NAME

wait, waitpid − wait for process to terminate

SYNOPSIS

#include <sys/types.h>
#include <sys/wait.h>

pid_t wait(int ****status*)**
pid_t waitpid(pid_t *pid***, int ****status***, int** *options***)**

DESCRIPTION

Wait causes its caller to delay until a signal is received or one of its child processes terminates. If any child has died since the last **wait**, return is immediate, returning the process id and exit status of one of the terminated children. If there are no children, return is immediate with the value −1 returned.

On return from a successful **wait** call, *status* is nonzero, and the high byte of *status* contains the low byte of the argument to **exit** supplied by the child process; the low byte of *status* contains the termination status of the process. A more precise definition of the *status* word is given in *<sys/wait.h>*. If **wait** can called with a null pointer argument to indicate that no status need be returned.

Waitpid provides an alternate interface for programs that must not block when collecting the status of child processes, or that wish to wait for one particular child. The pid parameter is the process ID of the child to wait for, −1 for any child. The *status* parameter is defined as above. The *options* parameter is used to indicate the call should not block if there are no processes that wish to report status (WNOHANG), and/or that children of the current process that are stopped due to a SIGTTIN, SIGTTOU, SIGTSTP, or SIGSTOP signal should also have their status reported (WUNTRACED). (Job control is not implemented for Minix, but these symbold and signals are.)

When the WNOHANG option is specified and no processes wish to report status, **waitpid** either returns 0 under some implementations, or −1 with **errno** set to **EAGAIN** under others. (Under Minix it returns 0.) The WNOHANG and WUNTRACED options may be combined by *or*'ing the two values.

NOTES

The call **wait(&***status***)** is equivalent to **waitpid(−1, &***status***, 0)**.

See **sigaction**(2) for a list of termination statuses (signals); 0 status indicates normal termination. A special status (0177) is returned for a stopped process that has not terminated and can be restarted; see **ptrace**(2). If the 0200 bit of the termination status is set, a core image of the process was produced by the system.

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

<sys/wait.h> defines a number of macros that operate on a status word:

WIFEXITED(*status***)**
True if normal exit.

WEXITSTATUS(*status***)**
Exit status if the process returned by a normal exit, zero otherwise.

WTERMSIG(*status***)**
Signal number if the process died by a signal, zero otherwise.

WIFSIGNALED(*status***)**
True if the process died by a signal.

WIFSTOPPED(*status***)**
True if the process is stopped. (Never true under Minix.)

WSTOPSIG(*status***)**
Signal number of the signal that stopped the process.

RETURN VALUE

If **wait** 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.

Waitpid returns −1 if there are no children not previously waited for, if the process that it wants to wait for doesn't exist, or if WNOHANG is specified and there are no stopped or exited children.

ERRORS

Wait 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 *status* argument points to an illegal address.

[EAGAIN] **Waitpid** is called with the WNOHANG option and no child has exited yet.

SEE ALSO

execve(2), **exit**(2), **sigaction**(2).