NAME

sudoers − list of which users may execute what

DESCRIPTION

The *sudoers* file is composed two types of entries: aliases (basically variables) and user specifications (which specify who may run what). The grammar of *sudoers* will be described below in Extended Backus-Naur Form (EBNF). Don't despair if you don't know what EBNF is, it is fairly simple and the definitions below are annotated.

Quick guide to EBNF

EBNF is a concise and exact way of describing the grammar of a language. Each EBNF definition is made up of *production rules*. Eg.

```
symbol ::= definition | alternate1 | alternate2 ...
```

Each *production rule* references others and thus makes up a grammar for the language. EBNF also contains the following operators, which many readers will recognize from regular expressions. Do not, however, confuse them with ''wildcard'' characters, which have different meanings.

`?` Means that the preceding symbol (or group of symbols) is optional. That is, it may appear once or not at all.

`*` Means that the preceding symbol (or group of symbols) may appear zero or more times.

`+` Means that the preceding symbol (or group of symbols) may appear one or more times.

Parentheses may be used to group symbols together. For clarity, we will use single quotes ('') to designate what is a verbatim character string (as opposed to a symbol name).

Aliases

There are four kinds of aliases: the `User_Alias`, `Runas_Alias`, `Host_Alias` and `Cmnd_Alias`.

```
Alias ::= 'User_Alias' = User_Alias (':' User_Alias)* |
          'Runas_Alias' = Runas_Alias (':' Runas_Alias)* |
          'Host_Alias' = Host_Alias (':' Host_Alias)* |
          'Cmnd_Alias' = Cmnd_Alias (':' Cmnd_Alias)*

User_Alias ::= NAME '=' User_List

Runas_Alias ::= NAME '=' Runas_User_List

Host_Alias ::= NAME '=' Host_List

Cmnd_Alias ::= NAME '=' Cmnd_List

NAME ::= [A-Z]([A-Z][0-9]_)*
```

Each *alias* definition is of the form

```
Alias_Type NAME = item1, item2, ...
```

where *Alias_Type* is one of `User_Alias`, `Runas_Alias`, `Host_Alias`, or `Cmnd_Alias`. A `NAME` is a string of upper case letters, numbers, and the underscore characters ('_'). A `NAME` **must** start with an upper case letter. It is possible to put several alias definitions of the same type on a single line, joined by a semicolon (':'). Eg.

```
Alias_Type NAME = item1, item2, item3 : NAME = item4, item5
```

The definitions of what constitutes a valid *alias* member follow.

```
User_List ::= User |
              User ',' User_List

User ::= '!'* username |
         '!'* '#'uid |
         '!'* '%'group |
         '!'* '+'netgroup |
         '!'* User_Alias
```

A User_List is made up of one or more usernames, uids (prefixed with '#'), System groups (prefixed with '%'), netgroups (prefixed with '+') and other aliases. Each list item may be prefixed with one or more '!' operators. An odd number of '!' operators negates the value of the item; an even number just cancel each other out.

```
Runas_List ::= Runas_User |
               Runas_User ',' Runas_List

Runas_User ::= '!'* username |
               '!'* '#'uid |
               '!'* '%'group |
               '!'* +netgroup |
               '!'* Runas_Alias
```

Likewise, a Runas_List has the same possible elements as a User_List, except that it can include a Runas_Alias, instead of a User_Alias.

```
Host_List ::= Host |
              Host ',' Host_List

Host ::= '!'* hostname |
         '!'* ip_addr |
         '!'* network(/netmask)? |
         '!'* '+'netgroup |
         '!'* Host_Alias
```

A Host_List is made up of one or more hostnames, IP addresses, network numbers, netgroups (prefixed with '+') and other aliases. Again, the value of an item may be negated with the '!' operator. If you do not specify a netmask with a network number, the netmask of the host's ethernet *interface*(s) will be used when matching. The netmask may be specified either in dotted quad notation (eg. 255.255.255.0) or CIDR notation (number of bits, eg. 24). A hostname may include shell-style wildcards (see 'Wildcards' section below), but unless the hostname command on your machine returns the fully qualified hostname, you'll need to use the *fqdn* option for wildcards to be useful.

```
Cmnd_List ::= Cmnd |
              Cmnd ',' Cmnd_List

commandname ::= filename |
                filename args |
                filename '""'

Cmnd ::= '!'* commandname |
         '!'* directory |
         '!'* Cmnd_Alias
```

A Cmnd_List is a list of one or more commandnames, directories, and other aliases. A commandname is a fully qualified filename which may include shell-style wildcards (see 'Wildcards' section below). A

simple filename allows the user to run the command with any arguments he/she wishes. However, you may also command line arguments (including wildcards). Alternately, you can specify "" to indicate that the command may only be run **without** command line arguments. A directory is a fully qualified pathname ending in a '/'. When you specify a directory in a Cmnd_List, the user will be able to run any file within that directory (but not in any subdirectories therein).

If a Cmnd has associated command line arguments, then the arguments in the Cmnd must match exactly those given by the user on the command line (or match the wildcards if there are any). Note that the following characters must be escaped with a '\' if they are used in command arguments: ',', ':', '=', '\'.

Defaults

Certain configuration options may be changed from their default values at runtime via one or more Default_Entry lines. These may affect all users on any host, all users on a specific host, or just a specific user. When multiple entries match, they are applied in order. Where there are conflicting values, the last value on a matching line takes effect.

```
Default_Type ::= 'Defaults' ||
                 'Defaults' ':' User ||
                 'Defaults' '@' Host

Default_Entry ::= Default_Type Parameter_List

Parameter ::= Parameter '=' Value ||
              '!'* Parameter ||
```

Parameters may be **flags**, **integer** values, or **strings**. Flags are implicitly boolean and can be turned off via the '!' operator. Some integer and string parameters may also be used in a boolean context to disable them. Values may be enclosed in double quotes (") when they contain multiple words. Special characters may be escaped with a backslash (\).

Flags:

long_otp_prompt
: When validating with a One Time Password scheme (**S/Key** or **OPIE**), a two-line prompt is used to make it easier to cut and paste the challenge to a local window. It's not as pretty as the default but some people find it more convenient. This flag is off by default.

ignore_dot
: If set, **sudo** will ignore '.' or '' (current dir) in $PATH; the $PATH itself is not modified. This flag is off by default.

mail_always
: Send mail to the *mailto* user every time a users runs **sudo**. This flag is off by default.

mail_no_user
: If set, mail will be sent to the *mailto* user if the invoking user is not in the *sudoers* file. This flag is on by default.

mail_no_host
: If set, mail will be sent to the *mailto* user if the invoking user exists in the *sudoers* file, but is not allowed to run commands on the current host. This flag is off by default.

mail_no_perms
: If set, mail will be sent to the *mailto* user if the invoking user allowed to use **sudo** but the command they are trying is not listed in their *sudoers* file entry. This flag is off by default.

tty_tickets
: If set, users must authenticate on a per-tty basis. Normally, **sudo** uses a directory in the ticket dir with the same name as the user running it. With this flag enabled, **sudo** will use a file named for the tty the user is logged in on in that directory. This flag is off by default.

lecture
: If set, a user will receive a short lecture the first time he/she runs **sudo**. This flag is on by default.

authenticate If set, users must authenticate themselves via a password (or other means of authentication) before they may run commands. This default may be overridden via the PASSWD and NOPASSWD tags. This flag is on by default.

root_sudo If set, root is allowed to run **sudo** too. Disabling this prevents users from ‘‘chaining’’ **sudo** commands to get a root shell by doing something like "sudo sudo /bin/sh". This flag is on by default.

log_host If set, the hostname will be logged in the (non-syslog) **sudo** log file. This flag is off by default.

log_year If set, the four-digit year will be logged in the (non-syslog) **sudo** log file. This flag is off by default.

shell_noargs If set and **sudo** is invoked with no arguments it acts as if the -s flag had been given. That is, it runs a shell as root (the shell is determined by the SHELL environment variable if it is set, falling back on the shell listed in the invoking user’s /etc/passwd entry if not). This flag is off by default.

set_home If set and **sudo** is invoked with the -s flag the HOME environment variable will be set to the home directory of the target user (which is root unless the -u option is used). This effectively makes the -s flag imply -H. This flag is off by default.

path_info Normally, **sudo** will tell the user when a command could not be found in their $PATH. Some sites may wish to disable this as it could be used to gather information on the location of executables that the normal user does not have access to. The disadvantage is that if the executable is simply not in the user’s $PATH, **sudo** will tell the user that they are not allowed to run it, which can be confusing. This flag is off by default.

fqdn Set this flag if you want to put fully qualified hostnames in the *sudoers* file. Ie: instead of myhost you would use myhost.mydomain.edu. You may still use the short form if you wish (and even mix the two). Beware that turning on *fqdn* requires **sudo** to make DNS lookups which may make **sudo** unusable if DNS stops working (for example if the machine is not plugged into the network). Also note that you must use the host’s official name as DNS knows it. That is, you may not use a host alias (CNAME entry) due to performance issues and the fact that there is no way to get all aliases from DNS. If your machine’s hostname (as returned by the hostname command) is already fully qualified you shouldn’t need to set *fqfn*. This flag is off by default.

insults If set, **sudo** will insult users when they enter an incorrect password. This flag is off by default.

requiretty If set, **sudo** will only run when the user is logged in to a real tty. This will disallow things like "rsh somehost sudo ls" since *rsh*(1) does not allocate a tty. Because it is not possible to turn of echo when there is no tty present, some sites may with to set this flag to prevent a user from entering a visible password. This flag is off by default.

env_editor If set, **visudo** will use the value of the EDITOR or VISUAL environment falling back on the default editor. Note that this may create a security hole as most editors allow a user to get a shell (which would be a root shell and not be logged).

rootpw If set, **sudo** will prompt for the root password instead of the password of the invoking user.

runaspw If set, **sudo** will prompt for the password of the user defined by the *runas_default* option (defaults to root) instead of the password of the invoking user.

targetpw If set, **sudo** will prompt for the password of the user specified by the -u flag (defaults to root) instead of the password of the invoking user.

set_logname Normally, **sudo** will set the LOGNAME and USER environment variables to the name of the target user (usually root unless the -u flag is given). However, since some programs (including the RCS revision control system) use LOGNAME to determine the real identity of the user, it may be desirable to change this behavior. This can be done by negating the

set_logname option.

use_loginclass
If set, **sudo** will apply the defaults specified for the target user's login class if one exists. Only available if **sudo** is configured with the --with-logincap option.

Integers:

passwd_tries The number of tries a user gets to enter his/her password before **sudo** logs the failure and exits. The default is 3.

Integers that can be used in a boolean context:

loglinelen Number of characters per line for the file log. This value is used to decide when to wrap lines for nicer log files. This has no effect on the syslog log file, only the file log. The default is 80 (use 0 or negate to disable word wrap).

timestamp_timeout
Number of minutes that can elapse before **sudo** will ask for a passwd again. The default is 5, set this to 0 to always prompt for a password.

passwd_timeout
Number of minutes before the **sudo** password prompt times out. The default is 5, set this to 0 for no password timeout.

umask Umask to use when running the root command. Set this to 0777 to not override the user's umask. The default is 0022.

Strings:

mailsub Subject of the mail sent to the *mailto* user. The escape %h will expand to the hostname of the machine. Default is ''*** SECURITY information for %h ***''.

badpass_message
Message that is displayed if a user enters an incorrect password. The default is ''Sorry, try again.'' unless insults are enabled.

timestampdir The directory in which **sudo** stores its timestamp files. The default is *@TIMEDIR@*.

passprompt The default prompt to use when asking for a password; can be overridden via the -p option or the SUDO_PROMPT environment variable. Supports two escapes: ''%u'' expands to the user's login name and ''%h'' expands to the local hostname. The default value is ''Password:''.

runas_default The default user to run commands as if the -u flag is not specified on the command line. This defaults to ''root''.

syslog_goodpri
Syslog priority to use when user authenticates successfully. Defaults to ''notice''.

syslog_badpri
Syslog priority to use when user authenticates unsuccessfully. Defaults to ''alert''.

editor Path to the editor to be used by **visudo**. The default is the path to vi on your system.

Strings that can be used in a boolean context:

logfile Path to the **sudo** log file (not the syslog log file). Setting a path turns on logging to a file, negating this option turns it off.

syslog Syslog facility if syslog is being used for logging (negate to disable syslog logging). Defaults to ''local2''.

mailerpath Path to mail program used to send warning mail. Defaults to the path to sendmail found at configure time.

mailerflags Flags to use when invoking mailer. Defaults to `-t`.

mailto Address to send warning and erorr mail to. Defaults to ‘‘root’’.

exempt_group
Users in this group are exempt from password and PATH requirements. This is not set by default.

secure_path Path used for every command run from **sudo**. If you don’t trust the people running **sudo** to have a sane PATH environment variable you may want to use this. Another use is if you want to have the ‘‘root path’’ be separate from the ‘‘user path.’’ This is not set by default.

verifypw This option controls when a password will be required when a user runs **sudo** with the **−v**. It has the following possible values:

```
all         All the user’s I<sudoers> entries for the
            current host must have the C<NOPASSWD>
            flag set to avoid entering a password.

any         At least one of the user’s I<sudoers> entries
            for the current host must have the
            C<NOPASSWD> flag set to avoid entering a
            password.

never       The user need never enter a password to use
            the B<-v> flag.

always      The user must always enter a password to use
            the B<-v> flag.
```

The default value is ‘all’.

listpw This option controls when a password will be required when a user runs **sudo** with the **−l**. It has the following possible values:

```
all         All the user’s I<sudoers> entries for the
            current host must have the C<NOPASSWD>
            flag set to avoid entering a password.

any         At least one of the user’s I<sudoers> entries
            for the current host must have the
            C<NOPASSWD> flag set to avoid entering a
            password.

never       The user need never enter a password to use
            the B<-l> flag.

always      The user must always enter a password to use
            the B<-l> flag.
```

The default value is ‘any’.

When logging via *syslog* (3), **sudo** accepts the following values for the syslog facility (the value of the **syslog** Parameter): **authpriv** (if your OS supports it), **auth**, **daemon**, **user**, **local0**, **local1**, **local2**, **local3**, **local4**, **local5**, **local6**, and **local7**. The following syslog priorities are supported: **alert**, **crit**, **debug**, **emerg**, **err**, **info**, **notice**, and **warning**.

User Specification

```
User_Spec ::= User_list Host_List '=' User_List Cmnd_Spec_List \
              (':' User_Spec)*

Cmnd_Spec_List ::= Cmnd_Spec |
                   Cmnd_Spec ',' Cmnd_Spec_List

Cmnd_Spec ::= Runas_Spec? ('NOPASSWD:' | 'PASSWD:')? Cmnd

Runas_Spec ::= '(' Runas_List ')'
```

A **user specification** determines which commands a user may run (and as what user) on specified hosts. By default, commands are run as **root** but this can be changed on a per-command basis.

Let's break that down into its constituent parts:

Runas_Spec

A `Runas_Spec` is simply a `Runas_List` (as defined above) enclosed in a set of parentheses. If you do not specify a `Runas_Spec` in the user specification, a default `Runas_Spec` of **root** will be used. A `Runas_Spec` sets the default for commands that follow it. What this means is that for the entry:

```
dgb    boulder = (operator) /bin/ls, /bin/kill, /usr/bin/who
```

The user **dgb** may run */bin/ls*, */bin/kill*, and */usr/bin/lprm* -- but only as **operator**. Eg.

```
    sudo -u operator /bin/ls.
```

It is also possible to override a `Runas_Spec` later on in an entry. If we modify the entry like so:

```
dgb    boulder = (operator) /bin/ls, (root) /bin/kill, /usr/bin/lprm
```

Then user **dgb** is now allowed to run */bin/ls* as **operator**, but */bin/kill* and */usr/bin/lprm* as **root**.

NOPASSWD and PASSWD

By default, **sudo** requires that a user authenticate him or herself before running a command. This behavior can be modified via the `NOPASSWD` tag. Like a `Runas_Spec`, the `NOPASSWD` tag sets a default for the commands that follow it in the `Cmnd_Spec_List`. Conversely, the `PASSWD` tag can be used to reverse things. For example:

```
ray    rushmore = NOPASSWD: /bin/kill, /bin/ls, /usr/bin/lprm
```

would allow the user **ray** to run */bin/kill*, */bin/ls*, and */usr/bin/lprm* as root on the machine rushmore as **root** without authenticating himself. If we only want **ray** to be able to run */bin/kill* without a password the entry would be:

```
ray    rushmore = NOPASSWD: /bin/kill, PASSWD: /bin/ls, /usr/bin/lprm
```

Note however, that the `PASSWD` tag has no effect on users who are in the group specified by the exempt_group option.

By default, if the `NOPASSWD` tag is applied to any of the entries for a user on the current host, he or she will be able to run `sudo -l` without a password. Additionally, a user may only run `sudo -v` without a password if the `NOPASSWD` tag is present for all a user's entries that pertain to the current host. This behavior may be overridden via the verifypw and listpw options.

Wildcards (aka meta characters):

sudo allows shell-style *wildcards* to be used in pathnames as well as command line arguments in the *sudoers* file. Wildcard matching is done via the **POSIX** `fnmatch(3)` routine. Note that these are *not* regular expressions.

`*` Matches any set of zero or more characters.

`?` Matches any single character.

`[...]` Matches any character in the specified range.

`[!...]` Matches any character **not** in the specified range.

`\x` For any character ‘‘x’’, evaluates to ‘‘x’’. This is used to escape special characters such as: ‘‘*’’, ‘‘?’’, ‘‘[’’, and ‘‘}’’.

Note that a forward slash (’/’) will **not** be matched by wildcards used in the pathname. When matching the command line arguments, however, as slash **does** get matched by wildcards. This is to make a path like:

```
/usr/bin/*
```

match `/usr/bin/who` but not `/usr/bin/X11/xterm`.

Exceptions to wildcard rules:

The following exceptions apply to the above rules:

`""` If the empty string `""` is the only command line argument in the *sudoers* entry it means that command is not allowed to be run with **any** arguments.

Other special characters and reserved words:

The pound sign (’#’) is used to indicate a comment (unless it occurs in the context of a user name and is followed by one or more digits, in which case it is treated as a uid). Both the comment character and any text after it, up to the end of the line, are ignored.

The reserved word **ALL** is a built in *alias* that always causes a match to succeed. It can be used wherever one might otherwise use a `Cmnd_Alias`, `User_Alias`, `Runas_Alias`, or `Host_Alias`. You should not try to define your own *alias* called **ALL** as the built in alias will be used in preference to your own. Please note that using **ALL** can be dangerous since in a command context, it allows the user to run **any** command on the system.

An exclamation point (’!’) can be used as a logical *not* operator both in an *alias* and in front of a `Cmnd`. This allows one to exclude certain values. Note, however, that using a `!` in conjunction with the built in `ALL` alias to allow a user to run ‘‘all but a few’’ commands rarely works as intended (see SECURITY NOTES below).

Long lines can be continued with a backslash (’\’) as the last character on the line.

Whitespace between elements in a list as well as specicial syntactic characters in a *User Specification* (’=’, ‘:’, ‘(’, ‘)’) is optional.

The following characters must be escaped with a backslash (’\’) when used as part of a word (eg. a username or hostname): ’@’, ‘!’, ‘=’, ‘:’, ‘,’, ‘(’, ‘)’, ‘\’.

EXAMPLES

Below are example *sudoers* entries. Admittedly, some of these are a bit contrived. First, we define our *aliases*:

```
# User alias specification
User_Alias     FULLTIMERS = millert, mikef, dowdy
User_Alias     PARTTIMERS = bostley, jwfox, crawl
User_Alias     WEBMASTERS = will, wendy, wim
```

```
# Runas alias specification
Runas_Alias    OP = root, operator
Runas_Alias    DB = oracle, sybase

# Host alias specification
Host_Alias     SPARC = bigtime, eclipse, moet, anchor :\
               SGI = grolsch, dandelion, black :\
               ALPHA = widget, thalamus, foobar :\
               HPPA = boa, nag, python
Host_Alias     CUNETS = 128.138.0.0/255.255.0.0
Host_Alias     CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
Host_Alias     SERVERS = master, mail, www, ns
Host_Alias     CDROM = orion, perseus, hercules

# Cmnd alias specification
Cmnd_Alias     DUMPS = /usr/bin/mt, /usr/sbin/dump, /usr/sbin/rdump,\
                       /usr/sbin/restore, /usr/sbin/rrestore
Cmnd_Alias     KILL = /usr/bin/kill
Cmnd_Alias     PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias     SHUTDOWN = /usr/sbin/shutdown
Cmnd_Alias     HALT = /usr/sbin/halt, /usr/sbin/fasthalt
Cmnd_Alias     REBOOT = /usr/sbin/reboot, /usr/sbin/fastboot
Cmnd_Alias     SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                        /usr/local/bin/tcsh, /usr/bin/rsh, \
                        /usr/local/bin/zsh
Cmnd_Alias     SU = /usr/bin/su
```

Here we override some of the compiled in default values. We want **sudo** to log via *syslog* (3) using the *auth* facility in all cases. We don't want to subject the full time staff to the **sudo** lecture, and user **millert** need not give a password. In addition, on the machines in the *SERVERS* `Host_Alias`, we keep an additional local log file and make sure we log the year in each log line since the log entries will be kept around for several years.

```
# Override builtin defaults
Defaults               syslog=auth
Defaults:FULLTIMERS    !lecture
Defaults:millert       !authenticate
Defaults@SERVERS       log_year, logfile=/var/log/sudo.log
```

The *User specification* is the part that actually determines who may run what.

```
root           ALL = (ALL) ALL
%wheel         ALL = (ALL) ALL
```

We let **root** and any user in group **wheel** run any command on any host as any user.

```
FULLTIMERS     ALL = NOPASSWD: ALL
```

Full time sysadmins (**millert**, **mikef**, and **dowdy**) may run any command on any host without authenticating themselves.

```
PARTTIMERS     ALL = ALL
```

Part time sysadmins (**bostley**, **jwfox**, and **crawl**) may run any command on any host but they must authenticate themselves first (since the entry lacks the `NOPASSWD` tag).

```
jack           CSNETS = ALL
```

The user **jack** may run any command on the machines in the *CSNETS* alias (the networks `128.138.243.0`, `128.138.204.0`, and `128.138.242.0`). Of those networks, only <128.138.204.0> has an explicit netmask (in CIDR notation) indicating it is a class C network. For the other networks in *CSNETS*, the local machine's netmask will be used during matching.

```
lisa            CUNETS = ALL
```

The user **lisa** may run any command on any host in the *CUNETS* alias (the class B network `128.138.0.0`).

```
operator        ALL = DUMPS, KILL, PRINTING, SHUTDOWN, HALT, REBOOT,\
                /usr/oper/bin/
```

The **operator** user may run commands limited to simple maintenance. Here, those are commands related to backups, killing processes, the printing system, shutting down the system, and any commands in the directory */usr/oper/bin/*.

```
joe             ALL = /usr/bin/su operator
```

The user **joe** may only *su* (1) to operator.

```
pete            HPPA = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root
```

The user **pete** is allowed to change anyone's password except for root on the *HPPA* machines. Note that this assumes *passwd* (1) does not take multiple usernames on the command line.

```
bob             SPARC = (OP) ALL : SGI = (OP) ALL
```

The user **bob** may run anything on the *SPARC* and *SGI* machines as any user listed in the *OP* `Runas_Alias` (**root** and **operator**).

```
jim             +biglab = ALL
```

The user **jim** may run any command on machines in the *biglab* netgroup. **Sudo** knows that ''biglab'' is a netgroup due to the '+' prefix.

```
+secretaries    ALL = PRINTING, /usr/bin/adduser, /usr/bin/rmuser
```

Users in the **secretaries** netgroup need to help manage the printers as well as add and remove users, so they are allowed to run those commands on all machines.

```
fred            ALL = (DB) NOPASSWD: ALL
```

The user **fred** can run commands as any user in the *DB* `Runas_Alias` (**oracle** or **sybase**) without giving a password.

```
john            ALPHA = /usr/bin/su [!-]*, !/usr/bin/su *root*
```

On the *ALPHA* machines, user **john** may su to anyone except root but he is not allowed to give *su* (1) any flags.

```
jen             ALL, !SERVERS = ALL
```

The user **jen** may run any command on any machine except for those in the *SERVERS* `Host_Alias` (master, mail, www and ns).

```
jill            SERVERS = /usr/bin/, !SU, !SHELLS
```

For any machine in the *SERVERS* `Host_Alias`, **jill** may run any commands in the directory /usr/bin/ except for those commands belonging to the *SU* and *SHELLS* `Cmnd_Aliases`.

```
steve           CSNETS = (operator) /usr/local/op_commands/
```

The user **steve** may run any command in the directory /usr/local/op_commands/ but only as user operator.

```
matt            valkyrie = KILL
```

On his personal workstation, valkyrie, **matt** needs to be able to kill hung processes.

```
WEBMASTERS      www = (www) ALL, (root) /usr/bin/su www
```

On the host www, any user in the *WEBMASTERS* `User_Alias` (will, wendy, and wim), may run any command as user www (which owns the web pages) or simply *su* (1) to www.

```
ALL             CDROM = NOPASSWD: /sbin/umount /CDROM,\
                /sbin/mount -o nosuid\,nodev /dev/cd0a /CDROM
```

Any user may mount or unmount a CD−ROM on the machines in the CDROM `Host_Alias` (orion, perseus, hercules) without entering a password. This is a bit tedious for users to type, so it is a prime candidate for encapsulating in a shell script.

SECURITY NOTES

It is generally not effective to ‘‘subtract’’ commands from `ALL` using the ‘!’ operator. A user can trivially circumvent this by copying the desired command to a different name and then executing that. For example:

```
    bill        ALL = ALL, !SU, !SHELLS
```

Doesn’t really prevent **bill** from running the commands listed in *SU* or *SHELLS* since he can simply copy those commands to a different name, or use a shell escape from an editor or other program. Therefore, these kind of restrictions should be considered advisory at best (and reinforced by policy).

CAVEATS

The *sudoers* file should **always** be edited by the **visudo** command which locks the file and does grammatical checking. It is imperative that *sudoers* be free of syntax errors since **sudo** will not run with a syntactically incorrect *sudoers* file.

When using netgroups of machines (as opposed to users), if you store fully qualified hostnames in the netgroup (as is usually the case), you either need to have the machine’s hostname be fully qualified as returned by the `hostname` command or use the *fqdn* option in *sudoers*.

FILES

```
/etc/sudoers            List of who can run what
/etc/group              Local groups file
/etc/netgroup           List of network groups
```

SEE ALSO

sudo (8), *visudo* (8), *su* (1), *fnmatch* (3).