NAME

vfork – spawn new process in a virtual memory efficient way

SYNOPSIS

#include <unistd.h>

pid_t vfork(void);

DESCRIPTION

vfork( ) can be used to create new processes without fully copying the address space of the old process. It is useful when the purpose of **fork( )** would have been to create a new system context for an **execve( )**. **vfork( )** differs from **fork( )** in that the child borrows the parent's memory and thread of control until a call to **execve( )** or an exit (either by a call to **_exit( )** (see **exit**(2)) or abnormally). The parent process is suspended while the child is using its resources.

vfork( ) returns 0 in the child's context and (later) the process ID (PID) of the child in the parent's context.

vfork( ) can normally be used just like **fork( )**. It does not work, however, to return while running in the child's context from the procedure which called **vfork( )** since the eventual return from **vfork( )** would then return to a no longer existent stack frame. Be careful, also, to call **_exit( )** rather than **exit**(3C) if you cannot **execve( )**, since **exit**(3C) will flush and close standard I/O channels, and thereby corrupt the parent processes standard I/O data structures. Even with **fork( )** it is wrong to call **exit**(3C) since buffered data would then be flushed twice.

RETURN VALUES

Upon successful completion, **vfork( )** returns a value of 0 to the child process and returns the process ID of the child process to the parent process. Otherwise, a value of –1 is returned to the parent process, no child process is created, and the global variable **errno** is set to indicate the error.

ERRORS

vfork( ) will fail and no child process will be created if one or more of the following are true:

EAGAIN The system-imposed limit on the total number of processes under execution would be exceeded. This limit is determined when the system is generated.

EAGAIN The system-imposed limit on the total number of processes under execution by a single user would be exceeded. This limit is determined when the system is generated.

ENOMEM There is insufficient swap space for the new process.

SEE ALSO

exec(2), **exit**(2), **fork**(2), **ioctl**(2), **wait**(2), **exit**(3C)

NOTES

vfork( ) is unsafe in multi-thread applications.

This function will be eliminated in a future release. The memory sharing semantics of **vfork( )** can be obtained through other mechanisms.

To avoid a possible deadlock situation, processes that are children in the middle of a **vfork( )** are never sent **SIGTTOU** or **SIGTTIN** signals; rather, output or *ioctl*s are allowed and input attempts result in an EOF indication.

On some systems, the implementation of **vfork( )** causes the parent to inherit register values from the child. This can create problems for certain optimizing compilers if **<unistd.h>** is not included in the source calling **vfork( )**.