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：201705130120 | 姓名：苑宗鹤 | | 班级： 1班 |
| 实验题目： 进程控制实验 | | | |
| 实验学时：2 | | 实验日期：2019/12/22 | |
| 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：  pc | | | |
| 软件环境：  Win10 clion  Linux ubuntu | | | |
| 实验内容与设计：  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。  实验代码：  #include**<unistd.h>** #include **<stdio.h>** #include**<sys/wait.h>** #include**<stdlib.h>** pid\_t pidarray[2]; **int** pslock = 0;  **typedef void** (\*sighandler\_t)(**int**);  **void** sigcat() { *// printf("%d Process continue\n", getpid());* }  **int** main() {  signal(**SIGINT**, (sighandler\_t) sigcat);  **while** (**true**) {  pidarray[0] = fork();  **if** (pidarray[0] > 0) { *//main thread* pidarray[1] = fork(); *//subthread2* }   **if** (pidarray[0] == 0) {*//subthread1* pause();  system(**"ls"**);  **break**;  } **else if** (pidarray[1] == 0) { *//subthread2* system(**"ps"**);  **break**;  } **else** {*//mainthread* **int** status;  waitpid(pidarray[1], &status, 0);   kill(pidarray[0], **SIGINT**);  waitpid(pidarray[0], &status, 0);   }  sleep(3);  }    **return** 0; }  运行结果： | | | |
| 结论分析与体会：  通过先pause ls进程 然后在ps进程完全执行结束后给ls进程发SIGINT信号恢复ls进程 执行ls  如果因为系统调度原因在主进程发SIGINT信号后 ls进程才进入pause则会导致ls进程无限等待  解决方法:在主进程中适当sleep | | | |