NAME

wait – check child process status

SYNTAX

#include <wait.h>

int **wait_nohang**(&*wstat*);
int **wait_stop**(&*wstat*);
int **wait_stopnohang**(&*wstat*);
int **wait_pid**(&*wstat*,*pid*);

int **wait_exitcode**(*wstat*);
int **wait_crashed**(*wstat*);
int **wait_stopped**(*wstat*);
int **wait_stopsig**(*wstat*);

int *pid*;
int *wstat*;

DESCRIPTION

wait_nohang looks for zombies (child processes that have exited). If it sees a zombie, it eliminates the zombie, puts the zombie's exit status into *wstat*, and returns the zombie's process ID. If there are several zombies, **wait_nohang** picks one. If there are children but no zombies, **wait_nohang** returns 0. If there are no children, **wait_nohang** returns -1, setting **errno** appropriately.

wait_stopnohang is similar to **wait_nohang**, but it also looks for children that have stopped.

wait_stop is similar to **wait_stopnohang**, but if there are children it will pause waiting for one of them to stop or exit.

wait_pid waits for child process *pid* to exit. It eliminates any zombie that shows up in the meantime, discarding the exit status.

wait_stop and **wait_pid** retry upon **error_intr**.

STATUS PARSING

If the child stopped, **wait_stopped** is nonzero; **wait_stopsig** is the signal that caused the child to stop.

If the child exited by crashing, **wait_stopped** is zero; **wait_crashed** is nonzero.

If the child exited normally, **wait_stopped** is zero; **wait_crashed** is zero; and **wait_exitcode** is the child's exit code.

SEE ALSO

wait(2), error(3)