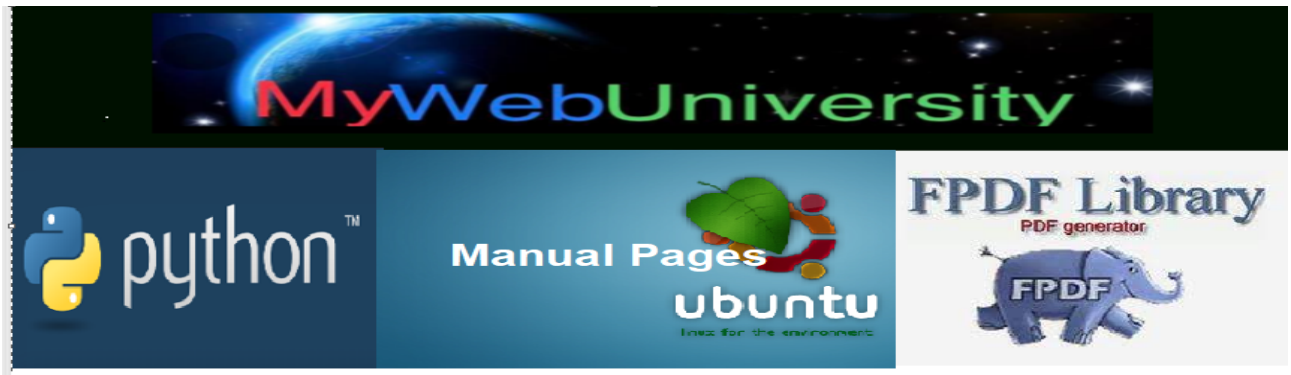

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command 'pthread_atfork.3'

$ man pthread_atfork.3

pthread_atfork(3) Library Functions Manual pthread_atfork(3)

NAME

pthread_atfork - register fork handlers

LIBRARY

POSIX threads library (libpthread, -lpthread)

SYNOPSIS

```
#include <pthread.h>

int pthread_atfork(typeof(void (void)) *prepare,
                   typeof(void (void)) *parent,
                   typeof(void (void)) *child);
```

DESCRIPTION

The pthread_atfork() function registers fork handlers that are to be executed when fork(2) is called by any thread in a process. The handlers are executed in the context of the thread that calls fork(2).

Three kinds of handler can be registered:

? prepare specifies a handler that is executed in the parent process before fork(2) pro? cessing starts.

? parent specifies a handler that is executed in the parent process after fork(2) process? ing completes.

? child specifies a handler that is executed in the child process after fork(2) processing completes.

Any of the three arguments may be NULL if no handler is needed in the corresponding phase of fork(2) processing.

RETURN VALUE

On success, pthread_atfork() returns zero. On error, it returns an error number.

pthread_atfork() may be called multiple times by a process to register additional handlers.

The handlers for each phase are called in a specified order: the prepare handlers are called in reverse order of registration; the parent and child handlers are called in the order of registration.

ERRORS

ENOMEM Could not allocate memory to record the fork handler list entry.

STANDARDS

POSIX.1-2008.

HISTORY

POSIX.1-2001.

NOTES

When fork(2) is called in a multithreaded process, only the calling thread is duplicated in the child process. The original intention of pthread_atfork() was to allow the child process to be returned to a consistent state. For example, at the time of the call to fork(2), other threads may have locked mutexes that are visible in the user-space memory du? plicated in the child. Such mutexes would never be unlocked, since the threads that placed the locks are not duplicated in the child. The intent of pthread_atfork() was to provide a mechanism whereby the application (or a library) could ensure that mutexes and other process and thread state would be restored to a consistent state. In practice, this task is gener? ally too difficult to be practicable.

After a fork(2) in a multithreaded process returns in the child, the child should call only async-signal-safe functions (see signal-safety(7)) until such time as it calls execve(2) to execute a new program.

POSIX.1 specifies that pthread_atfork() shall not fail with the error EINTR.

SEE ALSO

fork(2), atexit(3), pthreads(7)