Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'getrlimit.3p' command

$ man getrlimit.3p

GETRLIMIT(3P) POSIX Programmer's Manual GETRLIMIT(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

getrlimit, setrlimit ? control maximum resource consumption

SYNOPSIS

#include <sys/resource.h>

int getrlimit(int resource, struct rlimit *rlp);

int setrlimit(int resource, const struct rlimit *rlp);

DESCRIPTION

The getrlimit() function shall get, and the setrlimit() function shall set, limits on the consumption of a variety of resources.

Each call to either getrlimit() or setrlimit() identifies a specific resource to be operated upon as well as a resource limit. A resource limit is represented by an rlimit structure. The rlim_cur member speci? fies the current or soft limit and the rlim_max member specifies the maximum or hard limit. Soft limits may be changed by a process to any value that is less than or equal to the hard limit. A process may (ir? reversibly) lower its hard limit to any value that is greater than or equal to the soft limit. Only a process with appropriate privileges can

raise a hard limit. Both hard and soft limits can be changed in a sin?
gle call to setrlimit() subject to the constraints described above.
The value RLIM_INFINITY, defined in <sys/resource.h>, shall be consid?
ered to be larger than any other limit value. If a call to getrlimit()
returns RLIM_INFINITY for a resource, it means the implementation shall
not enforce limits on that resource. Specifying RLIM_INFINITY as any
resource limit value on a successful call to setrlimit() shall inhibit
enforcement of that resource limit.
The following resources are defined:
RLIMIT_CORE This is the maximum size of a core file, in bytes, that
may be created by a process. A limit of 0 shall prevent
the creation of a core file. If this limit is exceeded,
the writing of a core file shall terminate at this size.
RLIMIT_CPU This is the maximum amount of CPU time, in seconds, used
by a process. If this limit is exceeded, SIGXCPU shall
be generated for the process. If the process is catching
or ignoring SIGXCPU, or all threads belonging to that
process are blocking SIGXCPU, the behavior is unspeci?
fied.
RLIMIT_DATA This is the maximum size of a data segment of the
process, in bytes. If this limit is exceeded, the mal?
loc() function shall fail with errno set to [ENOMEM].
RLIMIT_FSIZE This is the maximum size of a file, in bytes, that may be
created by a process. If a write or truncate operation
would cause this limit to be exceeded, SIGXFSZ shall be
generated for the thread. If the thread is blocking, or
the process is catching or ignoring SIGXFSZ, continued
attempts to increase the size of a file from end-of-file
to beyond the limit shall fail with errno set to [EFBIG].
RLIMIT_NOFILE This is a number one greater than the maximum value that
the system may assign to a newly-created descriptor. If
this limit is exceeded, functions that allocate a file
descriptor shall fail with errno set to [EMFILE]. This

limit constrains the number of file descriptors that a process may allocate.

RLIMIT_STACK This is the maximum size of the initial thread's stack, in bytes. The implementation does not automatically grow the stack beyond this limit. If this limit is exceeded, SIGSEGV shall be generated for the thread. If the thread is blocking SIGSEGV, or the process is ignoring or catch? ing SIGSEGV and has not made arrangements to use an al? ternate stack, the disposition of SIGSEGV shall be set to SIG_DFL before it is generated.

RLIMIT_AS This is the maximum size of total available memory of the process, in bytes. If this limit is exceeded, the mal? loc() and mmap() functions shall fail with errno set to [ENOMEM]. In addition, the automatic stack growth fails with the effects outlined above.

When using the getrlimit() function, if a resource limit can be repre? sented correctly in an object of type rlim_t, then its representation is returned; otherwise, if the value of the resource limit is equal to that of the corresponding saved hard limit, the value returned shall be RLIM_SAVED_MAX; otherwise, the value returned shall be RLIM_SAVED_CUR.

When using the setrlimit() function, if the requested new limit is RLIM_INFINITY, the new limit shall be ``no limit''; otherwise, if the requested new limit is RLIM_SAVED_MAX, the new limit shall be the cor? responding saved hard limit; otherwise, if the requested new limit is RLIM_SAVED_CUR, the new limit shall be the corresponding saved soft limit; otherwise, the new limit shall be the requested value. In addi? tion, if the corresponding saved limit can be represented correctly in an object of type rlim_t then it shall be overwritten with the new limit.

The result of setting a limit to RLIM_SAVED_MAX or RLIM_SAVED_CUR is unspecified unless a previous call to getrlimit() returned that value as the soft or hard limit for the corresponding resource limit.

The determination of whether a limit can be correctly represented in an

object of type rlim_t is implementation-defined. For example, some im?
plementations permit a limit whose value is greater than RLIM_INFINITY
and others do not.

The exec family of functions shall cause resource limits to be saved.

RETURN VALUE

Upon successful completion, getrlimit() and setrlimit() shall return 0.
Otherwise, these functions shall return -1 and set errno to indicate
the error.

ERRORS

The getrlimit() and setrlimit() functions shall fail if:

EINVAL An invalid resource was specified; or in a setrlimit() call, the
new rlim_cur exceeds the new rlim_max.

EPERM The limit specified to setrlimit() would have raised the maximum
limit value, and the calling process does not have appropriate
privileges.

The setrlimit() function may fail if:

EINVAL The limit specified cannot be lowered because current usage is
already higher than the limit.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

If a process attempts to set the hard limit or soft limit for
RLIMIT_NOFILE to less than the value of {_POSIX_OPEN_MAX} from <lim?
its.h>, unexpected behavior may occur.

If a process attempts to set the hard limit or soft limit for
RLIMIT_NOFILE to less than the highest currently open file descriptor
+1, unexpected behavior may occur.

RATIONALE

It should be noted that RLIMIT_STACK applies ``at least'' to the stack
of the initial thread in the process, and not to the sum of all the
stacks in the process, as that would be very limiting unless the value
is so big as to provide no value at all with a single thread.

FUTURE DIRECTIONS

None.

SEE ALSO

exec, fork(), malloc(), open(), sigaltstack(), sysconf(), ulimit()

The Base Definitions volume of POSIX.1?2017, <stropts.h>, <sys_re?

source.h>