计算机科学与工程学院实验报告（首页）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | | **操作系统** | | | **班级** |  | | |
| **实验名称** | | **实验1实验环境与基础进程编程** | | | **指导教师** | |  | |
| **姓名** |  | | **学号** |  | **组号** | **-** | |
| **日期2022年12月1日** | | | | | | | |

1. 实验目的
   1. 熟悉Linux的基本命令
   2. 掌握vi工具的使用
   3. 掌握gcc编译工具的使用
   4. 掌握基本进程创建方法
2. 实验要求
3. 在linux环境下，熟悉基本命令，如cd, ls, rm, rmdir, chmod, cp,设置环境变量等。
4. 练习Linux vi文本编辑工具的使用，创建文件，修改文件，删除字符/行，拷贝粘贴字符/行，保存，强制退出等。
5. 使用Linux vi文本编辑工具编写一个简单进程创建程序，并进行编译，链接，输出指定文件，并运行。
6. 熟悉gcc/gdb工具的使用。
7. 基础知识准备（如果已经选修过Linux，则跳过本题目）
8. 基本命令：cd, ls, rm, rmdir, mkdir, touch, cp等命令。
9. vi编辑器的使用：创建文件，编辑文件，保存文件，退出，编辑模式与命令模式切换等。（可以在终端输入: vimtutor 根据教程进行学习。
10. 运行程序，解释现象

void main(){

int pid1=fork();

printf(“\*\*1\*\*\n”);

int pid2=fork();

printf(“\*\*2\*\*\n”);

if(pid1==0){int pid3=fork();printf(“\*\*3\*\*\n”);}

else printf(“\*\*4\*\*\n”);

}

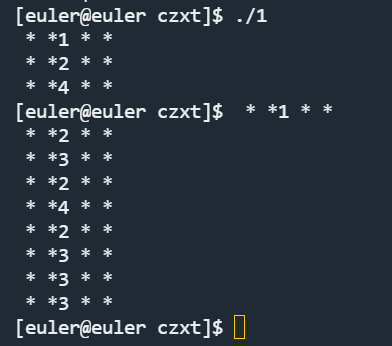
1. 编写一段程序，使用系统调用fork()创建两个子进程。当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符；父进程显示字符“a”；子进程分别显示字符“b”和字符“c”。试观察记录屏幕上的显示结果，并分析原因。
2. 说明

本实验应在教师指导下进行，尽量不要擅自进行，否则可能会破坏原有系统。

1. 运行结果及实验结果、实验体会

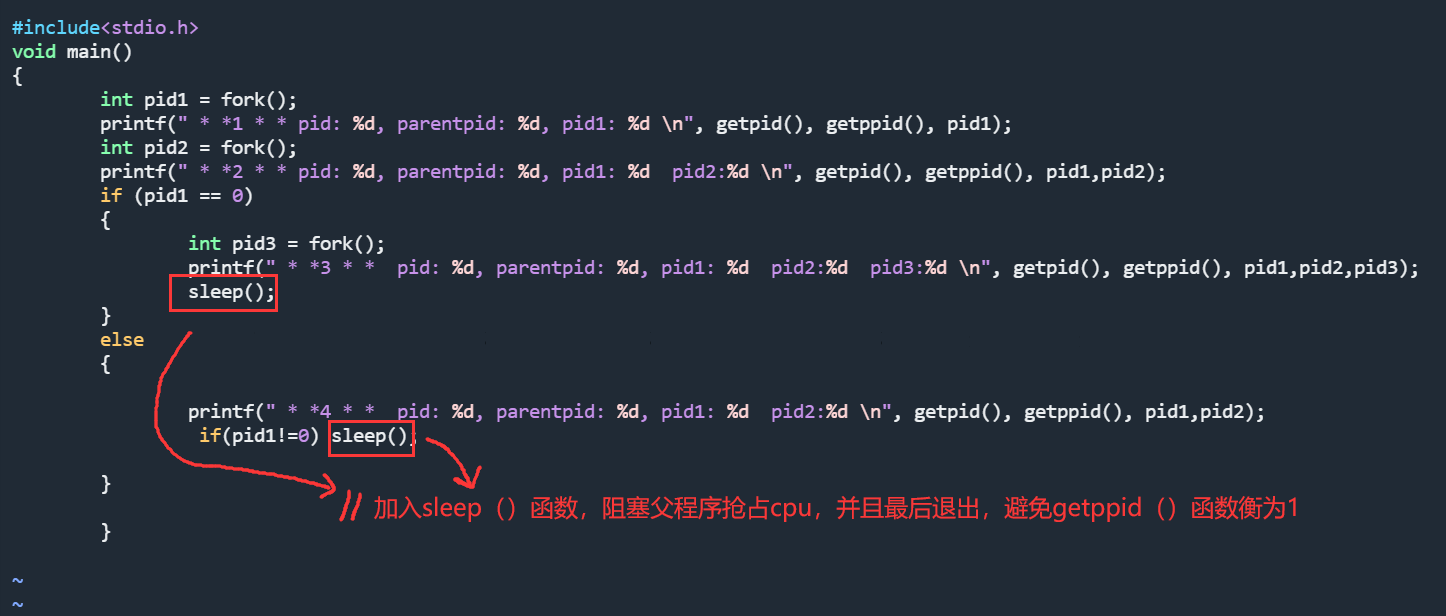
# 2.运行程序，解释现象

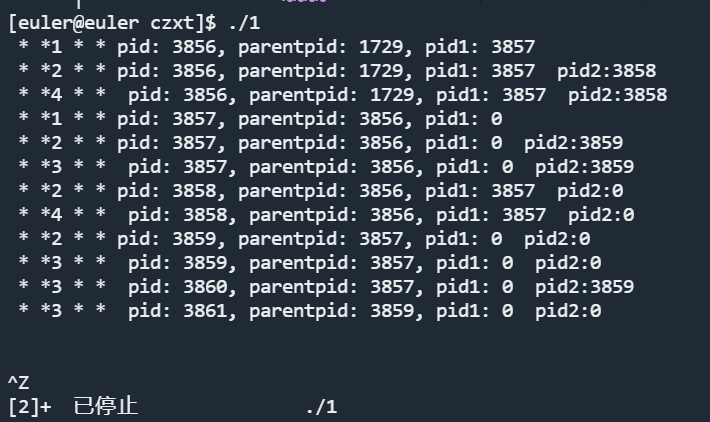
2.（1）．**原代码运行结果：**



**（2）现象解释：**

为了更清晰的搞懂fork函数的原理，我对原代码进行了改动：





改动后运行最后结果

改动后从而清楚得到每一次fork函数后，打印时程序的pid以及其parentpid，还有pid1，pid2的返回值。从fork函数原理我们得知：

fork函数的返回值有三种情况，

1.父程序返回值为非0。

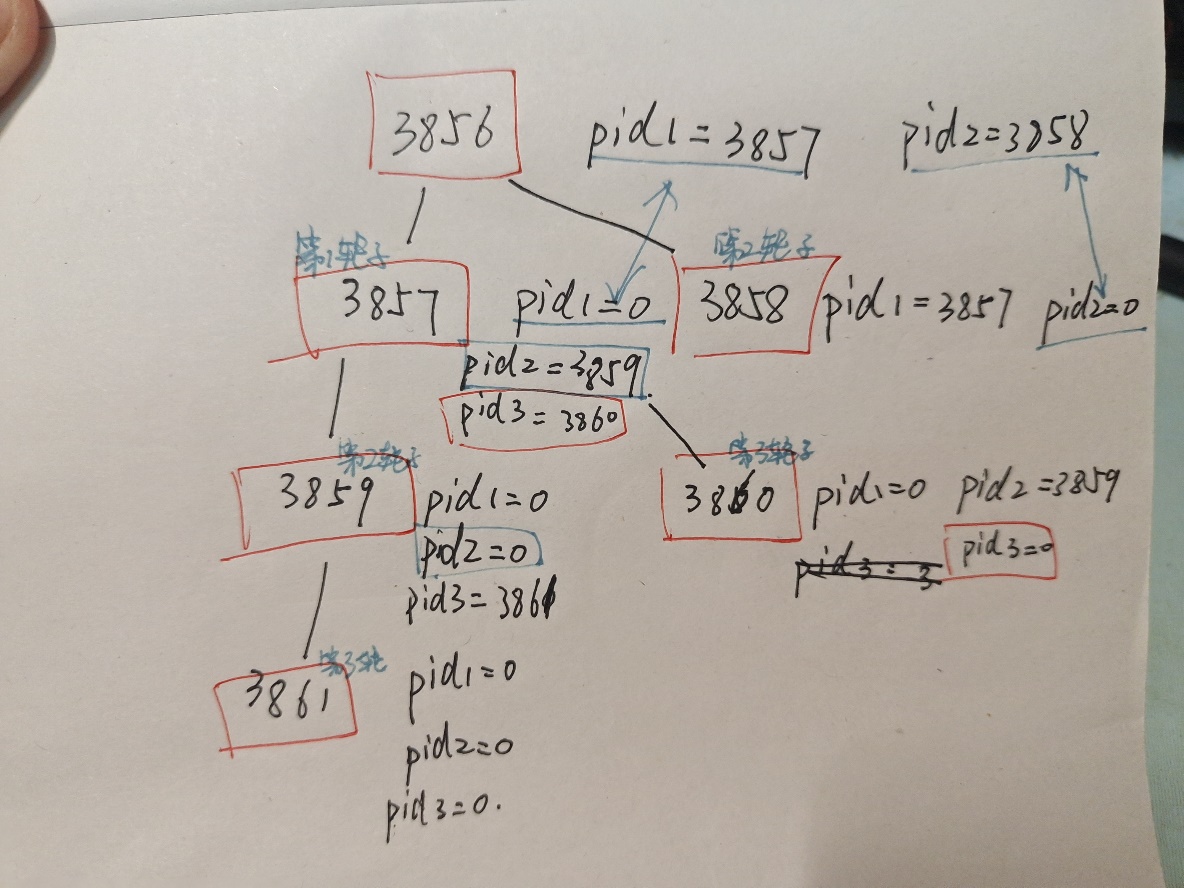
2.子程序返回值为0。

3.运行失败返回值<0。

由此我们得知pid1，pid2等原理得以解释：



具体步骤解释



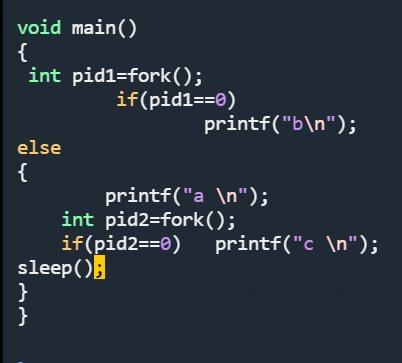
从此规律得知树状图如上

由于父进程与子进程同时抢占cpu，导致所以每一次编译时，数字排序各不相同的。

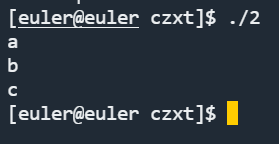
由于改动后的程序运用了sleep函数阻塞，所以程序编译需要ctrl+z才能结束。

# 3、编写一段程序，使用系统调用fork()创建两个子进程。当此程序运行时，在系统中有一个父进程和两个子进程活动。让每一个进程在屏幕上显示一个字符；父进程显示字符“a”；子进程分别显示字符“b”和字符“c”。试观察记录屏幕上的显示结果，并分析原因。

**3.（1）源代码：**



**（2）运行结果：**



**下面我将结合改动后的代码来分析运行结果原因。**

**（3）改动后源代码：**

*void main()*

*{*

*int pid1=fork();*

*if(pid1==0)*

*printf("b pid=%d pid1=%d ppid=%d \n ",getpid(),pid1,getppid());*

*else*

*{*

*printf("a pid=%d pid1=%d ppid=%d \n",getpid(),pid1,getppid());*

*int pid2=fork();*

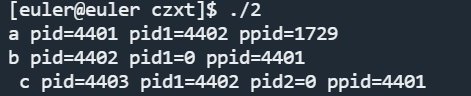
*if(pid2==0) printf("c pid=%d pid1=%d pid2=%d ppid=%d \n",getpid(),pid1,pid2,getppid());*

*sleep();*

*}*

*}*

**（4）改动后运行结果：**



**（5）结果分析：**

可见b与c的parentpid都是4401，即父程序都是4401。

第一次产生的4402pid1与父程序的 pid1=0，没有pid2。

第二次产生的4403pid1与父程序的pid1相同而代表第二轮的pid2为0。

从而清楚地得知：

打印b的子程序pid为4402是第一次fork产生的。

而打印c的子程序pid为4403，是条件语句内第二次fork产生的。

有一点值得注意的是，如果把pid2=fork（）语句写在else条件外，答案会出错，会得到两个c。

# 实验体会：

通过查阅资料了解到fork函数的原理与调用方法，再通过自己敲代码不断巩固，已经熟知了fork函数，父进程，子进程的基础原理，明白了会画树状图辅助理解的重要性，还有思维上的变通，才让我想清楚这一道题的难点，除了父生子，子也可当父这个难点外，其中一个难点在于，子程序大体上等同于父程序，所以pid1，pid2都有会相同或不同之处。关于更深奥的进程原理还要继续钻研学习。第二道题通过编译运行，了解到细微的不同也会导致题目的错误。此外就是vim指令运用的不熟练，在敲代码上消耗了一定时间。还有就是sleep（）函数作为调用getppid（）用于辅助理解，需要了解cpu抢占，还有阻塞相关知识。