NAME

exit, _exit – terminate process

SYNOPSIS

#include <unistd.h>

void _exit(int *status***);**

MT-LEVEL

_exit( ) is Async-Signal-Safe

DESCRIPTION

_exit( ) terminates the calling process with the following consequences:

All of the file descriptors open in the calling process are closed.

A **SIGCHLD** signal is sent to the calling process's parent process.

If the parent process of the calling process has "not" specified the **SA_NOCLDWAIT** flag (see **sigaction**(2)),

the calling process is transformed into a "zombie process." A zombie process is a process that only occupies a slot in the process table. It has no other space allocated either in user or kernel space. The process table slot that it occupies is partially overlaid with time accounting information (see **<sys/proc.h>**) to be used by the **times** function.

If the parent process of the calling process is executing **wait**(2) or **waitpid**(2), then it is notified of the calling process's termination and the low-order eight bits of *status* are made available to it.

If the parent process of the calling process is not executing a **wait**(2) or **waitpid**(2), *status* is saved for return to the parent process whenever the parent process executes an appropriate subsequent **wait**(2) or

waitpid(2).

Otherwise, if the parent process of the calling process has specified the **SA_NOCLDWAIT** flag,

status is unavailable to the parent.

The parent process ID of all of the calling process's existing child processes and zombie processes is set to 1. This means the initialization process (see **intro**(2)) inherits each of these processes.

Each attached shared memory segment is detached and the value of **shm_nattach** in the data structure associated with its shared memory identifier is decremented by 1. Memory mappings created in the process are unmapped.

For each semaphore for which the calling process has set a **semadj** value (see **semop**(2)), that **semadj** value is added to the **semval** of the specified semaphore.

If the process has process, text or data locks, an *unlock* is performed (see **plock**(3C) and **memcntl**(2)).

All open named semaphores in the process are closed as if by appropriate calls to **sem_close**(3R). All open message queues in the process are closed as if by appropriate calls to **mq_close**(3R). Any outstanding asynchronous I/O operations may be cancelled.

An accounting record is written on the accounting file if the system's accounting routine is enabled (see **acct**(2)).

If the process is a controlling process, **SIGHUP** is sent to the foreground process group of its controlling terminal and its controlling terminal is deallocated.

If the calling process has any stopped children whose process group will be orphaned when the calling process exits, or if the calling process is a member of a process group that will be orphaned when the calling process exits, that process group will be sent **SIGHUP** and **SIGCONT** signals.

SEE ALSO

acct(2), **intro**(2), **memcntl**(2), **semop**(2), **sigaction**(2), **times**(2), **wait**(2), **waitpid**(2), **exit**(3C), **plock**(3C), **mq_close**(3R), **sem_close**(3R), **signal**(5)