# man pages section 5: Standards, Environments, and Macros

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–0884–10
May 2003


030523@5943

# 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.

# Introduction

**NAME**

Intro – introduction to miscellany

**DESCRIPTION**

The Trusted Solaris privilege macros and modified PAM modules are described in this section.

**Note –** The printed *Trusted Solaris 8 HW 12/02 Reference Manual* includes only those man pages that have been modified or originate in the Trusted Solaris environment. Printed versions of unchanged `SunOS` 5.8 man pages are found in the *SunOS 5.8 Reference Manual*. To read the Solaris pages in AnswerBook2™, go to *man pages section 5*.

Trusted Solaris terms used on the man pages are defined in the `DEFINITIONS` section of the `Intro`(1) and `Intro`(2) man pages and explained further in the *Trusted Solaris User's Guide*, the *Trusted Solaris Developer's Guide* and the *Trusted Solaris Administrator's Procedures* guide.

Among the topics presented in this section are:

| | |
|---|---|
| Headers | The header (`.h`) files `fcntl`, `floatingpoint`, `math`, `langinfo`, `nl_types`, `siginfo`, `signal`, `stat`, `stdarg`, `types`, `ucontext`, `values`, `varargs`, and `wait` (on the `wstat` page) are described. |
| Environments | The user environment (`environ`), the subset of the user environment that depends on language and cultural conventions (`locale`), the large file compilation environment (`lfcompile`), and the transitional compilation environment (`lfcompile64`) are described. |
| Macros | The macros to format Reference Manual pages (`man` and `mansun`) as well as other text format macros (`me`, `mm`, and `ms`) are described. |
| Characters | Tables of character sets (`ascii`, `charmap`, `eqnchar`, and `iconv`), file format notation (`formats`), file name pattern matching (`fnmatch`), and regular expressions (`regex` and `regexp`) are presented. |
| FNS | Topics concerning the Federated Naming Service (`fns`, `fns_initial_context`, `fns_policies`, and `fns_references`) are discussed. |
| Standards | The POSIX (IEEE) Standards and the X/Open Specifications are described on the `standards` page. |

# Standards, Environments, and Macros

**NAME**

pam_tp_auth – authentication PAM module for Trusted Solaris

**SYNOPSIS**

```
/usr/lib/security/pam_tp_auth.so.1
```

**DESCRIPTION**

The Trusted Solaris service module for PAM, `/usr/lib/security/pam_tp_auth.so.1`, provides functionality for one PAM module: authentication. The `pam_tp_auth.so.1` module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

**Authentication Module**

The authentication component of `pam_tp_auth.so` provides a function that checks for the trusted path of the process that is requesting authentication.

The following options may be passed to the function:

`su_auth_check_on` Enforce authorization checks for `PAM_USER`.

**ATTRIBUTES**

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

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---|---|
| MT Level | MT-Safe with exceptions |

**Trusted Solaris 8 HW 12/02 Reference Manual**
**SunOS 5.8 Reference Manual**

`pam_tsol`(5)

`keylogin`(1), `libpam`(3LIB), `pam`(3PAM), `pam_authenticate`(3PAM), `pam_setcred`(3PAM), `pam.conf`(4), `attributes`(5)

**NOTES**

The interfaces in `libpam()` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

**NAME**

pam_tsol – authentication, account, and password management PAM modules for Trusted Solaris

**SYNOPSIS**

`/usr/lib/security/pam_tsol.so.1`

**DESCRIPTION**

The Trusted Solaris service module for PAM, `/usr/lib/security/pam_tsol.so.1`, provides functionality for three PAM modules: authentication, account management, and password management. The `pam_tsol.so.1` module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

**Authentication Module Management**

The Trusted Solaris authentication management component provides a function to verify the identity of a user, (`pam_sm_authenticate()`). This provides an additional check for role authentication. It prevents direct role logins and indirect role logins from nontrusted clients.

The following options may be passed to `pam_sm_authenticate()`:

| | |
|---|---|
| `secondary_login` | Indicates that the service is a secondary login. |
| `trustedpath` | Passes the information that the trusted path is set in the remote client process. |

**Account Management Module**

The Trusted Solaris account management component provides a function to perform account management, `pam_sm_acct_mgmt()`. The function checks whether the users account is locked, and if it is locked, `pam_sm_acct_mgmt()` denies access. It also checks whether the account is disabled and if so, checks whether the user is authorized to enable logins. If the user is authorized, it converses with the user and allows or disallows the user to log in, and enables or does not enable the account. It checks for the allowed label range for the user, and also checks whether the user is authorized for remote logins.

The following options may be passed to `pam_sm_acct_mgmt()`:

| | |
|---|---|
| `label_check_on` | Enforce label range check for `PAM_USER`. |
| `auth_check_on` | Enforce authorization checks for `PAM_USER`. |
| `enable_check_off` | Do not check whether logins are enabled. |

**Password Management Module**

The Trusted Solaris password management component provides a function, `pam_sm_chauth_tok()`, to change passwords, in the UNIX password database.

The following options may be passed to `pam_sm_chauth_tok()`:

| | |
|---|---|
| `enable_randomword` | Use the randomword generator in the system to generate password lists. A pluggable randomword function library could be installed in `/usr/lib/security/pam_rw.so`. |

**ATTRIBUTES**

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

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---|---|
| MT Level | MT-Safe with exceptions |



`pam_tp_auth`(5)

`keylogin`(1), `libpam`(3LIB), `pam`(3PAM), `pam_authenticate`(3PAM), `pam_setcred`(3PAM), `pam.conf`(4), `attributes`(5)

**NOTES**

The interfaces in `libpam()` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

**NAME**

pam_unix – authentication, account, session, and password management PAM modules for UNIX

**SYNOPSIS**

```
/usr/lib/security/pam_unix.so.1
```

**DESCRIPTION**

The UNIX service module for PAM, `/usr/lib/security/pam_unix.so.1`, provides functionality for all four PAM modules: authentication, account management, session management and password management. The `pam_unix.so.1` module is a shared object that can be dynamically loaded to provide the necessary functionality upon demand. Its path is specified in the PAM configuration file.

**Unix Authentication Module**

The UNIX authentication component provides functions to verify the identity of a user, (`pam_sm_authenticate()`) and to set user specific credentials (`pam_sm_setcred()`). `pam_sm_authenticate()` compares the user entered password with the password from the UNIX password database. If the passwords match, the user is authenticated. If the user also has secure RPC credentials and the secure RPC password is the same as the UNIX password, then the secure RPC credentials are also obtained.

The following options may be passed to the UNIX service module:

`check_retries` — If this option is passed, the UNIX authentication module maintains a count of failed logins per user. This count is kept in the lowest byte of the flag field in the `shadow` database. When this count exceeds or equals the allowed number of retries as defined by `RETRIES` in `/etc/default/login` (or a default value of 5), the user's account is locked, and the module returns `PAM_MAXTRIES`. The account is not locked if the user has `lock_after_retries` set to `no` in his `user_attr` entry, or if `LOCK_AFTER_RETRIES` is set to `NO` in `/etc/security/policy.conf`. If the count is less than `RETRIES` (or the default value of 5), the module returns `PAM_AUTH_ERR`. If authentication succeeds, the count is reset to `0`. When the user gets a new password, the account is unlocked and the failed login count is reset to `0`. This option works for local files and NIS+ only.

`debug` — `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn` — Turn off warning messages.

`use_first_pass` — It compares the password in the password database with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, it quits and does not prompt the user for a password. This option should

only be used if the authentication service is designated as *optional* in the `pam.conf` configuration file.

`try_first_pass` — It compares the password in the password database with the user's initial password (entered when the user authenticated to the first authentication module in the stack). If the passwords do not match, or if no password has been entered, prompt the user for a password. When prompting for the current password, the UNIX authentication module will use the prompt, "password:" unless one of the following scenarios occur:

1. The option `try_first_pass` is specified and the password entered for the first module in the stack fails for the UNIX module.
2. The option `try_first_pass` is not specified, and the earlier authentication modules listed in the `pam.conf` file have prompted the user for the password. In these two cases, the UNIX authentication module will use the prompt "SYSTEM password:". The `pam_sm_setcred()` function sets user specific credentials. If the user had secure RPC credentials, but the secure RPC password was not the same as the UNIX password, then a warning message is printed. If the user wants to get secure RPC credentials, then `keylogin(1)` needs to be run.

**Unix Account Management Module**

The UNIX account management component provides a function to perform account management, `pam_sm_acct_mgmt()`. The function retrieves the user's password entry from the UNIX password database and verifies that the user's account and password have not expired. The following options may be passed in to the UNIX service module:

`debug` — `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn` — Turn off warning messages.

**Unix Session Management Module**

The UNIX session management component provides functions to initiate `pam_sm_open_session()` and terminate `pam_sm_close_session()` UNIX sessions. For UNIX, `pam_open_session` updates the `/var/adm/lastlog` file. The account management module reads this file to determine the previous time the user logged in. The following options may be passed in to the UNIX service module:

`debug` — `syslog`(3C) debugging information at `LOG_DEBUG` level.

`nowarn` — Turn off warning messages. `pam_close_session` is a null function.

**Unix Password Management Module**

The UNIX password management component provides a function to change passwords `pam_sm_chauthtok()` in the UNIX password database. This module must be *required* in `pam.conf`. It cannot be *optional* or *sufficient*. The following options may be passed in to the UNIX service module:

| | |
|---|---|
| `debug` | `syslog`(3C) Debugging information at `LOG_DEBUG` level. |
| `nowarn` | Turn off warning messages. |
| `use_first_pass` | It compares the password in the password database with the user's old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, it quits and does not prompt the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, it quits and does not prompt the user for a new password. |
| `try_first_pass` | It compares the password in the password database with the user's old password (entered to the first password module in the stack). If the passwords do not match, or if no password has been entered, it prompts the user for the old password. It also attempts to use the new password (entered to the first password module in the stack) as the new password for this module. If the new password fails, it prompts the user for a new password. If the user's password has expired, the UNIX account module saves this information in the authentication handle using `pam_set_data()`, with a unique name, `SUNW_UNIX_AUTHOK_DATA`. The UNIX password module retrieves this information from the authentication handle using `pam_get_data()` to determine whether or not to force the user to update the user's password. |

**ATTRIBUTES**

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

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---|---|
| MT Level | MT-Safe with exceptions |

**SUMMARY OF TRUSTED SOLARIS CHANGES**

The Trusted Solaris environment adds the `check_retries` option for the UNIX authentication module.

**SunOS 5.8 Reference Manual**

`keylogin`(1), `libpam`(3LIB), `pam`(3PAM), `pam_authenticate`(3PAM), `pam_setcred`(3PAM), `syslog`(3C), `pam.conf`(4), `attributes`(5)

**NOTES** The interfaces in `libpam()` are MT-Safe only if each thread within the multi-threaded application uses its own PAM handle.

**NAME**

priv_macros – Test, assign, clear, or store a privilege or privilege set

**SYNOPSIS**

```
#include <tsol/priv.h>

PRIV_ASSERT (priv_set, priv_id)

PRIV_ISASSERT (priv_set, priv_id)

PRIV_EQUAL (set_a, set_b)

PRIV_EMPTY (priv_set)

PRIV_FILL (priv_set)

PRIV_ISEMPTY (priv_set)

PRIV_ISFULL (priv_set)

PRIV_CLEAR (priv_set, priv_id)

PRIV_INTERSECT (set_a, set_b)

PRIV_INVERSE (priv_set)

PRIV_ISSUBSET (set_a, set_b)

PRIV_UNION (set_a, set_b)

PRIV_TEST (priv_id,errno )

PRIV_XOR (set_a, set_b)

priv_set_t *priv_set, *set_a, *set_b;

priv_t priv_id;
```

**DESCRIPTION**

`PRIV_ASSERT` (*priv_set*, *priv_id*) asserts the *priv_id* privilege in the *priv_set.*

`PRIV_ISASSERT` (*priv_set*, *priv_id*) is nonzero if the *priv_id* privilege in *priv_set* is asserted; if not, the value is zero.

`PRIV_EQUAL` (*set_a, set_b*) is true if *set_a* and *set_b* are identical.

`PRIV_EMPTY` (*priv_set*) initializes a *priv_set* to the null set.

`PRIV_FILL` (*priv_set*) fills *priv_set.*

`PRIV_ISEMPTY` (*priv_set*) is nonzero if *priv_set* is a null set; if not, the value is zero.

`PRIV_ISFULL` (*priv_set*) is nonzero if *priv_set* is a full set; if not, the value is zero.

`PRIV_CLEAR` (*priv_set*, *priv_id*) clears the *priv_id* in *priv_set.*

`PRIV_INTERSECT` (*set_a, set_b*) stores the intersection of *set_a* and *set_b* in *set_b.*

`PRIV_INVERSE` (*priv_set*) stores the inverse of *priv_set* in *priv_set*.

`PRIV_ISSUBSET` (*set_a*, *set_b*) is nonzero if all privileges asserted in *set_a* are also asserted in *set_b* (that is, if *set_a* is a subset of *set_b*).

`PRIV_UNION` (*set_a*, *set_b*) stores the union of *set_a* and *set_b* in *set_b.*

`PRIV_TEST` (*priv_id*, *errno*) tests if *priv_id* is asserted in the effective set, and sets `errno` if not.

`PRIV_XOR` (*set_a*, *set_b*) stores the `EXCLUSIVE OR` of *set_a* and *set_b* in *set_b*.

**ATTRIBUTES**

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

| ATTRIBUTE TYPE | ATTRIBUTE VALUE |
|---|---|
| Availability | SUNWtsu |

**ERRORS**

The behavior of these macros is undefined if *priv_id* is less than one or greater than the constant `MAX_PRIV`.

**Trusted Solaris 8 HW 12/02 Reference Manual**
**SunOS 5.8 Reference Manual**

`getppriv`(2), `setppriv`(2)

`attributes`(5)

# Index