

**学生实验报告**

实验课名称：操作系统

实验项目名称：进程的创建、撤销和控制

专业名称：计算机科学与技术（交通信息工程）

班级：2016240203

学号：2016902094

学生姓名：瞿强鑫

教师姓名：刘晓春

2019**年** 5 **月** 22**日**

1. 实验名称

进程的创建、撤销和控制

1. 实验内容

1.进程的创建

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‘c’。试观察记录屏幕上的显示结果，并分析原因。

2.进程的控制

修改已编写的程序，将每个进程输出一个字符改为每个进程输出一句话，在观察晨旭执行时屏幕上出现的现象，并分析原因。

如果在程序中使用系统调用lockf()来给每一个进程加锁，可以实现进程间的互斥，观察并分析出现的现象。

3.进程的软中断通信。

要求：使用系统调用fork()创建两个子进程，再用系统调用signal()让父进程捕捉到中断信号后，父进程在用系统调用kill()向两个子进程发出信号，子进程捕捉到信号后分别输出下列信息后终止：

Child Process1 id Killed by Parent!

Child Process2 is Killed by Parent!

父进程等待两个子进程终止后，输出如下的信息后终止：

Parent Process is Killed!

在上面的程序中增加语句signal(SIGINT,SIG\_IGN)和signal(SIGQUIT,SIG\_IGN)，观察执行结果，并分析原因。

1. 实验原理和设计思路

本次实验以实验指导书为指导，按照流程学习，加深对进程概念的理解，明确进程和程序的区别，进一步认识并发执行的实质，分析进程争用资源的现象，学习解决进程互斥的方法，了解Linux系统中进程通信的基本原理。

1. 源代码

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

#include<signal.h>

int mark=1;

int process1()

{

int pid1,pid2;

pid1=fork();

if(pid1==-1)

printf("error!!!\n");

if(pid1 ==0)

printf("b");

else //father

{

pid2=fork();

if(pid2==-1)

printf("error!!!\n");

if(pid2 ==0)

printf("c");

else

printf("a");

}

exit(0);

}

int process2()

{

int pid1,pid2;

pid1=fork();

if(pid1==-1)

printf("error!!!\n");

if(pid1 ==0)

printf("I am first child:%d\n",getpid());

else //father

{

pid2=fork();

if(pid2==-1)

printf("error!!!\n");

if(pid2 ==0)

printf("I am second child:%d\n",getpid());

else

printf("I am parent:%d\n",getpid());

}

exit(0);

}

void process3()

{

int p1,p2;

p1=fork();

if(p1==0)

{

for(int i=0;i<5;++i)

{

lockf(1,1,0);

printf("I am first child:%d\n",getpid());

lockf(1,0,0);

sleep(1);

}

}

else

{

p2=fork();

if(p2==0)

{

for(int j=0;j<5;++j)

{

lockf(1,1,0);

printf("I am second child:%d\n",getpid());

lockf(1,0,0);

sleep(1);

}

}

else

{

lockf(1,1,0);

printf("I am parent:%d\n",getpid());

lockf(1,0,0);

}

}

}

int process4()

{

int p1,p2;

p1=fork();

if(p1==0)

{

lockf(1,1,0);

for(int i=0;i<5;++i)

{

printf("I am first child:%d\n",getpid());

sleep(1);

}

lockf(1,0,0);

}

else

{

p2=fork();

if(p2==0)

{

lockf(1,1,0);

for(int j=0;j<5;++j)

{

printf("I am second child:%d\n",getpid());

sleep(1);

}

lockf(1,0,0);

}

else

{

lockf(1,1,0);

printf("I am parent:%d\n",getpid());

lockf(1,0,0);

}

}

return 0;

}

void stop()

{

mark=0;

}

int process5()

{

int pid1,pid2,a;

pid1=fork();

if(pid1==-1)

printf("error!!!\n");

if(pid1 ==0)

{

signal(16,stop);

signal(SIGINT,SIG\_IGN);

//signal(SIGQUIT,SIG\_IGN);

while(mark);

printf("Child Process1 is killed by Parent！\n");

wait(&a);

exit(0);

}

else //father

{

pid2=fork();

if(pid2==-1)

printf("error!!!\n");

if(pid2 ==0)

{

signal(17,stop);

signal(SIGINT,SIG\_IGN);

//signal(SIGQUIT,SIG\_IGN);

while(mark);

printf("Child Process2 is killed by Parent！\n");

wait(&a);

exit(0);

}

else

{

signal(SIGINT,stop);

while(mark);

kill(pid1,16);

kill(pid2,17);

wait(&a);

sleep(1);

printf("Parent Process is killed！\n");

exit(0);

}

}

exit(0);

}

int main()

{

int choice;

printf("1.输出字符2.输出字符串3.lockf里面4.lockf外面5.信号\n");

scanf("%d",&choice);

switch(choice)

{

case 1:

process1();

break;

case 2:

process2();

break;

case 3:

process3();

break;

case 4:

process4();

break;

case 5:

process5();

break;

}

exit(0);

}

1. 结果截图

1.输出三个字符

如图所示，创建了两个子进程，父进程输出a，两个子进程输出bc我运行了好多次都是这个顺序。

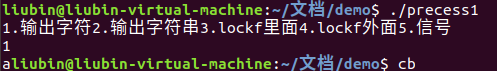


图1.输出字符

2.输出三句话，如图所示，和输出字符差不多，顺序也一样。

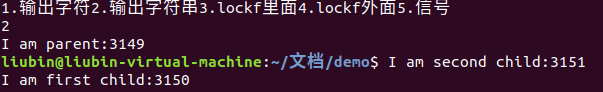


图2.输出语句

3.加lockf()函数对比

3.1 lockf加在循环打印中间，输出设备就在循环中被锁定，所以输出的信息是两个进程轮换输出的，如图所示。

for(int i=0;i<5;++i)

{

lockf(1,1,0);

printf("I am first child:%d\n",getpid());

lockf(1,0,0);

sleep(1);

}

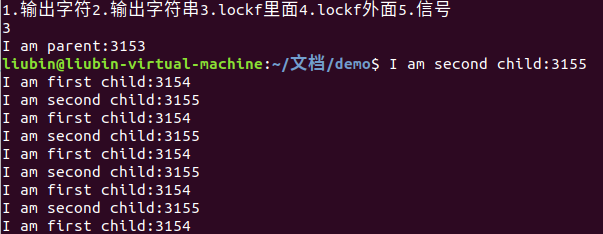


图3. lockf循环内

3.2 lockf加在循环打印外边，整个子进程中输出设备都被锁定，所以一个子进程中的内容先输出，再输出另一个子进程要用到输出设备的内容，如图所示。

lockf(1,1,0);

for(int i=0;i<5;++i)

{

printf("I am first child:%d\n",getpid());

sleep(1);

}

lockf(1,0,0);

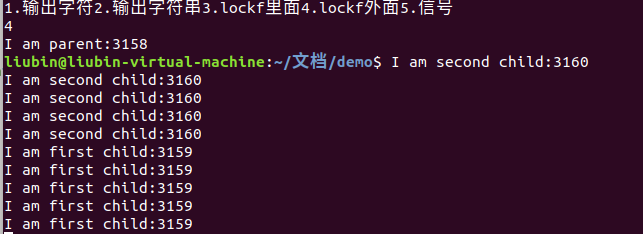


图4. lockf循环外

4.实现软中断

4.1先运行程序，利用终端向父进程发信号，如图所示，父进程接收信号后调用了stop()函数，我在向两个子进程发信号前加了一个死循环，while（mark），刚开始mark为1，执行stop()使mark=0，然后执行两个kill()函数，分别向两个子进程发信号，子进程接受后，打印信息并退出。等两个子程序结束后，父进程也退出,如图所示。

signal(SIGINT,stop);

while(mark);

kill(pid1,16);

kill(pid2,17);

signal(16,stop);

while(mark);

waitpid(pid1,&a,0);

waitpid(pid2,&a,0);

printf("Child Process1 is killed by Parent！\n");

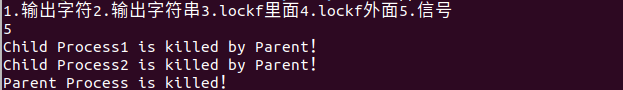


图5.软中断

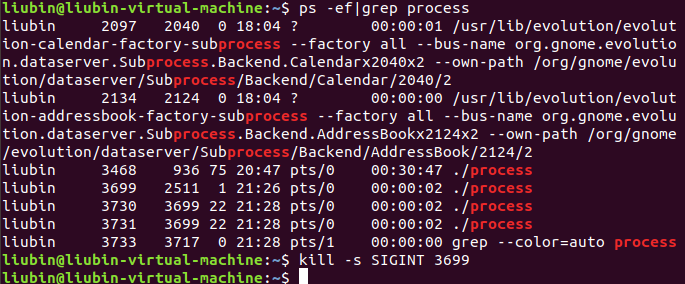


图6.发信号

4.2加上 signal(SIGINT,SIG\_IGN);后屏蔽了SIGINT信号，进程不能收到信号，一直处于运行状态。

signal(SIGINT,stop);

signal(SIGINT,SIG\_IGN);

signal(SIGQUIT,SIG\_IGN);

while(mark);

kill(pid1,16);

kill(pid2,17);

waitpid(pid1,&a,0);

waitpid(pid2,&a,0);

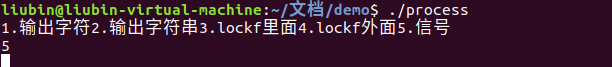


图7.加了 signal(SIGINT,SIG\_IGN)后

1. 编程体会

实验主要是linux环境出了问题，我一直没能装好环境，虚拟机安装好后在装镜像文件时老报错百度很久之后无果，实在不行只好用舍友的电脑来完成此次实验。

后来写程序对fork()函数理解不到位，后来在网上看到一篇好的文章才有了点理解，fork()就是创建了一个子进程和父进程的内容一样，看返回值0为子进程，大于1为父进程的id，小于0就是失败了。

第二部分时，开始时对lockf（）函数对参数不太理解，第一个参数文件描述符，为1对标准输出加锁，为0对标准输出加锁。

第三部分，运行程序后，按键盘没反应后，看网上按了ctrl+c后才有反应但不是想要的结果。后来用终端输入命令来发送SIGINT，才得到想要的结果。