fork (System Call)

NAME

fork – create a new process

SYNOPSIS
[crayon-6a9d57641a476536709872/]
DESCRIPTION

The fork() function shall create a new process. The new process (child process) shall be an exact copy of the calling process (parent process) except as detailed below:

The child process shall have a unique process ID.

The child process ID also shall not match any active process group ID.

The child process shall have a different parent process ID, which shall be the process ID of the calling process.

The child process shall have its own copy of the parent’s file descriptors. Each of the child’s file descriptors shall refer to the same open file description with the corresponding file descriptor of the parent.

The child process shall have its own copy of the parent’s open directory streams. Each open directory stream in the child process may share directory stream positioning with the corresponding directory stream of the parent.

The child process shall have its own copy of the parent’s message catalog descriptors.

The child process’ values of tms_utime, tms_stime, tms_cutime, and tms_cstime shall be set to 0.

The time left until an alarm clock signal shall be reset to zero, and the alarm, if any, shall be canceled; see alarm().

All semadj values shall be cleared.

File locks set by the parent process shall not be inherited by the child process.

The set of signals pending for the child process shall be initialized to the empty set.

Interval timers shall be reset in the child process.

Any semaphores that are open in the parent process shall also be open in the child process.

The child process shall not inherit any address space memory locks established by the parent process via calls to mlockall() or mlock().

Memory mappings created in the parent shall be retained in the child process. MAP_PRIVATE mappings inherited from the parent shall also be MAP_PRIVATE mappings in the child, and any modifications to the data in these mappings made by the parent prior to calling fork() shall be visible to the child. Any modifications to the data in MAP_PRIVATE mappings made by the parent after fork() returns shall be visible only to the parent. Modifications to the data in MAP_PRIVATE mappings made by the child shall be visible only to the child.

For the SCHED_FIFO and SCHED_RR scheduling policies, the child process shall inherit the policy and priority settings of the parent process during a fork() function. For other scheduling policies, the policy and priority settings on fork() are implementation-defined.

Per-process timers created by the parent shall not be inherited by the child process.

The child process shall have its own copy of the message queue descriptors of the parent. Each of the message descriptors of the child shall refer to the same open message queue description as the corresponding message descriptor of the

parent.

No asynchronous input or asynchronous output operations shall be inherited by the child process.

A process shall be created with a single thread. If a multi-threaded process calls fork(), the new process shall contain a replica of the calling thread and its entire address space, possibly including the states of mutexes and other resources. Consequently, to avoid errors, the child process may only execute async-signal-safe operations until such time as one of the exec functions is called.

Fork handlers may be established by means of the pthread_atfork() function in order to maintain application invariants across fork() calls.

When the application calls fork() from a signal handler and any of the fork handlers registered by pthread_atfork() calls a function that is not asynch-signal-safe, the behavior is undefined.

If the Trace option and the Trace Inherit option are both supported:

If the calling process was being traced in a trace stream that had its inheritance policy set to POSIX_TRACE_INHERITED, the child process shall be traced into that trace stream, and the child process shall inherit the parent’s mapping of trace event names to trace event type identifiers. If the trace stream in which the calling process was being traced had its inheritance policy set to POSIX_TRACE_CLOSE_FOR_CHILD, the child process shall not be traced into that trace stream. The inheritance policy is set by a call to the posix_trace_attr_setinherited() function.

If the Trace option is supported, but the Trace Inherit option is not supported:

The child process shall not be traced into any of the trace streams of its parent process.

If the Trace option is supported, the child process of a trace controller process shall not control the trace streams controlled by its parent process.

The initial value of the CPU-time clock of the child process shall be set to zero.

The initial value of the CPU-time clock of the single thread of the child process shall be set to zero.

All other process characteristics defined by IEEE Std 1003.1-2001 shall be the same in the parent and child processes. The inheritance of process characteristics not defined by IEEE Std 1003.1-2001 is unspecified by IEEE Std 1003.1-2001.

After fork(), both the parent and the child processes shall be capable of executing independently before either one terminates.

RETURN VALUE

Upon successful completion, fork() shall return 0 to the child process and shall return the process ID of the child process to the parent process. Both processes shall continue to execute from the fork() function. Otherwise, -1 shall be returned to the parent process, no child process shall be created, and errno shall be set to indicate the error.

ERRORS

The fork() function shall fail if:

[EAGAIN]
The system lacked the necessary resources to create another process, or the system-imposed limit on the total number of processes under execution system-wide or by a single user

{CHILD_MAX} would be exceeded.

The fork() function may fail if:

[ENOMEM]
Insufficient storage space is available.