NAME

login – sign on to the system

SYNOPSIS

login [–**p**] [–**d** *device*] [–**h** *hostname* [*terminal*] | –**r** *hostname*] [*name* [*environ* ...]]

AVAILABILITY

SUNWcsu

DESCRIPTION

You use the **login** command at the beginning of each terminal session to identify yourself 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.

login asks for your user name, if it is not supplied as an argument, and your password(s), 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 there are no lowercase characters in the first line of input processed, **login** assumes the connecting TTY is an uppercase-only terminal. It then sets the port's **termio**(7I) options to reflect this.

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 several incorrect login attempts, all attempts 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), the user is forced to changed the password. In this case the **/etc/nsswitch.conf** file is consulted to determine password repositories (see **nsswitch.conf**(4)). The password update configurations supported are limited to the following five cases.

- **passwd: files**
- **passwd: files nis**
- **passwd: files nisplus**
- **passwd: compat** (==> files nis)
- **passwd: compat** (==> files nisplus)
  passwd_compat: nisplus

Failure to comply with the configurations will prevent the user from logging onto the system because **passwd**(1) will fail. If you do not complete the login successfully within a certain period of time, it is likely that you will be silently disconnected.

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)).

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/*your-login-name*
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 **=** 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 root, edit the **/etc/default/login** file by inserting a ' # ' (pound-sign) before the **CONSOLE=/dev/console** entry. See **FILES** below.

SECURITY

login uses **pam**(3) 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
```

If there are no entries for the **login** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

When **login** is invoked through **rlogind** or **telnetd**, the service name used by PAM is **rlogin** or **telnet** respectively.

OPTIONS

–**d** *device* **login** accepts a device option, *device*. *device* is taken to be the path name of the TTY port **login** is to operate on. The use of the device option can be expected to improve **login** performance, since **login** will not need to call **ttyname**(3C). The –**d** option is available only to users whose **UID** and effective **UID** are root. Any other attempt to use –**d** will cause **login** to quietly exit.

–**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.

EXIT STATUS

0	success
non-zero	error.

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/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/utmp	accounting
/var/adm/wtmp	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 set, root can login on that device only. This will not prevent execution of remote commands with **rsh**(1). Comment out this line to allow login by root.

PASSREQ: Determines if login requires a 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**(3) **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**(3)). The default is 5.

SYNC_AGED_PASSWORD: If YES, then if any one of the user passwords has aged, then all passwords should be updated. If NO, only those passwords that have aged should be updated (see **pam**(3)). The default is YES.

ALLOW_AGED_PASSWORD:
If YES, then if any password ages, but the password can not be updated, then the user is still allowed in. If NO, then the user would not be able to login if the password is not updated (see **pam**(3)). The default is NO.

SEE ALSO

csh(1), **ksh**(1), **mail**(1), **mailx**(1), **newgrp**(1), **passwd**(1), **rlogin**(1), **rsh**(1), **sh**(1), **shell_builtins**(1), **telnet**(1), **admintool**(1M), **in.rlogind**(1M), **in.telnetd**(1M), **logins**(1M), **quota**(1M), **su**(1M), **syslogd**(1M), **useradd**(1M), **userdel**(1M), **pam**(3), **syslog**(3), **hosts.equiv**(4), **logindevperm**(4), **loginlog**(4), **nsswitch.conf**(4), **pam.conf**(4), **passwd**(4), **profile**(4), **shadow**(4), **environ**(5), **pam_unix**(5)

DIAGNOSTICS

Login incorrect The user name or the password cannot be matched.

Not on system console
Root 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.

WARNINGS

If you use the **CONSOLE** setting to disable root logins, you should arrange that remote command execution by root is also disabled. See **rsh**(1), **rcmd**(3N), and **hosts.equiv**(4) for further details.

NAME

passwd – change login password and password attributes

SYNOPSIS

passwd [*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 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*

AVAILABILITY

SUNWcsu

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) and **shadow**(4) for additional information. 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. Passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant. **PASSLENGTH** is found in **/etc/default/passwd** and by default is set to **6**.
- 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.

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.

If all requirements are met, by default, the **passwd** command will consult **/etc/nsswitch.conf** 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 5 cases. Failure to comply with these configurations will prevent users from logging onto the system.

- **passwd: files**
- **passwd: files nis**
- **passwd: files nisplus**
- **passwd: compat** (==> files nis)
- **passwd: compat** (==> files nisplus)
  passwd_compat: nisplus

In **files** case, super-users (for instance, real and effective uid equal to zero, 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 the super-user.

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* when the user is warned that their password is about to expire.

SECURITY

passwd uses **pam**(3) 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    password    required    /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

–r Specifies the repository to which an operation is applied. The supported repositories are **files**, **nis**, or **nisplus**.

–e Change the login shell.

–g Change the gecos (finger) information.

–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**(3N) 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 the super-user. It does not work at all for the **nis** repository which does not support password aging.

–a Show password attributes for all entries. Use only with the **–s** option; *name* must not be provided. For **nisplus** repository, this will show only the entries in the NIS+ passwd table in the local domain that the invoker is authorized to "read". For the **files** repository, this is restricted to the super-user.

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 Locks 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.
−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.

ENVIRONMENT

If any of the **LC_*** variables (**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.

FILES

/etc/oshadow Old shadow file.
/etc/passwd Password file.
/etc/shadow Shadow password file.
/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 the 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.

SEE ALSO

finger(1), **login**(1), **nispasswd**(1), **yppasswd**(1), **domainname**(1M), **eeprom**(1M), **id**(1M), **passmgmt**(1M), **pwconv**(1M), **su**(1M), **useradd**(1M), **userdel**(1M), **usermod**(1M), **crypt**(3C), **getpwnam**(3C), **getspnam**(3C), **nis_local_directory**(3N), **pam**(3), **loginlog**(4), **pam.conf**(4), **passwd**(4), **shadow**(4), **environ**(5), **pam_unix**(5)

NOTES

The **passwd** command replaces the **nispasswd** and **yppasswd** commands and should be used in their place.

Network administrators, who own the NIS+ password table, may change any password attributes.

NAME

in.ftpd, ftpd – file transfer protocol server

SYNOPSIS

in.ftpd [–**dl**] [–**t***timeout*]

AVAILABILITY

SUNWcsu

DESCRIPTION

in.ftpd is the Internet File Transfer Protocol (FTP) server process. The server is invoked by the Internet daemon **inetd**(1M) each time a connection to the FTP service (see **services**(4)) is made.

OPTIONS

–**d**	Debugging information is logged to the system log daemon **syslogd**(1M).
–**l**	Each FTP session is logged to the system log daemon **syslogd**(1M).
–**t***timeout*	Set the inactivity timeout period to *timeout* seconds. The FTP server will timeout an inactive session after 15 minutes.

Requests

The FTP server currently supports the following FTP requests; case is not distinguished.

ABOR	abort previous command
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CDUP	change to parent of current working directory
CWD	change working directory
DELE	delete a file
HELP	give help information
LIST	give list files in a directory (**ls** –**lg**)
MKD	make a directory
MODE	specify data transfer *mode*
NLST	give name list of files in directory (**ls**)
NOOP	do nothing
PASS	specify password
PASV	prepare for server-to-server transfer
PORT	specify data connection port
PWD	print the current working directory
QUIT	terminate session
RETR	retrieve a file
RMD	remove a directory
RNFR	specify rename-from file name
RNTO	specify rename-to file name

STOR store a file

STOU store a file with a unique name

STRU specify data transfer *structure*

TYPE specify data transfer *type*

USER specify user name

XCUP change to parent of current working directory

XCWD change working directory

XMKD make a directory

XPWD print the current working directory

XRMD remove a directory

The remaining FTP requests specified in RFC 959 are recognized, but not implemented.

The FTP server will abort an active file transfer only when the **ABOR** command is preceded by a Telnet Interrupt Process (IP) signal and a Telnet Synch signal in the command Telnet stream, as described in RFC 959.

in.ftpd interprets file names according to the globbing conventions used by **sh**(1). This allows users to utilize the metacharacters: * **? [ ]** { } ~

in.ftpd authenticates users according to four rules.

1) The user name must be in the password data base and have a password that is not null. A password must always be provided by the client before any file operations may be performed. The PAM framework (see SECURITY below) is used to verify that the correct password was entered.

2) If the user name appears in the file **/etc/ftpusers**, **ftp** access is denied.

3) **ftp** access is denied if the user's shell (from the password database) is not listed in the file **/etc/shells.** If the file **/etc/shells** does not exist, then the user's shell must be one of the following:

/usr/bin/sh	/usr/bin/csh	/usr/bin/ksh
/usr/bin/jsh	/bin/sh	/bin/csh
/bin/ksh	/bin/jsh	/sbin/sh
/sbin/jsh		

4) If the user name is anonymous or ftp, an entry for the user name *ftp* must be present in the password and shadow databases. The user is then allowed to log in by specifying any password — by convention this is given as the user's e-mail address (such as **user@host.Sun.COM**). Do not specify a valid shell in the password entry of the *ftp* user, and do not give it a valid password (use NP in the encrypted password field of the shadow file).

For anonymous ftp users, **in.ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot**(2) command to the home directory of the ftp user. In order that system security is not breached, it is recommended that the ftp subtree be constructed with care; the following rules are suggested.

~ftp Make the home directory owned by **root** and unwritable by anyone. This directory should not be on a file system mounted with the **nosuid** option.

~ftp/bin Make this directory owned by the super-user and unwritable by anyone. Make this a symbolic link to **~ftp/usr/bin** The program **ls**(1) must be present to support the list commands. This program should have mode 111.

~ftp/usr/lib Make this directory owned by the super-user and unwritable by anyone. Copy the following shared libraries from **/usr/lib** into this directory:

ld.so*
libc.so*
libdl.so*
libintl.so*
libw.so*
libnsl.so*
libsocket.so*
nss_nis.so*
nss_nisplus.so*
nss_dns.so*
nss_files.so*
straddr.so*

~ftp/etc Make this directory owned by the super-user and unwritable by anyone. Copies of the files **passwd**(4), **group**(4), and **netconfig**(4) must be present for the **ls**(1) command to work properly. These files should be mode 444.

~ftp/pub Make this directory mode 777 and owned by **ftp**. Users should then place files which are to be accessible via the anonymous account in this directory.

~ftp/dev Make this directory owned by the super-user and unwritable by anyone. First perform **ls –lL** on the device files listed below to determine their major and minor numbers, then use **mknod** to create them in this directory.

/dev/zero
/dev/tcp
/dev/udp
/dev/ticotsord

~ftp/usr/share/lib/zoneinfo
Make this directory mode 555 and owned by the super-user. Copy its contents from **/usr/share/lib/zoneinfo**. This enables **ls –l** to display time

and date stamps correctly.

SECURITY

in.ftpd uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the module to be used for **in.ftpd**. Here is a partial **pam.conf** file with entries for the **in.ftpd** command using the UNIX authentication, account management, and session management module.

```
ftp     auth      required    /usr/lib/security/pam_unix.so.1
ftp     account   required    /usr/lib/security/pam_unix.so.1
ftp     session   required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **ftp** service, then the entries for the "other" service will be used. Unlike **login**, **passwd**, and other commands, the **ftp** protocol will only support a single password. Using multiple modules will prevent **in.ftpd** from working properly.

EXAMPLES

To set up anonymous ftp with UNIX authentication, add the following entry to the **/etc/passwd** file. In this case, **/export/ftp** was chosen to be the anonymous ftp area, and the shell is the non-existent file **/nosuchshell**. This prevents users from logging in as the ftp user.

ftp:x:30000:30000:Anonymous FTP:/export/ftp:/nosuchshell

Add the following entry to /etc/shadow:

ftp:NP:6445::::::

The following is a shell script that will set up the anonymous ftp area. It presumes that names are resolved using NIS.

```
#!/bin/sh
# script to setup anonymous ftp area
#
# handle the optional command line argument
case $# in

  # the default location for the anon ftp comes from the passwd file
  0) ftphome="`grep '^ftp:' /etc/passwd | cut -d: -f6`"
     ;;

  1) if [ "$1" = "start" ]; then
       ftphome="`grep '^ftp:' /etc/passwd | cut -d: -f6`"
     else
       ftphome=$1
     fi
     ;;

  *) echo "Usage: $0 [anon-ftp-root]"
     exit 1
     ;;
esac
```

```
if [ -z "${ftphome}" ]; then
   echo "$0: ftphome must be non-null"
   exit 2
fi

# This script assumes that ftphome is neither / nor /usr so ...
if [ "${ftphome}" = "/" -o "${ftphome}" = "/usr" ]; then
   echo "$0: ftphome must not be / or /usr"
   exit 2
fi

# If ftphome does not exist but parent does, create ftphome
if [ ! -d ${ftphome} ]; then
    # lack of -p below is intentional
    mkdir ${ftphome}


fi
echo Setting up anonymous ftp area ${ftphome}

# Ensure that the /usr/bin directory exists
if [ ! -d ${ftphome}/usr/bin ]; then
    mkdir -p ${ftphome}/usr/bin
fi

cp /usr/bin/ls ${ftphome}/usr/bin
chmod 111 ${ftphome}/usr/bin/ls

# Now set the ownership and modes to match the man page
chown root ${ftphome}/usr/bin
chmod 555 ${ftphome}/usr/bin

# this may not be the right thing to do
# but we need the bin -> usr/bin link
if [ -r ${ftphome}/bin ]; then
    mv -f ${ftphome}/bin ${ftphome}/Obin
fi
ln -s usr/bin ${ftphome}

# Ensure that the /usr/lib and /etc directories exist
if [ ! -d ${ftphome}/usr/lib ]; then
    mkdir -p ${ftphome}/usr/lib
fi
if [ ! -d ${ftphome}/etc ]; then
    mkdir -p ${ftphome}/etc
```

```
fi

#Most of the following are needed for basic operation, except
#for libnsl.so, nss_nis.so, libsocket.so, and straddr.so which are
#needed to resolve NIS names.

cp /usr/lib/ld.so /usr/lib/ld.so.1 ${ftphome}/usr/lib

for lib in libc libdl libintl libw libnsl libsocket \
   nss_nis nss_nisplus nss_dns nss_files
do
   cp /usr/lib/${lib}.so.1 ${ftphome}/usr/lib
   rm -f ${ftphome}/usr/lib/${lib}.so
   ln -s ./${lib}.so.1 ${ftphome}/usr/lib/${lib}.so
done

cp /usr/lib/straddr.so.2 ${ftphome}/usr/lib
rm -f ${ftphome}/usr/lib/straddr.so
ln -s ./straddr.so.2 ${ftphome}/usr/lib/straddr.so

cp /etc/passwd /etc/group /etc/netconfig ${ftphome}/etc

# Copy timezone database
mkdir -p ${ftphome}/usr/share/lib/zoneinfo
(cd ${ftphome}/usr/share/lib/zoneinfo
  (cd /usr/share/lib/zoneinfo; find . -print | cpio -o) | cpio -imdu
  find . -print | xargs chmod 555
  find . -print | xargs chown root
)

chmod 555 ${ftphome}/usr/lib/*
chmod 444 ${ftphome}/etc/*

# Now set the ownership and modes
chown root ${ftphome}/usr/lib ${ftphome}/etc
chmod 555 ${ftphome}/usr/lib ${ftphome}/etc

# Ensure that the /dev directory exists
if [ ! -d ${ftphome}/dev ]; then
    mkdir -p ${ftphome}/dev
fi

# make device nodes. ticotsord and udp are necessary for
# 'ls' to resolve NIS names.
```

```
for device in zero tcp udp ticotsord
do
  line=‘ls -lL /dev/${device} | sed -e ’s/,//’‘
  major=‘echo $line | awk ’{print $5}’‘
  minor=‘echo $line | awk ’{print $6}’‘
  rm -f ${ftphome}/dev/${device}
  mknod ${ftphome}/dev/${device} c ${major} ${minor}
done

chmod 666 ${ftphome}/dev/*

## Now set the ownership and modes
chown root ${ftphome}/dev
chmod 555 ${ftphome}/dev

if [ ! -d ${ftphome}/pub ]; then
  mkdir -p ${ftphome}/pub
fi
chown ftp ${ftphome}/pub
chmod 777 ${ftphome}/pub
```

SEE ALSO

ftp(1), **ls**(1), **aset**(1M), **inetd**(1M), **mknod**(1M), **syslogd**(1M), **chroot**(2), **pam**(3), **getsockopt**(3N), **group**(4), **inetd.conf**(4), **netconfig**(4), **netrc**(4), **passwd**(4), **pam.conf**(4), **services**(4), **pam_unix**(5)

Postel, Jon, and Joyce Reynolds, *File Transfer Protocol (* FTP), RFC 959, Network Information Center, SRI International, Menlo Park, Calif., October 1985.

DIAGNOSTICS

in.ftpd logs various errors to **syslogd**, with a facility code of **daemon**.

Info Severity

These messages are logged only if the –**l** flag is specified.

FTPD: connection from *host* **at** *time*
: A connection was made to **ftpd** from the host *host* at the date and time *time*.

FTPD: User *user* **timed out after** *timeout* **seconds at** *time*
: The user *user* was logged out because they had not entered any commands after *timeout* seconds; the logout occurred at the date and time *time*.

Debug Severity

These messages are logged only if the –**d** flag is specified.

FTPD: command: *command*
: A command line containing *command* was read from the FTP client.

lost connection
: The FTP client dropped the connection.

<––– replycode
<––– replycode–
A reply was sent to the FTP client with the reply code *replycode*. The next message logged will include the message associated with the reply. If a – follows the reply code, the reply is continued on later lines.

NOTES

The anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

/etc/ftpusers contains a list of users who cannot access the system; the format of the file is one user name per line.

NAME

in.rlogind, rlogind – remote login server

SYNOPSIS

/usr/sbin/in.rlogind

AVAILABILITY

SUNWcsu

DESCRIPTION

in.rlogind is the server for the **rlogin**(1) program. The server provides a remote login facility. **in.rlogind** is invoked by **inetd**(1M) when a remote login connection is established.

First, the server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

Then, **in.rlogind** allocates a pseudo-terminal and manipulates file descriptors so that the slave half of the pseudo-terminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r** option. The login process then proceeds with the **pam**(3) authentication process (see SECURITY below) If automatic authentication fails, it reprompts the user to login.

The parent of the login process manipulates the master side of the pseudo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, a packet protocol is invoked to provide Ctrl-S/Ctrl-Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, **TERM**; see **environ**(4).

SECURITY

in.rlogind uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.rlogind**. Here is a partial **pam.conf** file with entries for the **rlogin** command using the "rhosts" and UNIX authentication modules, and the UNIX account, session management, and password management modules.

```
rlogin    auth      sufficient    /usr/lib/security/pam_rhosts_auth.so.1
rlogin    auth      required      /usr/lib/security/pam_unix.so.1
rlogin    account   required      /usr/lib/security/pam_unix.so.1
rlogin    session   required      /usr/lib/security/pam_unix.so.1
```

With this configuration, the server checks the client's source address. If an entry for the client exists in both **/etc/hosts** and **/etc/hosts.equiv**, a user logging in from the client is not prompted for a password. If the address is associated with a host for which no corresponding entry exists in **/etc/hosts**, the user is prompted for a password, regardless of whether or not an entry for the client is present in **/etc/hosts.equiv** (see **hosts**(4) and **hosts.equiv**(4)).

If there are no entries for the **rlogin** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords. Removing the "pam_rhosts_auth.so.1" entry will disable the **/etc/hosts.equiv** and **~/.rhosts** authentication protocol and the user would always be forced to type the password. The *sufficient* flag indicates that authentication through

the pam_rhosts_auth.so.1 module is "sufficient" to authenticate the user. Only if this authentication fails is the next authentication module used.

SEE ALSO

login(1), **rlogin**(1), **in.rshd**(1M), **inetd**(1M), **pam**(3), **environ**(4), **hosts**(4), **hosts.equiv**(4), **pam.conf**(4), **pam_rhosts_auth**(5), **pam_unix**(5)

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Try again.
A *fork* by the server failed.

/usr/bin/sh: ...
The user's login shell could not be started.

NOTES

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ''open'' environment.

A facility to allow all data exchanges to be encrypted should be present.

NAME

in.rshd, rshd – remote shell server

SYNOPSIS

in.rshd *host.port*

DESCRIPTION

in.rshd is the server for the **rsh**(1) program. The server provides remote execution facilities with authentication based on privileged port numbers.

in.rshd is invoked by **inetd**(1M) each time a shell service is requested, and executes the following protocol:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection. The client's host address (in hex) and port number (in decimal) are the arguments passed to **in.rshd**.

2) The server reads characters from the socket up to a null (**0**) byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address. If the address is associated with a host for which no corresponding entry exists in the host name data base (see **hosts**(4)), the server aborts the connection. Please refer to the SECURITY section below for more details.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the *server*'s machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the *client*'s machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) **in.rshd** then validates the user according to the following steps. The remote user name is looked up in the password file and a **chdir** is performed to the user's home directory. If the lookup fails, the connection is terminated. If the **chdir** fails, it does a **chdir** to **/** (root). If the user is not the super-user, (user ID 0), and if the pam_rhosts_auth PAM module is configured for authentication (see SECURITY below), the file **/etc/hosts.equiv** is consulted for a list of hosts considered equivalent. If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine. If this lookup fails, the connection is terminated.

9) A null byte is returned on the connection associated with the **stderr** and the

command line is passed to the normal login shell of the user. (The PATH variable is set to **/usr/bin**.) The shell inherits the network connections established by **in.rshd**.

SECURITY

in.rshd uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.rshd**. Here is a partial **pam.conf** file with entries for the **rsh** command using rhosts authentication, UNIX account management, and session management module.

```
rsh     auth     required     /usr/lib/security/pam_rhosts_auth.so.1
rsh     account  required     /usr/lib/security/pam_unix.so.1
rsh     session  required     /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **rsh** service, then the entries for the "other" service will be used. To maintain the authentication requirement for **in.rshd**, the rsh entry must always be configured with the pam_rhosts_auth.so.1 module. Multiple authentication modules can not be listed for the **rsh** service.

FILES

/etc/hosts.equiv

SEE ALSO

rsh(1), **inetd**(1M), **pam**(3), **hosts**(4), **pam.conf**(4), **pam_rhosts_auth**(5), **pam_unix**(5)

DIAGNOSTICS

The following diagnostic messages are returned on the connection associated with **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 in step 9 above (0 is returned above upon successful completion of all the steps prior to the command execution).

locuser too long
The name of the user on the client's machine is longer than 16 characters.

remuser too long
The name of the user on the remote machine is longer than 16 characters.

command too long
The command line passed exceeds the size of the argument list (as configured into the system).

Hostname for your address unknown.
No entry in the host name database existed for the client's machine.

Login incorrect.
No password file entry for the user name existed.

Permission denied.
The authentication procedure described above failed.

Can't make pipe.
The pipe needed for the **stderr** was not created.

Try again.
A *fork* by the server failed.

NOTES

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an "open" environment.

A facility to allow all data exchanges to be encrypted should be present.

NAME

in.telnetd, telnetd – DARPA TELNET protocol server

SYNOPSIS

/usr/sbin/in.telnetd

AVAILABILITY

SUNWcsu

DESCRIPTION

in.telnetd is a server that supports the DARPA standard TELNET virtual terminal protocol. **in.telnetd** is invoked in the internet server (see **inetd**(1M)), normally for requests to connect to the TELNET port as indicated by the **/etc/services** file (see **services**(4)).

in.telnetd operates by allocating a pseudo-terminal device for a client, then creating a login process which has the slave side of the pseudo-terminal as its standard input, output, and error. **in.telnetd** manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the login process.

When a TELNET session starts up, **in.telnetd** sends TELNET options to the client side indicating a willingness to do *remote echo* of characters, and to *suppress go ahead*. The pseudo-terminal allocated to the client is configured to operate in "cooked" mode, and with **XTABS**, **ICRNL**, and **ONLCR** enabled (see **termio**(7I)).

in.telnetd is willing to do: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **in.telnetd** is willing to have the remote client do: *binary*, *terminal type*, *terminal size*, *logout option*, and *suppress go ahead*.

in.telnetd also allows environment variables to be passed, provided that the client negotiates this during the initial option negotiation. The DISPLAY environment variable may be sent this way, either by the TELNET general environment passing methods, or via the XDISPLOC TELNET option. DISPLAY can be passed in the environment option during the same negotiation where XDISPLOC is used. Note that if you use both methods, use the same value for both. Otherwise, the results may be unpredictable.

These options are specified in Internet standards RFC 1096, RFC 1408, RFC 1571, and RFC 1572.

SECURITY

in.telnetd uses **pam**(3) 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 **in.telnetd**. Here is a partial **pam.conf** file with entries for the **telnet** command using the UNIX authentication, account management, session management, and password management modules.

```
telnet    auth      required    /usr/lib/security/pam_unix.so.1
telnet    account   required    /usr/lib/security/pam_unix.so.1
telnet    session   required    /usr/lib/security/pam_unix.so.1
telnet    password  required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **telnet** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted

for multiple passwords.

SEE ALSO

telnet(1), **pam**(3), **services**(4), **pam.conf**(4), **pam_unix**(5), **termio**(7I)

Alexander, S., *''TELNET Environment Option,''* RFC 1572, Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Borman, Dave, *''TELNET Environment Option,''* RFC 1408, Network Information Center, SRI International, Menlo Park, Calif., January 1993.

Borman, Dave, *''TELNET Environment Option Interoperability Issues,''* RFC 1571, Network Information Center, SRI International, Menlo Park, Calif., January 1994.

Crispin, Mark, *''TELNET Logout Option,''* RFC 727, Network Information Center, SRI International, Menlo Park, Calif., April 1977.

Marcy, G., *''TELNET X Display Location Option,''* RFC 1096, Network Information Center, SRI International, Menlo Park, Calif., March 1989.

Postel, Jon, and Joyce Reynolds, *''TELNET Protocol Specification,''* RFC 854, Network Information Center, SRI International, Menlo Park, Calif., May 1983.

Waitzman, D., *''TELNET Window Size Option,''* RFC 1073, Network Information Center, SRI International, Menlo Park, Calif., October 1988.

NOTES

Some TELNET commands are only partially implemented.

Binary mode has no common interpretation except between similar operating systems

The terminal type name received from the remote client is converted to lower case.

The *packet* interface to the pseudo-terminal should be used for more intelligent flushing of input and output queues.

in.telnetd never sends TELNET *go ahead* commands.

NAME

in.uucpd, uucpd – UUCP server

SYNOPSIS

/usr/sbin/in.uucpd [–**n**]

AVAILABILITY

SUNWbnuu

DESCRIPTION

in.uucpd is the server for supporting UUCP connections over networks.

in.uucpd is invoked by **inetd**(1M) when a UUCP connection is established (that is, a connection to the port indicated in the "uucp" service specification; see **services**(4)), and executes the following protocol:

1) The server prompts with **login:**. The **uucico**(1M) process at the other end must supply a username.

2) Unless the username refers to an account without a password, the server then prompts with **Password:**. The **uucico** process at the other end must supply the password for that account.

If the username is not valid, or is valid but refers to an account that does not have **/usr/lib/uucp/uucico** as its login shell, or if the password is not the correct password for that account, the connection is dropped. Otherwise, **uucico** is run, with the user ID, group ID, group set, and home directory for that account, with the environment variables **USER** and **LOGNAME** set to the specified username, and with a –**u** flag specifying the username. Unless the –**n** flag is specified, entries are made in **/var/adm/utmp, /var/adm/wtmp,** and **/var/adm/lastlog** for the username. **in.uucpd** must be invoked by a user with appropriate privilege (usually root) in order to be able to verify that the password is correct.

SECURITY

in.uucpd uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **in.uucpd**. Here is a partial **pam.conf** file with entries for **uucp** using the UNIX authentication, account management, and session management module.

```
uucp    auth      required    /usr/lib/security/pam_unix.so.1
uucp    account   required    /usr/lib/security/pam_unix.so.1
uucp    session   required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **uucp** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the peer may be prompted for multiple passwords.

FILES

/var/adm/utmp	accounting
/var/adm/wtmp	accounting
/var/adm/lastlog	time of last login

SEE ALSO

inetd(1M), **uucico**(1M), **pam**(3), **pam.conf**(4), **services**(4), **pam_unix**(5)

DIAGNOSTICS

All diagnostic messages are returned on the connection, after which the connection is closed.

user read
An error occurred while reading the username.

passwd read
An error occurred while reading the password.

Login incorrect.
The username is invalid or refers to an account with a login shell other than **/usr/lib/uucp/uucico**, or the password is not the correct password for the account.

NAME

init, telinit – process control initialization

SYNOPSIS

/sbin/init [**0123456abcQqSs**]

/etc/telinit [**0123456abcQqSs**]

AVAILABILITY

SUNWcsu

DESCRIPTION

init is a general process spawner. Its primary role is to create processes from information stored in the file **/etc/inittab**.

Run Level Defined

At any given time, the system is in one of eight possible run levels. A run level is a software configuration under which only a selected group of processes exists. Processes spawned by **init** for each of these run levels are defined in **/etc/inittab**. **init** can be in one of eight run levels, **0**–**6** and **S** or **s** (**S** and **s** are identical). The run level changes when a privileged user runs **/sbin/init**. This sends appropriate signals to the original **init** spawned by the operating system at boot time, saying which run level to invoke.

init and System Booting

When the system is booted, **init** is invoked and the following occurs. First, it reads **/etc/default/init** to set environment variables. This is typically where **TZ** (time zone) and locale-related environments such as **LANG** or **LC_CTYPE** get set.

init then looks in **/etc/inittab** for the **initdefault** entry (see **inittab**(4)). If the **initdefault** entry:

exists — **init** usually uses the run level specified in that entry as the initial run level to enter.

does not exist — **/etc/inittab**, **init** asks the user to enter a run level from the system console.

S or **s** — **init** goes to the single-user state. In this state, the system console device (**/dev/console**) is opened for reading and writing and the command **/sbin/su**, (see **su**(1M)), is invoked. Use either **init** or **telinit** to change the run level of the system. Note that if the shell is terminated (using an end-of-file), **init** only re-initializes to the single-user state if **/etc/inittab** does not exist.

0-6 — **init** enters the corresponding run level. Run levels **0**, **5**, and **6** are reserved states for shutting the system down. Run levels **2**, **3**, and **4** are available as multi-user operating states.

If this is the first time since power up that **init** has entered a run level other than single-user state, **init** first scans **/etc/inittab** for **boot** and **bootwait** entries (see **inittab**(4)). These entries are performed before any other processing of **/etc/inittab** takes place, providing that the run level entered matches that of the entry. In this way any special initialization of the operating system, such as mounting file systems, can take

place before users are allowed onto the system. **init** then scans **/etc/inittab** and executes all other entries that are to be processed for that run level.

To spawn each process in **/etc/inittab**, **init** reads each entry and for each entry that should be respawned, it forks a child process. After it has spawned all of the processes specified by **/etc/inittab**, **init** waits for one of its descendant processes to die, a **powerfail** signal, or a signal from another **init** or **telinit** process to change the system's run level. When one of these conditions occurs, **init** re-examines **/etc/inittab**.

inittab Additions

New entries can be added to **/etc/inittab** at any time; however, **init** still waits for one of the above three conditions to occur before re-examining **/etc/inittab**. To get around this, **init Q** or **init q** command wakes **init** to re-examine **/etc/inittab** immediately.

When **init** comes up at boot time and whenever the system changes from the single-user state to another run state, **init** sets the **ioctl**(2) states of the console to those modes saved in the file **/etc/ioctl.syscon**. **init** writes this file whenever the single-user state is entered.

Run Level Changes

When a run level change request is made, **init** sends the warning signal (**SIGTERM**) to all processes that are undefined in the target run level. **init** waits five seconds before forcibly terminating these processes by sending a kill signal (**SIGKILL**).

When **init** receives a signal telling it that a process it spawned has died, it records the fact and the reason it died in **/var/adm/utmp** and **/var/adm/wtmp** if it exists (see **who**(1)). A history of the processes spawned is kept in **/var/adm/wtmp.**

If **init** receives a **powerfail** signal (**SIGPWR**) it scans **/etc/inittab** for special entries of the type **powerfail** and **powerwait**. These entries are invoked (if the run levels permit) before any further processing takes place. In this way **init** can perform various cleanup and recording functions during the powerdown of the operating system.

/etc/defaults/init File

Default values can be set for the following flags in **/etc/default/init**. For example:
TZ=US/Pacific

TZ	Either specifies the timezone information (see **ctime**(3C)) or the name of a timezone information file **/usr/share/lib/zoneinfo**.
LC_CTYPE	Character characterization information.
LC_MESSAGES	Message translation.
LC_MONETARY	Monetary formatting information.
LC_NUMERIC	Numeric formatting information.
LC_TIME	Time formatting information.
LC_ALL	If set, all other **LC_*** environmental variables take-on this value.
LANG	If **LC_ALL** is not set, and any particular **LC_*** is also not set, the value of **LANG** is used for that particular environmental variable.

telinit

telinit, which is linked to **/sbin/init**, is used to direct the actions of **init**. It takes a one-character argument and signals **init** to take the appropriate action.

SECURITY

init uses **pam**(3) for session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the session management module to be used for **init**. Here is a partial **pam.conf** file with entries for **init** using the UNIX session management module.

```
init    session    required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **init** service, then the entries for the "other" service will be used.

OPTIONS

0 Go into firmware.

1 Put the system in system administrator mode. All file systems are mounted. Only a small set of essential kernel processes are left running. This mode is for administrative tasks such as installing optional utility packages. All files are accessible and no users are logged in on the system.

2 Put the system in multi-user mode. All multi-user environment terminal processes and daemons are spawned. This state is commonly referred to as the multi-user state.

3 Extend multi-user mode by making local resources available over the network.

4 Is available to be defined as an alternative multi-user environment configuration. It is not necessary for system operation and is usually not used.

5 Shut the machine down so that it is safe to remove the power. Have the machine remove power, if possible.

6 Stop the operating system and reboot to the state defined by the **initdefault** entry in **/etc/inittab**.

a, **b**, **c** process only those **/etc/inittab** entries having the **a**, **b**, or **c** run level set. These are pseudo-states, which may be defined to run certain commands, but which do not cause the current run level to change.

Q, **q** Re-examine **/etc/inittab**.

S, **s** Enter single-user mode. When this occurs, the terminal that executed this command becomes the system console. This is the only run level that doesn't require the existence of a properly formatted **/etc/inittab** file. If this file does not exist, then by default, the only legal run level that **init** can enter is the single-user mode. When the system comes up to **S** or **s**, file systems for users' files are not mounted and only essential kernel processes are running. When the system comes down to **S** or **s**, all mounted file systems remain mounted, and all processes started by **init** that should only be running in multi-user mode are killed. In addition, any process that has a **utmp** entry will be killed. This last condition insures that all port monitors

started by the SAC are killed and all services started by these port monitors, including **ttymon** login services, are killed. Other processes not started directly by **init** will remain running. For example, **cron** remains running.

FILES

/etc/inittab	controls process dispatching by init
/var/adm/utmp	accounting information
/var/adm/wtmp	history of all logins since file was last created
/etc/ioctl.syscon	
/dev/console	system console device
/etc/default/init	environment variables.

SEE ALSO

login(1), **sh**(1), **stty**(1), **who**(1), **shutdown**(1M), **su**(1M), **ttymon**(1M), **ioctl**(2), **kill**(2), **ctime**(3C), **pam**(3), **inittab**(4), **pam.conf**(4), **utmp**(4), **utmpx**(4), **pam_unix**(5), **termio**(7I)

DIAGNOSTICS

If **init** finds that it is respawning an entry from **/etc/inittab** more than ten times in two minutes, assumes that there is an error in the command string in the entry, and generates an error message on the system console. It will then refuse to respawn this entry until either five minutes has elapsed or it receives a signal from a user-spawned **init** or **telinit**. This prevents **init** from eating up system resources when someone makes a typographical error in the **inittab** file, or a program is removed that is referenced in **/etc/inittab**.

NOTES

init and **telinit** can be run only by a privileged user.

The **S** or **s** state must not be used indiscriminately in **/etc/inittab**. When modifying this file, it is best to avoid adding this state to any line other than **initdefault**.

If a default state is not specified in the **initdefault** entry in **/etc/inittab**, state **6** is entered. Consequently, the system will loop by going to firmware and rebooting continuously.

If the **utmp** file cannot be created when booting the system, the system will boot to state "**s**" regardless of the state specified in the **initdefault** entry in **/etc/inittab**. This can occur if the **/var** file system is not accessible.

NAME

rpc.rexd, rexd – RPC-based remote execution server

SYNOPSIS

/usr/sbin/rpc.rexd [–**s**]

AVAILABILITY

SUNWnisu

DESCRIPTION

rpc.rexd is the Sun RPC server for remote program execution. This daemon is started by **inetd**(1M) whenever a remote execution request is made.

For non-interactive programs, the standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, in a fashion that is similar to the login sessions provided by **rlogin**(1). This daemon may use NFS to mount file systems specified in the remote execution request.

SECURITY

rpc.rexd uses **pam**(3) for account and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **rpc.rexd**. Here is a partial **pam.conf** file with **rpc.rexd** entries for account and session management using the UNIX module.

```
rpc.rexd   account    required    /usr/lib/security/pam_unix.so.1
rpc.rexd   session    required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **rpc.rexd** service, then the entries for the "other" service will be used. **rpc.rexd** uses the **getpwuid()** call to determine whether the given user is a legal user.

OPTIONS

–**s** Secure. When specified, requests must have valid DES credentials. If the request does not have a DES credential it is rejected. The default publickey credential is rejected. Only newer **on**(1) commands send DES credentials.

If access is denied with an authentication error, you may have to set your publickey with the **chkey**(1) command.

Specifying the –**s** option without presenting secure credentials will result in an error message: **Unix too weak auth (DesONly)!**

FILES

/dev/pts*n*	pseudo-terminals used for interactive mode
/etc/passwd	authorized users
/tmp_rex/rexd??????	temporary mount points for remote file systems.

SEE ALSO

chkey(1), **on**(1), **rlogin**(1), **inetd**(1M), **pam**(3), **inetd.conf**(4), **pam.conf**(4), **publickey**(4), **pam_unix**(5)

DIAGNOSTICS

Diagnostic messages are normally printed on the console, and returned to the requestor.

NOTES Root cannot execute commands using **rexd** client programs such as **on**(1).

NAME

sac – service access controller

SYNOPSIS

sac –**t** *sanity_interval*

/usr/lib/saf/sac

DESCRIPTION

The Service Access Controller (SAC) is the overseer of the server machine. It is started when the server machine enters multiuser mode. The SAC performs several important functions as explained below.

Customizing the SAC *environment*. When **sac** is invoked, it first looks for the per-system configuration script **/etc/saf/_sysconfig**. **sac** interprets **_sysconfig** to customize its own environment. The modifications made to the SAC environment by **_sysconfig** are inherited by all the children of the SAC. This inherited environment may be modified by the children.

Starting port monitors. After it has interpreted the **_sysconfig** file, the **sac** reads its administrative file **/etc/saf/_sactab**. **_sactab** specifies which port monitors are to be started. For each port monitor to be started, **sac** forks a child (see **fork**(2)) and creates a **utmp** entry with the *type* field set to **LOGIN_PROCESS**. Each child then interprets its per-port monitor configuration script **/etc/saf/***pmtag***/_config**, if the file exists. These modifications to the environment affect the port monitor and will be inherited by all its children. Finally, the child process **exec**s the port monitor, using the command found in the **_sactab** entry. (See **sacadm**; this is the command given with the –**c** option when the port monitor is added to the system.)

Polling port monitors to detect failure. The –**t** option sets the frequency with which **sac** polls the port monitors on the system. This time may also be thought of as half of the maximum latency required to detect that a port monitor has failed and that recovery action is necessary.

Administrative functions. The Service Access Controller represents the administrative point of control for port monitors. Its administrative tasks are explained below.

When queried (**sacadm** with either –**l** or –**L**), the Service Access Controller returns the status of the port monitors specified, which **sacadm** prints on the standard output. A port monitor may be in one of six states:

ENABLED	The port monitor is currently running and is accepting connections. See **sacadm**(1M) with the –**e** option.
DISABLED	The port monitor is currently running and is not accepting connections. See **sacadm** with the –**d** option, and see **NOTRUNNING**, below.
STARTING	The port monitor is in the process of starting up. **STARTING** is an intermediate state on the way to **ENABLED** or **DISABLED**.
FAILED	The port monitor was unable to start and remain running.

STOPPING The port monitor has been manually terminated but has not completed its shutdown procedure. **STOPPING** is an intermediate state on the way to **NOTRUNNING**.

NOTRUNNING The port monitor is not currently running. (See **sacadm** with **–k**.) This is the normal "not running" state. When a port monitor is killed, all ports it was monitoring are inaccessible. It is not possible for an external user to tell whether a port is not being monitored or the system is down. If the port monitor is not killed but is in the **DISABLED** state, it may be possible (depending on the port monitor being used) to write a message on the inaccessible port telling the user who is trying to access the port that it is disabled. This is the advantage of having a **DISABLED** state as well as the **NOTRUNNING** state.

When a port monitor terminates, the SAC removes the **utmp** entry for that port monitor.

The SAC receives all requests to enable, disable, start, or stop port monitors and takes the appropriate action.

The SAC is responsible for restarting port monitors that terminate. Whether or not the SAC will restart a given port monitor depends on two things:

- The restart count specified for the port monitor when the port monitor was added by **sacadm**; this information is included in **/etc/saf/***pmtag***/_sactab**.
- The number of times the port monitor has already been restarted.

SECURITY

sac uses **pam**(3) for session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the session management module to be used for **sac**. Here is a partial **pam.conf** file with entries for **sac** using the UNIX session management module.

```
sac     session     required     /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **sac** service, then the entries for the "other" service will be used.

OPTIONS

–t *sanity_interval* Sets the frequency (*sanity_interval*) with which **sac** polls the port monitors on the system.

FILES

/etc/saf/_sactab
/etc/saf/_sysconfig
/var/adm/utmp
/var/saf/_log

SEE ALSO

pmadm(1M), **sacadm**(1M), **fork**(2), **pam**(3), **pam.conf**(4), **pam_unix**(5)

NAME

su – become super-user or another user

SYNOPSIS

su [–] [*username* [*arg* ...]]

AVAILABILITY

SUNWcsu

DESCRIPTION

su allows one to become another user without logging off. The default user *name* is **root** (super-user).

To use **su**, the appropriate password must be supplied (unless the invoker is already **root**). If the password is correct, **su** creates a new shell process that has the real and effective user ID, group IDs, and supplementary group list set to those of the specified *username*. The new shell will be the shell specified in the shell field of *username's* password file entry (see **passwd**(4)). If no shell is specified, **/usr/bin/sh** is used (see **sh**(1)). To return to normal user ID privileges, type an EOF character (CTRL-D) to exit the new shell.

Any additional arguments given on the command line are passed to the new shell. When using programs such as **sh**, an *arg* of the form **–c** *string* executes *string* using the shell and an arg of **–r** gives the user a restricted shell.

If the first argument to **su** is ' - ' (dash), the environment will be changed to what would be expected if the user actually logged in as the specified user. This is accomplished by invoking the program used as the shell with a first argument value whose initial character is ' - ' (dash), thus simulating a login. If the first argument to **su** is not ' - ' (dash), the environment is passed along unchanged, with the exception of **$PATH**, which is controlled by **PATH** and **SUPATH** in **/etc/default/su**.

All attempts to become another user using **su** are logged in the log file **/var/adm/sulog** (see **sulog**(4)).

SECURITY

su uses **pam**(3) for authentication, account management, and session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **su**. Here is a partial **pam.conf** file with entries for the **su** command using the UNIX authentication, account management, and session management module.

```
su      auth      required    /usr/lib/security/pam_unix.so.1
su      account   required    /usr/lib/security/pam_unix.so.1
su      session   required    /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **su** service, then the entries for the "other" service will be used. If multiple authentication modules are listed, then the user may be prompted for multiple passwords.

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute:

```
example% su bin
```

To become user **bin** but change the environment to what would be expected if **bin** had originally logged in, execute:

example% su − bin

To execute *command* with the temporary environment and permissions of user **bin**, type:

example% su − bin −c "*command args*"

ENVIRONMENT

If any of the **LC_*** variables (**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 **su** 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 are set in the environment, the "C" (U.S. style) locale determines how **su** behaves.

LC_CTYPE Determines how **su** handles characters. When **LC_CTYPE** is set to a valid value, **su** can display and handle text and filenames containing valid characters for that locale. **su** can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. **su** 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).

FILES

$HOME/.profile user's login commands for sh and ksh

/etc/passwd system's password file

/etc/profile system-wide sh and ksh login commands

/var/adm/sulog log file

/etc/default/su the default parameters that reside here are:

- **SULOG**: If defined, all attempts to **su** to another user are logged in the indicated file.
- **CONSOLE**: If defined, all attempts to **su** to **root** are logged on the console.
- **PATH**: Default path. (**/usr/bin:**)
- **SUPATH**: Default path for a user invoking **su** to **root**. (**/usr/sbin:/usr/bin**)
- **SYSLOG**: Determines whether the **syslog**(3) **LOG_AUTH** facility should be used to log all **su** attempts. **LOG_NOTICE** messages are generated for **su**'s to root, **LOG_INFO** messages are generated for **su**'s to other users, and **LOG_CRIT** messages are generated for failed **su** attempts.

SEE ALSO **csh**(1), **env**(1), **ksh**(1), **login**(1), **sh**(1), **syslogd**(1M), **pam**(3), **syslog**(3), **pam.conf**(4), **passwd**(4), **profile**(4), **sulog**(4), **environ**(5), **pam_unix**(5)

NAME

ttymon – port monitor for terminal ports

SYNOPSIS

/usr/lib/saf/ttymon
/usr/lib/saf/ttymon –**g** [–**d** *device*] [–**h**] [–**t** *timeout*] [–**l** *ttylabel*] [–**p** *prompt*] [–**m** *modules*] [–**T** *termtype*]

DESCRIPTION

ttymon is a STREAMS-based TTY port monitor. Its function is to monitor ports, to set terminal modes, baud rates, and line disciplines for the ports, and to connect users or applications to services associated with the ports. Normally, **ttymon** is configured to run under the Service Access Controller, **sac**(1M), as part of the Service Access Facility (SAF). It is configured using the **sacadm**(1M) command. Each instance of **ttymon** can monitor multiple ports. The ports monitored by an instance of **ttymon** are specified in the port monitor's administrative file. The administrative file is configured using the **pmadm**(1M) and **ttyadm**(1M) commands. When an instance of **ttymon** is invoked by the **sac** command, it starts to monitor its ports. For each port, **ttymon** first initializes the line disciplines, if they are specified, and the speed and terminal settings. For ports with entries in **/etc/logindevperm**, device owner, group and permissions are set. (See **logindevperm**(4).) The values used for initialization are taken from the appropriate entry in the TTY settings file. This file is maintained by the **sttydefs**(1M) command. Default line disciplines on ports are usually set up by the **autopush**(1M) command of the Autopush Facility.

ttymon then writes the prompt and waits for user input. If the user indicates that the speed is inappropriate by pressing the BREAK key, **ttymon** tries the next speed and writes the prompt again. When valid input is received, **ttymon** interprets the per-service configuration file for the port, if one exists, creates a **utmp** entry if required (see **utmp**(4)), establishes the service environment, and then invokes the service associated with the port. Valid input consists of a string of at least one non-newline character, terminated by a carriage return. After the service terminates, **ttymon** cleans up the **utmp** entry, if one exists, and returns the port to its initial state.

If *autobaud* is enabled for a port, **ttymon** will try to determine the baud rate on the port automatically. Users must enter a carriage return before **ttymon** can recognize the baud rate and print the prompt. Currently, the baud rates that can be determined by *autobaud* are **110**, **1200**, **2400**, **4800**, and **9600**.

If a port is configured as a bidirectional port, **ttymon** will allow users to connect to a service, and, if the port is free, will allow **uucico**(1M), **cu**(1C), or **ct**(1C) to use it for dialing out. If a port is bidirectional, **ttymon** will wait to read a character before it prints a prompt.

If the *connect-on-carrier* flag is set for a port, **ttymon** will immediately invoke the port's associated service when a connection request is received. The prompt message will not be sent.

If a port is disabled, **ttymon** will not start any service on that port. If a disabled message is specified, **ttymon** will send out the disabled message when a connection request is received. If **ttymon** is disabled, all ports under that instance of **ttymon** will also be disabled.

SERVICE INVOCATION

The service **ttymon** invokes for a port is specified in the **ttymon** administrative file. **ttymon** will scan the character string giving the service to be invoked for this port, looking for a **%d** or a **%%** two-character sequence. If **%d** is found, **ttymon** will modify the service command to be executed by replacing those two characters by the full path name of this port (the device name). If **%%** is found, they will be replaced by a single **%**.

When the service is invoked, file descriptor **0**, **1**, and **2** are opened to the port device for reading and writing. The service is invoked with the user ID, group ID and current home directory set to that of the user name under which the service was registered with **ttymon**. Two environment variables, **HOME** and **TTYPROMPT,** are added to the service's environment by **ttymon**. **HOME** is set to the home directory of the user name under which the service is invoked. **TTYPROMPT** is set to the prompt string configured for the service on the port. This is provided so that a service invoked by **ttymon** has a means of determining if a prompt was actually issued by **ttymon** and, if so, what that prompt actually was.

See **ttyadm**(1M) for options that can be set for ports monitored by **ttymon** under the Service Access Controller.

SECURITY

ttymon uses **pam**(3) for session management. The PAM configuration policy, listed through **/etc/pam.conf**, specifies the modules to be used for **ttymon**. Here is a partial **pam.conf** file with entries for **ttymon** using the UNIX session management module.

```
ttymon  session  required  /usr/lib/security/pam_unix.so.1
```

If there are no entries for the **ttymon** service, then the entries for the "other" service will be used.

OPTIONS

–g A special invocation of **ttymon** is provided with the **–g** option. This form of the command should only be called by applications that need to set the correct baud rate and terminal settings on a port and then connect to **login** service, but that cannot be pre-configured under the SAC. The following combinations of options can be used with **–g**:

–d *device* *device* is the full path name of the port to which **ttymon** is to attach. If this option is not specified, file descriptor **0** must be set up by the invoking process to a TTY port.

–h If the -h flag is not set, **ttymon** will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed.

–l *ttylabel* *ttylabel* is a link to a speed and TTY definition in the **ttydefs** file. This definition tells **ttymon** at what speed to run initially, what the initial

TTY settings are, and what speed to try next if the user indicates that the speed is inappropriate by pressing the BREAK key. The default speed is 9600 baud.

–**m** *modules* When initializing the port, **ttymon** will pop all modules on the port, and then push *modules* in the order specified. *modules* is a comma-separated list of pushable modules. Default modules on the ports are usually set up by the Autopush Facility.

–**p** *prompt* Allows the user to specify a prompt string. The default prompt is **Login:** .

–**t** *timeout* Specifies that **ttymon** should exit if no one types anything in *timeout* seconds after the prompt is sent.

–**T** *termtype* Sets the **TERM** environment variable to *termtype.*

ENVIRONMENT

If any of the **LC_*** variables (**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 **ttymon** 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 **ttymon** behaves.

LC_CTYPE

Determines how **ttymon** handles characters. When **LC_CTYPE** is set to a valid value, **ttymon** can display and handle text and filenames containing valid characters for that locale. **ttymon** can display and handle Extended Unix Code (EUC) characters where any individual character can be 1, 2, or 3 bytes wide. **ttymon** can also handle EUC characters of 1, 2, or more column widths. In the "C" locale, only characters from ISO 8859-1 are valid.

FILES

/etc/logindevperm

SEE ALSO

ct(1C), **cu**(1C), **autopush**(1M), **pmadm**(1M), **sac**(1M), **sacadm**(1M), **sttydefs**(1M), **ttyadm**(1M), **uucico**(1M), **pam**(3), **logindevperm**(4), **pam.conf**(4), **utmp**(4), **environ**(5), **pam_unix**(5)

System Administration Guide, Volume II

NOTES

If a port is monitored by more than one **ttymon**, it is possible for the **ttymon**s to send out prompt messages in such a way that they compete for input.