山东大学 计算机科学与技术 学院

操作系统 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201900161140 | 姓名：张文浩 | | 班级： 19人工智能 |
| 实验题目：进程控制实验 | | | |
| 实验学时：2 | | 实验日期： 4.9 | |
| 实验目的：  加深对于进程并发执行概念的理解。实践并发进/线程的创建和控制方法。观 察和体验进程的动态特性。进一步理解进程生命期期间创建、变换、撤销状态变换 的过程。掌握进程控制的方法，了解父子进程间的控制和协作关系。练习 Linux 系 统中进/线程创建与控制有关的系统调用的编程和调试技术。 | | | |
| 硬件环境： 自带的笔记本电脑windows | | | |
| 软件环境：Vmware的ubuntu虚拟机 | | | |
| 实验步骤与内容：   1. 先根据实验指导书完成实例的程序案例进行学习。 2. 进行独立实验： 3. 参考以上示例程序中建立并发进程的方法，编写一个多进程并发执行程序。父进程每隔 3 秒重复建立两个子进程，首先创建的让其执行 ls 命令，之后创建执行让其执行 ps 命令，并控制 ps 命令总在 ls 命令之前执行。 4. 先建立头文件，头文件中包含一个进程自定义的键盘中断信号处理函数代码如下： 5. 通过代码调用头文件，代码具体如下： 6. 新建mypctl.c文件编写程序，在main函数中先自定义申请一个中断函数，储存两个要用的进程： 7. 先创建ls的进程，在子进程中先使用pause（）等待父进程的kill，然后在子进程中再创建一个子进程防止被覆盖掉，在新建的子进程中装入ls进程： 8. 在父进程中创建ps的子进程，父进程要先等待子进程运行结束，使ps命令的进程装入后使用kill使ls的子进程启动。最后使用sleep休眠3秒。 9. 创建makefile     10.最后使用gmake，./mypctl运行，运行结果如下： | | | |
| 结论分析与体会：  在本次试验中，反映出进程在创建新进程时的情况，体现了父进程等待直到某个或全部子进程执行完，也体现了新进程的地址空间的两种可能，分别是子进程是父进程的复制品和子进程装入另一个新程序。因为之前没有接触过关于进程的命令编程，所以这次实验一开始入到了很多困难，但通过反复阅读实验指导书，在网上找了很多博文一点点学习，最终尝试了许多次，终于成功。通过这次实验，收获了很多，看了许多相关博客，对于fork()和父子进程的关系的认识更加清晰。 | | | |

附源代码

#include"mypctl.h"

int main(int argc, char \*argv[]){

int status1,status2;

signal(SIGINT,(sighandler\_t)sigcat);

char\* args1[] = {"/bin/ls","-a",NULL};

char\* args2[] = {"/bin/ps","-a",NULL};

int pid1 = fork();

int count = 0;

if(pid1 < 0){

printf("创建进程失败\n");

}

if(pid1 == 0){

while(1)

{

pause();

printf("我是子进程%d,我的父进程是%d\n",getpid(),getppid());

printf("开始执行任务：ls\n");

int pid = fork();

if(pid < 0){

printf("创建进程失败\n");

}

if(pid == 0){

printf("ls %d号进程启动\n",getpid());

status1 = execve(args1[0],args1,NULL);//装入执行程序

}

}

}

if(pid1 > 0){

while(1)

{

printf("\n我是父进程%d, 现在是第%d次执行\n",getpid(),count);

printf("开始执行任务： ps\n");

int pid2 = fork();

if(pid2 > 0){

waitpid(pid2,&status2,0);//等待pid2结束

kill(pid1,SIGINT);//运行pid1

sleep(3);

printf("第%d次执行结束，休眠3秒\n",count++);

printf("\n");

}

if(pid2 < 0){

printf("进城创建失败\n");

}

if(pid2 == 0){

printf("ps %d号进程启动\n",getpid());

status2 = execve(args2[0],args2,NULL);

}

}

}

return EXIT\_SUCCESS;

}