.*

max The maximum number of days the password is valid for *username.*

warn The number of days relative to *max* before the password expires that the *username* will be warned.

OPTIONS

The following options are supported:

`–g` Changes the gecos (finger) information.

`–h` Changes the home directory.

–`s` Changes the login shell. By default, only the NIS+ administrator can change the login shell. User will be prompted for the new login shell.

–`a` Shows the password attributes for all entries. This will show only the entries in the NIS+ passwd table in the local domain that the invoker is authorized to "read".

–`d` [*username*] Displays password attributes for the caller or the user specified if the invoker has the right privileges.

–`D` *domainname* Consults the `passwd.org_dir` table in `domainname`. If this option is not specified, the default `domainname` returned by `nis_local_directory()` will be used. This `domainname` is the same as that returned by `domainname`(1M).

Privileged User Options

Only a privileged user can use the following options:

–`f` Forces the user to change password at the next login by expiring the password for *username*.

–`l` Locks the password entry for *username*. Subsequently, `login`(1) would disallow logins with this NIS+ password entry.

–`n` *min* Sets minimum field for *username*. The *min* field contains the minimum number of days between password changes for *username*. If *min* is greater than *max*, the user may not change the password. Always use this option with the –`x` option, unless *max* is set to -1 (aging turned off). In that case, *min* need not be set.

–`x` *max* Set maximum field for *username*. The *max* field contains the number of days that the password is valid for *username*. The aging for *username* will be turned off immediately if *max* is set to -1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.

–`w` *warn* Sets *warn* field for *username*. The *warn* field contains the number of days before the password expires that the user will be warned whenever he or she attempts to login.

SUMMARY OF TRUSTED SOLARIS CHANGES

The `nispasswd` command is restricted in the Trusted Solaris environment. A user or role changes passwords by selecting the Change Password option from the Trusted Path menu in the CDE front panel. Authorized administrative roles can change another user’s password through the User Accounts tool in the Solaris Management Console.

EXIT STATUS

The following exit values are returned:

0 Success.

1 Permission denied.

2 Invalid combination of options.

3 Unexpected failure. NIS+ passwd table unchanged.

4 NIS+ passwd table missing.

5 NIS+ is busy. Try again later.

6 Invalid argument to option.

7 Aging is disabled.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWnisu

SEE ALSO

Trusted Solaris 8 Reference Manual

`login`(1), `passwd`(1), `nsswitch.conf`(4), `shadow`(4),

SunOS 5.8 Reference Manual

`keylogin`(1), `nis+`(1), `nistbladm`(1), `rlogin`(1), `domainname`(1M), `nisserver`(1M), `getpwnam`(3C), `getspnam`(3C), `nis_local_directory`(3NSL), `passwd`(4), `attributes`(5)

NOTES

The use of `nispasswd` is STRONGLY discouraged. Even though it is a hardlink to `passwd`(1), its operation is subtly different and not desirable in a modern NIS+ domain.

In particular, `nispasswd` will not attempt to contact the `rpc.nispasswdd` daemon running on the NIS+ master. It will instead attempt to do the updates by itself via the NIS+ API. For this to work, the permissions on the password data need to be modified from the default as set up by the nisserver setup script (see `nisserver`(1M)).

Using `passwd`(1) with the `−r nisplus` option will achieve the same result and will be consistent across all the different name services available. This is the recommended way to change the password in NIS+.

The login program, file access display programs (for example, '`ls −l`'), and network programs that require user passwords (for example, `rlogin`(1), `ftp`(1), and so on) use the standard `getpwnam`(3C) and `getspnam`(3C) interfaces to get password information. These programs will get the NIS+ password information, that is modified by `nispasswd`, only if the `passwd:` entry in the /etc/nsswitch.conf file includes nisplus. See nsswitch.conf(4) for more details.

NAME

passwd – Change login password and password attributes

SYNOPSIS

passwd [–r files | –r ldap | –r nis | –r nisplus][*name*]

passwd [–r files] [–egh] [*name*]

passwd [–r files] –s [–a]

passwd [–r files] –s [*name*]

passwd [–r files] [–d | –l][–f] [–n *min*] [–w *warn*] [–x *max*] *name*

passwd –r ldap [–egh] [*name*]

passwd –r nis [–egh] [*name*]

passwd –r nisplus [–egh] [–D *domainname*] [*name*]

passwd –r nisplus –s [–a]

passwd –r nisplus [–D *domainname*] –s [*name*]

passwd –r nisplus [–l] [–f] [–n *min*] [–w *warn*] [–x *max*] [–D *domainname*] *name*

DESCRIPTION

The passwd command changes the password or lists password attributes associated with the user’s login *name*. Additionally, privileged users may use passwd to install or change passwords and attributes associated with any login *name*.

When used to change a password, passwd prompts everyone for their old password, if any. It then prompts for the new password twice. When the old password is entered, passwd checks to see if it has "aged" sufficiently. If "aging" is insufficient, passwd terminates; see pwconv(1M), nistbladm(1), and shadow(4) for additional information.

When LDAP, NIS, or NIS+ is in effect on a system, passwd changes the NIS or NIS+ database. The NIS or NIS+ password may be different from the password on the local machine. If NIS or NIS+ is running, use passwd –r to change password information on the local machine.

The pwconv command creates and updates /etc/shadow with information from /etc/passwd. pwconv relies on a special value of ’x’ in the password field of /etc/passwd. This value of ’x’ indicates that the password for the user is already in /etc/shadow and should not be modified.

If aging is sufficient, a check is made to ensure that the new password meets construction requirements. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated for, at most, two more times.

Passwords must be constructed to meet the following requirements:

- Each password must have PASSLENGTH characters, where PASSLENGTH is defined in /etc/default/passwd and is set to 6. Only the first eight characters are significant.
- Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" refers to all upper or lower case letters.
- Each password must differ from the user's login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.
- New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

If all requirements are met, by default, the passwd command will consult nsswitch.conf(4) to determine in which repositories to perform password update. It searches the passwd and passwd_compat entries. The sources (repositories) associated with these entries will be updated. However, the password update configurations supported are limited to the following cases. Failure to comply with the configurations will prevent users from logging onto the system. The password update configurations are:

- `passwd: files`
- `passwd: files ldap`
- `passwd: files nis`
- `passwd: files nisplus`
- `passwd: compat` (==> files nis)
- `passwd: compat` (==> files ldap)

  `passwd_compat: ldap`
- `passwd: compat` (==> files nisplus)

  `passwd_compat: nisplus`

Network administrators, who own the NIS+ password table, may change any password attributes.

In the files case, administrative roles (for instance, real and effective uid equal to 0, see id(1M) and su(1M)) may change any password; hence, passwd does not prompt privileged users for the old password. Privileged users are not forced to comply with password aging and password construction requirements. A privileged user can create a null password by entering a carriage return in response to the prompt for a new password. (This differs from passwd –d because the "password" prompt will still be displayed.)

Any user may use the `–s` option to show password attributes for his or her own login *name*, provided they are using the `–r nisplus` argument. Otherwise, the `–s` argument is restricted to an administrative role.

The format of the display will be:

name status mm/dd/yy min max warn

or, if password aging information is not present,

name status

where

name The login ID of the user.

status The password status of *name*: `PS` stands for passworded or locked, `LK` stands for locked, and `NP` stands for no password.

mm/dd/yy The date password was last changed for *name*. (Note that all password aging dates are determined using Greenwich Mean Time (Universal Time) and therefore may differ by as much as a day in other time zones.)

min The minimum number of days required between password changes for *name*. `MINWEEKS` is found in `/etc/default/passwd` and is set to `NULL`.

max The maximum number of days the password is valid for *name*. `MAXWEEKS` is found in `/etc/default/passwd` and is set to `NULL`.

warn The number of days relative to *max* before the password expires and the *name* will be warned.

Security

`passwd` uses `pam`(3PAM) for password management. The PAM configuration policy, listed through `/etc/pam.conf`, specifies the password modules to be used for `passwd`. Here is a partial `pam.conf` file with entries for the `passwd` command using the UNIX password module:

```
passwd required password /usr/lib/security/pam_unix.so.1
```

If there are no entries for the `passwd` service, then the entries for the "other" service will be used. If multiple password modules are listed, then the user may be prompted for multiple passwords.

OPTIONS

The following options are supported:

`–r` Specifies the repository to which an operation is applied. The supported repositories are `files`, `ldap`, or `nisplus`.

`–e` Change the login shell. For the `files` repository, this only works for the superuser. Normal users may change the `ldap` or `nisplus` repository. The choice of shell is limited by the requirements of `getusershell`(3C). If the user currently has a shell that is not allowed by `getusershell`, only an administrative role may change it.

`–g` Change the gecos (finger) information. For the `files` repository, this only works for administrative roles. Normal users may change the `ldap` or `nisplus` repository.

`–h` Change the home directory.

`–D` *domainname* Consult the `passwd.org_dir` table in `domainname`. If this option is not specified, the default `domainname` returned by `nis_local_directory`(3NSL) will be used. This domain name is the same as that returned by `domainname`(1M).

`–s` *name* Show password attributes for the login *name*. For the `nisplus` repository, this works for everyone. However for the `files` repository, this only works for an administrative role.

`–a` Show password attributes for all entries. Use only with the `–s` option; *name* must not be provided. For the `nisplus` repository, this will show only the entries in the NIS+ password table in the local domain that the invoker is authorized to "read". For the `files` repository, this is restricted to an administrative role.

Privileged User Options

Only a privileged user can use the following options:

`–f` Force the user to change password at the next login by expiring the password for *name*.

`–l` Lock password entry for *name*.

`–n` *min* Set minimum field for *name*. The *min* field contains the minimum number of days between password changes for *name*. If *min* is greater than *max*, the user may not change the password. Always use this option with the `–x` option, unless *max* is set to `–1` (aging turned off). In that case, *min* need not be set.

–w *warn* Set warn field for *name.* The *warn* field contains the number of days before the password expires and the user is warned. This option is not valid if password aging is disabled.

–x *max* Set maximum field for *name.* The *max* field contains the number of days that the password is valid for *name.* The aging for *name* will be turned off immediately if *max* is set to –1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.

–d Deletes password for *name.* The login *name* will not be prompted for password. It is only applicable to the files repository.

OPERANDS

name User login name

ENVIRONMENT VARIABLES

If any of the LC_* variables, that is, LC_CTYPE, LC_MESSAGES, LC_TIME, LC_COLLATE, LC_NUMERIC, and LC_MONETARY (see environ(5)) are not set in the environment, the operational behavior of passwd for each corresponding locale category is determined by the value of the LANG environment variable. If LC_ALL is set, its contents are used to override both the LANG and the other LC_* variables. If none of the above variables is set in the environment, the "C" (U.S. style) locale determines how passwd behaves.

LC_CTYPE Determines how passwd handles characters. When LC_CTYPE is set to a valid value, passwd can display and handle text and filenames containing valid characters for that locale. passwd can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. passwd can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

LC_MESSAGES Determines how diagnostic and informative messages are presented. This includes the language and style of the messages, and the correct form of affirmative and negative responses. In the "C" locale, the messages are presented in the default form found in the program itself (in most cases, U.S. English).

EXIT STATUS

The passwd command exits with one of the following values:

0 Success

1 Permission denied

2 Invalid combination of options

3	Unexpected failure; password file unchanged
4	Unexpected failure; password file(s) missing
5	Password file(s) busy; try again later
6	Invalid argument to option
7	Aging option disabled

FILES

`/etc/oshadow`

`/etc/shells`	List of shells on the system	
`/etc/passwd`	Password file	
`/etc/shadow`	Shadow password file	
`/etc/default/login`	`RETRIES`	The number of times a user or role account can enter the wrong password before the account is locked. Assigning a number to `RETRIES` overrides the system default of `5`.

`/etc/default/passwd`

Default values can be set for the following flags in `/etc/default/passwd`. For example: `MAXWEEKS=26`

`MAXWEEKS`	Maximum time period that password is valid
`MINWEEKS`	Minimum time period before the password can be changed
`PASSLENGTH`	Minimum length of password, in characters
`WARNWEEKS`	Time period until warning of date of password’s ensuing expiration

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

PASSLENGTH in /etc/default/passwd is set to 6. A RETRIES value in /etc/default/login can be set to override the system default number of 5, which is the maximum number of times a user or role account can enter the wrong password before the account is locked. If a user or role account's user_attr(4) sets the value of lock_after_retries to no, then the account is not locked. The account is locked by the insertion of the LK string in the account's status field in passwd(4). The security administrator can reopen a locked account only by assigning a new password to the account.

The passwd command is restricted in the Trusted Solaris environment. A user or role changes passwords by selecting the Change Password option from the Trusted Path menu in the CDE front panel. Authorized administrative roles can change another user's password through the User Accounts tool in the Solaris Management Console.

SEE ALSO

Trusted Solaris 8 Reference Manual

login(1), nispasswd(1), eeprom(1M), smuser(1M), su(1M), nsswitch.conf(4), shadow(4), pam_unix(5)

SunOS 5.8 Reference Manual

finger(1), nistbladm(1), yppasswd(1), domainname(1M), id(1M), passmgmt(1M), pwconv(1M), crypt(3C), getpwnam(3C), getspnam(3C), getusershell(3C), nis_local_directory(3NSL), pam(3PAM), loginlog(4), pam.conf(4), passwd(4), attributes(5), environ(5), pam_ldap(5)

NAME

pattr – Get the viewable process attribute flags

SYNOPSIS

/usr/bin/pattr [–x] [*pid*…]

DESCRIPTION

pattr, a proc tools command, displays the viewable process attribute flags of the pattr process or of a process specified by *pid.* Those flags that cannot be viewed normally can be viewed with privilege. The process attribute flags are a collection of security flags:

```
Trusted path flag
Privilege debugging flag
Network token Mapping Process flag
Label view flags (external view or internal view)
Label translation flags
Part of diskless boot flag
Part of cut and paste selection agent flag
Part of Trusted Printing system flag
Part of automount flag
```

When the –x option is not specified, the output displays pairs of *Name* (*n* bits): *Value* as shown in the EXAMPLES section.

OPTIONS

–x Print process attribute flags in a hex format.

RETURN VALUES

pattr exits with one of the following values:

0 Success.

1 Failure.

EXAMPLES

EXAMPLE 1 pattr Display

When pattr is invoked within the Trusted Path, the display can look like this:

```
host% pattr
6872:
Trusted Path (1 bit):              Enabled
Privilege Debugging (1 bit):       Disabled
Label Translation (15 bits):       0x0
Label View (2 bits):               Internal
Token Mapper (1 bit):              Disabled
Diskless Boot (1 bit):             Disabled
Selection Agent (1 bit):           Disabled
Printing System (1 bit):           Disabled
Automounter(1 bit):                Disabled
```

Without the Trusted Path attribute, the Label Translation flag does not display, and the Trusted path flag shows as Disabled.

EXAMPLE 2 pattr –x Display

When pattr is invoked with the –x option, the display looks like this:

```
host% pattr -x
8533:   0x40003
```

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

SEE ALSO

Trusted Solaris 8 Reference Manual

proc(4), getpattr(2), setpattr(2)

SunOS 5.8 Reference Manual

attributes(5)

NAME

pclear – Get process clearance

SYNOPSIS

/usr/bin/pclear [*pid*…]

/usr/bin/pclear –l [*pid*…]

/usr/bin/pclear –L [*pid*…]

DESCRIPTION

`pclear`, a proc tools command, displays the process clearance, the clearance at which the process is running. If no *pid* is specified, the clearance of the `pclear` command is returned. The information is displayed in the form

pid: *clearance*

OPTIONS

–l Display the clearance in short form. This option is the default.

–L Display the clearance in long form.

RETURN VALUES

`pclear` exits with one of these values:

0 Successful completion.

1 Unsuccessful completion because of usage error.

2 Inability to translate clearance.

3 Inability to allocate memory.

EXAMPLES

```
host% pclear
4676: TS A B
```

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

SEE ALSO

Trusted Solaris 8 Reference Manual

`proc`(1), `getclearance`(2)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–`r`] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–`rxlF`] [*pid* | *core*]…

/usr/bin/pldd [–`F`] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–`F`] [*pid* | *core*]…

/usr/bin/pfiles [–`F`] *pid*…

/usr/bin/pwdx [–`F`] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–`v`] *pid*…

/usr/bin/ptree [–`a`] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop`	Stop each process (`PR_REQUESTED` stop).
`prun`	Set each process running (inverse of `pstop`).
`pwait`	Wait for all of the specified processes to terminate.
`ptree`	Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.
`ptime`	Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.
`pattr`	Get the viewable process attribute flags. See the `pattr`(1) man page for more information.
`pclear`	Get the process clearance. See the `pclear`(1) man page for more information.
`plabel`	Get the label of a process. See the `plabel`(1) man page for more information.
`ppriv`	Get the effective privileges of a process. See the `ppriv`(1) man page for more information.
`pprivtest`	Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r`	(`pflags` only) If the process is stopped, display its machine registers.
`–r`	(`pmap` only) Print the process' reserved addresses.
`–x`	(`pmap` only) Print resident / shared / private mapping details.
`–l`	(`pmap` only) Print unresolved dynamic linker map names.
`–a`	(`ptree` only) All; include children of process 0.
`–v`	(`pwait` only) Verbose; report terminations to standard output.
`–F`	Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the `–F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0` Successful operation.

non-zero An error has occurred.

FILES

`/proc/*` Process files

`/usr/proc/lib/*` `proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

plabel – Get the label of a process

SYNOPSIS

/usr/bin/plabel [–`iIlLsS`] [*pid*…]

DESCRIPTION

`plabel`, a proc tools command, gets the label of a process. If the *pid* is not specified, the label displayed is that of the `plabel` command. When output options are not specified, the format of the label display reflects the label display options set by the administrator. If the command specifies conflicting options, `plabel` command usage is displayed. Conflicting options include –`i` and –`I`, –`s` and –`S`, and –`l` and –`L`.

OPTIONS

–`i` Get the information label associated with the process, `ADMIN_LOW`, and display it. Information Labels (ILs) are obsolete. See `NOTES`.

–`I` Get the information label associated with the process, `ADMIN_LOW`, and display it. Information Labels (ILs) are obsolete. See `NOTES`.

–`l` Get the CMW label associated with the process, and display that label in short form. The IL portion of the label displays as `ADMIN_LOW`.

–`L` Get the CMW label associated with the process, and display that label in long form. The IL portion of the label displays as `ADMIN_LOW`.

–`s` Get the sensitivity label associated with the process, and display that label in short form.

–`S` Get the sensitivity label associated with the process, and display that label in long form.

RETURN VALUES

`plabel` exits with one of these values:

`0` Successful completion.

`1` Unsuccessful completion because of a usage error.

`2` Inability to translate label.

`3` Inability to allocate memory.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

SEE ALSO

Trusted Solaris 8 Reference Manual

`proc`(1), `getcmwplabel`(2)

SunOS 5.8 Reference Manual

attributes(5)

NOTES

Information labels (ILs) are not supported in Trusted Solaris 7 and later releases. Trusted Solaris software interprets any ILs on communications and files from systems running earlier releases as ADMIN_LOW.

- Objects still have CMW labels, and CMW labels still include the IL component: IL[SL]; however, the IL component is fixed at ADMIN_LOW.

  As a result, Trusted Solaris 7 and later releases have the following characteristics:

  - ILs do not display in window labels; SLs (Sensitivity Labels) display alone within brackets.
  - ILs do not float.
  - Setting an IL on an object has no effect.
  - Getting an object's IL will always return ADMIN_LOW.
  - Although certain utilities, library functions, and system calls can manipulate IL strings, the resulting ILs cannot be set on any objects.
  - Sensitivity labels, not information labels, display on printer banners.
  - IL–related privileges are no longer used.
  - In auditing, the ilabel token is recorded as ADMIN_LOW, when it is recorded. The audit event numbers 519 (AUE_OFLOAT), 520 (AUE_SFLOAT), and 9036 (AUE_iil_change) continue to be reserved, but those events are no longer recorded.

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–`r`] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–`rxlF`] [*pid* | *core*]…

/usr/bin/pldd [–`F`] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–`F`] [*pid* | *core*]…

/usr/bin/pfiles [–`F`] *pid*…

/usr/bin/pwdx [–`F`] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–`v`] *pid*…

/usr/bin/ptree [–`a`] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process’ reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–`r`] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–`rxlF`] [*pid* | *core*]…

/usr/bin/pldd [–`F`] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–`F`] [*pid* | *core*]…

/usr/bin/pfiles [–`F`] *pid*…

/usr/bin/pwdx [–`F`] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–`v`] *pid*…

/usr/bin/ptree [–`a`] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

ppriv – Get the effective privileges of a process

SYNOPSIS

/usr/bin/ppriv [–a] [*pid*…]

DESCRIPTION

ppriv, a proc tools command, gets the effective privilege set of the process specified by *pid.* With the –a option, ppriv gets all privilege sets of the process. If no *pid* is specified, the privileges of the ppriv command are displayed.

When all the privileges are effective, the display is simply all:

```
$ ppriv 789

all
```

OPTIONS

–a Display all privilege sets of the process whose process ID is specified. If no process ID is specified, the privilege sets of the ppriv command are displayed.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

RETURN VALUES

ppriv exits with one of the following values:

0 Successful completion.

1 Unsuccessful completion.

EXAMPLES

EXAMPLE 1 ppriv with no *pid*

If no *pid* has been specified, the effective privileges of ppriv are displayed:

```
$ ppriv
5771: proc_mac_read, proc_owner
```

With the –a option, all privilege sets of ppriv are displayed:

```
$ ppriv –a
5756:
Effective: proc_mac_read, proc_owner
Permitted: proc_mac_read, proc_owner
Inheritable: none
Saved: none
```

EXAMPLE 2 ppriv with more than one *pid*

If several *pid*s are specified, their effective privileges are displayed:

```
$ ppriv 5741 5756 5475
5741: sys_trans_label
5756: proc_mac_read, proc_owner
5475: No such process
```

With the –a option, all privilege sets of *pid* are displayed:

```
$ ppriv –a 5741 443
5741:
Effective: sys_trans_label
Permitted: sys_trans_label
Inheritable: none
Saved: none
443:
Effective: net_mac_read
Permitted: net_mac_read
Inheritable: none
Saved: none
```

SEE ALSO

Trusted Solaris 8 Reference Manual

proc(1), pprivtest(1), getppriv(2)

SunOS 5.8 Reference Manual

attributes(5)

NAME

pprivtest – Test effective privilege set of the process

SYNOPSIS

/usr/bin/pprivtest [–e] [–s] [–p *pid*] *priv_names*

DESCRIPTION

`pprivtest`, a proc tools command, tests whether the *priv_names* privileges are a subset of the effective set of the process. *priv_names* is one of these:

- A comma-separated list of privilege names, as reported by `ppriv`
- A comma-separated list of numeric privilege IDs as found in `</usr/include/sys/tsol/priv_names.h>`
- The keyword `all` to indicate all privileges

Without the –e (equal) option, the specified privileges are checked as a subset of the process privileges. `pprivtest` reports those privileges that are specified in *priv_names* but not found in the process. The –e option additionally reports privileges that the file has, but that were not specified in the `pprivtest` command.

OPTIONS

–p *pid* Test the privilege set of the process specified by the process ID. If no process ID is specified, test the privilege set of the `pprivtest` command.

–e Test whether the specified privileges are equal to the effective privileges of the process.

–s Use silent mode to suppress outputs. (This option is useful in shell scripts that need only the return value.)

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

RETURN VALUES

`pprivtest` exits with one of these values:

`0` All of the specified privileges are in the effective set.

With the –e option, the specified privileges are equal to the effective set of the process.

`1` At least one of the specified privileges is not in the effective set of the process.

With the –e option, the specified privileges are not equal to the effective set of the process.

EXAMPLES

EXAMPLE 1 `pprivtest -e` Equal Privilege Test

Use this command to test if the current process' privileges are exactly equal to the specified privileges:

```
  example% pprivtest –e p1,p2
```

EXAMPLE 2 `pprivtest` Output

If the process privileges did not match exactly, the output could be in this example format:

```
  example% 1298:missing:p2:extra:p3
```

SEE ALSO

Trusted Solaris 8 Reference Manual

`proc`(1), `ppriv`(1), `priv_name`(4)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

profiles – print rights profiles for a user

SYNOPSIS

profiles [–l] [*user* …]

DESCRIPTION

The profiles command prints on standard output the names of the rights profiles on your local system that have been assigned to you or to the optionally-specified user or role name. Profiles are a bundling mechanism used to enumerate the commands, CDE actions, and authorizations needed to perform a specific function. Along with each listed executable are the process attributes, such as the effective user and group IDs, with which the process runs when started by a privileged command interpreter. The profile shells are pfcsh, pfksh, and pfexec. See the pfexec(1) man page. Profiles can contain other profiles defined in prof_attr(4).

Multiple profiles can be combined to construct the appropriate access control. When profiles are assigned, the authorizations are added to the existing set. If the same command appears in multiple profiles, the first occurrence, as determined by the ordering of the profiles, is used for process-attribute settings. For convenience, a wild card can be specified to match all commands.

When profiles are interpreted, the profile list is loaded from user_attr(4). If any default profile is defined in /etc/security/policy.conf (see policy.conf(4)), the list of default profiles will be added to the list loaded from user_attr(4). Matching entries in prof_attr(4) provide the authorizations list, and matching entries in exec_attr(4) provide the commands list.

OPTIONS

–l Lists the commands in each profile followed by the special process attributes such as user and group IDs.

EXAMPLES

EXAMPLE 1 Sample output

The output of the profiles command has the following form:

```
example% profiles tester01 tester02
tester01 : Audit Management, All Commands
tester02 : Device Management, All Commands
example%
```

EXAMPLE 2 Using the list option

```
example% profiles -l tester01 tester02
tester01 :
    Audit Management:
      /usr/sbin/audit          euid=root
      /usr/sbin/auditconfig    euid=root    egid=sys
    All Commands:
      *
tester02 :
      Device Management:
        /usr/bin/allocate:       euid=root
        /usr/bin/deallocate:     euid=root
      All Commands
        *
  example%
```

EXIT STATUS

The following exit values are returned:

0 Successful completion.

1 An error occurred.

FILES

/etc/user_attr	Local source of extended attributes associated with users and roles.
/etc/security/auth_attr	Local source for authorization names and descriptions.
/etc/security/policy.conf	Provides the security policy configuration for user-level attributes.
/etc/security/prof_attr	Local source for rights profile names, descriptions, and other attributes of profiles.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

CDE actions can be assigned to rights profiles. To affect all name services, not just files, use the smprofile(1M) command instead of the profiles command.

SEE ALSO

Trusted Solaris 8 Reference Manual

auths(1), roles(1), smprofile(1M), exec_attr(4), prof_attr(4), user_attr(4)

SunOS 5.8 Reference Manual

pfexec(1), getprofattr(3SECDB), policy.conf(4), attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the `–F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–`r`] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–`rxlF`] [*pid* | *core*]…

/usr/bin/pldd [–`F`] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–`F`] [*pid* | *core*]…

/usr/bin/pfiles [–`F`] *pid*…

/usr/bin/pwdx [–`F`] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–`v`] *pid*…

/usr/bin/ptree [–`a`] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the `–F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of /proc (see proc(4)). Most of them take a list of process-ids (*pid*); those that do also accept /proc/ *nnn* as a process-id, so the shell expansion /proc/* can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see core(4)); those that do accept a list of either process ID s or names of core files or both.

pflags — Print the /proc tracing flags, the pending and held signals, and other /proc status information for each lwp in each process.

pcred — Print the credentials (effective, real, saved UID s and GID s) of each process.

pmap — Print the address space map of each process.

pldd — List the dynamic libraries linked into each process, including shared objects explicitly attached using dlopen(3DL) . See also ldd(1) .

psig — List the signal actions of each process. See signal(3HEAD) .

pstack — Print a hex+symbolic stack trace for each lwp in each process.

pfiles — Report fstat(2) and fcntl(2) information for all open files in each process.

pwdx — Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using rlogin(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –F flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the gcore(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for pstack . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (librtld_db) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

0	Successful operation.
non-zero	An error has occurred.

FILES

/proc/*	Process files
/usr/proc/lib/*	proc tools supporting files

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: pattr(1) , pclear(1) , plabel(1) , ppriv(1) , and pprivtest(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–`r`] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–`rxlF`] [*pid* | *core*]…

/usr/bin/pldd [–`F`] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–`F`] [*pid* | *core*]…

/usr/bin/pfiles [–`F`] *pid*…

/usr/bin/pwdx [–`F`] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–`v`] *pid*…

/usr/bin/ptree [–`a`] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0` Successful operation.

non-zero An error has occurred.

FILES

`/proc/*` Process files

`/usr/proc/lib/*` `proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

`pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , `pprivtest`(1) , `fcntl`(2) , `fstat`(2) , `proc`(4)

SunOS 5.8 Reference Manual

`ldd`(1) , `ps`(1) , `pwd`(1) , `rlogin`(1) , `time`(1) , `truss`(1) , `wait`(1) , `dlopen`(3DL) , `signal`(3HEAD) , `core`(4) , `attributes`(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of /proc (see proc(4)). Most of them take a list of process-ids (*pid*); those that do also accept /proc/ *nnn* as a process-id, so the shell expansion /proc/* can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see core(4)); those that do accept a list of either process ID s or names of core files or both.

pflags — Print the /proc tracing flags, the pending and held signals, and other /proc status information for each lwp in each process.

pcred — Print the credentials (effective, real, saved UID s and GID s) of each process.

pmap — Print the address space map of each process.

pldd — List the dynamic libraries linked into each process, including shared objects explicitly attached using dlopen(3DL) . See also ldd(1) .

psig — List the signal actions of each process. See signal(3HEAD) .

pstack — Print a hex+symbolic stack trace for each lwp in each process.

pfiles — Report fstat(2) and fcntl(2) information for all open files in each process.

pwdx — Print the current working directory of each process.

`pstop` Stop each process (`PR_REQUESTED` stop).

`prun` Set each process running (inverse of `pstop`).

`pwait` Wait for all of the specified processes to terminate.

`ptree` Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.

`ptime` Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.

`pattr` Get the viewable process attribute flags. See the `pattr`(1) man page for more information.

`pclear` Get the process clearance. See the `pclear`(1) man page for more information.

`plabel` Get the label of a process. See the `plabel`(1) man page for more information.

`ppriv` Get the effective privileges of a process. See the `ppriv`(1) man page for more information.

`pprivtest` Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r` (`pflags` only) If the process is stopped, display its machine registers.

`–r` (`pmap` only) Print the process' reserved addresses.

`–x` (`pmap` only) Print resident / shared / private mapping details.

`–l` (`pmap` only) Print unresolved dynamic linker map names.

`–a` (`ptree` only) All; include children of process 0.

`–v` (`pwait` only) Verbose; report terminations to standard output.

`–F` Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –F flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

proc, pflags, pcred, pmap, pldd, psig, pstack, pfiles, pwdx, pstop, prun, pwait, ptree, ptime – Proc tools

SYNOPSIS

/usr/bin/pflags [–r] [*pid* | *core*]…

/usr/bin/pcred [*pid* | *core*]…

/usr/bin/pmap [–rxlF] [*pid* | *core*]…

/usr/bin/pldd [–F] [*pid* | *core*]…

/usr/bin/psig *pid*…

/usr/bin/pstack [–F] [*pid* | *core*]…

/usr/bin/pfiles [–F] *pid*…

/usr/bin/pwdx [–F] *pid*…

/usr/bin/pstop *pid*…

/usr/bin/prun *pid*…

/usr/bin/pwait [–v] *pid*…

/usr/bin/ptree [–a] [[*pid* | *user*]…]

/usr/bin/ptime *command* [*arg*…]

DESCRIPTION

The proc tools are utilities that exercise features of `/proc` (see `proc`(4)). Most of them take a list of process-ids (*pid*); those that do also accept `/proc/` *nnn* as a process-id, so the shell expansion `/proc/*` can be used to specify all processes in the system. Some of the proc tools can also be applied to core files (see `core`(4)); those that do accept a list of either process ID s or names of core files or both.

`pflags` Print the `/proc` tracing flags, the pending and held signals, and other `/proc` status information for each lwp in each process.

`pcred` Print the credentials (effective, real, saved UID s and GID s) of each process.

`pmap` Print the address space map of each process.

`pldd` List the dynamic libraries linked into each process, including shared objects explicitly attached using `dlopen`(3DL) . See also `ldd`(1) .

`psig` List the signal actions of each process. See `signal`(3HEAD) .

`pstack` Print a hex+symbolic stack trace for each lwp in each process.

`pfiles` Report `fstat`(2) and `fcntl`(2) information for all open files in each process.

`pwdx` Print the current working directory of each process.

`pstop`	Stop each process (`PR_REQUESTED` stop).
`prun`	Set each process running (inverse of `pstop`).
`pwait`	Wait for all of the specified processes to terminate.
`ptree`	Print the process trees containing the specified *pid* s or *user* s, with child processes indented from their respective parent processes. An argument of all digits is taken to be a process-id, otherwise it is assumed to be a user login name. Default is all processes.
`ptime`	Time the *command* , like `time`(1) , but using microstate accounting for reproducible precision. Unlike `time`(1) , children of the command are not timed.
`pattr`	Get the viewable process attribute flags. See the `pattr`(1) man page for more information.
`pclear`	Get the process clearance. See the `pclear`(1) man page for more information.
`plabel`	Get the label of a process. See the `plabel`(1) man page for more information.
`ppriv`	Get the effective privileges of a process. See the `ppriv`(1) man page for more information.
`pprivtest`	Test the effective privilege set of a process. See the `pprivtest`(1) man page for more information.

OPTIONS

See the individual Trusted Solaris process manual pages for the options that they support. The following options are supported for Solaris process utilities:

`–r`	(`pflags` only) If the process is stopped, display its machine registers.
`–r`	(`pmap` only) Print the process' reserved addresses.
`–x`	(`pmap` only) Print resident / shared / private mapping details.
`–l`	(`pmap` only) Print unresolved dynamic linker map names.
`–a`	(`ptree` only) All; include children of process 0.
`–v`	(`pwait` only) Verbose; report terminations to standard output.
`–F`	Force; grab the target process even if another process has control.

USAGE

These proc tools stop their target processes while inspecting them and reporting the results: `pfiles` , `pldd` , `pmap` , and `pstack` . A process can do nothing while it is stopped. A process can do nothing while it is stopped. Thus, for example, if the X server is inspected by one of these proc tools running in a window

under the X server's control, the whole window system can become deadlocked because the proc tool would be attempting to print its results to a window that cannot be refreshed. Logging in from from another system using `rlogin`(1) and killing the offending proc tool would clear up the deadlock in this case.

Caution should be exercised when using the –`F` flag. Imposing two controlling processes on one victim process can lead to chaos. Safety is assured only if the primary controlling process, typically a debugger, has stopped the victim process and the primary controlling process is doing nothing at the moment of application of the proc tool in question.

Some of the proc tools can also be applied to core files, as shown by the synopsis above. A core file is a snapshot of a process's state and is produced by the kernel prior to terminating a process with a signal or by the `gcore`(1) utility. Some of the proc tools may need to derive the name of the executable corresponding to the process which dumped core or the names of shared libraries associated with the process. These files are needed, for example, to provide symbol table information for `pstack` . If the proc tool in question is unable to locate the needed executable or shared library, some symbol information will be unavailable for display. Similarly, if a core file from one operating system release is examined on a different operating system release, the run-time link-editor debugging interface (`librtld_db`) may not be able to initialize. In this case, symbol information for shared libraries will not be available.

EXIT STATUS

The following exit values are returned:

`0`	Successful operation.
non-zero	An error has occurred.

FILES

`/proc/*`	Process files
`/usr/proc/lib/*`	`proc` tools supporting files

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu (32-bit) SUNWtsxu (64-bit)

SUMMARY OF TRUSTED SOLARIS CHANGES

The Trusted Solaris environment provides additional utilities for obtaining a process' security attributes. See their man pages for a full description: `pattr`(1) , `pclear`(1) , `plabel`(1) , `ppriv`(1) , and `pprivtest`(1) .

SEE ALSO

Trusted Solaris 8 Reference Manual

pattr(1) , pclear(1) , plabel(1) , ppriv(1) , pprivtest(1) , fcntl(2) , fstat(2) , proc(4)

SunOS 5.8 Reference Manual

ldd(1) , ps(1) , pwd(1) , rlogin(1) , time(1) , truss(1) , wait(1) , dlopen(3DL) , signal(3HEAD) , core(4) , attributes(5)

NAME

rm, rmdir – Remove directory entries

SYNOPSIS

/usr/bin/rm [–f] [–i] *file*…

/usr/bin/rm –rRM [–f] [–i] *dirname*… [*file*…]

/usr/xpg4/bin/rm [–fiRr] *file*…

/usr/bin/rmdir [–ps] *dirname*…

DESCRIPTION

/usr/bin/rm and /usr/xpg4/bin/rm

The `rm` utility removes the directory entry specified by each *file* argument. If a file has no write permission and the standard input is a terminal, the full set of permissions (in octal) for the file are printed followed by a question mark. This is a prompt for confirmation. If the answer begins with `y` (for yes), the file is deleted, otherwise the file remains.

If *file* is a symbolic link, the link will be removed, but the file or directory to which it refers will not be deleted. Users do not need write permission to remove a symbolic link, provided they have write permissions in the directory.

If multiple *file* s are specified and removal of a *file* fails for any reason, `rm` will write a diagnostic message to standard error, do nothing more to the current *file* , and go on to any remaining *file* s.

If the standard input is not a terminal, the utility will operate as if the –f option is in effect.

/usr/bin/rmdir

The `rmdir` utility will remove the directory entry specified by each *dirname* operand, which must refer to an empty directory.

Directories will be processed in the order specified. If a directory and a subdirectory of that directory are specified in a single invocation of `rmdir` , the subdirectory must be specified before the parent directory so that the parent directory will be empty when `rmdir` tries to remove it.

If a specified directory is a single-level directory, the directory is not removed. SLD s may be removed by first removing all files in the SLD s, then removing the multilevel directory containing the SLD s.

OPTIONS

The following options are supported for `/usr/bin/rm` and `/usr/xpg4/bin/rm` :

–r Recursively remove directories and subdirectories in the argument list. The directory will be emptied of files and removed. The user is normally prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting, however, if the –f option is used, or if the standard input is not a terminal and the –i option is not used.

Symbolic links that are encountered with this option will not be traversed.

If the removal of a non-empty, write-protected directory is attempted, the utility will always fail (even if the `−f` option is used), resulting in an error message.

`−R` Same as `−r` option.

/usr/bin/rm

The following options are supported for `/usr/bin/rm` only:

`−f` Remove all files (whether write-protected or not) in a directory without prompting the user. In a write-protected directory, however, files are never removed (whatever their permissions are), but no messages are displayed. If the removal of a write-protected directory is attempted, this option will not suppress an error message.

`−i` Interactive. With this option, `rm` prompts for confirmation before removing any files. It overrides the `−f` option and remains in effect even if the standard input is not a terminal.

`−M` When this option is used with the recursive option (`−R`), `rm` processes all accessible SLD s as it descends multilevel directories.

/usr/xpg4/bin/rm

The following options are supported for `/usr/xpg4/bin/rm` only:

`−f` Do not prompt for confirmation. Do not write diagnostic messages or modify the exit status in the case of non-existent operands. Any previous occurrences of the `−i` option will be ignored.

`−i` Prompt for confirmation. Any occurrences of the `−f` option will be ignored.

/usr/bin/rmdir

The following options are supported for `/usr/bin/rmdir` only:

`−p` Allow users to remove the directory *dirname* and its parent directories which become empty. A message is printed to standard error if all or part of the path could not be removed.

`−s` Suppress the message printed on the standard error when `−p` is in effect.

OPERANDS

The following operands are supported:

file A path name of a directory entry to be removed.

dirname A path name of an empty directory to be removed.

USAGE

See `largefile`(5) for the description of the behavior of `rm` and `rmdir` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

/usr/bin/rm and/usr/xpg4/bin/rm

The following command:

```
example% rm a.out core
```

removes the directory entries: `a.out` and `core` .

The following command:

```
example% rm –rf junk
```

removes the directory `junk` and all its contents, without prompting.

/usr/bin/rmdir

If a directory `a` in the current directory is empty except that it contains a directory `b` and `a/b` is empty except that it contains a directory `c` ,

```
example% rmdir –p a/b/c
```

removes all three directories.

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `rm` and `rmdir` : `LC_COLLATE` , `LC_CTYPE` , `LC_MESSAGES` , and `NLSPATH` .

EXIT STATUS

The following exit values are returned:

`0` If the `–f` option was not specified, all the named directory entries were removed; otherwise, all the existing named directory entries were removed.

`>0` An error occurred.

ATTRIBUTES

/usr/bin/rm and /usr/bin/rmdir

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

/usr/xpg4/bin/rm

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

The –M option for rm processes all accessible SLD s in multilevel directories. If a directory specified for rmdir is an SLD , it is not removed.

SEE ALSO

Trusted Solaris 8 Reference Manual

rmdir(2) , unlink(2)

SunOS 5.8 Reference Manual

attributes(5) , environ(5)

DIAGNOSTICS

All messages are generally self-explanatory.

It is forbidden to remove the files " . " and " . . " in order to avoid the consequences of inadvertently doing something like the following:

```
rm –r .*
```

NOTES

A double hyphen (–-) permits the user to mark the end of any command line options explicity, allowing rm to recognize file arguments that begin with a hyphen (–). As an aid to BSD migration, rm accepts – as a synonym for –- . This migration aid may disappear in a future release. If a –- and a – both appear on the same command line, the second one is interpreted as a file.

NAME

rm, rmdir – Remove directory entries

SYNOPSIS

/usr/bin/rm [−`f`] [−`i`] *file…*

/usr/bin/rm −`rRM` [−`f`] [−`i`] *dirname…* [*file…*]

/usr/xpg4/bin/rm [−`fiRr`] *file…*

/usr/bin/rmdir [−`ps`] *dirname…*

DESCRIPTION

/usr/bin/rm and /usr/xpg4/bin/rm

The `rm` utility removes the directory entry specified by each *file* argument. If a file has no write permission and the standard input is a terminal, the full set of permissions (in octal) for the file are printed followed by a question mark. This is a prompt for confirmation. If the answer begins with `y` (for yes), the file is deleted, otherwise the file remains.

If *file* is a symbolic link, the link will be removed, but the file or directory to which it refers will not be deleted. Users do not need write permission to remove a symbolic link, provided they have write permissions in the directory.

If multiple *file* s are specified and removal of a *file* fails for any reason, `rm` will write a diagnostic message to standard error, do nothing more to the current *file* , and go on to any remaining *file* s.

If the standard input is not a terminal, the utility will operate as if the −`f` option is in effect.

/usr/bin/rmdir

The `rmdir` utility will remove the directory entry specified by each *dirname* operand, which must refer to an empty directory.

Directories will be processed in the order specified. If a directory and a subdirectory of that directory are specified in a single invocation of `rmdir` , the subdirectory must be specified before the parent directory so that the parent directory will be empty when `rmdir` tries to remove it.

If a specified directory is a single-level directory, the directory is not removed. SLD s may be removed by first removing all files in the SLD s, then removing the multilevel directory containing the SLD s.

OPTIONS

The following options are supported for `/usr/bin/rm` and `/usr/xpg4/bin/rm` :

−`r` Recursively remove directories and subdirectories in the argument list. The directory will be emptied of files and removed. The user is normally prompted for removal of any write-protected files which the directory contains. The write-protected files are removed without prompting, however, if the −`f` option is used, or if the standard input is not a terminal and the −`i` option is not used.

Symbolic links that are encountered with this option will not be traversed.

If the removal of a non-empty, write-protected directory is attempted, the utility will always fail (even if the `−f` option is used), resulting in an error message.

`−R` Same as `−r` option.

/usr/bin/rm

The following options are supported for `/usr/bin/rm` only:

`−f` Remove all files (whether write-protected or not) in a directory without prompting the user. In a write-protected directory, however, files are never removed (whatever their permissions are), but no messages are displayed. If the removal of a write-protected directory is attempted, this option will not suppress an error message.

`−i` Interactive. With this option, `rm` prompts for confirmation before removing any files. It overrides the `−f` option and remains in effect even if the standard input is not a terminal.

`−M` When this option is used with the recursive option (`−R`), `rm` processes all accessible SLD s as it descends multilevel directories.

/usr/xpg4/bin/rm

The following options are supported for `/usr/xpg4/bin/rm` only:

`−f` Do not prompt for confirmation. Do not write diagnostic messages or modify the exit status in the case of non-existent operands. Any previous occurrences of the `−i` option will be ignored.

`−i` Prompt for confirmation. Any occurrences of the `−f` option will be ignored.

/usr/bin/rmdir

The following options are supported for `/usr/bin/rmdir` only:

`−p` Allow users to remove the directory *dirname* and its parent directories which become empty. A message is printed to standard error if all or part of the path could not be removed.

`−s` Suppress the message printed on the standard error when `−p` is in effect.

OPERANDS

The following operands are supported:

file A path name of a directory entry to be removed.

dirname A path name of an empty directory to be removed.

USAGE

See `largefile`(5) for the description of the behavior of `rm` and `rmdir` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

/usr/bin/rm and/usr/xpg4/bin/rm

The following command:

```
example% rm a.out core
```

removes the directory entries: a.out and core .

The following command:

```
example% rm −rf junk
```

removes the directory junk and all its contents, without prompting.

/usr/bin/rmdir

If a directory a in the current directory is empty except that it contains a directory b and a/b is empty except that it contains a directory c ,

```
example% rmdir −p a/b/c
```

removes all three directories.

ENVIRONMENT VARIABLES

See environ(5) for descriptions of the following environment variables that affect the execution of rm and rmdir : LC_COLLATE , LC_CTYPE , LC_MESSAGES , and NLSPATH .

EXIT STATUS

The following exit values are returned:

0 If the −f option was not specified, all the named directory entries were removed; otherwise, all the existing named directory entries were removed.

>0 An error occurred.

ATTRIBUTES

/usr/bin/rm and /usr/bin/rmdir

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

/usr/xpg4/bin/rm

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWxcu4
CSI	enabled

SUMMARY OF TRUSTED SOLARIS CHANGES

The –M option for rm processes all accessible SLD s in multilevel directories. If a directory specified for rmdir is an SLD , it is not removed.

SEE ALSO

Trusted Solaris 8 Reference Manual

rmdir(2) , unlink(2)

SunOS 5.8 Reference Manual

attributes(5) , environ(5)

DIAGNOSTICS

All messages are generally self-explanatory.

It is forbidden to remove the files " . " and " . . " in order to avoid the consequences of inadvertently doing something like the following:

```
rm –r .*
```

NOTES

A double hyphen (–-) permits the user to mark the end of any command line options explicity, allowing rm to recognize file arguments that begin with a hyphen (–). As an aid to BSD migration, rm accepts – as a synonym for –- . This migration aid may disappear in a future release. If a –- and a – both appear on the same command line, the second one is interpreted as a file.

NAME

roles – print roles granted to a user

SYNOPSIS

roles [*user* …]

DESCRIPTION

The command `roles` prints on standard output the roles on your local system that you or the optionally-specified user have been granted. Roles are special accounts that correspond to a functional responsibility rather than to an actual person (referred to as a normal user).

Each user may have zero or more roles. Roles have most of the attributes of normal users and are identified like normal users in `passwd`(4) and `shadow`(4). Each role must have an entry in the `user_attr`(4) file that identifies it as a role. Roles can have their own authorizations and profiles. See `auths`(1) and `profiles`(1).

Roles are not allowed to log into a system as a primary user. Instead, a user must log in as him or herself and assume the role. The actions of a role are attributable to the normal user. When auditing is enabled, the audited events of the role contain the audit ID of the original user who assumed the role.

Roles must have valid passwords and one of the shells that interprets profiles: either `pfcsh`, `pfksh`, or `pfsh`. See `pfexec`(1).

Roles are assumed through the Trusted Path menu. Successful assumption requires knowledge of the role's password and membership in the role. Role assignments are specified in `user_attr`(4).

EXAMPLES

EXAMPLE 1 Sample output

The output of the `roles` command has the following form:

```
example% roles tester01 tester02
tester01 : admin
tester02 : secadmin, root
example%
```

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`1` An error occurred.

FILES

`/etc/user_attr` Local source of extended attributes associated with users and roles.

`/etc/security/auth_attr` Local source for authorization names and descriptions.

`/etc/security/prof_attr` Local source for rights profile names, descriptions, and other attributes of profiles.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

Roles are assumed through the Trusted Path menu rather than the `su` command. To affect all name services, not just files, use the `smrole`(1M) command instead of the `roles` command.

SEE ALSO

Trusted Solaris 8 Reference Manual

`auths`(1), `profiles`(1), `smrole`(1M), `su`(1M), `getauusernam`(3BSM), `auth_attr`(4), `user_attr`(4)

SunOS 5.8 Reference Manual

`pfexec`(1), `rlogin`(1), `passwd`(4), `shadow`(4), `attributes`(5)

NAME

setfattrflag – Sets the file’s security attribute flags

SYNOPSIS

/usr/bin/setfattrflag `–m` [`–p` 0 | 1] *filename…*

/usr/bin/setfattrflag `–p` 0 | 1 [`–t`] *filename…*

DESCRIPTION

`setfattrflag` sets the security attributes flags of *filename*. For `setfattrflag` to successfully set directory flags, *filename* must be a directory. For `setfattrflag` to successfully set file-related flags, *filename* must be a file. At least one option is required. Setting a file’s public object security attribute flag requires the `file_audit` privilege. If the owner of the invoking process is not the owner of the file, the `file_owner` privilege is also required. At least one option is required.

This command works only on Trusted Solaris file systems. When used on other file systems (such as UFS), the command returns an error message.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

OPTIONS

`–m` Set the MLD flag on the directory. Once set, this flag cannot be cleared.

`–p` Set the file’s public object security attribute flag. A zero clears the flag, and a `1` sets the flag.

`–t` If *filename* is an MLD, translate to the underlying single-level directory. By default, `setfattrflag` does not translate multilevel directories to underlying single-level directories. This option is not allowed with the `–m` option.

RETURN VALUES

`setfattrflag` exits with one of the following values:

`0` Successful completion.

`1` Unsuccessful completion.

SEE ALSO

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

setfpriv – Change the privilege sets associated with a file

SYNOPSIS

/usr/bin/setfpriv {–s | –m | –d }–a *privseta* –f *privsetf file*…

DESCRIPTION

setfpriv changes the privilege sets of a file or files. The setfpriv command needs the file_setpriv privilege to succeed. Only the owner of a file can change the privilege sets associated with that file unless the command has the file_owner privilege. The user must have MAC write permission. DAC write permission is not required.

Refer to setfpriv(2) for a complete description of conditions to satisfy and privileges needed to execute this command.

The –s option sets the privileges to the entries specified on the command line. The –d option deletes one or more specified privileges from the file's privilege set. The –m option adds one or more specified privileges to the file's privilege set. One and only one of the options –s,–m, or –d must be specified.

The –a option specifies that a set of allowed privileges is to be set. The –f option specifies that a set of forced privileges is to be set. *privseta* and *privsetf* are one of these:

- A comma-separated list of privilege names as found in /usr/lib/tsol/locale/*locale_name*/priv_name. See the priv_desc(4) man page.
- A comma-separated list of numeric privilege IDs as found in </usr/include/sys/tsol/priv_names.h>.
- The keyword all to indicate all privileges.
- The keyword none to indicate an empty privilege set.

One or both of the options –a and –f must be specified, each followed by a privilege set. No white space may exist in a privilege-set list.

An attempt to assert a privilege in a file's forced set is denied unless that privilege is also asserted in the file's allowed set. All privileges cleared from a file's allowed set are automatically cleared from the file's forced set. It is not an error to attempt to clear a privilege from a set in which it is already cleared.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

EXAMPLES

EXAMPLE 1 Set all allowed privileges on a file

Setting privileges in the forced set requires that those privileges be set in the file's allowed set.

```
example% setfpriv –s –a all file1
```

Both the file's allowed and forced privilege sets can be set at the same time. To set all allowed privileges and a set of forced privileges on a file:

```
example% setfpriv –s –a all –f p1,p2,p3 file1
```

EXAMPLE 2 Set some allowed privileges on a file

```
example% setfpriv –s –a p1,p2,p3 file2
```

EXAMPLE 3 Add forced privileges to a file

```
example% setfpriv –m –f p1,p2,p3 file3
```

EXAMPLE 4 Delete privileges from a forced set on a file

```
example% setfpriv –d –f p1,p2,p3 file4
```

EXAMPLE 5 Set allowed privileges on one file from those of another

```
example% setfpriv –s –a'getfpriv –s –a file4' file5
```

RETURN VALUES

`setfpriv` exits with one of the following values:

0 Successful completion.

1 Unsuccessful completion.

SEE ALSO

Trusted Solaris 8 Reference Manual

`getfpriv`(1), `testfpriv`(1), `getfpriv`(2), `setfpriv`(2), `priv_desc`(4)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

setlabel – Sets the CMW label for files

SYNOPSIS

setlabel [–s][–h] *newlabel filename*…

DESCRIPTION

`setlabel` sets the CMW label associated with each *filename.* Unless *newlabel* and *filename* have been specified, no labels will be set. Incremental changes to labels are supported.

Refer to `setcmwlabel`(2) for a complete description of the conditions required to satisfy, and the privileges needed to execute this command.

Users may enter a label in plain text in the following form:

```
{ + } { classification } { { + | - } word } ...
```

Items in curly brackets are optional. A vertical bar (|) represents a choice between two items. Items followed by an ellipsis may be repeated zero or more times. Leading and trailing whitespace is ignored. Items may be separated by blanks, tabs, commas or slashes (/).

The system always displays labels in uppercase. Users may enter labels in any combination of uppercase and lowercase.

The classification part of the label must be a valid classification name as defined in `label_encodings`(4). Classification names may contain embedded blanks or punctuation, if they are so defined in the `label_encodings` file. Short and long forms of classification names may be used interchangeably.

The words *compartments* and *markings* used in labels must be valid words as defined in `label_encodings`. Words may contain embedded blanks or punctuation if they are so defined in `label_encodings`.

Short and long forms of words may be used interchangeably. Words may be specified in any order; however, they are processed left to right, so that where words conflict with each other, the word furthest to the right takes precedence.

EXTENDED DESCRIPTION

Plus and minus signs may be used when modifying an existing label. They turn on or off the compartments and markings associated with the words.

A CMW label is represented in characters in the form:

```
{ information label } { [sensitivity label] }
```

or

*

Information Labels (ILs) are obsolete. See NOTES.

Items in curly brackets are optional. Leading and trailing white space is ignored. Items may be separated by blanks, tabs, commas, or slashes (/).

The special case where the CMW label is an asterisk (*) represents a CMW label whose SL is to be set equal to its IL.

OPTIONS

No options Set the information label portion and the sensitivity label portion of the CMW label. Information Labels (ILs) are obsolete. See NOTES.

`–h` Set the label of the symbolic link.

`–s` Set the sensitivity label portion of the CMW label.

RETURN VALUES

`setlabel` exits with one of the following values:

`0` Successful completion.

`1` Usage error.

`2` Error in getting, setting or translating the label.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

USAGE

On the command line, enclose the label in double quotes unless the label you are entering is only one word. Without quotes, a second word or letter separated by a space is interpreted as a second argument. Labels containing the characters [and] should be in quotes to suppress the shell's use of those characters in filename substitution.

```
% setlabel –s SECRET somefile
% setlabel "[SECRET]" somefile
```

Use any combination of upper and lowercase letters. You may separate items in a label with blanks, tabs, commas or slashes (/). Do not use any other punctuation.

```
% setlabel "ADMIN_LOW[ts a b]" somefile
% setlabel "admin_low[ts,a,b]" somefile
% setlabel "admin_low[ts/a b]" somefile
```

When entering a full CMW label, enter the IL first, followed by the SL in brackets.

```
information label[sensitivity label]
```

Information Labels (ILs) are obsolete. See NOTES.

When entering an SL with a command option that sets the SL, you do not need to use brackets around the SL.

```
% setlabel –s " TOP SECRET A B" somefile
```

EXAMPLES

EXAMPLE 1 To set an SL

To set *somefile*'s SL to SECRET A:

```
example% setlabel "[Secret a]" somefile
```

EXAMPLE 2 To turn on or turn off a compartment

To turn on compartment B in *somefile*'s SL:

```
example% setlabel –s  +b somefile
```

To turn off compartment A in *somefile*'s SL:

```
example% setlabel –s -- –A somefile
```

NOTES

If an incremental change is being made to an existing label and the first character of the label is a hyphen (–), a preceding double-hyphen (--) is required; the double-hyphen must follow any of the –i, –s, and –h options. (See the examples.)

Information labels (ILs) are not supported in Trusted Solaris 7 and later releases. Trusted Solaris software interprets any ILs on communications and files from systems running earlier releases as ADMIN_LOW.

Objects still have CMW labels, and CMW labels still include the IL component: IL[SL]; however, the IL component is fixed at ADMIN_LOW.

As a result, Trusted Solaris 7 has the following characteristics:

- ILs do not display in window labels; SLs (Sensitivity Labels) display alone within brackets.
- ILs do not float.
- Setting an IL on an object has no effect, and getting an object's IL will always return ADMIN_LOW.
- Although certain utilities, library functions, and system calls can manipulate IL strings, the resulting ILs are always ADMIN_LOW, and cannot be set on any objects.
- Options related to information labels in the label_encodings(4) file can be ignored:

```
Markings Name= Marks;
Float Process Information Label;
```

SEE ALSO

Trusted Solaris 8 Reference Manual

setcmwlabel(2)

SunOS 5.8 Reference Manual

attributes(5)

NAME

nca, snca – the Solaris Network Cache and Accelerator (NCA)

DESCRIPTION

The Solaris Network Cache and Accelerator (“ NCA ”) is a kernel module designed to provide improved web server performance. The kernel module, `ncakmod` , services HTTP requests. To improve the performance of servicing HTTP requests, the NCA kernel module maintains an in-kernel cache of web pages. If the NCA kernel module cannot service the request itself, it passes the request to the `http` daemon (`httpd`) by means of a private interface. The logging facility, `ncalogd` , logs all requests. This private interface uses the Solaris Doors RPC mechanism. See, for example, `door_create`(3DOOR) , `door_call`(3DOOR) , and `door_bind`(3DOOR) .

The NCA cache consistency is maintained by honoring HTTP headers dealing with a given content type and expiration date, much the same way as a proxy cache.

The NCA is disabled in the Trusted Solaris environment.

For configuration information, see *System Administration Guide, Volume 3* .

NCA is intended to be run on a dedicated webserver. Running other large processes while running NCA may cause undesirable behavior.

FILES

`/etc/nca/ncakmod.conf`	Lists configuration parameters for NCA .
`/etc/nca/ncalogd.conf`	Lists configuration parameters for NCA logging.
`/etc/nca/nca.if`	Lists the physical interfaces on which NCA will run.
`/etc/hostname.{}{0-9}`	Lists all physical interfaces configured on the server.
`/etc/hosts`	Lists all host names associated with the server. Entries in this file must match with entries in `/etc/hostname.{}{0–9}` for NCA to function.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWncar
Interface Stability	Evolving

SUMMARY OF TRUSTED SOLARIS CHANGES

The Network Cache and Accelerator kernel module is disabled in the Trusted Solaris environment.

SEE ALSO

Trusted Solaris 8 Reference Manual

`door_create`(3DOOR) , `nca.if`(4)

SunOS 5.8 Reference Manual

`door_bind`(3DOOR) , `door_call`(3DOOR) , `ncakmod.conf`(4) , `attributes`(5)

System Administration Guide, Volume 3

NAME

tar – Create tape archives and add or extract files

SYNOPSIS

tar c [bBefFhiloPpTvwX [0-7]] [*block*] [*tarfile*] [*exclude-file*] {–`I` *include-file* |
–`C` | "directory | file" | *file* }…

tar r [bBefFhilPpTvw [0-7]] [*block*] {–`I` *include-file* | –`C` *directory* | *file* | file }…

tar t [BedfFhilTvX [0-7]] [*tarfile*] [*exclude-file*] {–`I` *include-file* | *file* }…

tar u [bBefFhilPpTvw [0-7]] [*block*] [*tarfile*] *file*…

tar x [BedfFhilmopTvwX [0-7]] [*tarfile*] [*exclude-file*] [*file*…]

DESCRIPTION

The `tar` command archives and extracts files to and from a single file called a *tarfile*. A tarfile is usually a magnetic tape, but it can be any file. `tar`'s actions are controlled by the *key* argument. The *key* is a string of characters containing exactly one function letter (`c`, `r`, `t`, `u`, or `x`) and zero or more function modifiers (letters or digits), depending on the function letter used. The *key* string contains no SPACE characters. Function modifier arguments are listed on the command line in the same order as their corresponding function modifiers appear in the *key* string.

The –`I` *include-file*, –`C` *directory file*, and *file* arguments specify which files or directories are to be archived or extracted. In all cases, appearance of a directory name refers to the files and, recursively, to subdirectories of that directory. Arguments appearing within braces (`{ }`) indicate that one of the arguments must be specified.

The `tar` command provides the functionality to create, update, list the table of contents, and extract a tarfile that contains extended Trusted Solaris security attributes, MLD and SLD information. The `tar` command also provides the compatibility support to list the table of contents and extract a Trusted Solaris 1.2 tarfile onto a Trusted Solaris 2.5.1 or 7 system. Two new function modifiers, `T` and `d`, are added to support these functions; see below for their descriptions.

The `tar` command operates on a single file called the tarfile. The tarfile is essentially a sequence of the archived files. Each archived file contains the information that is needed to restore a file. When the tarfile contains Trusted Solaris extended security attributes, MLD and SLD information, each archived file is preceded by its own ancillary file, which holds the extended security attributes, MLD and SLD information.

Without privileges, the `tar` command works within the Trusted Solaris security policy, which is enforced by the file system. When invoked by an ordinary user without privileges, `tar` works at a single sensitivity label and can be used only to create a tarfile at the sensitivity label of the current workspace.

OPTIONS

The following options are supported:

–`I` *include-file* Opens *include-file* containing a list of files, one per line, and treats it as if each file appeared separately on the command line. Be careful of trailing white spaces. Also beware of leading white spaces, since, for each line in the included file, the entire line (apart from the newline) will be used to match against the initial string of files to include. In the case where excluded files (see `X` function modifier) are also specified, they take precedence over all included files. If a file is specified in both the *exclude-file* and the *include-file* (or on the command line), it will be excluded.

–`C` *directory file* Performs a `chdir` (see `cd`(1)) operation on *directory* and performs the `c` (create) or `r` (replace) operation on *file*. Use short relative path names for *file*. If *file* is ‘`.`’, archive all files in *directory*. This option enables archiving files from multiple directories not related by a close common parent.

OPERANDS

The following operands are supported:

file A path name of a regular file or directory to be archived (when the `c`, `r` or `u` functions are specified), extracted (`x`) or listed (`t`). When *file* is the path name of a directory, the action applies to all of the files and (recursively) subdirectories of that directory.

When a file is archived, and the `E` flag (see `Function Modifiers`) is not specified, the filename cannot exceed 256 characters. In addition, it must be possible to split the name between parent directory names so that the prefix is no longer than 155 characters and the name is no longer than 100 characters. If `E` is specified, a name of up to `PATH_MAX` characters may be specified.

For example, a file whose basename is longer than 100 characters could not be archived without using the `E` flag. A file whose directory portion is 200 characters and whose basename is 50 characters could be archived (without using `E`) if a slash appears in the directory name somewhere in character positions 151-156.

Function Letters

The function portion of the key is specified by one of the following letters:

`c` Create. Writing begins at the beginning of the tarfile, instead of at the end.

`r` Replace. The named *file*s are written at the end of the tarfile. A file created with extended headers must be updated with extended headers (see `E` flag under `Function Modifiers`). A file created without extended headers cannot be modified with extended headers.

`t` Table of Contents. The names of the specified files are listed each time they occur in the tarfile. If no *file* argument is given, the names of all files in the tarfile are listed. With the `v` function modifier, additional information for the specified files is displayed.

`u` Update. The named *file*s are written at the end of the tarfile if they are not already in the tarfile, or if they have been modified since last written to that tarfile. An update can be rather slow. A tarfile created on a 5.x system cannot be updated on a 4.x system. A file created with extended headers must be updated with extended headers (see `E` flag under `Function Modifiers`). A file created without extended headers cannot be modified with extended headers.

`x` Extract or restore. The named *file*s are extracted from the tarfile and written to the directory specified in the tarfile, relative to the current directory. Use the relative path names of files and directories to be extracted. If a named file matches a directory whose contents has been written to the tarfile, this directory is recursively extracted. The owner, modification time, and mode are restored (if possible); otherwise, to restore owner, tar must be run with user ID of 0. Character-special and block-special devices (created by `mknod`(1M)) can only be extracted when the tar program has asserted the sys_devices privilege. If no *file* argument is given, the entire content of the tarfile is extracted. If the tarfile contains several files with the same name, each file is written to the appropriate directory, overwriting the previous one. Filename substitution wildcards cannot be used for extracting files from the archive; rather, use a command of the form:

```
tar xvf . . . /dev/rmt/0` tar tf . . . /dev/rmt/0 | grep 'pattern'`
```

When extracting tapes created with the `r` or `u` functions, directory modification times may not be set correctly. These same functions cannot be used with many tape drives due to tape drive limitations such as the absence of backspace or append capabilities.

When using the `r`, `u`, or `x` functions or the `X` function modifier, the named files must match exactly the corresponding files in the *tarfile.* For example, to extract `./`*thisfile,* you must specify `./`*thisfile,* and not *thisfile.* The `t` function displays how each file was archived.

Function Modifiers

The characters below may be used in conjunction with the letter that selects the desired function.

`b` Blocking Factor. Use when reading or writing to raw magnetic archives (see `f` below). The *block* argument specifies the number of 512-byte tape blocks to be included in each read or write operation performed on the tarfile. The minimum is 1, the default is 20. The maximum value is a function of the amount of memory available and the blocking requirements of the specific tape device involved (see mtio(7I) for details.) The maximum cannot exceed INT_MAX/512 (4194303).

When a tape archive is being read, its actual blocking factor will be automatically detected, provided that it is less than or equal to the nominal blocking factor (the value of the *block* argument, or the default value if the b modifier is not specified). If the actual blocking factor is greater than the nominal blocking factor, a read error will result. See Example 5 in EXAMPLES.

B Block. Force tar to perform multiple reads (if necessary) to read exactly enough bytes to fill a block. This function modifier enables tar to work across the Ethernet, since pipes and sockets return partial blocks even when more data is coming. When reading from standard input, '–', this function modifier is selected by default to ensure that tar can recover from short reads.

d The function modifier d indicates the tarfile is in Trusted Solaris 1.2 format. This function letter is not valid for the function letters c, r, or u. When this function modifier is used with the function letter t to display tarfile's contents, the tar program processes the input tarfile according to the Trusted Solaris 1.2 format. If the function modifier T is also specified, then the contents of the Trusted Solaris 1.2 tarfile is displayed with a line for each ancillary file and a line for each archived file. The line for an ancillary file has the same filename as its corresponding archived file, but it is suffixed by the string "(A)".

When this function modifier is used with the function letter x to extract a tarfile, the tar program processes the input tarfile according to the Trusted Solaris 1.2 format. If the function modifier T is also specified, the appropriate MLD , SLD information and extended security attributes (which are valid on Trusted Solaris 2.5.1 and 7 systems) are used to restore each archived file.

e Error. Exit immediately with a positive exit status if any unexpected errors occur. The SYSV3 environment variable overrides the default behavior. (See ENVIRONMENT section below.)

E Write a tarfile with extended headers. (Used with c, r, or u options; ignored with t or x options.) When a tarfile is written with extended headers, the modification time is maintained with a granularity of microseconds rather than seconds. In addition, filenames no longer than PATH_MAX characters that could not be archived without E, and file sizes greater than `8GB`, are supported. The `E` flag is required whenever the larger files and/or files with longer names, or whose `UID/GID` exceed `2097151`, are to be archived, or if time granularity of microseconds is desired.

`f` File. Use the *tarfile* argument as the name of the tarfile. If `f` is specified, `/etc/default/tar` is not searched. If `f` is omitted, `tar` will use the device indicated by the `TAPE` environment variable, if set; otherwise, it will use the default values defined in `/etc/default/tar`. If the name of the tarfile is '–', `tar` writes to the standard output or reads from the standard input, whichever is appropriate. `tar` can be used as the head or tail of a pipeline. `tar` can also be used to move hierarchies with the command:

```
example% cd fromdir; tar cf – .| (cd todir; tar xfBp –)
```

`F` With one `F` argument, `tar` excludes all directories named `SCCS` and `RCS` from the tarfile. With two arguments, `FF`, `tar` excludes all directories named `SCCS` and `RCS`, all files with `.o` as their suffix, and all files named `errs`, `core`, and `a.out`. The `SYSV3` environment variable overrides the default behavior. (See `ENVIRONMENT VARIABLES` section below.)

`h` Follow symbolic links as if they were normal files or directories. Normally, `tar` does not follow symbolic links.

`i` Ignore directory checksum errors.

`k` *size* Requires `tar` to use the size argument as the size of an archive in kilobytes. This is useful when the archive is intended for a fixed size device such as floppy disks. Large files are then split across volumes if they do not fit in the specified size.

`l` Link. Output error message if unable to resolve all links to the files being archived. If `l` is not specified, no error messages are printed.

`m` Modify. The modification time of the file is the time of extraction. This function modifier is valid only with the `x` function.

`n` The file being read is a non-tape device. Reading of the archive is faster since `tar` can randomly seek around the archive.

`o` Ownership. Assign to extracted files the user and group identifiers of the user running the program, rather than those on tarfile. This is the default behavior for users when tar is not being run with the user ID of 0. If the `o` function modifier is not set and the tar command's user ID is 0, the extracted files will take on the group and user identifiers of the files on tarfile (see `chown`(1) for more information). The `o` function modifier is only valid with the `x` function.

`p` Restore the named files to their original modes, and ACLs if applicable, ignoring the present `umask`(1). This is the default behavior if invoked by the user ID of 0 with the `x` function letter specified. If tar is invoked with the user ID of 0, SETUID and sticky information are also extracted, and files are restored with their original owners and permissions, rather than owned by root. When this function modifier is used with the `c` function, ACLs are created in the tarfile along with other information. Errors will occur when a tarfile with ACLs is extracted by previous versions of `tar`.

`P` Suppress the addition of a trailing "/" on directory entries in the archive.

`T` When this modifier is used with the function letter `c`, `r`, or `u` for creating, replacing or updating a tarfile, the extended security attributes, MLD and SLD information associated with each archived file are stored in the tarfile. The `tar` command also traverses any MLD it encounters. Hence, SLDs dominated by the `tar` process's sensitivity label are walked, or all SLDs are walked with certain privileges.

Specifying `T` implies the function modifier `p`.

When used with the function letter `t`, the `tarfile` content is displayed with a line for each ancillary file and a line for each archived file. The line for an ancillary file has the same filename as its corresponding archived file, but it is suffixed by the string "(A)".

When used with the function letter `x` for extracting a `tarfile`, the tar program attempts to restore each archived file using the MLD and SLD information, and the extended security attributes.

`q` Stop after extracting the first occurrence of the named file. `tar` will normally continue reading the archive after finding an occurrence of a file.

`v` Verbose. Output the name of each file preceded by the function letter. With the `t` function, `v` provides additional information about the tarfile entries. The listing is similar to the format produced by the `−l` option of the `ls`(1) command.

`w` What. Output the action to be taken and the name of the file, then await the user's confirmation. If the response is affirmative, the action is performed; otherwise, the action is not performed. This function modifier cannot be used with the `t` function.

`X` Exclude. Use the *exclude-file* argument as a file containing a list of relative path names for files (or directories) to be excluded from the tarfile when using the functions `c`, `x`, or `t`. Be careful of trailing white spaces. Also beware of leading white spaces, since, for each line in the excluded file, the entire line (apart from the newline) will be used to match against the initial string of files to exclude. Multiple `X` arguments may be used, with one *exclude-file* per argument. In the case where included files (see `−I` *include-file* option) are also specified, the excluded files take precedence over all included files. If a file is specified in both the *exclude-file* and the *include-file* (or on the command line), it will be excluded.

`[0-7]` Select an alternative drive on which the tape is mounted. The default entries are specified in `/etc/default/tar`. If no digit or `f` function modifier is specified, the entry in `/etc/default/tar` with digit "`0`" is the default.

USAGE

See `largefile`(5) for the description of the behavior of `tar` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

The automatic determination of the actual blocking factor may be fooled when reading from a pipe or a socket (see the `B` function modifier below).

1/4" streaming tape has an inherent blocking factor of one 512-byte block. It can be read or written using any blocking factor.

This function modifier works for archives on disk files and block special devices, among others, but is intended principally for tape devices.

For information on `tar` header format, see `archives`(4).

EXAMPLES

EXAMPLE 1 Using the tar Command to Create an Archive of Your Home Directory

The following is an example using `tar` to create an archive of your home directory on a tape mounted on drive `/dev/rmt/0`:

```
example% cd
example% tar cvf /dev/rmt/0 .
messages from  tar
```

The `c` function letter means create the archive; the `v` function modifier outputs messages explaining what `tar` is doing; the `f` function modifier indicates that the tarfile is being specified (`/dev/rmt/0` in this example). The dot (`.`) at the end of the command line indicates the current directory and is the argument of the `f` function modifier.

Display the table of contents of the tarfile with the following command:

```
example% tar tvf /dev/rmt/0
```

The output will be similar to the following for the POSIX locale:

```
rw–r––r––   1677/40    2123    Nov  7 18:15 1985    ./test.c
…
example%
```

The columns have the following meanings:

- column 1 is the access permissions to `./test.c`
- column 2 is the *user-id*/*group-id* of `./test.c`
- column 3 is the size of `./test.c` in bytes
- column 4 is the modification date of `./test.c`. When the `LC_TIME` category is not set to the POSIX locale, a different format and date order field may be used.
- column 5 is the name of `./test.c`

To extract files from the archive:

```
example% tar xvf /dev/rmt/0
messages from tar
example%
```

If there are multiple archive files on a tape, each is separated from the following one by an EOF marker. To have `tar` read the first and second archives from a tape with multiple archives on it, the *non-rewinding* version of the tape device name must be used with the `f` function modifier, as follows:

```
example% tar xvfp /dev/rmt/0n read first archive from tape
messages from tar example% tar xvfp /dev/rmt/0n read second archive from tape
messages from tar example%
```

Note that in some earlier releases, the above scenario did not work correctly, and intervention with `mt`(1) between `tar` invocations was necessary. To emulate the old behavior, use the non-rewind device name containing the letter `b` for BSD behavior. See the `Close Operations` section of the `mtio`(7I) manual page.

EXAMPLE 2 Using tar to Archive Files from /usr/include and from /etc to Default Tape Drive 0:

To archive files from `/usr/include` and from `/etc` to default tape drive `0`:

```
example% tar c –C /usr include –C /etc .
```

The table of contents from the resulting tarfile produces output like the following:

```
include/
include/a.out.h
and all the other files in /usr/include . . .
./chown and all the other files in /etc
```

To extract all files in the `include` directory:

```
example% tar xv include
x include/, 0 bytes, 0 tape blocksand all files under include . . .
```

EXAMPLE 3 Using tar to Transfer Files Across the Network

The following is an example using `tar` to transfer files across the network. First, here is how to archive files from the local machine (`example`) to a tape on a remote system (`host`):

```
example% tar cvfb  –  20 files |  rsh host dd of=/dev/rmt/0  obs=20b
  messages from tar
example%
```

In the example above, we are *creating* a *tarfile* with the `c` key letter, asking for *verbose* output from `tar` with the `v` function modifier, specifying the name of the output *tarfile* using the `f` function modifier (the standard output is where the *tarfile* appears, as indicated by the '–' sign), and specifying the blocksize (`20`) with the `b` function modifier. If you want to change the blocksize, you must change the blocksize arguments both on the `tar` command *and* on the `dd` command.

EXAMPLE 4 Using tar to Retrieve Files From a Tape on the Remote System Back to the Local System:

The following is an example that uses `tar` to retrieve files from a tape on the remote system back to the local system:

```
example% rsh –n host dd if=/dev/rmt/0 bs=20b | tar xvBfb – 20 files
messages from tar
example%
```

In the example above, we are *extracting* from the *tarfile* with the `x` key letter, asking for *verbose output from* `tar` with the `v` function modifier, telling `tar` it is reading from a pipe with the `B` function modifier, specifying the name of the input *tarfile* using the `f` function modifier (the standard input is where the *tarfile* appears, as indicated by the '–' sign), and specifying the blocksize (`20`) with the `b` function modifier.

EXAMPLE 5 Creating an Archive of the Home Directory on /dev/rmt/0 with a Blocking Factor of 19

The following example creates an archive of the home directory on `/dev/rmt/0` with an actual blocking factor of `19`:

```
example% tar cvfb /dev/rmt/0 19 $HOME
```

To recognize this archive's actual blocking factor without using the `b` function modifier:

```
example% tar tvf /dev/rmt/0
tar: blocksize = 19
. . .
```

To recognize this archive's actual blocking factor using a larger nominal blocking factor:

```
example% tar tvf /dev/rmt/0 30
tar: blocksize = 19
```

```
  . . .
```

To attempt to recognize this archive’s actual blocking factor using a nominal blocking factor that is too small:

```
  example% tar tvf /dev/rmt/0 10
  tar: tape read error
```

EXAMPLE 6 Creating a tar File with Extended Security Attributes

The following example uses `tar` to create a tarfile of the *tartest* directory and save the extended security attributes, MLD and SLD information.

```
  example% cd
  example% tar cvfT onetarfile tartest
```

The output will be similar to the following:

```
  a tartest/(A) 1K
  a tartest/ 0K
  a tartest/file1(A) 1K
  a tartest/file1 0K
  a tartest/mld1/(A) 1K
  a tartest/mld1/ 0K
  a tartest/mld1/(A) 1K
  a tartest/mld1/ 0K
  a tartest/mld1/file50(A) 1K
  a tartest/mld1/file50 1K
  …
```

The `c` function letter means create the archive; the `v` function modifier outputs messages explaining what `tar` is doing; the `f` function modifier indicates that the name of the `tarfile` to be created (`onetarfile` in this example). The `T` function modifier indicates that the extended security attributes, MLD and SLD information for each archived file are stored in the `tarfile`. The `tartest` is the name of the directory from which to create the `tarfile`.

The lines that end with (A) are the ancillary files for each archived file.

Display the table of contents of the tarfile (onetarfile in this example) with the following command:

```
example% tar tvfT onetarfile
```

The output will be similar to the following:

```
drwxr-xr-x 35436/10 54 Nov 11 17:07 1996 tartest/(A)
drwxr-xr-x+35436/10   0 Nov 11 17:07 1996 tartest/
-rw-r--r-- 35436/10  64 Nov 11 10:40 1996 tartest/file1(A)
-rw-r--r--+35436/10   0 Nov 11 10:40 1996 tartest/file1
drwxr-xr-x 35436/10 82 Nov 11 11:44 1996 tartest/mld1/(A)
drwxr-xr-x+35436/10 0 Nov 11 11:44 1996 tartest/mld1/
drwxr-xr-x 35436/10  87 Nov 11 11:33 1996 tartest/mld1/(A)
drwxr-xr-x+35436/10   0 Nov 11 11:33 1996 tartest/mld1/
```

```
-rw-r--r-- 35436/10 106 Nov 11 11:06 1996 tartest/mld1/file50(A)
-rw-r--r--+35436/10  17 Nov 11 11:06 1996 tartest/mld1/file50
…
```

The lines that end with (A) are ancillary files for each archived file.

Extract files from the tarfile (`onetarfile` in this example) with the following command:

```
example% tar xvfT onetarfile
```

The output will be similar to the following:

```
x tartest/(A), 54 bytes, 1 tape blocks
x tartest/, 0 bytes, 0 tape blocks
x tartest/file1(A), 64 bytes, 1 tape blocks
x tartest/file1, 0 bytes, 0 tape blocks
x tartest/mld1/(A), 82 bytes, 1 tape blocks
x tartest/.MLD.mld1/, 0 bytes, 0 tape blocks
x tartest/mld1/(A), 87 bytes, 1 tape blocks
x tartest/.MLD.mld1/.SLD.0/, 0 bytes, 0 tape blocks
x tartest/mld1/file50(A), 106 bytes, 1 tape blocks
x tartest/.MLD.mld1/.SLD.0/file50, 17 bytes, 1 tape blocks
…
```

The lines that end with (A) are ancillary files for each archived file.

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `tar`: `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_TIME`, `TZ`, and `NLSPATH`.

EXIT STATUS

The following exit values are returned:

`0` Successful completion.

`>0` An error occurred.

SUMMARY OF TRUSTED SOLARIS CHANGES

`tar` provides a function modifier `T` for creating, processing, and extracting a tarfile containing the extended security attributes, and MLD and SLD information. When an MLD is encountered in creating or updating a tarfile, the MLD is traversed according to the `tar` process's sensitivity label and privileges.

In addition, `tar` provides another function modifier for processing and extracting a tarfile created on a Trusted Solaris 1.2 system. The function modifier `d` can be used only with the function letters `t` and `x`.

MAC restrictions apply when `tar` is used. Appropriate privileges may be required to override access checks that are enforced for the create, update and extract operations.

For creating or updating a tarfile, one or more of the following privileges may be required: file_mac_read, file_mac_write, file_mac_search, file_dac_read, file_dac_write, file_dac_search, or sys_trans_label.

The extended security attributes that require privileges to restore, are restored when the appropriate privileges are present. Hence, to successfully extract files from a tarfile and restore the extended security attributes, one or more of the following privileges may be required: file_mac_read, file_mac_write, file_dac_read, file_dac_write, file_setdac, file_setid, file_chown, file_owner, file_downgrade_sl, file_upgrade_sl, file_setpriv, file_audit, sys_devices, or sys_trans_label.

FILES

/dev/rmt/[0-7][b][n]
/dev/rmt/[0-7]l[b][n]
/dev/rmt/[0-7]m[b][n]
/dev/rmt/[0-7]h[b][n]
/dev/rmt/[0-7]u[b][n]
/dev/rmt/[0-7]c[b][n]
/etc/default/tar

Settings may look like this:

```
archive0=/dev/rmt/0
archive1=/dev/rmt/0n
archive2=/dev/rmt/1
archive3=/dev/rmt/1n
archive4=/dev/rmt/0
archive5=/dev/rmt/0n
archive6=/dev/rmt/1
archive7=/dev/rmt/1n
```

/tmp/tar*

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	Enabled

SEE ALSO

Trusted Solaris 8 Reference Manual

chown(1), label_encodings(4)

SunOS 5.8 Reference Manual

ar(1), basename(1), cd(1), cpio(1), csh(1), dirname(1), ls(1), mt(1), pax(1), setfacl(1), umask(1), mknod(1M), vold(1M), archives(4), attributes(5), environ(5), largefile(5), mtio(7I)

DIAGNOSTICS

Diagnostic messages are output for bad key characters and tape read / write errors, and for insufficient memory to hold the link tables.

NOTES

There is no way to access the *n*th occurrence of a file.

Tape errors are handled ungracefully.

When the Volume Management daemon is running, accesses to floppy devices through the conventional device names (for example, `/dev/rdiskette`) may not succeed. See `vold`(1M) for further details.

The `tar` archive format allows UIDs and GIDs up to `2097151` to be stored in the archive header. Files with UIDs and GIDs greater than this value will be archived with the UID and GID of `60001`.

If an archive is created that contains files whose names were created by processes running in multiple locales, a single locale that uses a full 8-bit codeset (for example, the `en_US` locale) should be used both to create the archive and to extract files from the archive.

Notes for function modifier `T` and `d`:

For Trusted Solaris 1.2, Trusted Solaris 2.5.1, and Trusted Solaris 7 tarfiles, a compatible `label_encodings`(4) file is expected between the time the tarfile is created or updated and the time the tarfile is extracted.

When a Trusted Solaris 1.2 tarfile is restored on a Trusted Solaris 2.5.1 system, the label `SYSTEM_HIGH` is mapped to the label `ADMIN_HIGH`, and the label `SYSTEM_LOW` is mapped to the label `ADMIN_LOW`. In addition, the privileges and file audit mask are not used for the restored files because their formats are not compatible with Trusted Solaris 2.5.1 and 7’s equivalent security attributes.

If the name of the linked file in a symbolic link contains explicitly adorned MLD names and / or SLD names, it may no longer be a valid pathname after extraction. The reason is that the MLD adornment and SLD name at the time the tarfile is created or updated might be different than they are at the time the tarfile is extracted. At extraction time, `tar` attempts to update the link pathname of the symbolic link with the proper MLD adornment and SLD name. If `tar` fails, an error message is issued. Users need to perform any corrections themselves after the extraction is done.

Extracting a Trusted Solaris 2.5.1 tarfile on a standard Solaris 2.5 system may cause directory-checksum errors. Use the `−i` option, which ignores directory-checksum errors, to get around this problem.

NAME

testfpriv – Check or test the privilege sets associated with a file

SYNOPSIS

/usr/bin/testfpriv [–s] [[–e] [–a *privseta*]] [[–e] –f *privsetf*] *filename*

DESCRIPTION

`testfpriv` checks or tests the privilege sets of a file or files. The command must have MAC read permission.

privseta and *privsetf* are one of these:

- A comma-separated list of privilege names as reported by `getfpriv`
- A comma-separated list of numeric privilege IDs as found in `</usr/include/sys/tsol/priv_names.h>`
- The keyword `all` to indicate all privileges

No whitespace may exist in either list.

Without the –e (equal) option, the specified set of privileges is checked as a subset of the forced or the allowed privileges specified on the command line. The `testfpriv` function reports those privileges that are specified in *privseta* and *privsetf* but not found in the allowed or forced sets of the file. The –e option also reports privileges that the file has but that were not specified in the `testfpriv` command.

The privilege sets of each named file are checked according to options described in the next section.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWtsu

OPTIONS

–a Test whether *privseta* is either equal to or a subset of the allowed set of *filename*.

–e Test the equality of *privset* and the privilege set of *filename*.

–f Test whether *privsetf* is either equal to or a subset of the forced set of *filename*.

–s Use silent mode to suppress output. (This option is useful in shell scripts that need only the return value.)

RETURN VALUES

`testfpriv` exits with one of these values:

0 Specified privileges are in the allowed or the forced set of the file. With the –e option, the specified privileges are equal to the allowed set or the forced set of the file.

1 The specified privileges are not in the allowed set of the file, or (with –e) the allowed set of the file contains privileges not specified in this command.

2 The specified privileges are not in the forced set of the file, or (with –e) the forced set of the file contains privileges not specified in this command.

3 Both the allowed and forced sets have mismatches as described for return values 1 and 2.

4 `testfpriv` completed unsuccessfully.

EXAMPLES

EXAMPLE 1 Determine privileges in the forced set of a file

To determine if a set of privileges is in the forced set of a file, use this command:

```
example%testfpriv –f p1,p2,p3 file1
```

If all the specified privileges are in the forced set of the file, no output is returned. If any of the privileges is not in the forced set of the file, the function displays the missing privilege(s). For example,

```
example% file1:missing:p2
```

EXAMPLE 2 Test a file’s forced and allowed sets

To test if a file’s forced and allowed sets are exactly equal to the specified privileges, use this command:

```
example%testfpriv -e -f p1 -e -a p2 file2
```

If the file’s privileges did not match the specified privileges exactly, the output could be in this format:

```
example% file3:forced:extra:p3:allowed:missing:p2:extra:p4
```

EXAMPLE 3 Test both the allowed and the forced sets

For example, use this command to test for all bits on in the allowed set, and whether only *p1* and *p2* are present in the forced set:

```
example% testfpriv -s -e -a all -f p1,p2 file4
```

Because this example uses the silent mode, no output is returned. The returned exit value demonstrates the result.

SEE ALSO

Trusted Solaris 8 Reference Manual

`getfpriv`(1), `setfpriv`(1), `getfpriv`(2), `setfpriv`(2)

SunOS 5.8 Reference Manual

`attributes`(5)

NAME

find, tfind – Find files

SYNOPSIS

find *path… expression*

tfind *path… expression*

DESCRIPTION

The `find` utility recursively descends the directory hierarchy for each *path* , seeking files that match a Boolean *expression* written in the primaries given below.

`find` can descend to arbitrary depths in a file hierarchy and will not fail due to path length limitations (unless a *path* operand specified by the application exceeds `PATH_MAX` requirements).

The `tfind` command supports execution of commands in restricted environments defined by the profile-shell mechanism. `tfind` contains all the functionality of `find` , except for the expressions `–exec` *command* and `–ok` *command* . For these expressions `tfind` invokes *command* through the profile shell (`pfexec`(1)).

OPERANDS

The following operands are supported:

path A path name of a starting point in the directory hierarchy.

expression The first argument that starts with a – , or is a `!` or a `(` , and all subsequent arguments will be interpreted as an *expression* made up of the following primaries and operators. In the descriptions, wherever *n* is used as a primary argument, it will be interpreted as a decimal integer optionally preceded by a plus (`+`) or minus (`-`) sign, as follows:

+*n*	more than *n*
n	exactly *n*
`–n`	less than *n*

Expressions

Valid expressions are:

`–atime` *n* True if the file was accessed *n* days ago. The access time of directories in *path* is changed by `find` itself.

`–cpio` *device* Always true; write the current file on *device* in `cpio` format (5120-byte records).

`–ctime` *n* True if the file's status was changed *n* days ago.

`–depth` Always true; causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when `find` is used with `cpio`(1) to transfer files that are contained in directories without write permission.

`–exec` *command* True if the executed *command* returns a zero value as exit status. The end of *command* must be punctuated by an escaped semicolon. A command argument `{}` is replaced by the current path name. If issued from `tfind` , the command is invoked through a profile shell (`pfsh`).

`–follow` Always true; causes symbolic links to be followed. When following symbolic links, `find` keeps track of the directories visited so that it can detect infinite loops; for example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the `–type l` expression.

`–fstype` *type* True if the filesystem to which the file belongs is of type *type* .

`–group` *gname* True if the file belongs to the group *gname* . If *gname* is numeric and does not appear in the `/etc/group` file, or in the NIS / NIS+ tables, it is taken as a group ID .

`–inum` *n* True if the file has inode number *n* .

`–links` *n* True if the file has *n* links.

`–local` True if the file system type is not a remote file system type as defined in the `/etc/dfs/fstypes` file. `nfs` is used as the default remote filesystem type if the `/etc/dfs/fstypes` file is not present. Note that `–local` will descend the hierarchy of non-local directories. See EXAMPLES for an example of how to search for local files without descending.

`–ls` Always true; prints current path name together with its associated statistics. These include (respectively):

- inode number
- size in kilobytes (1024 bytes)
- protection mode
- number of hard links

- user
- group
- size in bytes
- modification time.

If the file is a special file the size field will instead contain the major and minor device numbers.

If the file is a symbolic link the pathname of the linked-to file is printed preceded by ‘ → ’. The format is identical to that of **ls –gilds** (see ls(1B)). Note: Formatting is done internally, without executing the ls program.

–M
In all multilevel directories (MLD) encountered, search single-level directories (SLD s) that are dominated by the sensitivity label of the process. However, if the effective privilege set of the process contains the file_mac_read and file_mac_search privileges, search all SLD s. The file system enforces all underlying DAC policies and privilege interpretations.

If –M is *not* specified and *path* points to an adorned MLD, traverse only this MLD ’s SLD s. For all other MLD s encountered, automatically translate to the SLD at the sensitivity label of the process even if find is run with all privileges.

If –M is *not* specified and *path* points to an unadorned MLD , for this and all other MLD s encountered, automatically translate to the SLD at the sensitivity label of the process even if find is run with all privileges.

If –M is *not* specified and *path* does not point to an MLD , for all MLD s encountered, automatically translate to the SLD at the sensitivity label of the process even if find is run with all privileges.

–mount
Always true; restricts the search to the file system containing the directory specified. Does not list mount points to other file systems.

–mtime
True if the file’s data was modified *n* days ago.

n

`–name` *pattern*
: True if *pattern* matches the current file name. Normal shell file name generation characters (see `sh`(1)) may be used. A backslash (`\\`) is used as an escape character within the pattern. The pattern should be escaped or quoted when `find` is invoked from the shell.

`–ncpio` *device*
: Always true; write the current file on *device* in `cpio –c` format (5120 byte records).

`–newer` *file*
: True if the current file has been modified more recently than the argument *file* .

`–nogroup`
: True if the file belongs to a group not in the `/etc/group` file, or in the NIS / NIS+ tables.

`–nouser`
: True if the file belongs to a user not in the `/etc/passwd` file, or in the NIS / NIS+ tables.

`–ok` *command*
: Like `–exec` except that the generated command line is printed with a question mark first and is executed only if the user responds by typing **`y`** . If issued from `tfind` , `command` is invoked through a profile shell (`pfsh`).

`–perm` [–] *mode*
: The *mode* argument is used to represent file mode bits. It will be identical in format to the < *symbolic* mode > operand described in `chmod`(1) , and will be interpreted as follows. To start, a template will be assumed with all file mode bits cleared. An *op* symbol of:

`+` Will set the appropriate mode bits in the template.

– Will clear the appropriate bits.

`=` Will set the appropriate mode bits, without regard to the contents of the file mode creation mask of a process.

The *op* symbol of – cannot be the first character of *mode* ; this restriction avoids ambiguity with the optional leading hyphen. Because the initial mode is all bits off, there are no symbolic modes that need to use – as the first character.

If the hyphen is omitted, the primary will evaluate as true when the file permission bits exactly match the value of the resulting template.

Otherwise, if *mode* is prefixed by a hyphen, the primary will evaluate as true if at least all the bits in the resulting template are set in the file permission bits.

`−perm` [−] *onum* True if the file permission flags exactly match the octal number *onum* . [See `chmod`(1) .] If *onum* is prefixed by a minus sign (`−`), only the bits that are set in *onum* are compared with the file-permission flags, and the expression evaluates true if they match.

`−print` Always true; causes the current path name to be printed.

`−prune` Always yields true. Do not examine any directories or files in the directory structure below the *pattern* just matched. (See `EXAMPLES`). Specifying `−depth` overrides the `−prune` option, which will have no effect.

`−size` *n* [`c`] True if the file is *n* blocks long (512 bytes per block). If *n* is followed by a `c` , the size is in bytes.

`−type` *c* True if the type of the file is *c* , where *c* is `b` , `c` , `d` , `D` , `f` , `l` , `m` , `p` , or `s` for block special file, character special file, directory, door, plain file, symbolic link, MLD , FIFO(named pipe), or socket, respectively.

`−user` *uname* True if the file belongs to the user *uname* . If *uname* is numeric and does not appear as a login name in the `/etc/passwd` file, it is taken as a user ID .

True if the file belongs to the user *uname* . If *uname* is numeric and does not appear as a login name in the `/etc/passwd` file, or in the NIS / NIS+ tables, it is taken as a user ID .

`−xdev` Same as the `−mount` primary.

Complex Expressions

The primaries may be combined using the following operators (in order of decreasing precedence):

1) (*expression*) True if the parenthesized expression is true. (Parentheses are special to the shell and must be escaped.)

2) ! *expression* — The negation of a primary (! is the unary *not* operator).

3) *expression* [–a] *expression* — Concatenation of primaries (the AND operation is implied by the juxtaposition of two primaries).

4) *expression* –o *expression* — Alternation of primaries (–o is the OR operator).

Note: When you use `find` in conjunction with `cpio` , if you use the `–L` option with `cpio` , then you must use the `–follow` expression with `find` and vice versa, otherwise there will be undesirable results.

If no *expression* is present, `–print` is used as the expression. Otherwise, if the given expression does not contain any of the primaries `–exec` , `–ok` , or `–print` , the given expression will be effectively replaced by

(*given_expression*) `–print`

The `–user` , `–group` , and `–newer` primaries each will evaluate their respective arguments only once. Invocation of *command* specified by `–exec` or `–ok` does not affect subsequent primaries on the same file.

USAGE

See `largefile`(5) for the description of the behavior of `find` when encountering files greater than or equal to 2 Gbyte (2^{31} bytes).

EXAMPLES

EXAMPLE 1 Writing out the hierarchy directory

The following commands are equivalent:

```
  example% find .  example% find . –print
```

They both write out the entire directory hierarchy from the current directory.

EXAMPLE 2 Removing files

Remove all files in your home directory named `a.out` or `*.o` that have not been accessed for a week:

```
  example% find $HOME \\( -name a.out -o -name *.ó \\) \\
  -atime +7 -exec rm {} \\;
```

EXAMPLE 3 Printing all file names but skipping SCCS directories

Recursively print all file names in the current directory and below, but skipping SCCS directories:

```
example% find . –name SCCS –prune –o –print
```

EXAMPLE 4 Printing all file names and the SCCS directory name

Recursively print all file names in the current directory and below, skipping the contents of SCCS directories, but printing out the SCCS directory name:

```
example% find . –print –name SCCS –prune
```

EXAMPLE 5 Testing for the newer file

The following command is roughly equivalent to the –`nt` extension to `test`(1) :

```
example$ if [ -n "$(find file1 -prune -newer file2)"
]; then   printf %s\\\
 "file1 is newer than file2"
```

EXAMPLE 6 Selecting a file using 24–hour mode

The descriptions of –`atime` , –`ctime` , and –`mtime` use the terminology *n* "24-hour periods". For example, a file accessed at 23:59 will be selected by:

```
example% find . –atime –1 –print
```

at 00:01 the next day (less than 24 hours later, not more than one day ago). The midnight boundary between days has no effect on the 24-hour calculation.

EXAMPLE 7 Finding files by a literal in their names

Find files with "abc" in their names; search all SLD s dominated by the sensitivity label as the `find` process:

```
example% find begin_path –M –type f –name '*abc*'
```

EXAMPLE 8 Traversing directories by sensitivity label

Find MLD s with "xyz" in their names; search all SLD s dominated by the sensitivity label as the `find` process:

```
example% find begin_path –M –type m –name '*xyz*'
```

EXAMPLE 9 Removing files with "abc" in their names

Remove files with "abc" in their names; begin at the current directory and perform the removal through a profile shell (`pfsh`).

```
example% tfind . –type f –name '*abc*' –exec rm {} \\;
```

EXAMPLE 10 Printing files matching a user's permission mode

Recursively print all file names whose permission mode exactly matches read, write, and execute access for user, and read and execute access for group and other:

```
example% find . –perm u=rwx,g=rx,o=rx
```

The above could alternatively be specified as follows:

```
example% find . –perm a=rwx,g-w,o-w
```

EXAMPLE 11 Printing files with write access for other

Recursively print all file names whose permission includes, but is not limited to, write access for other:

```
example% find . -perm -o+w
```

EXAMPLE 12 Printing local files without descending non-local directories

```
example% find . ! -local -prune -o -print
```

ENVIRONMENT VARIABLES

See `environ`(5) for descriptions of the following environment variables that affect the execution of `find` : `LC_COLLATE` , `LC_CTYPE` , `LC_MESSAGES` , `LC_TIME` , and `LC_ALL` , and `NLSPATH` .

EXIT STATUS

The following exit values are returned:

`0` All *path* operands were traversed successfully.

`>0` An error occurred.

SUMMARY OF TRUSTED SOLARIS CHANGES

Modifications to the `find` command deal with multilevel directories. A new `–M` option enables traversing MLD s. A new argument (`m`) for the `–type` option enables selecting the MLD type.

FILES

`/etc/passwd` Password file

`/etc/group` Group file

`/etc/dfs/fstypes` File that registers distributed file system packages

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu
CSI	enabled

SEE ALSO

Trusted Solaris 8 Reference Manual

`chmod`(1) , `stat`(2)

SunOS 5.8 Reference Manual

`cpio`(1) , `ls`(1) , `pfexec`(1) , `sh`(1) , `test`(1) , `umask`(2) , `attributes`(5) , `environ`(5) , `largefile`(5)

WARNINGS

The following options are obsolete and will not be supported in future releases:

–`cpio` *device* Always true; write the current file on *device* in `cpio` format (5120-byte records).

–`ncpio` *device* Always true; write the current file on *device* in `cpio` –`c` format (5120 byte records).

NOTES

When using `find` to determine files modified within a range of time, one must use the –`time` argument *before* the –`print` argument; otherwise, `find` will give all files.

NAME

uname – Print name of current system

SYNOPSIS

uname [–aimnprsvX]

uname [–S *system_name*]

DESCRIPTION

The uname utility prints information about the current system on the standard output. When options are specified, symbols representing one or more system characteristics will be written to the standard output. If no options are specified, uname prints the current operating system's name. The options print selected information returned by uname(2), sysinfo(2), or both.

OPTIONS

The following options are supported:

–a Print basic information currently available from the system.

–i Print the name of the hardware implementation (platform).

–m Print the machine hardware name (class). Use of this option is discouraged; use uname –p instead. See NOTES section below.

–n Print the node name (the node name is the name by which the system is known to a communications network).

–p Print the current host's ISA or processor type.

–r Print the operating system release level.

–s Print the name of the operating system. This is the default.

–v Print the operating system version.

–X Print expanded system information, one information element per line, as expected by SCO UNIX. The displayed information includes:

- system name, node, release, version, machine, and number of CPUs
- BusType, Serial, and Users (set to "unknown" in Solaris)
- OEM# and Origin# (set to 0 and 1, respectively)

–S *system_name* The node name may be changed by specifying a *system_name* argument. The *system_name* argument is restricted to SYS_NMLN characters. SYS_NMLN is an implementation-specific value defined in <sys/utsname.h>.

To succeed with the –S option in the Trusted Solaris environment, this command needs the sys_net_config privilege. If a user other than root attempts this option, the command also needs the file_dac_read, file_dac_write, file_mac_read, and file_mac_write privileges to update the /etc/nodename file.

EXAMPLES

EXAMPLE 1 Using The uname Command

The following command:

```
example% uname −sr
```

prints the operating system name and release level, separated by one SPACE character.

ENVIRONMENT VARIABLES

SYSV3 This variable is used to override the default behavior of uname. This is necessary to make it possible for some INTERACTIVE UNIX Systems and SCO UNIX programs and scripts to work properly. Many scripts use uname to determine the OS type or the version of the OS to ensure software is compatible with that OS. Setting SYSV3 to an empty string will make uname print the following default values:

```
nodename nodename 3.2 2 i386
```

The individual elements that uname displays can also be modified by setting SYSV3 in the following format:

os,sysname,node,rel,ver,mach

os	Operating system (IUS or SCO)
sysname	System name
node	Nodename as displayed by the −n option
rel	Release level as displayed by the −r option
ver	Version number as displayed by the −v option
mach	Machine name as displayed by −m option

Do not put spaces between the elements. If an element is omitted, the current system value will be used.

See environ(5) for descriptions of the following environment variables that affect the execution of uname: LC_CTYPE, LC_MESSAGES, and NLSPATH.

EXIT STATUS

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

To succeed with the `–S` option, this command needs the `sys_net_config` privilege. If a user other than root attempts this option, the command also needs the `file_dac_read`, `file_dac_write`, `file_mac_read`, and `file_mac_write` privileges to update the `/etc/nodename` file.

SEE ALSO

Trusted Solaris 8 Reference Manual

`sysinfo`(2)

SunOS 5.8 Reference Manual

`arch`(1), `isalist`(1), `uname`(2), `attributes`(5), `environ`(5)

NOTES

Independent software vendors (ISVs) and others who need to determine detailed characteristics of the platform on which their software is either being installed or executed should use the `uname` command.

To determine the operating system name and release level, use `uname –sr`. To determine only the operating system release level, use `uname –r`. Note that operating system release levels are not guaranteed to be in *x.y* format, such as 5.3, 5.4, 5.5, and so on; future releases may be in the *x.y.z* format, such as 5.3.1, 5.3.2, 5.4.1 and so on.

In SunOS 4.*x* releases, the `arch`(1) command was often used to obtain information similar to that obtained by using the `uname` command. The `arch`(1) command output "sun4" was often incorrectly interpreted to signify a SunOS SPARC system. If hardware platform information is desired, use `uname –sp`.

The `arch –k` and `uname –m` commands return equivalent values; however, the use of either of these commands by third party programs is discouraged, as is the use of the `arch` command in general. To determine the machine’s Instruction Set Architecture (ISA or processor type), use `uname` with the `–p` option.

NAME

vacation – Reply to mail automatically

SYNOPSIS

vacation [–I]

vacation [–a *alias*] [–f *database_file*] [–j] [–m *message_file*] [–s *sender*] [–t*N*] *username*

DESCRIPTION

The `vacation` utility automatically replies to incoming mail.

Installation

The installation consists of an interactive program which sets up `vacation`'s basic configuration.

To install `vacation`, type it with no arguments on the command line. The program creates a `.vacation.msg` file, which contains the message that is automatically sent to all senders when `vacation` is enabled, and starts an editor for you to modify the message. (See `USAGE` section.) Which editor is invoked is determined by the `VISUAL` or `EDITOR` environment variable, or `vi`(1) if neither of those environment variables are set.

A `.forward` file is also created if one does not exist in your home directory. Once created, the `.forward` file will contain a line of the form:

```
\username, "|/usr/bin/vacation username"
```

One copy of an incoming message is sent to the *username* and another copy is piped into `vacation`.

If a `.forward` file is present in your home directory, it will ask whether you want to remove it, which disables `vacation` and ends the installation.

The program automatically creates `.vacation.pag` and `.vacation.dir`, which contain a list of senders when `vacation` is enabled.

Activation and Deactivation

The presence of the `.forward` file determines whether or not `vacation` is disabled or enabled. To disable `vacation`, remove the `.forward` file, or move it to a new name.

Initialization

The –I option clears the `vacation` log files, `.vacation.pag` and `.vacation.dir`, erasing the list of senders from a previous `vacation` session. (See `OPTIONS` section).

Additional Configuration

`vacation` provides configuration options that are not part of the installation, these being –a, –f, –j, –m, –s, and –t. (See `OPTIONS` section).

OPTIONS

The following options are supported:

–I Initialize the `.vacation.pag` and `.vacation.dir` files and enables `vacation`. If the –I flag is not specified, and a *user* argument is given, `vacation` reads the first line from the standard input (for a `From:` line, no colon). If absent, it produces an error message.

Options `–a`, `–f`, `–j`, `–m`, `–t`, and `–s` are configuration options to be used in conjunction with `vacation` in the `.forward` file, not on the command line. For example,

```
\username, "|/usr/bin/vacation –t1m username"
```

repeats replies to the sender every minute.

`–a` *alias* Indicates that *alias* is one of the valid aliases for the user running `vacation`, so that mail addressed to that alias generates a reply.

`–f` *file* Uses *file* instead of `.vacation` as the base name for the database file.

`–j` Does not check whether the recipient appears in the `To:` or the `Cc:` line. Warning: use of this option can result in vacation replies being sent to mailing lists and other inappropriate places; its use is therefore strongly discouraged.

`–m` *file* Uses *file* instead of `.vacation.msg` as the message to send for the reply.

`–s` *sender* Replies to *sender* instead of the value read from the UNIX `From` line of the incoming message.

`–t`*N* Changes the interval between repeat replies to the same sender. The default is 1 week. A trailing `s`, `m`, `h`, `d`, or `w` scales *N* to seconds, minutes, hours, days, or weeks, respectively.

USAGE

Files

`.vacation.msg` should include a header with at least a `Subject:` line (it should not include a `From:` or a `To:` line). For example:

```
Subject: I am on vacation
I am on vacation until July 22.  If you have something urgent,
please contact Jo Jones (jones@fB0).
    --Jonni
```

If the string `$SUBJECT` appears in the `.vacation.msg` file, it is replaced with the subject of the original message when the reply is sent; thus, a `.vacation.msg` file such as

```
Subject: I am on vacation
I am on vacation until July 22.
```

```
Your mail regarding "$SUBJECT" will be read when I return.
If you have something urgent, please contact
Jo Jones (jones@fB0).
    --Jonni
```

will include the subject of the message in the reply.

No message is sent if the `To:` or the `Cc:` line does not list the user to whom the original message was sent or one of a number of aliases for them, if the initial `From:` line includes the string –`REQUEST@`, or if a `Precedence: bulk` or `Precedence: junk` line is included in the header.

`vacation` will also not respond to mail from either `postmaster` or `Mailer-Daemon`.

FILES

`~/.forward`	File that replies to sender when user is on vacation.
`~/.vacation.msg`	File that contains body of the message sent to sender.

A list of senders is kept in the `dbm` format files `.vacation.pag` and `.vacation.dir` in your home directory. These files are `dbm` files and cannot be viewed directly with text editors.

ATTRIBUTES

See `attributes`(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Availability	SUNWcsu

SUMMARY OF TRUSTED SOLARIS CHANGES

Enable `vacation` processing at every label at which you want to receive mail and respond with a vacation message.

SEE ALSO

Trusted Solaris 8 Reference Manual

`sendmail`(1M)

SunOS 5.8 Reference Manual

`vi`(1), `dbm`(3UCB), `getusershell`(3C), `aliases`(4), `shells`(4), `attributes`(5)

Index