实验三 Linux进程控制2

**学号： 20110031020 姓名： 许健 年级： 20计科**

1. **实验目的**
2. 进一步了解进程控制的各项原理及功能。
3. 了解wait( )、exit( )、lockf( )等系统调用的使用方法。
4. **实验预备知识及实验环境搭建**
   1. **实验预备知识**
5. wait()系统调用：wait( )会暂时停止目前进程的执行，直到有信号来到或子进程结束。如果在调用wait()时子进程已经结束，则wait()会立即返回子进程结束状态值。子进程的结束状态值会由参数status返回，而子进程的进程识别码也会一快返回。如果不在意结束状态值，则参数status 可以设成NULL。子进程的结束状态值请参考waitpid()。返回值方面，如果执行成功则返回子进程识别码（PID），如果有错误发生则返回-1。具体内容参考表 1。

表 1 wait系统调用的具体信息

|  |  |
| --- | --- |
| 相关函数 | waitpid、fork |
| 表头文件 | #include<sys/types.h>  #include<sys/wait.h> |
| 定义函数 | pid\_t wait (int \* status); |

1. exit()系统调用：不像fork那么难理解，从exit的名字就能看出，这个系统调用是用来终止一个进程的。无论在程序中的什么位置，只要执行到exit系统调用，进程就会停止剩下的所有操作，清除包括PCB在内的各种数据结构，并终止本进程的运行。exit系统调用带有一个整数类型的参数status，我们可以利用这个参数传递进程结束时的状态，比如说，该进程是正常结束的，还是出现某种意外而结束的，一般来说，0表示没有意外的正常结束；其他的数值表示出现了错误，进程非正常结束。exit系统调用的具体信息如表 2所示。

表 2 exit系统调用的具体信息

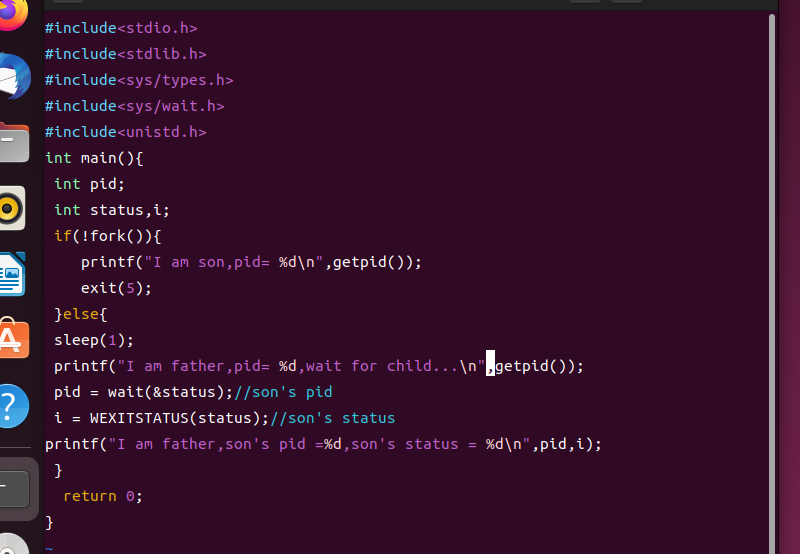
|  |  |
| --- | --- |
| 表头文件 | #include<stdlib.h> |
| 定义函数 | void exit(int status); |

1. lockf：利用系统调用lockf（fd，mode，size），对指定区域（有size指示）进行加锁或解锁，以实现进程的同步或互斥。其中，fd是文件描述字；mode是锁定方式，mode=1表示加锁，mode=0表示解锁；size是指定文件fd的指定区域，用0表示从当前位置到文件结尾（注：有些Linux系统是locking（fd，mode，size））。
   1. **实验环境搭建：**

Ubuntu或CentOS，带有vi编辑器和gcc编译器。

1. **使用wait系统调用**
   1. **实验要求**
2. 编写一C/C++语言程序（程序名为testwait.c/testwait.cpp），使用系统调用wait()控制进程的阻塞。
3. 多次连续反复运行这个程序，观察屏幕显示结果的顺序，试简单分析其原因。
4. 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。
   1. 实验截图:

代码:

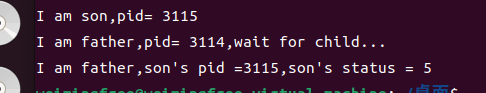


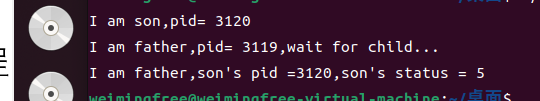
建立一个子进程

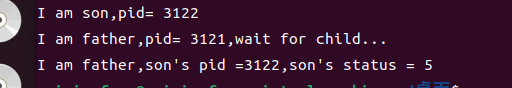
对于子进程来说，打印出子进程的pid，并以5的状态结束子进程

对于父进程来说，暂停1s后，打印出父进程的pid，等待子进程结束，打印出子进程的pid和返回的状态

结果：

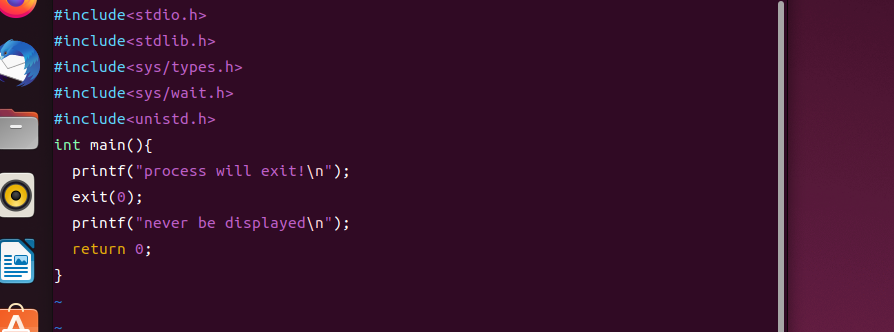




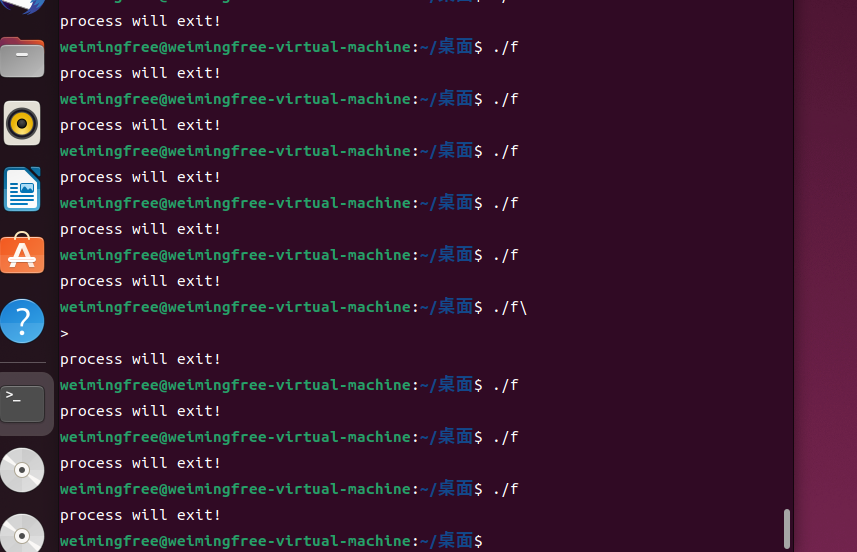


1. **使用exit系统调用**
   1. **实验要求**
2. 编写一C/C++语言程序（程序名为testexit.c/testexit.cpp），使用系统调用exit( )尝试对进程进行终止操作。
3. 多次连续反复运行这个程序，观察屏幕显示结果的顺序，试简单分析其原因。
4. 可以使用实验报告模板中所推荐的代码实现，但是要求为代码添加注释，对代码关键逻辑步骤进行解释。在代码头部添加如代码 1所示式样的头部版权声明。使用星号、井号、等号、破折号等各类符号对版权声明添加边框，并拼出19os的式样。

4.2实验截图:

代码：  


结果：



可以看到在exit()之后的那行代码永远打印不出来

1. **使用lockf系统调用**
   1. **实验要求**

1) 编写一C/C++语言程序（程序名为testlockf.c/testlockf.cpp），使用系统调 用lockf( )尝试对程序的指定区域进行加解锁操作。

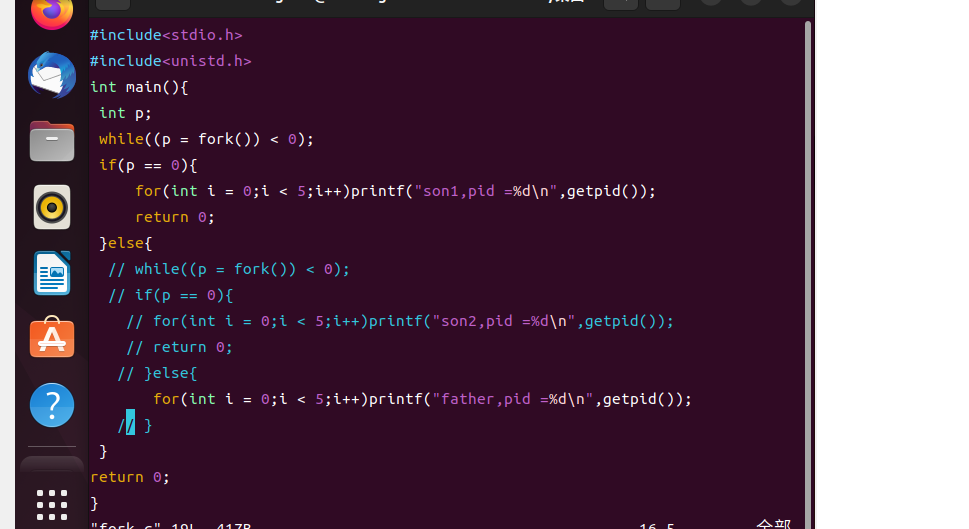
2) 多次连续反复运行这个程序，观察屏幕显示结果的顺序，试简单分析其 原因。

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。

5.2实验截图:

1)

代码:



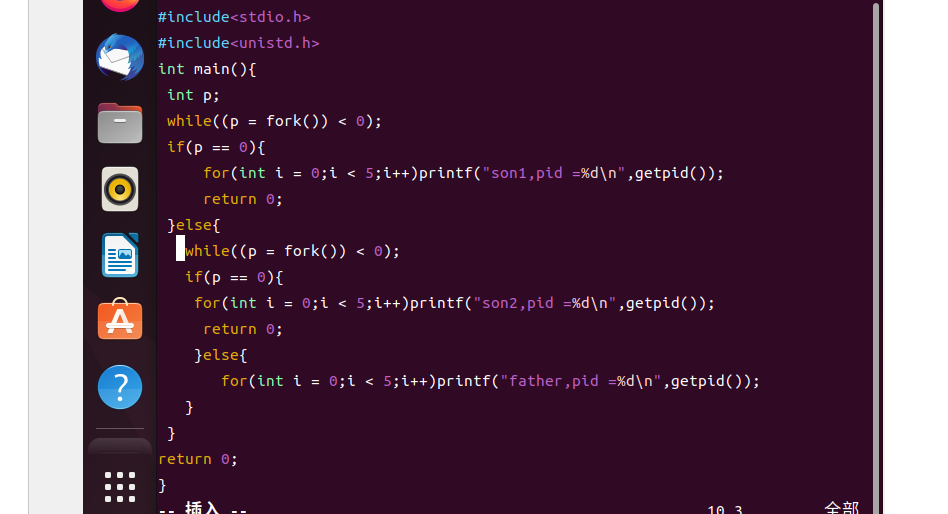
截图：

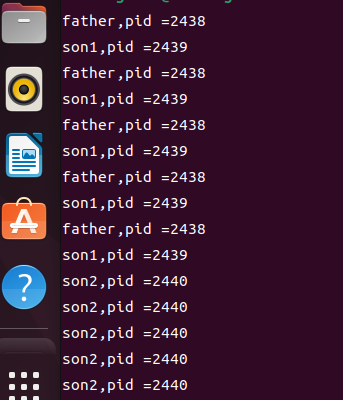


如上图所示，当仅有两个进程时，即使没有lockf，出现交叉打印的几率也很小。

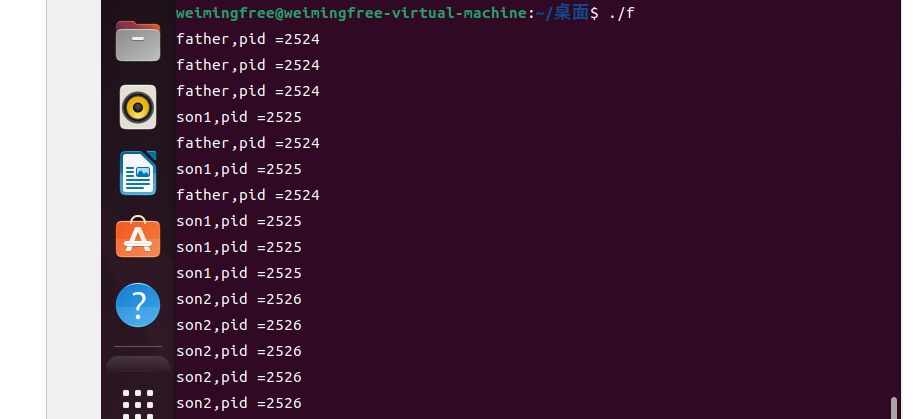
2）

代码：

  
截图：



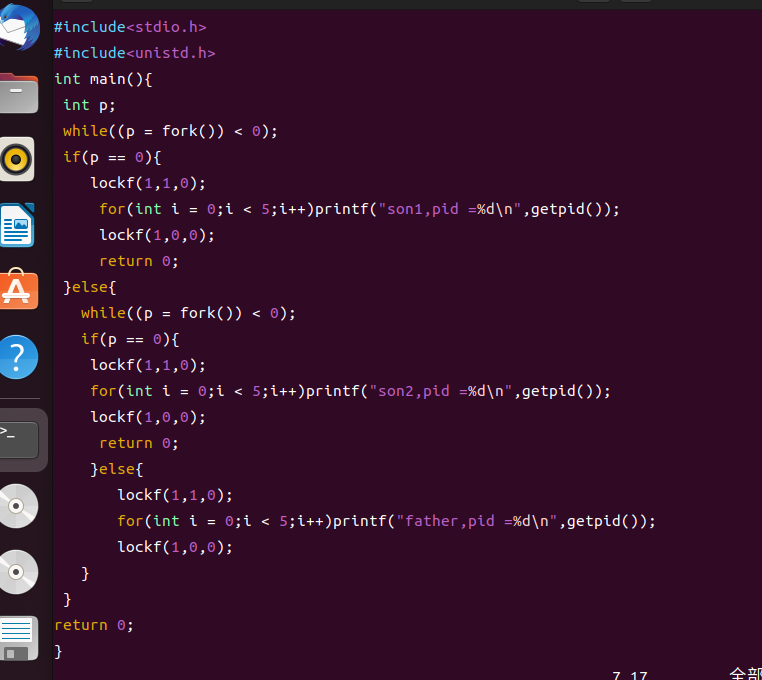




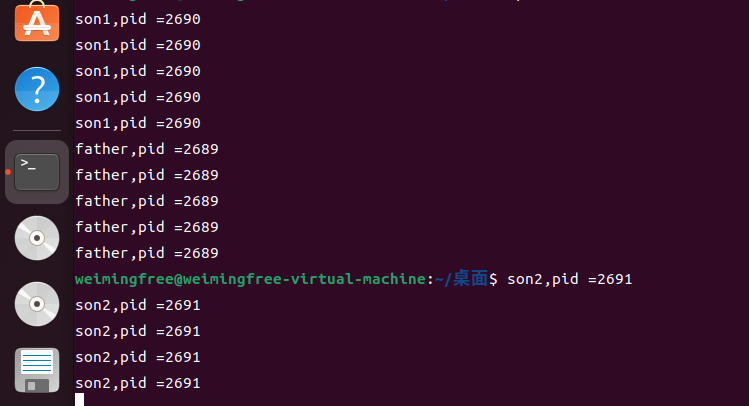
如上图所示，如果有三个进程，在未加上锁之前，出现交叉打印的几率大大增加。

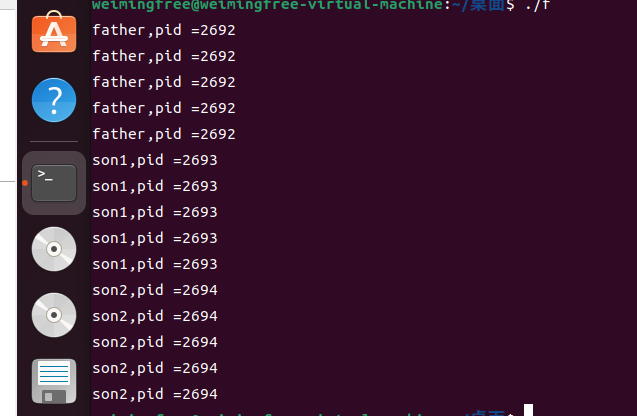
3）

代码:



结果:





当三个进程都上锁之后，就不会出现交叉打印的现象了