山东大学 软件 学院

**操作系统** 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100300157 | 姓名：孙荣骏 | | 班级：21.3 |
| 实验编号：实验3 | | | |
| 实验题目：进程调度算法实验 | | | |
| 实验学时：2 | | 实验日期：2022.4.28 | |
| 实验目的：  加深对进程调度概念的理解，体验进程调度机制的功能，了解Linux系统中进程  调度策略的使用方法。 练习进程调度算法的编程和调试技术 | | | |
| 硬件环境：  宿主机  机型：联想拯救者r7000p2021  CPU：AMD R7 5800H  内存：16G  虚拟机  RAM：4GB | | | |
| 软件环境：  虚拟机：Ubuntu 16.04  宿主机：win10 | | | |
| 实验步骤与内容：  **独立实验：**      主要实验代码及注释如下：    #include <stdio.h>  #include <stdlib.h>  #include <sched.h>  #include <sys/time.h>  #include <sys/resource.h>  #include "os3.h"  int main(int argc, char \*argv[])  {  int pidParent, pidChild; // 父进程和子进程的进程号    signal(SIGINT, (sighandler\_t)sigint); // SIGINT信号，优先级加１  signal(SIGTSTP, (sighandler\_t)sigtstp); // SIGTSTP信号，优先级减１    pidParent = getpid();  pidChild = fork(); // 创建子进程    // 父子进程循环输出  if (pidChild < 0)  {  printf("子进程创建失败！\n");  exit(EXIT\_FAILURE);  }  else if (pidChild > 0)  {  // 父进程输出  for (int i = 0; i < 10; i++)  {  printf("我是父进程%d，策略为%d，优先级为%d\n",  pidParent, sched\_getscheduler(pidParent), getpriority(PRIO\_PROCESS, 0));  sleep(4);  }  }  else  {  // 子进程输出  pidChild = getpid();  for (int i = 0; i < 10; i++)  {  printf("我是子进程%d，策略为%d，优先级为%d\n",  getpid(), sched\_getscheduler(getpid()), getpriority(PRIO\_PROCESS, 0));  sleep(4);  }  }    return EXIT\_SUCCESS;  }  #ifndef EXP3\_H\_INCLUDED  #define EXP3\_H\_INCLUDED  #include <sys/types.h>  #include <wait.h>  #include <unistd.h>  #include <signal.h>  #include <stdio.h>  #include <stdlib.h>  // 进程自定义的时间信号  typedef void (\*sighandler\_t)(int);  void sigint()  {  // 进程收到SIGINT信号（在控制台输入Ctrl+C），优先数加１  int pri = 0;  pri = getpriority(PRIO\_PROCESS, 0); // 获取当前进程的优先级  if (pri >= -20 && pri < 19)  pri += 1;  printf("当前进程为%d，优先级+1\n", getpid());  setpriority(PRIO\_PROCESS, getpid(), pri);  }  void sigtstp()  {  // 进程收到SIGTSTP信号（在控制台输入Ctrl+Z），优先数减１  int pri = 0;  pri = getpriority(PRIO\_PROCESS, 0); //　获取当前进程的优先级  if (pri > -20 && pri <= 19)  pri -= 1;  printf("当前进程为%d，优先级-1\n", getpid());  setpriority(PRIO\_PROCESS, getpid(), pri);  }  #endif // EXP3\_H\_INCLUDED  os1: os3.o  gcc os3.o -o os3  os1.o: os3.c os3.h  gcc -g -c os3.c  .PHONY: clean  clean:  rm os3 \*.o | | | |
| 结论分析与体会：  在本次试验中，我加深了对于进程调度算法的理解 ，并且熟悉了对进程优先级的调整。  通过复习课本，我巩固了对进程调度算法的理解  进程调度算法包括先来先服务，时间片轮转，多级反馈轮转法，优先级法，短作业优先法，最高响应比优先法。  我还了解了进程优先级减少的操作需要在根环境下运行。 | | | |