第三次上机　邓依伊　201411212043

1. 修改实验二的程序，将每个进程输出一个字符改为每个进程输出一句话，观察分析显示结果；

实验二程序修改如下，每个进程输出字符为一句话：

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
    int p1,p2;
    p1 = fork();
    if (p1==-1)
        printf("error");
    else if (p1==0)
        printf("b:%d  father:%d\n",getpid(),getppid());
    else {
        p2 = fork();
        if (p2==0)
            printf("c:%d  father:%d\n",getpid(),getppid());
        else {
            printf("a:%d\n",getpid());
            sleep(1);
        }
    }
}
```

输出结果如下：

```
dengyiyideMacBook-Air:Desktop Ivy11D$ gcc exp2.c
dengyiyideMacBook-Air:Desktop Ivy11D$ ./a.out
a:2804
b:2805  father:2804
c:2806  father:2804
dengyiyideMacBook-Air:Desktop Ivy11D$ ./a.out
a:2807
b:2808  father:2807
c:2809  father:2807
dengyiyideMacBook-Air:Desktop Ivy11D$ ./a.out
a:2810
b:2811  father:2810
c:2812  father:2810
```

1. 如果在父进程fork之前，输出一句话，这句话后面不加“\n”或加“\n”，结果有什么不同，为什么?

在父进程fork之前，输出一句“This is exp.3”，后面加“\n”：

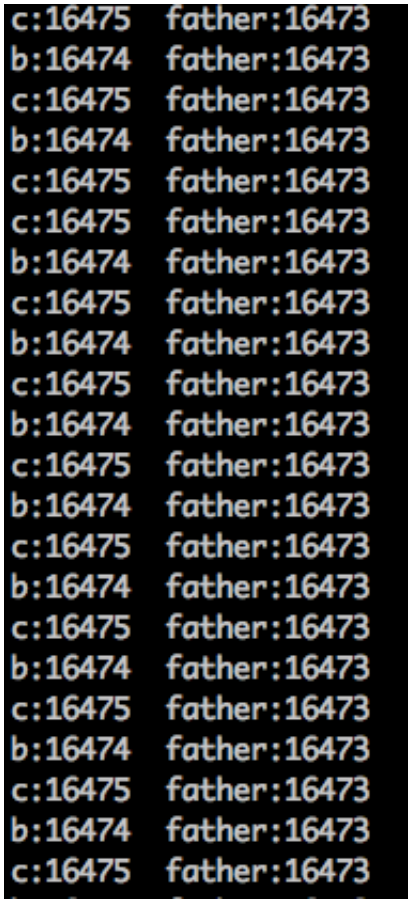
```
This is exp.3
a:2836
b:2837  father:2836
c:2838  father:1
```

后面不加“\n”：

```
This is exp.3a:2842
This is exp.3b:2843  father:2842
This is exp.3c:2844  father:2842
```

1. 如果在程序中使用系统调用lockf来给每一个进程加锁，可以实现进程之间的互斥。将lockf加在输出语句前后运行试试；将一条输出语句变成多条输出语句，将lockf语句放在循环语句外部或内部试试， 观察分析显示结果。

首先，这是不加lockf语句并每个进程循环打印50次语句的结果，输出结果中可以看见b和c进程的输出之间有打断：



将lockf语句改为放在循环语句外部后，三个进程的输出之间没有穿插，a,b,c三个进程各按顺序输出50行。

1. 以上各种情况都多运行几次，观察每次运行结果是否都一致？为什么？

lockf语句放在循环语句内部时，相当于没有写lockf语句，三个进程输出多个语句的穿插情况是随机的，所以此时运行几次的结果不一定完全一致。

而lockf语句放在循环语句外部时，一个进程在逐行打印时其他进程无法进行打印，只有等其语句全部打印完了才进行解锁，才能由下一个进程打印，这时多次运行的结果是一样的。

思考与体会

当父进程fork子进程后，父进程和子进程如何执行程序的?

fork之后，操作系统会复制一个与父进程完全相同的子进程，虽说是父子关系，但是在操作系统看来，他们更像兄弟关系，这2个进程共享代码空间，但是数据空间是互相独立的，子进程数据空间中的内容是父进程的完整拷贝，指令指针也完全相同，子进程拥有父进程当前运行到的位置（两进程的程序计数器pc值相同，也就是说，子进程是从fork返回处开始执行的），但有一点不同，如果fork成功，子进程中fork的返回值是0，父进程中fork的返回值是子进程的进程号，如果fork不成功，父进程会返回错误。