exit(3) Library Functions Manual exit(3)

NAME

exit - cause normal process termination

LIBRARY

Standard C library (libc, -lc)

SYNOPSIS

```
#include <stdlib.h>

[[noreturn]] void exit(int status);
```

DESCRIPTION

The exit() function causes normal process termination and the least significant byte of status (i.e., status & 0xFF) is returned to the parent (see wait(2)).

All functions registered with atexit(3) and on_exit(3) are called, in the reverse order of their registration. (It is possible for one of these functions to use atexit(3) or on_exit(3) to register an addi? tional function to be executed during exit processing; the new regis? tration is added to the front of the list of functions that remain to be called.) If one of these functions does not return (e.g., it calls

functions is called, and further exit processing (in particular, flush?
ing of stdio(3) streams) is abandoned. If a function has been regis?
tered multiple times using atexit(3) or on_exit(3), then it is called
as many times as it was registered.

All open stdio(3) streams are flushed and closed. Files created by
tmpfile(3) are removed.

The C standard specifies two constants, EXIT_SUCCESS and EXIT_FAILURE,
that may be passed to exit() to indicate successful or unsuccessful
termination, respectively.

RETURN VALUE

The exit() function does not return.

ATTRIBUTES

For an explanation of the terms used in this section, see attrib?
utes(7).

??
??

? Interface ? Attribute ? Value ?

??

? exit() ? Thread safety ? MT-Unsafe race:exit ?

??

??

The exit() function uses a global variable that is not protected, so it is not thread-safe.

STANDARDS

C11, POSIX.1-2008.

HISTORY

C89, POSIX.1-2001, SVr4, 4.3BSD.

NOTES

The behavior is undefined if one of the functions registered using atexit(3) and on_exit(3) calls either exit() or longjmp(3). Note that a call to execve(2) removes registrations created using atexit(3) and on_exit(3).

The use of EXIT_SUCCESS and EXIT_FAILURE is slightly more portable (to non-UNIX environments) than the use of 0 and some nonzero value like 1 or -1. In particular, VMS uses a different convention.

such as the GNU C library have also adopted); see the file <sysex? its.h>.

After exit(), the exit status must be transmitted to the parent process. There are three cases:

? If the parent has set SA_NOCLDWAIT, or has set the SIGCHLD handler to SIG_IGN, the status is discarded and the child dies immediately.

? If the parent was waiting on the child, it is notified of the exit status and the child dies immediately.

? Otherwise, the child becomes a "zombie" process: most of the process resources are recycled, but a slot containing minimal information about the child process (termination status, resource usage statis? tics) is retained in process table. This allows the parent to sub? sequently use waitpid(2) (or similar) to learn the termination sta? tus of the child; at that point the zombie process slot is released.

If the implementation supports the SIGCHLD signal, this signal is sent to the parent. If the parent has set SA_NOCLDWAIT, it is undefined whether a SIGCHLD signal is sent.

Signals sent to other processes

is the controlling terminal of the session, then each process in the foreground process group of this controlling terminal is sent a SIGHUP signal, and the terminal is disassociated from this session, allowing it to be acquired by a new controlling process.

If the exit of the process causes a process group to become orphaned, and if any member of the newly orphaned process group is stopped, then a SIGHUP signal followed by a SIGCONT signal will be sent to each process in this process group. See setpgid(2) for an explanation of orphaned process groups.

Except in the above cases, where the signalled processes may be chil? dren of the terminating process, termination of a process does not in general cause a signal to be sent to children of that process. How? ever, a process can use the prctl(2) PR_SET_PDEATHSIG operation to arrange that it receives a signal if its parent terminates.

SEE ALSO

_exit(2), get_robust_list(2), setpgid(2), wait(2), atexit(3), on_exit(3), tmpfile(3)