NAME

sudo, **sudoedit** - execute a command as another user

SYNOPSIS

sudo -h | **-K** | **-k** | **-L** | **-V**
sudo -v [**-AknS**] [**-a** *auth_type*] [**-g** *group name* | *#gid*] [**-p** *prompt*] [**-u** *user name* | *#uid*]
sudo -l[*l*] [**-AknS**] [**-a** *auth_type*] [**-g** *group name* | *#gid*] [**-p** *prompt*] [**-U** *user name*]
[**-u** *user name* | *#uid*] [*command*]
sudo [**-AbEHnPS**] [**-a** *auth_type*] [**-C** *fd*] [**-c** *class* | -] [**-g** *group name* | *#gid*] [**-p** *prompt*] [**-r** *role*]
[**-t** *type*] [**-u** *user name* | *#uid*] [**VAR**=*value*] **-i** | **-s** [*command*]
sudoedit [**-AnS**] [**-a** *auth_type*] [**-C** *fd*] [**-c** *class* | -] [**-g** *group name* | *#gid*] [**-p** *prompt*]
[**-u** *user name* | *#uid*] file ...

DESCRIPTION

sudo allows a permitted user to execute a *command* as the superuser or another user, as specified by the *sudoers* file. See the *COMMAND EXECUTION* section below for more details.

sudo determines who is an authorized user by consulting the file */etc/sudoers*. By running **sudo** with the **-v** option, a user can update the time stamp without running a *command*. If authentication is required, **sudo** will exit if the user’s password is not entered within a configurable time limit. The default password prompt timeout is 5 minutes.

When invoked as **sudoedit**, the **-e** option (described below), is implied.

The options are as follows:

-A Normally, if **sudo** requires a password, it will read it from the user’s terminal. If the **-A** (*askpass*) option is specified, a (possibly graphical) helper program is executed to read the user’s password and output the password to the standard output. If the SUDO_ASKPASS environment variable is set, it specifies the path to the helper program. Otherwise, the value specified by the *askpass* option in sudoers(5) is used. If no askpass program is available, **sudo** will exit with an error.

-a *type* The **-a** (*authentication type*) option causes **sudo** to use the specified authentication type when validating the user, as allowed by */etc/login.conf*. The system administrator may specify a list of sudo-specific authentication methods by adding an "auth-sudo" entry in */etc/login.conf*. This option is only available on systems that support BSD authentication.

-b The **-b** (*background*) option tells **sudo** to run the given command in the background. Note that if you use the **-b** option you cannot use shell job control to manipulate the process.

Most interactive commands will fail to work properly in background mode.

-C *fd* Normally, **sudo** will close all open file descriptors other than standard input, standard output and standard error. The **-C** (*close from*) option allows the user to specify a starting point above the standard error (file descriptor three). Values less than three are not permitted. This option is only available when the administrator has enabled the *closefrom_override* option in sudoers(5).

-c *class* The **-c** (*class*) option causes **sudo** to run the specified command with resources limited by the specified login class. The *class* argument can be either a class name as defined in */etc/login.conf*, or a single '-' character. Specifying a *class* of **-** indicates that the command should be run restricted by the default login capabilities for the user the command is run as. If the *class* argument specifies an existing user class, the command must be run as root, or the **sudo** command must be run from a shell that is already root. This option is only available on systems with BSD login classes.

-E The **-E** (*preserve environment*) option will override the *env_reset* option in sudoers(5). It is only available when either the matching command has the SETENV tag or the *setenv* option is set in sudoers(5). **sudo** will return an error if the **-E** option is specified and the user does not have permission to preserve the environment.

-e The **-e** (*edit*) option indicates that, instead of running a command, the user wishes to edit one or more files. In lieu of a command, the string "sudoedit" is used when consulting the *sudoers* file. If the user is authorized by *sudoers*, the following steps are taken:

1. Temporary copies are made of the files to be edited with the owner set to the invoking user.

2. The editor specified by the SUDO_EDITOR, VISUAL or EDITOR environment variables (in that order) is run to edit the temporary files. If none of SUDO_EDITOR, VISUAL or EDITOR are set, the first program listed in the *editor* sudoers(5) option is used.

3. If they have been modified, the temporary files are copied back to their original location and the temporary versions are removed.

If the specified file does not exist, it will be created. Note that unlike most commands run by *sudo*, the editor is run with the invoking user's environment unmodified. If, for some reason, **sudo** is unable to update a file with its edited version, the user will receive a warning and the edited copy will remain in a temporary file.

-g *group* Normally, **sudo** runs a command with the primary group set to the one specified by the password database for the user the command is being run as (by default, root). The **-g** (*group*) option causes **sudo** to run the command with the primary group set to *group* instead. To specify a *gid* instead of a *group name,* use *#gid*. When running commands as a *gid,* many shells require that the '#' be escaped with a backslash ('\'). If no **-u** option is specified, the command will be run as the invoking user (not root). In either case, the primary group will be set to *group*.

-H The **-H** (*HOME*) option option sets the HOME environment variable to the home directory of the target user (root by default) as specified by the password database. The default handling of the HOME environment variable depends on sudoers(5) settings. By default, **sudo** will set HOME if *env_reset* or *always_set_home* are set, or if *set_home* is set and the **-s** option is specified on the command line.

-h The **-h** (*help*) option causes **sudo** to print a short help message to the standard output and exit.

-i [*command*]
The **-i** (*simulate initial login*) option runs the shell specified by the password database entry of the target user as a login shell. This means that login-specific resource files such as *.profile* or *.login* will be read by the shell. If a command is specified, it is passed to the shell for execution via the shell's **-c** option. If no command is specified, an interactive shell is executed. **sudo** attempts to change to that user's home directory before running the shell. It also initializes the environment to a minimal set of variables, similar to what is present when a user logs in. The *Command environment* section below documents in detail how the **-i** option affects the environment in which a command is run.

-K The **-K** (sure *kill*) option is like **-k** except that it removes the user's time stamp file entirely and may not be used in conjunction with a command or other option. This option does not require a password.

-k [*command*]
When used alone, the **-k** (*kill*) option to **sudo** invalidates the user's time stamp file. The next time **sudo** is run a password will be required. This option does not require a password and was added to allow a user to revoke **sudo** permissions from a *.logout* file.

When used in conjunction with a command or an option that may require a password, the **-k** option will cause **sudo** to ignore the user's time stamp file. As a result, **sudo** will prompt for a password (if one is required by *sudoers*) and will not update the user's time stamp file.

-L The **-L** (*list* defaults) option will list the parameters that may be set in a *Defaults* line along with a short description for each. This option will be removed from a future version of **sudo**.

-l[**l**] [*command*]
If no *command* is specified, the **-l** (*list*) option will list the allowed (and forbidden) commands for the invoking user (or the user specified by the **-U** option) on the current host. If a *command* is specified and is permitted by *sudoers*, the fully-qualified path to the command is displayed along with any command line arguments. If *command* is specified but not allowed, **sudo** will exit with a status value of 1. If the **-l** option is specified with an *l* argument (i.e. **-ll**), or if **-l** is specified multiple times, a longer list format is used.

-n The **-n** (*non-interactive*) option prevents **sudo** from prompting the user for a password. If a password is required for the command to run, **sudo** will display an error message and exit.

-P The **-P** (*preserve group vector*) option causes **sudo** to preserve the invoking user's group vector unaltered. By default, **sudo** will initialize the group vector to the list of groups the target user is in. The real and effective group IDs, however, are still set to match the target user.

-p *prompt* The **-p** (*prompt*) option allows you to override the default password prompt and use a custom one. The following percent ('%') escapes are supported:

%H
expanded to the host name including the domain name (on if the machine's host name is fully qualified or the *fqdn* option is set in sudoers(5))

%h
expanded to the local host name without the domain name

%p
expanded to the name of the user whose password is being requested (respects the *rootpw*, *targetpw*, and *runaspw* flags in sudoers(5))

%U
expanded to the login name of the user the command will be run as (defaults to root unless the **-u** option is also specified)

%u
expanded to the invoking user's login name

%%
: two consecutive '%' characters are collapsed into a single '%' character

The prompt specified by the **-p** option will override the system password prompt on systems that support PAM unless the *passprompt_override* flag is disabled in *sudoers*.

-r *role*
: The **-r** (*role*) option causes the new (SELinux) security context to have the role specified by *role*.

-S
: The **-S** (*stdin*) option causes **sudo** to read the password from the standard input instead of the terminal device. The password must be followed by a newline character.

-s [*command*]
: The **-s** (*shell*) option runs the shell specified by the SHELL environment variable if it is set or the shell as specified in the password database. If a command is specified, it is passed to the shell for execution via the shell's **-c** option. If no command is specified, an interactive shell is executed.

-t *type*
: The **-t** (*type*) option causes the new (SELinux) security context to have the type specified by *type*. If no type is specified, the default type is derived from the specified role.

-U *user*
: The **-U** (*other user*) option is used in conjunction with the **-l** option to specify the user whose privileges should be listed. Only root or a user with the ALL privilege on the current host may use this option.

-u *user*
: The **-u** (*user*) option causes **sudo** to run the specified command as a user other than *root*. To specify a *uid* instead of a *user name*, *#uid*. When running commands as a *uid*, many shells require that the '#' be escaped with a backslash ('\'). Note that if the *targetpw* Defaults option is set (see sudoers(5)), it is not possible to run commands with a uid not listed in the password database.

-V
: The **-V** (*version*) option causes **sudo** to print its version string and exit. If the invoking user is already root the **-V** option will display the arguments passed to configure when **sudo** was built as well a list of the defaults **sudo** was compiled with as well as the machine's local network addresses.

-v
: When given the **-v** (*validate*) option, **sudo** will update the user's time stamp file, authenticating the user's password if necessary. This extends the **sudo** timeout for another 5 minutes (or whatever the timeout is set to in *sudoers*) but does not run a command.

-- The **--** option indicates that **sudo** should stop processing command line arguments.

Environment variables to be set for the command may also be passed on the command line in the form of **VAR**=*value*, e.g. **LD_LIBRARY_PATH**=*/usr/local/pkg/lib*. Variables passed on the command line are subject to the same restrictions as normal environment variables with one important exception. If the *setenv* option is set in *sudoers*, the command to be run has the SETENV tag set or the command matched is ALL, the user may set variables that would otherwise be forbidden. See sudoers(5) for more information.

Authentication and logging

sudo requires that most users authenticate themselves by default. A password is not required if the invoking user is root, if the target user is the same as the invoking user, or if the authentication has been disabled for the user or command in the *sudoers* file. Unlike su(1), when **sudo** requires authentication, it validates the invoking user's credentials, not the target user's (or root's) credentials. This can be changed via the *rootpw*, *targetpw* and *runaspw* Defaults entries in *sudoers*.

If a user who is not listed in *sudoers* tries to run a command via **sudo**, mail is sent to the proper authorities. The address used for such mail is configurable via the *mailto sudoers* Defaults entry and defaults to root.

Note that mail will not be sent if an unauthorized user tries to run **sudo** with the **-l** or **-v** option. This allows users to determine for themselves whether or not they are allowed to use **sudo**.

If **sudo** is run by root and the SUDO_USER environment variable is set, its value will be used to determine who the actual user is. This can be used by a user to log commands through **sudo** even when a root shell has been invoked. It also allows the **-e** option to remain useful even when invoked via a sudo-run script or program. Note, however, that the *sudoers* lookup is still done for root, not the user specified by SUDO_USER.

sudo uses time stamp files for credential caching. Once a user has been authenticated, the time stamp is updated and the user may then use sudo without a password for a short period of time (5 minutes unless overridden by the *timeout* option). By default, **sudo** uses a tty-based time stamp which means that there is a separate time stamp for each of a user's login sessions. The *tty_tickets* option can be disabled to force the use of a single time stamp for all of a user's sessions.

sudo can log both successful and unsuccessful attempts (as well as errors) to syslog(3), a log file, or both. By default, **sudo** will log via syslog(3) but this is changeable via the *syslog* and *logfile* Defaults settings.

sudo also supports logging a command's input and output streams. I/O logging is not on by default but can be enabled using the *log_input* and *log_output* Defaults flags as well as the LOG_INPUT and LOG_OUTPUT command tags.

Command environment

Since environment variables can influence program behavior, **sudo** provides a means to restrict which variables from the user's environment are inherited by the command to be run. There are two distinct ways *sudoers* can be configured to handle with environment variables.

By default, the *env_reset* option is enabled. This causes commands to be executed with a new, minimal environment. On AIX (and Linux systems without PAM), the environment is initialized with the contents of the */etc/environment* file. On BSD systems, if the *use_loginclass* option is enabled, the environment is initialized based on the *path* and *setenv* settings in */etc/login.conf*. The new environment contains the TERM, PATH, HOME, MAIL, SHELL, LOGNAME, USER, USERNAME and SUDO_* variables in addition to variables from the invoking process permitted by the *env_check* and *env_keep* options. This is effectively a whitelist for environment variables.

If, however, the *env_reset* option is disabled, any variables not explicitly denied by the *env_check* and *env_delete* options are inherited from the invoking process. In this case, *env_check* and *env_delete* behave like a blacklist. Since it is not possible to blacklist all potentially dangerous environment variables, use of the default *env_reset* behavior is encouraged.

In all cases, environment variables with a value beginning with () are removed as they could be interpreted as **bash** functions. The list of environment variables that **sudo** allows or denies is contained in the output of "sudo -V" when run as root.

Note that the dynamic linker on most operating systems will remove variables that can control dynamic linking from the environment of setuid executables, including **sudo**. Depending on the operating system this may include _RLD*, DYLD_*, LD_*, LDR_*, LIBPATH, SHLIB_PATH, and others. These type of variables are removed from the environment before **sudo** even begins execution and, as such, it is not possible for **sudo** to preserve them.

As a special case, if **sudo**'s **-i** option (initial login) is specified, **sudo** will initialize the environment regardless of the value of *env_reset*. The DISPLAY, PATH and TERM variables remain unchanged; HOME, MAIL, SHELL, USER, and LOGNAME are set based on the target user. On AIX (and Linux systems without PAM), the contents of */etc/environment* are also included. On BSD systems, if the *use_loginclass* option is enabled, the *path* and *setenv* variables in */etc/login.conf* are also applied. All other environment variables are removed.

Finally, if the *env_file* option is defined, any variables present in that file will be set to their specified values as long as they would not conflict with an existing environment variable.

COMMAND EXECUTION

When **sudo** executes a command, the *sudoers* file specifies the execution envionment for the command. Typically, the real and effective uid and gid are set to match those of the target user, as specified in the password database, and the group vector is initialized based on the group database (unless the **-P** option was specified).

The *sudoers* file settings affect the following execution parameters:

- real and effective user ID
- real and effective group ID
- supplementary group IDs
- the environment list
- SELinux role and type
- Solaris project
- Solaris privileges
- BSD login class
- file creation mode mask (umask)

See the *Command environment* section for details on how the environment list is constructed.

Process model

If **sudo** has been configured with PAM support or if I/O logging is enabled, **sudo** must wait until the command has completed before it will exit. In the case of PAM, **sudo** must remain running so that it can close the PAM session when the command is finished. If neither PAM nor I/O logging are configured, **sudo** will execute the command without calling fork(2). In either case, **sudo** sets up the execution environment as described above, and calls the execve system call (potentially in a child process). If I/O logging is enabled, a new pseudo-terminal ("pty") is created and a second **sudo** process is used to relay job control signals between the user's existing pty and the new pty the command is being run in. This extra process makes it possible to, for example, suspend and resume the command. Without it, the command would be in what POSIX terms an "orphaned process group" and it would not receive any job control signals.

Signal handling

If the command is run as a child of the **sudo** process (due to PAM or I/O logging), **sudo** will relay signals it receives to the command. Unless the command is being run in a new pty, the SIGHUP, SIGINT and SIGQUIT signals are not relayed unless they are sent by a user process, not the kernel. Otherwise, the command would receive SIGINT twice every time the user entered control-C. Some signals, such as SIGSTOP and SIGKILL, cannot be caught and thus will not be relayed to the command. As a general rule, SIGTSTP should be used instead of SIGSTOP when you wish to suspend a command being run by **sudo**.

As a special case, **sudo** will not relay signals that were sent by the command it is running. This prevents the command from accidentally killing itself. On some systems, the reboot(8) command sends SIGTERM to all non-system processes other than itself before rebooting the systyem. This prevents **sudo** from relaying the SIGTERM signal it received back to reboot(8), which might then exit before the system was actually rebooted, leaving it in a half-dead state similar to single user mode. Note, however, that this check only applies to the command run by **sudo** and not any other processes that the command may create. As a result, running a script that calls reboot(8) or shutdown(8) via **sudo** may cause the system to end up in this undefined state unless the reboot(8) or shutdown(8) are run using the **exec**() family of functions instead of **system**() (which interposes a shell between the command and the calling process).

EXIT VALUE

Upon successful execution of a program, the exit status from *sudo* will simply be the exit status of the program that was executed.

Otherwise, **sudo** exits with a value of 1 if there is a configuration/permission problem or if **sudo** cannot execute the given command. In the latter case the error string is printed to the standard error. If **sudo** cannot stat(2) one or more entries in the user's PATH, an error is printed on stderr. (If the directory does not exist or if it is not really a directory, the entry is ignored and no error is printed.) This should not happen under normal circumstances. The most common reason for stat(2) to return "permission denied" is if you are running an automounter and one of the directories in your PATH is on a machine that is currently unreachable.

LOG FORMAT

sudo can log events using either syslog(3) or a simple log file. In each case the log format is almost identical.

Accepted command log entries

Commands that sudo runs are logged using the following format (split into multiple lines for readability):

```
date hostname progname: username : TTY=ttyname ; PWD=cwd ; \
    USER=runasuser ; GROUP=runasgroup ; TSID=logid ; \
    ENV=env_vars COMMAND=command
```

Where the fields are as follows:

date	The date the command was run. Typically, this is in the format "MMM, DD, HH:MM:SS". If logging via syslog(3), the actual date format is controlled by the syslog daemon. If logging to a file and the *log_year* option is enabled, the date will also include the year.
hostname	The name of the host **sudo** was run on. This field is only present when logging via syslog(3).
progname	The name of the program, usually *sudo* or *sudoedit*. This field is only present when logging via syslog(3).
username	The login name of the user who ran **sudo**.
ttyname	The short name of the terminal (e.g. "console", "tty01", or "pts/0") **sudo** was run on, or "unknown" if there was no terminal present.
cwd	The current working directory that **sudo** was run in.
runasuser	The user the command was run as.
runasgroup	The group the command was run as if one was specified on the command line.
logid	An I/O log identifier that can be used to replay the command's output. This is only present when the *log_input* or *log_output* option is enabled.
env_vars	A list of environment variables specified on the command line, if specified.
command	The actual command that was executed.

Messages are logged using the locale specified by *sudoers_locale*, which defaults to the "C" locale.

Denied command log entries

If the user is not allowed to run the command, the reason for the denial will follow the user name. Possible reasons include:

user NOT in sudoers

The user is not listed in the *sudoers* file.

user NOT authorized on host

The user is listed in the *sudoers* file but is not allowed to run commands on the host.

command not allowed

The user is listed in the *sudoers* file for the host but they are not allowed to run the specified command.

3 incorrect password attempts

The user failed to enter their password after 3 tries. The actual number of tries will vary based on the number of failed attempts and the value of the *passwd_tries sudoers* option.

a password is required

The **-n** option was specified but a password was required.

sorry, you are not allowed to set the following environment variables

The user specified environment variables on the command line that were not allowed by *sudoers*.

Error log entries

If an error occurs, **sudo** will log a message and, in most cases, send a message to the administrator via email. Possible errors include:

parse error in /etc/sudoers near line N

sudo encountered an error when parsing the specified file. In some cases, the actual error may be one line above or below the line number listed, depending on the type of error.

problem with defaults entries

The *sudoers* file contains one or more unknown Defaults settings. This does not prevent **sudo** from running, but the *sudoers* file should be checked using **visudo**.

timestamp owner (username): No such user

The time stamp directory owner, as specified by the *timestampowner* setting, could not be found in the password database.

unable to open/read /etc/sudoers

The *sudoers* file could not be opened for reading. This can happen when the *sudoers* file is located on a remote file system that maps user ID 0 to a different value. Normally, **sudo** tries to open *sudoers* using group permissions to avoid this problem.

unable to stat /etc/sudoers
: The */etc/sudoers* file is missing.

/etc/sudoers is not a regular file
: The */etc/sudoers* file exists but is not a regular file or symbolic link.

/etc/sudoers is owned by uid N, should be 0
: The *sudoers* file has the wrong owner.

/etc/sudoers is world writable
: The permissions on the *sudoers* file allow all users to write to it. The *sudoers* file must not be world-writable, the default file mode is 0440 (readable by owner and group, writable by none).

/etc/sudoers is owned by gid N, should be 1
: The *sudoers* file has the wrong group ownership.

unable to open /var/adm/sudo/username/ttyname
: *sudoers* was unable to read or create the user's time stamp file.

unable to write to /var/adm/sudo/username/ttyname
: *sudoers* was unable to write to the user's time stamp file.

unable to mkdir to /var/adm/sudo/username
: *sudoers* was unable to create the user's time stamp directory.

Notes on logging via syslog

By default, *sudoers* logs messages via syslog(3). The *date*, *hostname*, and *progname* fields are added by the syslog daemon, not *sudoers* itself. As such, they may vary in format on different systems.

On most systems, syslog(3) has a relatively small log buffer. To prevent the command line arguments from being truncated, **sudo** will split up log messages that are larger than 960 characters (not including the date, hostname, and the string "sudo"). When a message is split, additional parts will include the string "(command continued)" after the user name and before the continued command line arguments.

Notes on logging to a file

If the *logfile* option is set, *sudoers* will log to a local file, such as */var/log/sudo*. When logging to a file, *sudoers* uses a format similar to syslog(3), with a few important differences:

1. The *progname* and *hostname* fields are not present.

2. If the *log_year sudoers* option is enabled, the date will also include the year.

3. Lines that are longer than *loglinelen* characters (80 by default) are word-wrapped and continued on the next line with a four character indent. This makes entries easier to read for a human being, but makes it more difficult to use grep(1) on the log files. If the *loglinelen sudoers* option is set to 0 (or negated with a '!'), word wrap will be disabled.

SECURITY NOTES

sudo tries to be safe when executing external commands.

To prevent command spoofing, **sudo** checks "." and "" (both denoting current directory) last when searching for a command in the user's PATH (if one or both are in the PATH). Note, however, that the actual PATH environment variable is *not* modified and is passed unchanged to the program that **sudo** executes.

sudo will check the ownership of its time stamp directory (*/var/adm/sudo* by default) and ignore the directory's contents if it is not owned by root or if it is writable by a user other than root. On systems that allow non-root users to give away files via chown(2), if the time stamp directory is located in a world-writable directory (e.g., */tmp*), it is possible for a user to create the time stamp directory before **sudo** is run. However, because **sudo** checks the ownership and mode of the directory and its contents, the only damage that can be done is to "hide" files by putting them in the time stamp dir. This is unlikely to happen since once the time stamp dir is owned by root and inaccessible by any other user, the user placing files there would be unable to get them back out.

sudo will not honor time stamps set far in the future. Time stamps with a date greater than current_time + 2 * TIMEOUT will be ignored and sudo will log and complain. This is done to keep a user from creating his/her own time stamp with a bogus date on systems that allow users to give away files if the time stamp directory is located in a world-writable directory.

On systems where the boot time is available, **sudo** will ignore time stamps that date from before the machine booted.

Since time stamp files live in the file system, they can outlive a user's login session. As a result, a user may be able to login, run a command with **sudo** after authenticating, logout, login again, and run **sudo** without authenticating so long as the time stamp file's modification time is within 5 minutes (or whatever the timeout is set to in *sudoers*). When the *tty_tickets sudoers* option is enabled, the time stamp has per-tty granularity but still may outlive the user's session. On Linux systems where the devpts filesystem is used, Solaris systems with the devices filesystem, as well as other systems that utilize a devfs filesystem that monotonically increase the inode number of devices as they are created (such as Mac OS X), **sudo** is able to determine when a tty-based time stamp file is stale and will ignore it. Administrators should not rely on this feature as it is not universally available.

Please note that **sudo** will normally only log the command it explicitly runs. If a user runs a command such as sudo su or sudo sh, subsequent commands run from that shell are not subject to **sudo**'s security policy. The same is true for commands that offer shell escapes (including most editors). If I/O logging is enabled, subsequent commands will have their input and/or output logged, but there will not be traditional logs for those commands. Because of this, care must be taken when giving users access to commands via **sudo** to verify that the command does not inadvertently give the user an effective root shell. For more information, please see the *PREVENTING SHELL ESCAPES* section in sudoers(5).

To prevent the disclosure of potentially sensitive information, **sudo** disables core dumps by default while it is executing (they are re-enabled for the command that is run).

For information on the security implications of *sudoers* entries, please see the *SECURITY NOTES* section in sudoers(5).

ENVIRONMENT

sudo utilizes the following environment variables:

EDITOR	Default editor to use in **-e** (sudoedit) mode if neither SUDO_EDITOR nor VISUAL is set.
MAIL	In **-i** mode or when *env_reset* is enabled in *sudoers*, set to the mail spool of the target user.
HOME	Set to the home directory of the target user if **-i** or **-H** are specified, *env_reset* or *always_set_home* are set in *sudoers*, or when the **-s** option is specified and *set_home* is set in *sudoers*.
PATH	Set to a sane value if the *secure_path* option is set in the *sudoers* file.
SHELL	Used to determine shell to run with **-s** option.
SUDO_ASKPASS	Specifies the path to a helper program used to read the password if no terminal is available or if the **-A** option is specified.
SUDO_COMMAND	Set to the command run by sudo.

SUDO_EDITOR	Default editor to use in **-e** (sudoedit) mode.
SUDO_GID	Set to the group ID of the user who invoked sudo.
SUDO_PROMPT	Used as the default password prompt.
SUDO_PS1	If set, PS1 will be set to its value for the program being run.
SUDO_UID	Set to the user ID of the user who invoked sudo.
SUDO_USER	Set to the login name of the user who invoked sudo.
USER	Set to the target user (root unless the **-u** option is specified).
VISUAL	Default editor to use in **-e** (sudoedit) mode if SUDO_EDITOR is not set.

FILES

/etc/sudoers	List of who can run what
/var/adm/sudo	Directory containing time stamps
/etc/environment	Initial environment for **-i** mode on AIX and Linux systems

EXAMPLES

Note: the following examples assume suitable sudoers(5) entries.

To get a file listing of an unreadable directory:

```
$ sudo ls /usr/local/protected
```

To list the home directory of user yaz on a machine where the file system holding ~yaz is not exported as root:

```
$ sudo -u yaz ls ~yaz
```

To edit the *index.html* file as user www:

```
$ sudo -u www vi ~www/htdocs/index.html
```

To view system logs only accessible to root and users in the adm group:

```
$ sudo -g adm view /var/log/syslog
```

To run an editor as jim with a different primary group:

```
$ sudo -u jim -g audio vi ~jim/sound.txt
```

To shut down a machine:

```
$ sudo shutdown -r +15 "quick reboot"
```

To make a usage listing of the directories in the /home partition. Note that this runs the commands in a sub-shell to make the cd and file redirection work.

```
$ sudo sh -c "cd /home ; du -s * | sort -rn > USAGE"
```

SEE ALSO

grep(1), su(1), stat(2), login_cap(3), passwd(5), sudoers(5), sudoreplay(8), visudo(8)

HISTORY

See the HISTORY file in the **sudo** distribution (http://www.sudo.ws/sudo/history.html) for a brief history of sudo.

AUTHORS

Many people have worked on **sudo** over the years; this version consists of code written primarily by:

Todd C. Miller

See the CONTRIBUTORS file in the **sudo** distribution (http://www.sudo.ws/sudo/contributors.html) for an exhaustive list of people who have contributed to **sudo**.

CAVEATS

There is no easy way to prevent a user from gaining a root shell if that user is allowed to run arbitrary commands via **sudo**. Also, many programs (such as editors) allow the user to run commands via shell escapes, thus avoiding **sudo**'s checks. However, on most systems it is possible to prevent shell escapes with **sudo ' s** *noexec* functionality. See the sudoers(5) manual for details.

It is not meaningful to run the cd command directly via sudo, e.g.,

```
$ sudo cd /usr/local/protected
```

since when the command exits the parent process (your shell) will still be the same. Please see the *EXAMPLES* section for more information.

Running shell scripts via **sudo** can expose the same kernel bugs that make setuid shell scripts unsafe on some operating systems (if your OS has a /dev/fd/ directory, setuid shell scripts are generally safe).

BUGS

If you feel you have found a bug in **sudo**, please submit a bug report at http://www.sudo.ws/sudo/bugs/

SUPPORT

Limited free support is available via the sudo-users mailing list, see http://www.sudo.ws/mailman/listinfo/sudo-users to subscribe or search the archives.

DISCLAIMER

sudo is provided "AS IS" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. See the LICENSE file distributed with **sudo** or http://www.sudo.ws/sudo/license.html for complete details.