

NAME

`sus` - super user shell

SYNOPSIS

```
sus [-CIMPRSVXabdfhklnt] [-E filename] [-N priority] [-c
dir] [-e filename] [-p user[@host]] [-u user] [-g group] [-U
user] [-G group] cmd [arguments...]
```

DESCRIPTION

Sus is a setuid program which allows the user who ran *sus* (the source user) to run a single command as a target user, typically the super user. It consults a configuration file to determine if the source user is allowed to execute the desired command as the target user. By default, the target user is the super user (root).

AUTHENTICATION

If the command is allowed and authentication is required the user will be prompted for a password. *Sus* supports a private password file, the standard system password databases or PAM.

If the user has successfully invoked *sus* within a set period of time (typically 5 minutes), the password prompt is skipped and the command is executed without further authentication.

If standard input is a tty style device, *sus* will issue a prompt before reading the password otherwise no prompt is issued.

OPTIONS

- A `prompt_string`
Override the default string used for prompting for a password. Note that if PAM is being used and issues multiple prompts (rare), all prompts will be set to this value.
- C Show the defines being passed to the internal CPP type preprocessor.
- E `filename`
Edit the specified file as the target user. The file is opened as root and locked via `fcntl(2)` before the editor is invoked. Editor is determined by the `VISUAL` or `EDITOR` environment variable. Except for the locking of the file, this is EXACTLY equivalent to invoking *sus* with `$VISUAL` or `$EDITOR` as the command, and *filename* as the argument so all checks against the configuration file apply as normal.

- G group
Specify an effective group for the target command. The group must match against the contents of the ALLOWED_EFFECTIVE_GROUPS macro. Can be a group name or a group id. The group must exist.
- I Do not initialize the supplementary groups of the target user. Only allowed if the macro ALLOW_NO_SUPPLEMENTARY_GROUPS evaluates to true.
- M Show the values of all macros currently defined when a command would be either accepted or denied. Only valid when in pretend mode.
- N priority
Run the command at the specified nice(2) level.
- P Run in promiscuous mode. See the section below. This turns on -a and -n and requires a non-root target user to be specified via -u.
- R Print resource usage information upon exit.
- S Simulate a login as the target user.
The target user must match a LOGIN class entry as the command in the configuration file. See the description of the LOGIN class below.
- T Do not use the controlling tty for reading passwords.
Use stdin instead. No prompt will be issued and there is no echo suppression.
- U user
Specify an effective user for the target command. The effective user specified must match against the ALLOWED_EFFECTIVE_USERS macro and must exist. Can be a numeric user id or a username.
- V Print version information and exit.
- X Copy the users XAUTHORITY file to /tmp, change the ownership to the target user and set the environment variable XAUTHORITY to point to the new location for the target command. In other words, try to make X11 work for the new user back to the existing display server. This has security implications, use with care. If the target user can read a users home directory, then X11 will work if the XAUTHORITY environment variable is simply passed through to the target environment. The copy of the users XAUTHORITY file is removed after the command terminates only if the command is not backgrounded and sus is configured to wait for the target

- command to terminate. The feature is disabled if the macro NO_X11 is true.
- a Always do the authentication step.
 - b Run the command in background. The command becomes a session leader. This is not permitted if the macro NO_BACKGROUNDING evaluates to True.
 - c directory
Perform a chdir to the specified directory before running the command. May be specified multiple times and the calls to chdir(2) are executed in the order specified. If any fail, no command is run. This is done as the target user just before any command is run.
 - d Enable debugging output. This option is only available if *sus* was compiled with debugging support enabled which is the default. Also, the source user must be root. This produces a lot of output and is probably only useful to the author. Works best if its the first argument otherwise some initialization happens before the flag is activated.
 - e filename
Edit file as invoking user. The filename specified must match a SUSEDIT class entry as the command in the configuration file. See the description of the SUSEDIT class below. Filename is opened as root and locked via fcntl(2) locking, then copied to a temporary file. The ownership of the temporary file is then changed to the invoking user. \$VISUAL or \$EDITOR is then invoked on the temporary file as the invoking user with the original command environment. After the editor process terminates, if the file is modified, it is copied back to the original location. NOTE: the copy is done in place, so if the machine crashes or other problems occur it is possible to corrupt the original file. If everthing is successfull, the temporary file is removed. NOTE: *sus* uses /tmp for the temporary location of the directory containing the file being edited. This can be overridden by the TMPDIR macro. The directory created is not removed since it is unique to each user and can be reused by subsequent calls to *sus*. TMPDIR could be set to the user home directory in the control file.
 - f List the control file after preprocessing (super user only).
 - g group
Run the command as this real group. The group must match against the macro ALLOWED_REAL_GROUPS. Can be a

- numeric group id or a group name.
- h Print usage information and exit.
 - k Invalidate the user's timestamp. This command does not require a password. This is slightly problematical if you set the value of `TIMESTAMP_DIR` more than once in the control file. The value in effect at the end of the control file is used.
 - l List the commands (actually the expressions used for matching against commands) that the current user can run.
 - n Never update the timestamps.
 - p `user[@hostname]`
Pretend mode. Pretend the invoking user is `user` and pretend the current host is `hostname` (super user only). Test mode is also turned on by this option. `Hostname` can be a name (short or a FQDN) or an IP address (v4 or V6). The results of simulating a machine other than the current one depend heavily on name service answers and may not produce the same results as running `sus` on the actual machine itself.
 - s Invoke a shell. Try to get a shell from either the `SHELL` environment variable, the users `passwd(5)` entry or a built in default (usually `/bin/sh`).
 - t Test mode. Simply print out if the command is allowed. No command is ever run in this mode.
 - u `user`
Run the command as this real user. If promiscuous mode is not specified, the target user must match against the macro `ALLOWED_REAL_USERS`. The specified user must exist. Can be a numerid user id or a username.
 - Terminates option processing. Remaining arguments are the command to run.

CONFIGURATION

General

Any configuration file is a UNIX text file.

It (and any file it includes) must be owned by the root account and only writable by the owner. Lines containing only whitespace are ignored. The file supports C style comments.

The configuration file is first fed through a CPP style preprocessor (without creating another process). See the section on PREPROCESSING.

Syntax

After preprocessing, the configuration file contains lines with the the following syntax:

```

directive ::= user_list : command_list
             ::= FAIL

user_list ::= full_user_expression{, ...}

command_list ::= command_expression{, ...}

full_user_expression ::= user_expression{@host_expression}

user_expression ::= class_expression

command_expression ::= class_expression{ ...}

host_expression ::= class_expression

class_expression ::= file_class
                    ::= group_class
                    ::= user_class
                    ::= proc_class
                    ::= host_class
                    ::= match_class
                    ::= search_class
                    ::= string_class
                    ::= not_class
                    ::= and_class
                    ::= or_class
                    ::= susedit_class
                    ::= login_class
                    ::= none_class
                    ::= intcmp_class
                    ::= string

user_class ::= USER(selector_list)
             ::= USERNAME(selector_list)
             ::= UID(selector_list)

file_class ::= FILE(selector_list)
             ::= DIR(selector_list)
             ::= RFILE(selector_list)
             ::= LFILE(selector_list)
             ::= LRFILE(selector_list)

group_class ::= GROUP(selector_list)

```

```

        ::= GROUPID(selector_list)
        ::= GROUPNAME(selector_list)

proc_class ::= PROC(selector_list)
             ::= RPROC(selector_list)

host_class ::= HOST(selector_list)

not_class  ::= NOT(class_expression)

and_class ::= AND(class_expression{, ..})

or_class  ::= OR(class_expression{, ...})

match_class ::= MATCH(string)

search_class ::= SEARCH(string)

search_class ::= STRING(string)
             ::= ISTRING(string)

susedit_class ::= SUSEDIT(class_expression)

login_class  ::= LOGIN(class_expression)

none_class   ::= NONE(class_expression{, ...})

intcmp_class ::= EQ(integer)
             ::= NE(integer)
             ::= GT(integer)
             ::= LT(integer)
             ::= GE(integer)
             ::= LE(integer)

selector_list ::= selector_argument{,selector_argument}

selector_argument ::= attribute_name = attribute
                   ::= attribute_name != attribute

attribute ::= class_expression

```

FAIL

If a line in the control file contains the word **FAIL** only, the rest of the control file is ignored.

USER Class

The user class supports the following attribute selectors.

Attribute Name	Matches
exists	true if user exists, false otherwise
uid	user id of the user
name	user name of the user
gid	primary group id of the user
ingroup	all users groups
gecos	gecos field for user
dir	home directory of user
home	home directory of user
shell	shell of user

The USER class will try to match against a username or a userid. The UID class will only match a userid (wholly numeric) but supports all the same attributes as the USER class. The USERNAME class supports all the attributes of the USER class, but will only try to match against a string which is not wholly numeric.

GROUP Class

The group class supports the following attribute selectors.

Attribute Name	Matches
exists	true if group exists, false otherwise
gid	gid of group
name	name of group
members	members of group

The GROUP class will try to match against a groupname or a groupid. The GID class will only match a groupid (wholly numeric) but supports all the same attributes as the GROUP class. The GROUPNAME class supports all the attributes of the GROUP class, but will only try to match against a string which is not wholly numeric.

FILE Class

The FILE class supports the following attribute selectors.

Attribute Name	Matches
exists	true if file exists, false otherwise
symlink	true if a symlink was traversed, false otherwise
name	name of file
realname	real pathname of file if available
type	type of the file

nlink	number of hard links
uid	user id of the file
gid	group id of the file
dev	device on which file resides
rdev	raw device on which file resides
size	size of the file in bytes
mode	file mode (octal, no leading zero)

File types are: reg (regular files), dir (directories), chr (character special), blk (block special), door (solaris only), socket and fifo (named pipe).

The class RFILE supports all the same attributes as class FILE, but if match is not found with the file an attempt is made to match with the files parent directory and so on up the file system tree to the root. The search terminates as soon as a match is found.

The class DIR supports all the same attributes as class FILE, but all matching is performed on the parent directory of the file object referenced rather than the file object itself.

PROC Class

The PROC class supports the following attribute selectors.

Attribute Name	Matches
exists	true if process exists, false otherwise
pid	process id of the process
ppid	parent process id of the process
uid	user id of the process
puid	user id of parent process
euid	effective user id of process
gid	group id of process
pgid	group id of parent process
egid	effective group id of process
sid	session id of process
argc	number of arguments of process
ingroup	all the processes groups

The class RPROC supports all the same attributes as class PROC, but if a match is not found with the file an attempt is made to match with with the processes parent and so on up the process tree to the root. The search terminates as soon as a match is found.

HOST Class

The HOST class supports the following attribute selectors.

Attribute Name	Matches
exists	true if host exists, "false" otherwise
name	matches name(s) of the host
ip	matches IP addresses (v4 or v6)

CIDR addresses (v4 or V6) are supported. Host names are always lower case.

SUSEDIT Class

The SUSEDIT class supports no attributes. The expression it contains is matched against the filename argument supplied via the `-e` option. If a match is found, the user may edit the filename so specified as themselves. The file to be edited must exist. See the description of the `-e` option above. If the `-e` option is not in effect, this class always returns false.

LOGIN Class

The LOGIN class supports no attributes. The expression it contains is matched against the target user (root or the user supplied via the `-u` option). If a match is found, the user may invoke a login shell as the target user. If the `-S` option has not been specified, this class always returns false.

MATCH Class

The MATCH class supports no attributes. The expression it contains is used as an anchored, extended regular expression.

SEARCH Class

The SEARCH class is identical to the MATCH class above, except the expression is treated as an unanchored, extended regular expression.

STRING class

The STRING class takes no attributes. The expression it contains is used as a simple string and is matched exactly.

ISTRING class

The ISTRING class is identical to class STRING, except the match is made case invarient.

NOT Class

The NOT class supports no attributes. It simply negates the *class_expression* it contains.

AND Class

The AND class supports no attributes. It simply evaluates the *class_expressions* it contains, and returns the conjunction.

OR Class

Similar to the AND class above, except returns the disjunction.

INTCMP Class

This class provides (albeit in a very cumbersome fashion) the ability to perform integer comparisons. The *class_expression* must be a an integer (see the section on integers below) otherwise an error is generated. The class may be called by several names. The name used controls the type of comparison performed (LT for less than, GE for greater than or equal to and so on). If a string which cannot be converted to an integer is used for comparison, the result is always false.

NONE Class

Any time this class is used, processing stops immediately with an error message.

Strings

If a *class_expression* does not match the forms of the *class_expressions* listed above, it is considered a string. Normally, this is exactly equivalent to using MATCH(string) ie the string is considered an anchored, extended regular expression. However, 2 other options are available via a compile time switch. The strings may be equivalent to STRING(string) or NONE(string).

If a string contains whitespace, parentheses, backslash or double quotes, they must quoted. A single backslash always removed any special meaning from the preceeding character. Double quotes quote whatever they contain. A backslash can be used to quote a double quote inside other double quotes.

Integers

In cases where a string in the control file is meant to represent an integer constant the conversion to an integer value is performed by the UNIX `strtoll(3)` utility. Decimal, octal and hexadecimal formats are supported.

The last non-whitespace character of the string can be used to specify a multiplier. The multiplier values are:

Multiplier	Value multiplied by
B	1
K	1024
M	1024 * 1024
G	1024 * 1024 * 1024
T	1024 * 1024 * 1024 * 1024
P	1024 * 1024 * 1024 * 1024 * 1024
E	1024 * 1024 * 1024 * 1024 * 1024 * 1024
s	1
m	60
h	60 * 60
d	60 * 60 * 24
w	60 * 60 * 24 * 7
y	60 * 60 * 24 * 365

When using hexadecimal, multipliers which are valid hexadecimal digits cannot be used.

Booleans

In cases where a string represents a boolean flag, the value of the boolean is TRUE if its value starts with the characters 'T', 't', 'Y' or 'y'. In all other cases (including not defined), the value is defined to be FALSE.

Semantic Issues

If a string is deemed to represent an IP address it is converted to IP addresses using `inet_pton(3)`. All IP addresses are represented internally as IPv6 addresses. IPv4 addresses can be used in the configuration file.

Not all classes can be used anywhere a *class_expression* is allowed. *Sus* does not allow comparisons where it thinks they do not make sense. For example, when matching the current user against the *user_expression* it is illegal to use the PROC class. The NOT, OR, AND, SEARCH, MATCH and STRING are always allowed.

When matching a token from the configuration file against the known classes defined above, if no match is found the token is treated as either MATCH(token) or STRING(token) or NONE(token) depending on a compile time switch.

See examples.

OPERATION

A user invokes *sus* specifying the command they wish to run, (and optionally the user and/or the effective user and/or the effective group they wish to run the command as) and the arguments to that command, if any.

The configuration file is read 1 line at a time. The source user and the current host are matched against each "full_user_expression" on the current line. If a match is not found, the line is ignored.

If a match is found, the requested command and arguments are matched against each *command_expression* on the current line. If no match is found, the line is ignored. If a match is found (the user is allowed to run the command) *sus* will invoke the command on behalf of the source user as the requested target user (usually the super user).

TARGET COMMAND ENVIRONMENT

Sus can be configured to completely replace the environment before running the command, allow the environment or parts thereof to be passed through to the command or to remove only selected environment variables before running the target command.

System resource limits are all removed or set to values defined in the control file before command execution. Signal handling is set to default for all signals.

TIMESTAMPS

Whenever *sus* is successfully invoked, it may attempt to create a timestamp file, usually in the root directory.

If the timestamp file exists, the time it was created is extracted and if sufficiently close to the current time but not in the future it is considered valid. If valid, no authentication is required by the user. Macros defined in the control plus some options control details of timestamp file location and how long a timestamp is considered valid as well as whether a timestamp file is created.

PROMISCUOUS MODE

This mode is only allowed when a target user other than root is specified. The authentication password, which is always required, is the password for the target user rather than the invoking user. It is most useful for CGI scripts where a WWW user enters their own password into a form which then allows them to run a CGI script as themselves, for example to change their own password. *Sus* never allows a command to be run as root in promiscuous mode. The macro `PROMISCUOUS` is defined if the user is using this mode. All the usual checks apply.

LOGGING

Sus can log all invocations by directly updating a log file or via `syslog(3)`. All log entries contain the time of invocation, the user name of the invoking user, the current directory and the command requested, including arguments. If the macro, `WAIT_FOR_CHILD` is defined and true in the configuration file, *sus* will run the requested command as a child, and log some accounting information when the command terminates.

PREPROCESSING

The configuration file is preprocessed by a CPP style macro preprocessor. See the accompanying man pages on `SUSCPP(1)` for details of operation. Note that `SUSCPP` is set up in CPP compatibility mode but this can be changed in the control file.

In cases where a macro value is a string, white space is removed unless quoted. This allows string macros to easily span multiple lines. Note that the preprocessor interprets some characters specially before they are seen by the *sus* parser. In particular, unquoted backslash characters should be used with care.

Macros defined via `#define` commands in the configuration file may be used to control the operation of *sus*. These macros and their types and meanings are:

ALLOW_INVALID_SHELL

(boolean) Normally *sus* will not allow either the invoking user nor the target user to have an invalid shell as defined by `/etc/shells`. If this macro is set to true, the checks for invalid shells are disabled. Default false.

LOG_OWNER

(string) The user id or user name of the owner of the

log file. Default root.

LOG_GROUP
(string) The group name or group id of the group of the log file. Default root.

LOG_PERMS
(integer) The permissions on the log file. Default 0600.

LOG_TO_SYSLOG
(boolean) If true, log via the syslog daemon. Default true.

LOG_FACILITY
(string) The syslog log facility to use for syslog style logging. Default LOG_AUTH.

LOG_LEVEL
(string) The syslog log level to use for syslog style logging. Default LOG_NOTICE

LOG_IDENT
(string) The identity to use for syslog style logging. Default "sus".

LOG_FILE
(string) The name of the log file. Default None.

TIMESTAMP_TTL
(time) The time a timestamp remains valid (time between requiring authentication). Default 5 minutes.

UMASK
(integer) The umask set before a command is run. Default 077.

TMPDIR
(string) Directory used for temporary files. Default "/tmp".

WAIT_FOR_CHILD
(boolean) If true, run the command as a child and wait for termination and log usage information. If false, the command overlays (via exec(2)) the instance of sus. Default false.

CHILD_WAIT_TIMEOUT
(time) If WAIT_FOR_CHILD is true, sets a maximum amount of time to wait for the child process to terminate. If exceeded, behaviour depends on KILL_CHILD_ON_WAIT_TIMEOUT below. Default forever.

KILL_CHILD_ON_WAIT_TIMEOUT

(boolean) If true, if `CHILD_WAIT_TIMEOUT` is exceeded while waiting for the child process, kill the child process and log an error message. If false, only a warning log message is generated and `sus` continues to wait. Default false.

TIMESTAMP_DIR

(string) Directory to contain timestamp. Default `"/`.

WRITE_TIMESTAMP

(boolean) If true, attempt to write the timestamp, else do not. Default false.

TIMESTAMP_SCOPE

(string) This macro can be used to tighten control on whether a timestamp is considered valid. Can have the values `USER`, `SHELL`, `SESSION`, `CONTROL_TTY` or `TTY`. If the scope is `USER`, a timestamp is valid for any instance of `sus` where the source user is the same as the source user who created the timestamp. For `SHELL`, the timestamp is only valid if the parent process id of the instance of `sus` matches the one which created the timestamp. Generally, this means the timestamp is valid in the current shell. For `SESSION`, the timestamp is only valid if the session id (as returned by `getsid(2)`) of the current instance of `sus` matches the session id of the instance which created the timestamp. For the value `CONTROL_TTY`, raw device id of the controlling tty must be the same, for the value `TTY` the raw device id of `stdin` (if a tty type device) must be the same. Default `USER`.

ENV_PRESERVE

(string list) Comma separated list of regular expressions. Environment variables matching any of the regular expressions are preserved for the invocation of the target command. This list is used first. The default is `TERM`, `TZ`, `TIMEZONE`, `HZ`, `DISPLAY`, `EDITOR`, `VISUAL`, `TAPE`, `XAUTHORITY`, `LPDEST`, `PRINTER`, `MANPATH`, `LANG`, `LC_TYPE`, `LC_NUMERIC`, `LC_TIME`, `LC_COLLATE`, `LC_MONETARY`, `LC_MESSAGES`, `LC_ALL`, `LM_LICENSE_FILE` and `TERMINFO`.

ENV_DELETE

(string list) Comma separated list of regular expressions. Environment variables matching any of the regular expressions are deleted from the environment before the invocation of the target command. This list is used second. The default is everything.

ENV_ADD

(string list) Comma separated list of environment

strings to add to the environment before the invocation of the target command. This list is used last. Default is to set `PS1='$ '`, `PS2='# '` and `SHELL=/bin/false`.

IN_DIRECTORY

(string) The directory in which the command must be executed. Default NULL.

SKIP_VERIFICATION

(boolean) If true, verification of the user is turned off and the timestamp are not examined or updated. Note that all other checks are still performed. Default false.

NO_BACKGROUNDING

(boolean) If true, commands may not be placed in the background via the `-b` option. Default false.

ALLOW_LOGIN_SHELL

(boolean) If true, user is allowed to use the option to invoke a login shell as the target user. All other checks apply as normal. Default false.

ALLOWED_REAL_USERS

(string list) If a target user is specified via the `-u` option (see **OPTIONS**) and promiscuous mode is not specified, then the contents of this macro are used to match against the target user to determine if the command is allowed. If this macro is not defined, the requested command is not permitted. The format of the macro contents are the same as `user_list` (see **CONFIGURATION**). Default NULL.

ALLOWED_EFFECTIVE_USERS

(string list) As for **ALLOWED_REAL_USERS** above, except the contents of the macro are used if the user specified a target effective user via the `-U` option (see **OPTIONS**). Default NULL.

ALLOWED_REAL_GROUPS

(string list) As for **ALLOWED_REAL_USERS** above, except the contents of the macro are used if the user specified a target real group via the `-g` option (see **OPTIONS**). The format of the macro contents are the same as for `group_list` (see **CONFIGURATION**). Default NULL.

ALLOWED_EFFECTIVE_GROUPS

(string list) As for **ALLOWED_REAL_GROUPS** above, except the contents of the macro are used if the user specified a target effective group via the `-G` option (see **OPTIONS**). Default NULL.

BLOCK_CHILD_FORK

(boolean) If defined and true, any child process will be blocked from executing the fork(2) system call. See MAX_CHILD_FORKS. Solaris only. Default false.

BLOCK_CHILD_EXEC

(boolean) If defined and true, any child process will be blocked from executing the exec(2) family of system calls. See MAX_CHILD_EXECS. Solaris only. Default false.

MAX_CHILD_FORKS

(integer) If set and BLOCK_CHILD_FORKS is true, this defines an upper bound on the number of forks(2) calls a child may perform before being killed. Default 0.

MAX_CHILD_EXECS

(integer) If set and BLOCK_CHILD_EXECS is true, this defines an upper bound on the number of exec(2) type system calls a child may perform before being killed. Must be at least 1. NOTE: use of the feature enforces the command name being an absolute path. Default 1.

RLIMITS

RLIMIT_CPU RLIMIT_FSIZE RLIMIT_DATA RLIMIT_STACK
RLIMIT_VMEM RLIMIT_CORE RLIMIT_RSS RLIMIT_NPROC
RLIMIT_MEMLOCK RLIMIT_AS RLIMIT_NOFILE (integer except
RLIMIT_CPU which is time) If set, set the corresponding resource limit to this value before running the target command. See rlimit(2). Not all resource limits are supported on all platforms. Default RLIM_INFINITY except RLIMIT_STACK which is 8 megabytes.

USE_PAM

(boolean) If defined and true and PAM support is compiled in, use the PAM routines for user authentication. If false, authenticate via a standard password/shadow file. Default true.

USE_SHADOW

(boolean) If defined and true and shadow password file support is compiled in, expect to find hashed passwords information in the shadow password file. Default true.

PASSWD_FILE

(string) The name of the password file which holds password records for authenticating the user. This is only used if PAM is not being used. The default is to use the systems getpwXXX routines to access password records. This file must look like a standard UNIX

password file but does not have to the system password file. Default NULL.

SHADOW_FILE

(string) The name of the password file used to hold shadow password user records if shadow password files are enabled and being used. This is only used if PAM is not being used. The default is to use the systems `getspXXX` routines to access user shadow information. This file must look like a standard UNIX shadow file but does not have to be the system shadow file. Default NULL.

CLOSE_FILES_BEFORE_EXEC

(boolean) If true, close all files except `stdin`, `stdout` and `stderr` before performing the `exec(2)` of the target command. Default true.

NO_X11

Disable the `-X` command line option. Default false.

The following macros are predefined by *sus*. They can be overridden on the control file if you want.

PROMISCUOUS

Defined to be true if the user is using promiscuous mode. Can be used to select or ignore sections of the configuration file.

ANY_USER

A class expression to match any user in the password database.

NO_USER

The reverse of **ANY_USER**.

ANY_GROUP

A class expression which will match any group in the group database.

NO_GROUP

The reverse of **ANY_GROUP**.

ANY_HOST

A class expression to match any host which exists according to `gethostbyXXX(3)` routines.

NO_HOST

The reverse of **ANY_HOST**.

ANY_FILE

A class expression which will match any existing file.

NO_FILE

The reverse of ANY_FILE.

ANY_PROCESS

A class expression which will match any existing process.

NO_PROCESS

The reverse of ANY_PROCESS.

ANY_COMMAND

A class expression to match any command.

NO_COMMAND

The reverse of ANY_COMMAND.

ANY_ARGUMENT

A class expression to match any single argument.

NO_ARGUMENTS

A special string which does not match any so far unmatched arguments on the target command.

ANY_ARGUMENTS

A special string which matches any so far unmatched arguments on the target command.

SOURCE_USERNAME

The user name of the invoking user.

SOURCE_USERID

The user id of the invoking user.

SOURCE_HOME

The home directory of the invoking user.

SOURCE_PGID

The process group id of the invoking process.

SOURCE_SID

The session id of the invoking process.

TARGET_USERNAME

The username of the target user.

TARGET_USERID

The userid of the target user.

TARGET_HOME

The home directory of the target user.

HOURL The current hour (0-23).

MINUTE

The current minute (0-59).

DAYNAME

The day of the week (locales full weekday name from strftime(3)).

DAY The day of the week as an integer (Sunday = 1)

HOSTNAME

As returned by uname(2).

SYSNAME

As returned by uname(2).

RELEASE

As returned by uname(2).

VERSION

As returned by uname(2).

MACHINE

As returned by uname(2).

SRPC_DOMAIN

As returned by sysinfo(2). Solaris only.

CWD The current directory as returned by getcwd(3).

HOSTNAMES

A regular expression which contains the names and aliases and IP addresses obtained from the system via getipnodebyXXX(3) routines for the target host. If no target host is specified via the -p option, the name returned by uname(2) is used to perform the lookups. NOTE: on linux, gethostbyXXX may be used instead of getipnodesbyXXX.

DEFAULT_ENV_PRESERVE

The default value of ENV_PRESERVE. See above.

DEFAULT_ENV_DELETE

The default value of ENV_DELETE. See above.

DEFAULT_ENV_ADD

The default value of ENV_ADD. See above.

RETURN VALUES If the target program is successfully executed and the macro `WAIT_FOR_CHILD` is true, *sus* returns the exit value of the child program unless a timeout occurs. If `WAIT_FOR_CHILD` is false, *sus* calls `exec(2)` directly, so the exit value is the exit value of the invoked command.

In all other cases, *sus* returns a non-zero value and prints a message to indicate the error.

SECURITY

sus makes no attempt to avoid bugs in any of the underlying OS utilities, such as library calls or system calls. If your platform has problems with any of the libraries *sus* uses, you may have security issues.

Also, and perhaps more importantly, *sus* does not attempt to avoid race conditions. For example, if you allow a user to use the `chown` command on files in a directory in which they have write permissions, then they can probably easily create a symlink to a file outside that directory and fool *sus* into thinking the target of the `chown` is ok even if you use the `realpath` option to try to prevent this from happening. Most if not all of these problems can be overcome by simply thinking carefully about the construction of the control file. For example, in the above example, insisting the `-h` option is present on the call to `chown` solves the problem.

sus will refuse to operate unless `PATH` is defined in the control file but it does no checks on the sanity of the `PATH` you supply. Set it carefully.

Many programs, including most editors, allow the user to create subshells. *sus* sets the `SHELL` environment variable to `/bin/false` prior to executing the command as a small attempt to stop this behaviour but in general this does not solve the problem. Under `SOLARIS`, *sus* can control calls to `fork/exec` system calls to overcome this problem.

In general, *sus* is only as the control file. Make sure you understand exactly what privileges you are bestowing when you create or edit the file.

PAM

Sus can be built to use `PAM` to authenticate the calling user. However, `PAM` is used to authenticate the password only. All other aspects of the user's account which `PAM` may or may not be capable of handling are ignored. For example, restrictions on when a user can log on which are enforced in

PAM will not be honoured by *sus*. I think this is the correct behaviour. PAM is used by *sus* for authentication only, not for policing accounts.

EXAMPLE CONFIGURATION

See the sample configuration file distributed with the source code.

SEE ALSO

suscpp(1), *regex*(5)

DISCLAIMER *sus* 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 *sus* for details.

AUTHOR

Peter Gray
University of Wollongong
pdg@uow.edu.au