NAME

wait3, wait4 – wait for process to terminate or stop

SYNOPSIS

```
#include <sys/wait.h>
#include <sys/time.h>
#include <sys/resource.h>

pid_t wait3(int *statusp, int options,
        struct rusage *rusage);

pid_t wait4(pid_t pid, int *statusp, int options,
        struct rusage *rusage);
```

DESCRIPTION

wait3( ) delays its caller until a signal is received or one of its child processes terminates or stops due to tracing. If any child process has died or stopped due to tracing and this has not already been reported, return is immediate, returning the process ID and status of one of those children. If that child process has died, it is discarded. If there are no children, –1 is returned immediately. If there are only running or stopped but reported children, the calling process is blocked.

If *statusp* is not a **NULL** pointer, then on return from a successful **wait3( )** call, the status of the child process is stored in the integer pointed to by *statusp*. **statusp* indicates the cause of termination and other information about the terminated process in the following manner:

- If the low-order 8 bits of ***statusp** are equal to 0177, the child process has stopped; the 8 bits higher up from the low-order 8 bits of ***statusp** contain the number of the signal that caused the process to stop. See **signal**(5).
- If the low-order 8 bits of ***statusp** are non-zero and are not equal to 0177, the child process terminated due to a signal; the low-order 7 bits of ***statusp** contain the number of the signal that terminated the process. In addition, if the low-order seventh bit of ***statusp** (that is, bit 0200) is set, a ''core image'' of the process was produced; see **signal**(5).
- Otherwise, the child process terminated due to an **exit( )** call; the 8 bits higher up from the low-order 8 bits of ***statusp** contain the low-order 8 bits of the argument that the child process passed to **exit( )**; see **exit**(2).

options is constructed from the bitwise inclusive OR of zero or more of the following flags, defined in the header **<sys/wait.h>**:

WNOHANG
: Execution of the calling process is not suspended if status is not immediately available for any child process.

WUNTRACED
: The status of any child processes that are stopped, and whose status has not yet been reported since they stopped, are also reported to the requesting process.

If *rusage* is not a **NULL** pointer, a summary of the resources used by the terminated process and all its children is returned. Only the user time used and the system time used are currently available. They are returned in the **ru_utime** and **ru_stime**, members of the rusage structure respectively.

When the **WNOHANG** option is specified and no processes have status to report, **wait3( )** returns 0. The **WNOHANG** and **WUNTRACED** options may be combined by ORing the two values.

wait4( ) is an extended interface. With a *pid* argument of 0, it is equivalent to **wait3( )**. If *pid* has a nonzero value, then **wait4( )** returns status only for the indicated process ID, but not for any other child processes. The status can be evaluated using the macros defined by **wstat**(5).

RETURN VALUES

If **wait3( )** or **wait4( )** returns due to a stopped or terminated child process, the process ID of the child is returned to the calling process. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

If **wait3( )** or **wait4( )** return due to the delivery of a signal to the calling process, a value of –1 is returned and **errno** is set to EINTR. If **WNOHANG** was set in *options*, it has at least one child process specified by *pid* for which status is not available, and status is not available for any process specified by *pid*, a value of zero is returned. Otherwise, a value of –1 is returned, and **errno** is set to indicate the error.

wait3( ) and **wait4( )** return 0 if **WNOHANG** is specified and there are no stopped or exited children, and return the process ID of the child process if they return due to a stopped or terminated child process. Otherwise, they return a value of –1 and sets **errno** to indicate the error.

ERRORS

wait3( ) or **wait4( )** will fail and return immediately if one or more of the following are true:

ECHILD	The calling process has no existing unwaited-for child processes.
EFAULT	The *statusp* or *rusage* arguments point to an illegal address.
EINTR	The function was interrupted by a signal. The value of the location pointed to by *statusp* is undefined.
EINVAL	The value of *options* is not valid.

wait4( ) may set **errno** to:

ECHILD	The process specified by *pid* does not exist or is not a child of the calling process.

wait3( ), and **wait4( )** will terminate prematurely, return –1, and set **errno** to EINTR upon the arrival of a signal whose **SA_RESTART** bit in its flags field is not set (see **sigaction**(2)).

SEE ALSO

kill(1), **exit**(2), **wait**(2), **waitid**(2), **waitpid**(2), **getrusage**(3C), **signal**(3C), **proc**(4), **signal**(5), **wstat**(5)

NOTES

If a parent process terminates without waiting on its children, the initialization process (process ID = 1) inherits the children.

wait3( ), and **wait4( )** are automatically restarted when a process receives a signal while awaiting termination of a child process, unless the **SA_RESTART** bit is not set in the flags for that signal.