NAME

limits - set or display process resource limits

SYNOPSIS

limits [**-C** *class* | **-P** *pid* | **-U** *user*] [**-SHB**] [**-ea**] [**-bcdfklmnopstuvw** [*val*]]
limits [**-C** *class* | **-U** *user*] [**-SHB**] [**-bcdfklmnopstuvw** [*val*]] [**-E**] [[*name=value* ...] *command*]

DESCRIPTION

The **limits** utility either prints or sets kernel resource limits, and may optionally set environment variables like env(1) and run a program with the selected resources. Three uses of the **limits** utility are possible:

limits [*limitflags*] [*name=value* ...] *command*

This usage sets limits according to *limitflags*, optionally sets environment variables given as *name=value* pairs, and then runs the specified *command*.

limits [*limitflags*]

This usage determines values of resource settings according to *limitflags*, does not attempt to set them and outputs these values to standard output. By default, this will output the current kernel resource settings active for the calling process. Using the **-C** *class* or **-U** *user* options, you may also display the current resource settings modified by the appropriate login class resource limit entries from the login.conf(5) login capabilities database.

limits -e [*limitflags*]

This usage determines values of resource settings according to *limitflags*, but does not set them. Like the previous usage, it outputs these values to standard output, except that it will emit them in **eval** format, suitable for the calling shell. If the shell is known (i.e., it is one of **sh**, **csh**, **bash**, **tcsh**, **ksh**, **pdksh** or **rc**), **limits** emits **limit** or **ulimit** commands in the format understood by that shell. If the name of the shell cannot be determined, then the **ulimit** format used by sh(1) is used.

This is very useful for setting limits used by scripts, or prior launching of daemons and other background tasks with specific resource limit settings, and provides the benefit of allowing global configuration of maximum resource usage by maintaining a central database of settings in the login class database.

Within a shell script, **limits** will normally be used with eval within backticks as follows:

```
eval ‘limits -e -C daemon‘
```

which causes the output of **limits** to be evaluated and set by the current shell.

The value of *limitflags* specified in the above contains one or more of the following options:

-C *class* Use current resource values, modified by the resource entries applicable for the login class *class*.

-U *user* Use current resource values, modified by the resource entries applicable to the login class the *user* belongs to. If user does not belong to any class, then the resource capabilities for the "default" class are used, if it exists, or the "root" class if the user is a superuser account.

-P *pid* Select or set limits for the process identified by the *pid*.

-S Select display or setting of "soft" (or current) resource limits. If specific limits settings follow this switch, only soft limits are affected unless overridden later with either the **-H** or **-B** options.

-H Select display or setting of "hard" (or maximum) resource limits. If specific limits settings follow this switch, only hard limits are affected until overridden later with either the **-S** or **-B** options.

-B Select display or setting of both "soft" (current) or "hard" (maximum) resource limits. If specific limits settings follow this switch, both soft and hard limits are affected until overridden later with either the **-S** or **-H** options.

-e Select "eval mode" formatting for output. This is valid only in display mode and cannot be used when running a command. The exact syntax used for output depends upon the type of shell from which **limits** is invoked.

-b [*val*] Select or set the *sbsize* resource limit.

-c [*val*] Select or set (if *val* is specified) the *coredumpsize* resource limit. A value of 0 disables core dumps.

-d [*val*] Select or set (if *val* is specified) the *datasize* resource limit.

-f [*val*] Select or set the *filesize* resource limit.

-k [*val*] Select or set the *kqueues* resource limit.

-l [*val*] Select or set the *memorylocked* resource limit.

-m [*val*] Select or set the *memoryuse* size limit.

-n [*val*] Select or set the *openfiles* resource limit. The system-wide limit on the maximum number of open files per process can be viewed by examining the *kern.maxfilesperproc* sysctl(8) variable. The total number of simultaneously open files in the entire system is limited to the value displayed by the *kern.maxfiles* sysctl(8) variable.

-o [*val*] Select or set the *umtxp* resource limit. The limit determines the maximal number of the process-shared locks which may be simultaneously created by the processes owned by the user, see pthread(3).

-p [*val*] Select or set the *pseudoterminals* resource limit.

-s [*val*] Select or set the *stacksize* resource limit.

-t [*val*] Select or set the *cputime* resource limit.

-u [*val*] Select or set the *maxproc* resource limit. The system-wide limit on the maximum number of processes allowed per UID can be viewed by examining the *kern.maxprocperuid* sysctl(8) variable. The maximum number of processes that can be running simultaneously in the entire system is limited to the value of the *kern.maxproc* sysctl(8) variable.

-v [*val*] Select or set the *virtualmem* resource limit. This limit encompasses the entire VM space for the user process and is inclusive of text, data, bss, stack, brk(2), sbrk(2) and mmap(2)’d space.

-w [*val*] Select or set the *swapuse* resource limit.

Valid values for *val* in the above set of options consist of either the string "infinity", "inf", "unlimited" or "unlimit" for an infinite (or kernel-defined maximum) limit, or a numeric value optionally followed by a suffix. Values which relate to size default to a value in bytes, or one of the following suffixes may be used as a multiplier:

b 512 byte blocks.
k kilobytes (1024 bytes).
m megabytes (1024*1024 bytes).
g gigabytes.
t terabytes.

The *cputime* resource defaults to a number of seconds, but a multiplier may be used, and as with size

values, multiple values separated by a valid suffix are added together:

s	seconds.
m	minutes.
h	hours.
d	days.
w	weeks.
y	365 day years.

-E Cause **limits** to completely ignore the environment it inherits.

-a Force all resource settings to be displayed even if other specific resource settings have been specified. For example, if you wish to disable core dumps when starting up the Usenet News system, but wish to set all other resource settings as well that apply to the "news" account, you might use:

```
eval ‘limits -U news -aBec 0‘
```

As with the setrlimit(2) call, only the superuser may raise process "hard" resource limits. Non-root users may, however, lower them or change "soft" resource limits within to any value below the hard limit. When invoked to execute a program, the failure of **limits** to raise a hard limit is considered a fatal error.

EXIT STATUS

The **limits** utility exits with EXIT_FAILURE if usage is incorrect in any way; i.e., an invalid option, or set/display options are selected in the same invocation, **-e** is used when running a program, etc. When run in display or eval mode, **limits** exits with a status of EXIT_SUCCESS. When run in command mode and execution of the command succeeds, the exit status will be whatever the executed program returns.

EXAMPLES

Show current stack size limit:

```
$ limits -s
Resource limits (current):
            stacksize          524288 kB
```

Try to run ls(1) with 1 byte of *datasize* limit:

```
$ limits -d 1b ls
Data segment size exceeds process limit
```

Abort trap

Produce ‘eval mode’ output to limit *sbsize* to 1 byte. Output obtained when command is run from sh(1):

```
$ limits -e -b 1b
ulimit -b 512;
```

Same as above from csh(1)

```
% limits -e -b 1b
limit -h sbsize 512;
limit sbsize 512;
```

SEE ALSO

csh(1), env(1), limit(1), sh(1), getrlimit(2), setrlimit(2), login_cap(3), login.conf(5), rctl(8), sysctl(8)

HISTORY

The **limits** utility first appeared in FreeBSD 2.1.7.

AUTHORS

The **limits** utility was written by David Nugent <*davidn@FreeBSD.org*>.

BUGS

The **limits** utility does not handle commands with equal (‘=’) signs in their names, for obvious reasons.

The **limits** utility makes no effort to ensure that resource settings emitted or displayed are valid and settable by the current user. Only a superuser account may raise hard limits, and when doing so the FreeBSD kernel will silently lower limits to values less than specified if the values given are too high.