

NAME

vfork - create a new process without copying the address space

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <unistd.h>
```

```
pid_t
```

```
vfork(void);
```

DESCRIPTION

Since this function is hard to use correctly from application software, it is recommended to use `posix_spawn(3)` or `fork(2)` instead.

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

The **vfork()** system call returns 0 in the child's context and (later) the pid of the child in the parent's context.

Many problems can occur when replacing `fork(2)` with **vfork()**. For example, it does not work to return while running in the child's context from the procedure that called **vfork()** since the eventual return from **vfork()** would then return to a no longer existent stack frame. Also, changing process state which is partially implemented in user space such as signal handlers with `libthr(3)` will corrupt the parent's state.

Be careful, also, to call `_exit(2)` rather than `exit(3)` if you cannot `execve(2)`, since `exit(3)` will flush and close standard I/O channels, and thereby mess up the parent processes standard I/O data structures. (Even with `fork(2)` it is wrong to call `exit(3)` since buffered data would then be flushed twice.)

RETURN VALUES

Same as for `fork(2)`.

SEE ALSO

`_exit(2)`, `execve(2)`, `fork(2)`, `rfork(2)`, `sigaction(2)`, `wait(2)`, `exit(3)`, `posix_spawn(3)`

HISTORY

The **vfork()** system call appeared in 3BSD.

BUGS

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(2) calls are allowed and input attempts result in an end-of-file indication.