

**NAME**

wait – wait for process to terminate

**SYNOPSIS**

(wait = 7.)

**sys wait**

(process ID in r0)

(status in r1)

**wait(status)**

**int \*status;**

**DESCRIPTION**

*Wait* causes its caller to delay until one of its child processes terminates. If any child has died since the last *wait*, return is immediate; if there are no children, return is immediate with the error bit set (resp. with a value of -1 returned). The normal return yields the process ID of the terminated child (in r0). In the case of several children several *wait* calls are needed to learn of all the deaths.

If no error is indicated on return, the r1 high byte (resp. the high byte stored into *status* ) contains the low byte of the child process r0 (resp. the argument of *exit* ) when it terminated. The r1 (resp. *status* ) low byte contains the termination status of the process. See signal (II) for a list of termination statuses (signals); 0 status indicates normal termination. A special status (0177) is returned for a stopped process which has not terminated and can be restarted. See ptrace (II). 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.

**SEE ALSO**

exit (II), fork (II), signal (II)

**DIAGNOSTICS**

The error bit (c-bit) is set if there are no children not previously waited for. From C, a returned value of -1 indicates an error.