实验二 Linux进程控制1

**学号： 19020011038 姓名： 岳宇轩 年级： 2019**

**一.使用fork系统调用创建多个子进程**

**1.具体要求与步骤**

1. 编写一C/C++语言程序（程序名为fork.c/fork.cpp），使用系统调用fork()创建两个子进程。当程序运行时，系统中有一个父进程和两个子进程在并发执行。父亲进程执行时屏幕显示“I am father”，儿子进程执行时屏幕显示“I am son”，女儿进程执行时屏幕显示“I am daughter”。
2. 多次连续反复运行这个程序，观察屏幕显示结果的顺序，直至出现不一样的情况为止。记下这种情况，试简单分析其原因。
3. 可以使用实验报告模板中所推荐的代码实现，但是要求为代码添加注释，对代码关键逻辑步骤进行解释。在代码头部添加如代码1所示式样的头部版权声明。使用星号、井号、等号、破折号等各类符号对版权声明添加边框，并拼出19os的式样。
4. **实验结果截图**

过程说明：

创建exp2-1.c文件，写入代码并添加头部版权注释

在终端执行gcc exp2-1.c -o exp2-1命令进行编译

执行编译生成的文件./exp2-1

多次执行，发现在I am son! 和 I am daughter!的出现顺序不同

不同顺序分析原因：进程并发执行

程序一共创建了三个进程，分别是父进程，儿子进程和女儿进程。由于父进程是在程序一开始运行就创建了的，所以先输出了 “I am father! ” .后来，父进程先执行创建儿子进程的操作，这时父进程和儿子进程并发执行：父进程执行的操作是创建女儿进程，儿子进程执行的操作是判断儿子进程是否正在执行。

因为只有当儿子进程获得处理机时，才可进行是否正在执行的判断（此时一定在执行），所以相当于判断儿子进程是否获得处理机。换句话说，只有儿子进程获得了处理机，才能执行判断是否正在执行的这一行代码。

当父进程完成女儿进程的创建之后，父进程，儿子进程和女儿进程并发执行。

1.在父进程和儿子进程并发执行时，如果父进程先获得处理机资源创建了女儿进程，而女儿进程在被创建后，一直占据处理处理机资源，完成了判断女儿进程是否执行的操作，然后才由儿子进程获取处理机资源，此时输出为：

I am father!

I am daughter!

I am son!

2在父进程和儿子进程并发执行时，如果儿子进程先获得处理机资源，执行其内容，判断儿子进程正在获得处理机，然后交由父进程获得处理机资源，创建女儿进程，然后由女儿进程判断其正在获得处理机资源，此时输出为：

I am father!

I am son!

I am daughter!

不同的顺序体现了进程的并发执行

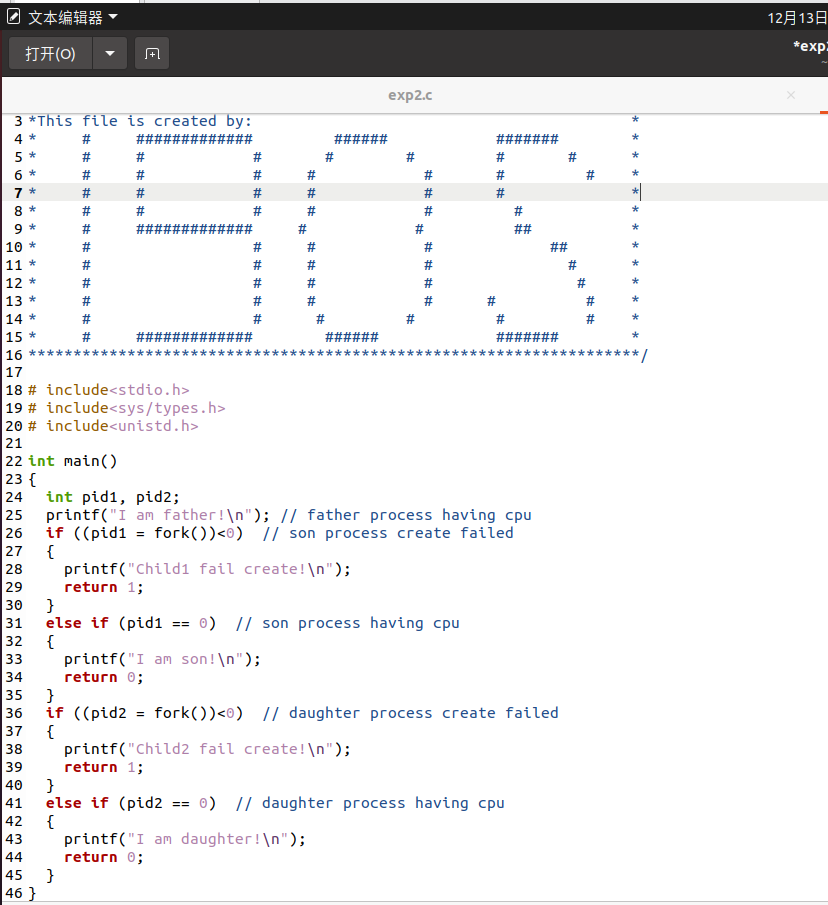


图 1 代码文件

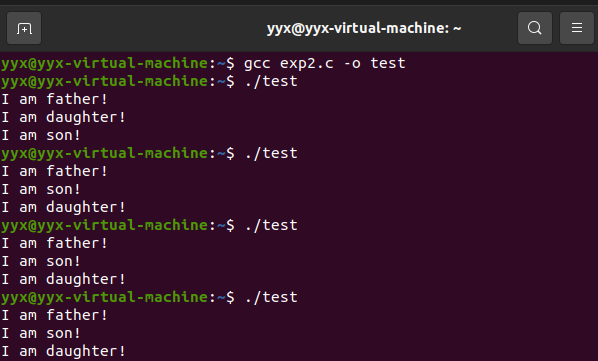


图 2 出现不同顺序

**二.使用fork系统调用创建多级子进程**

**1.具体要求与步骤**

1. 编写一C/C++语言程序（程序名为fork.c），使用系统调用fork( )创建一个子进程，然后在子进程中再创建子子进程。当程序运行时，系统中有一个父进程、一个子进程和一个孙子进程在并发执行。父亲进程执行时屏幕显示“I am father”，儿子进程执行时屏幕显示“I am son”，孙子进程执行时屏幕显示“c”。
2. 多次连续反复运行这个程序，观察屏幕显示结果的顺序，直至出现不一样的情况为止。记下这种情况，试简单分析其原因。
3. 可以使用实验报告模板中所推荐的代码实现，但是要求为代码添加注释，对代码关键逻辑步骤进行解释。在代码头部添加如代码1所示式样的头部版权声明。使用星号、井号、等号、破折号等各类符号对版权声明添加边框，并拼出19os的式样。

**2.实验结果截图**

代码过程说明：

首先在主函数中输出I am father，表示主函数正在执行（即父进程）。

使用fork函数创建子进程1，如果函数返回值<0则表示创建失败。

在子进程中需要：

1. 判断子进程是否正在被执行，如果是则请求控制台输出的IO；获得IO资源后则向控制台输出I am son
2. 创建孙子进程;如果fork返回<0表示创建失败。

在孙子进程中需要：

1. 判断孙子进程是否正在被执行，如果是则请求控制台输出的IO；获得IO资源后则向控制台输出c

不同顺序原因分析：进程争用临界资源，要互斥的访问临界资源

首先，父进程在程序一开始就被创建，此时IO资源未被占用，故直接输出了I am father

出现不同顺序的原因主要在于：在子进程中，孙子进程被创建后直接抢占了临界资源：控制台输出的IO，直接输出了c，在孙子进程释放资源后，子进程才获得了控制台输出的IO资源，输出I am son。此时输出顺序为：

I am father

c

I am son

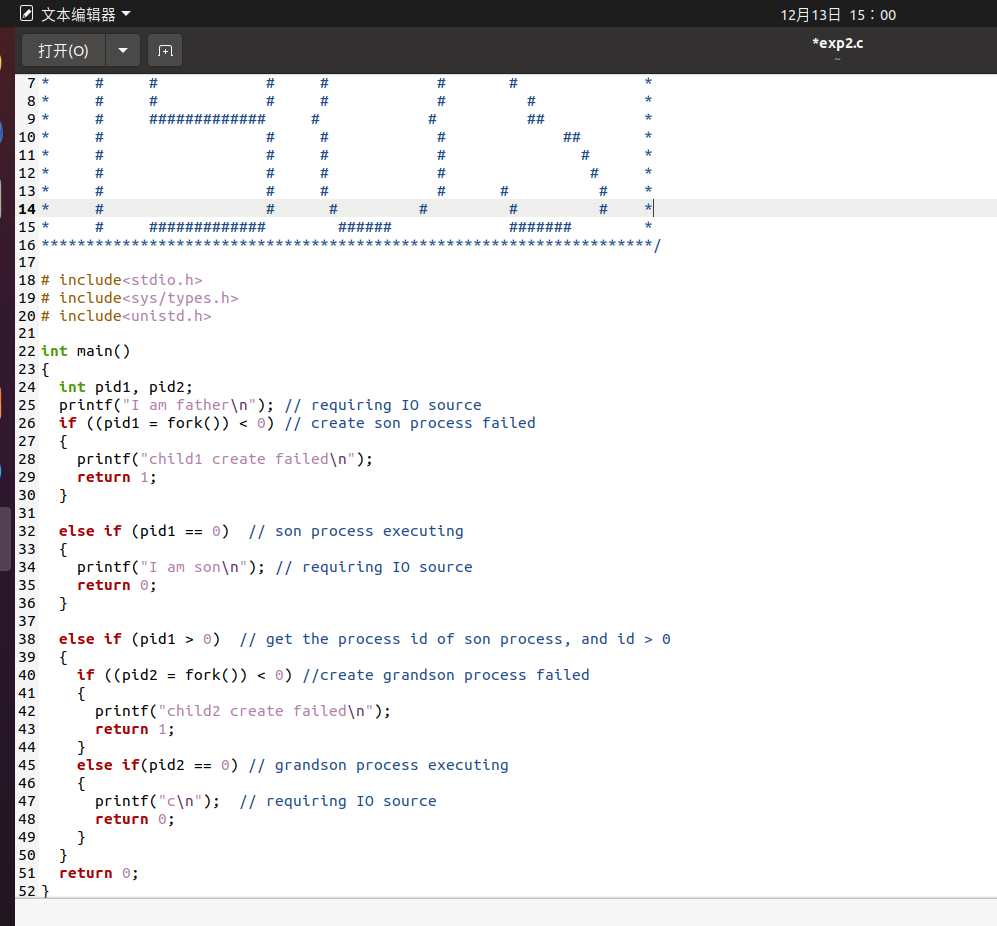
如果在子进程先抢占了控制台输出的IO资源，这时输出顺序就变为：

I am father

I am son

c

不同的顺序体现了进程对临界资源的互斥访问

图 3 程序代码截图

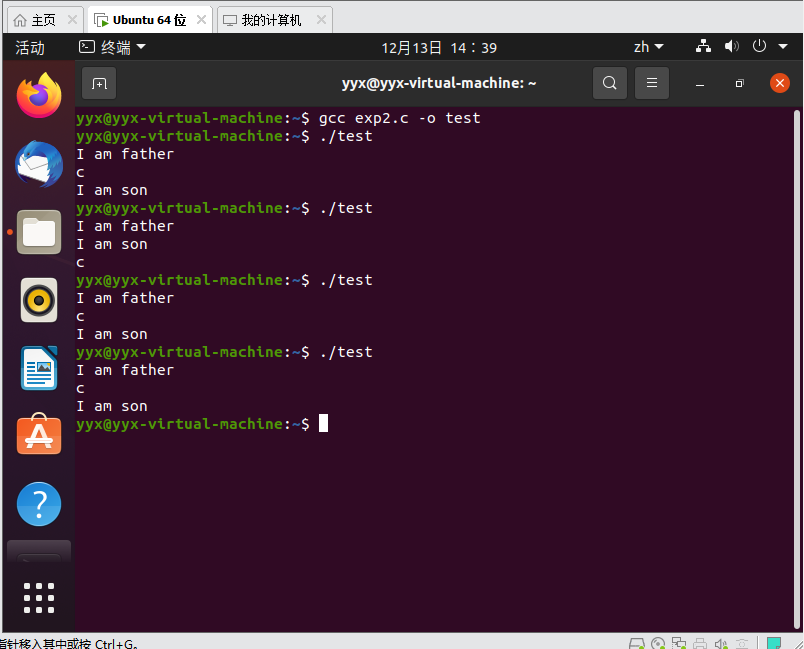


图 4 不同顺序结果