## 所需头文件：

#include <sys/types.h>

#include <unistd.h>

## ****pid\_t vfork(void);****

功能：

vfork() 函数和 fork() 函数（[fork()如何使用，](http://blog.csdn.net/tennysonsky/article/details/45165811)请[点此链接](http://blog.csdn.net/tennysonsky/article/details/45165811)）一样都是在已有的进程中创建一个新的进程，但它们创建的子进程是有区别的。

参数：

无

返回值：

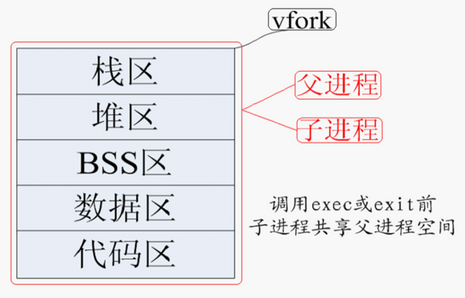
成功：子进程中返回 0，父进程中返回子进程 ID。pid\_t，为无符号整型。

失败：返回 -1。

## ****fork() 与 vfock() 都是创建一个进程，那它们有什么区别呢？****

vfork()创建的子进程与父进程共享进程空间

vfork()创建的子进程执行exec（进程替换）或exit（退出进程）后父进程才执行



## 验证：通过 vfork() 创建的子进程会执行完后，才到父进程执行：

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

int main(int argc, char \*argv[])

{

    pid\_t pid;

    pid = vfork();  // 创建进程

    if(pid < 0){ // 出错

        perror("vfork");

    }

    if(0 == pid){ // 子进程

        sleep(3); // 延时 3 秒

        printf("i am son\n");

        \_exit(0); // 退出子进程，必须

    }else if(pid > 0){ // 父进程

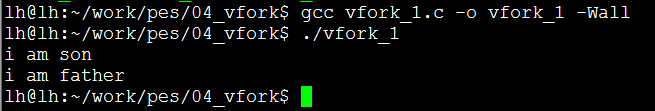
        printf("i am father\n");

    }

    return 0;

}

上面的代码，已经让子进程延时 3 s，结果还是子进程运行结束后，父进程才执行，运行结果如下：



vfork() 保证子进程先运行，在它调用 exec（进程替换） 或 exit（退出进程）之后父进程才可能被调度运行。

用 vfork() 创建进程，子进程里一定要调用 exec（进程替换） 或 exit（退出进程），否则，程序会出问题，没有意义。