NAME

getrusage − get resource usage

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/resource.h>

int getrusage(int *who***, struct rusage *****usage***);**

DESCRIPTION

getrusage() returns resource usage measures for *who*, which can be one of the following:

RUSAGE_SELF
: Return resource usage statistics for the calling process, which is the sum of resources used by all threads in the process.

RUSAGE_CHILDREN
: Return resource usage statistics for all children of the calling process that have terminated and been waited for. These statistics will include the resources used by grandchildren, and further removed descendants, if all of the intervening descendants waited on their terminated children.

RUSAGE_THREAD (since Linux 2.6.26)
: Return resource usage statistics for the calling thread. The **_GNU_SOURCE** feature test macro must be defined (before including *any* header file) in order to obtain the definition of this constant from *<sys/resource.h>*.

The resource usages are returned in the structure pointed to by *usage*, which has the following form:

```
struct rusage {
    struct timeval ru_utime; /* user CPU time used */
    struct timeval ru_stime; /* system CPU time used */
    long   ru_maxrss;        /* maximum resident set size */
    long   ru_ixrss;         /* integral shared memory size */
    long   ru_idrss;         /* integral unshared data size */
    long   ru_isrss;         /* integral unshared stack size */
    long   ru_minflt;        /* page reclaims (soft page faults) */
    long   ru_majflt;        /* page faults (hard page faults) */
    long   ru_nswap;         /* swaps */
    long   ru_inblock;       /* block input operations */
    long   ru_oublock;       /* block output operations */
    long   ru_msgsnd;        /* IPC messages sent */
    long   ru_msgrcv;        /* IPC messages received */
    long   ru_nsignals;      /* signals received */
    long   ru_nvcsw;         /* voluntary context switches */
    long   ru_nivcsw;        /* involuntary context switches */
};
```

Not all fields are completed; unmaintained fields are set to zero by the kernel. (The unmaintained fields are provided for compatibility with other systems, and because they may one day be supported on Linux.) The fields are interpreted as follows:

ru_utime
: This is the total amount of time spent executing in user mode, expressed in a *timeval* structure (seconds plus microseconds).

ru_stime
: This is the total amount of time spent executing in kernel mode, expressed in a *timeval* structure (seconds plus microseconds).

ru_maxrss (since Linux 2.6.32)
This is the maximum resident set size used (in kilobytes). For **RUSAGE_CHILDREN**, this is the resident set size of the largest child, not the maximum resident set size of the process tree.

ru_ixrss (unmaintained)
This field is currently unused on Linux.

ru_idrss (unmaintained)
This field is currently unused on Linux.

ru_isrss (unmaintained)
This field is currently unused on Linux.

ru_minflt
The number of page faults serviced without any I/O activity; here I/O activity is avoided by "reclaiming" a page frame from the list of pages awaiting reallocation.

ru_majflt
The number of page faults serviced that required I/O activity.

ru_nswap (unmaintained)
This field is currently unused on Linux.

ru_inblock (since Linux 2.6.22)
The number of times the filesystem had to perform input.

ru_oublock (since Linux 2.6.22)
The number of times the filesystem had to perform output.

ru_msgsnd (unmaintained)
This field is currently unused on Linux.

ru_msgrcv (unmaintained)
This field is currently unused on Linux.

ru_nsignals (unmaintained)
This field is currently unused on Linux.

ru_nvcsw (since Linux 2.6)
The number of times a context switch resulted due to a process voluntarily giving up the processor before its time slice was completed (usually to await availability of a resource).

ru_nivcsw (since Linux 2.6)
The number of times a context switch resulted due to a higher priority process becoming runnable or because the current process exceeded its time slice.

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EFAULT
usage points outside the accessible address space.

EINVAL
who is invalid.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
getrusage()	Thread safety	MT-Safe

STANDARDS

POSIX.1-2001, POSIX.1-2008, SVr4, 4.3BSD. POSIX.1 specifies **getrusage**(), but specifies only the fields *ru_utime* and *ru_stime*.

RUSAGE_THREAD is Linux-specific.

NOTES

Resource usage metrics are preserved across an **execve**(2).

Before Linux 2.6.9, if the disposition of **SIGCHLD** is set to **SIG_IGN** then the resource usages of child processes are automatically included in the value returned by **RUSAGE_CHILDREN**, although POSIX.1-2001 explicitly prohibits this. This nonconformance is rectified in Linux 2.6.9 and later.

The structure definition shown at the start of this page was taken from 4.3BSD Reno.

Ancient systems provided a **vtimes**() function with a similar purpose to **getrusage**(). For backward compatibility, glibc (up until Linux 2.32) also provides **vtimes**(). All new applications should be written using **getrusage**(). (Since Linux 2.33, glibc no longer provides an **vtimes**() implementation.)

See also the description of */proc/*pid*/stat* in **proc**(5).

SEE ALSO

clock_gettime(2), **getrlimit**(2), **times**(2), **wait**(2), **wait4**(2), **clock**(3)