NAME

_exit - terminate the calling process

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

void
_exit(*int status*);

DESCRIPTION

The **_exit**() system call terminates a process with the following consequences:

- All of the descriptors open in the calling process are closed. This may entail delays, for example, waiting for output to drain; a process in this state may not be killed, as it is already dying.

- If the parent process of the calling process has an outstanding wait(2) call or catches the SIGCHLD signal, it is notified of the calling process's termination and the *status* is set as defined by wait(2).

- The parent process-ID of all of the calling process's existing child processes are set to the process-ID of the calling process's reaper; the reaper (normally the initialization process) inherits each of these processes (see procctl(2), init(8) and the *DEFINITIONS* section of intro(2)).

- If the termination of the process causes any process group to become orphaned (usually because the parents of all members of the group have now exited; see "orphaned process group" in intro(2)), and if any member of the orphaned group is stopped, the SIGHUP signal and the SIGCONT signal are sent to all members of the newly-orphaned process group.

- If the process is a controlling process (see intro(2)), the SIGHUP signal is sent to the foreground process group of the controlling terminal, and all current access to the controlling terminal is revoked.

Most C programs call the library routine exit(3), which flushes buffers, closes streams, unlinks temporary files, etc., before calling **_exit**().

RETURN VALUES

The **_exit**() system call can never return.

SEE ALSO

fork(2), sigaction(2), wait(2), exit(3), init(8)

STANDARDS

The **_exit**() system call is expected to conform to IEEE Std 1003.1-1990 ("POSIX.1").

HISTORY

The **_exit**() function appeared in Version 7 AT&T UNIX.