实验4 进程调度

1. 实验目的
2. 通过实验加深对操作系统中进程调度的理解；
3. 掌握常用进程调度算法的具体实现。
4. 预备知识
5. Linux 中有一个总的调度结构，称之为调度器类（scheduler class）

* 允许不同的可动态添加的调度算法并存，总调度器根据调度器类的优先顺序，依次挑选调度器类中的进程进行调度。
* 确定调度器类后，再使用该调度器类的调度算法（调度策略）进行内部调度。
* 调度器类的优先级顺序为：

Stop\_Task > Real\_Time > Fair > Idle\_Task

其中，Fair和Real\_time最常用，分别采用CFS（完全公平调度算法）调度算法的默认调度类和实时调度类。

1. 实验内容
2. 编写C程序模拟实现单处理机系统中的进程调度算法，实现对多个进程的调度模拟，要求采用常见进程调度算法（如先来先服务、时间片轮转和优先级调度等算法）进行模拟调度。
3. **数据结构设计**

PCB：结构体

就绪队列：链表，每个节点为进程PCB

进程状态

1. **调度算法设计**

具体调度算法：FCFS、SJF、PR

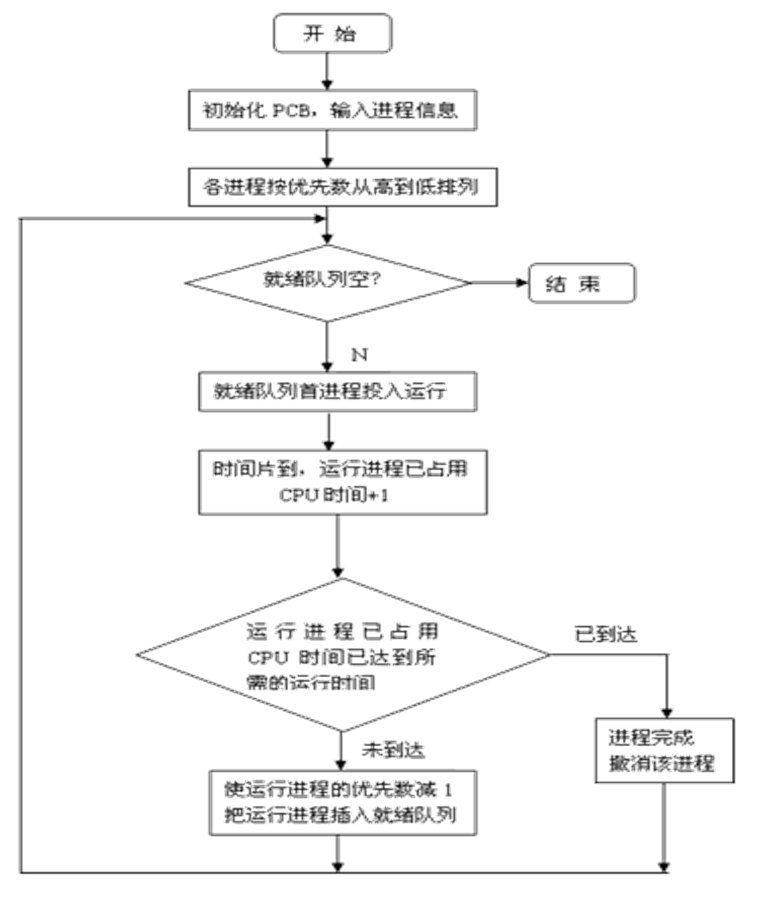
涉及多种操作：排序、链表操作

1. **程序输出设计**

调度进程的顺序、每个进程的起始时间、终止时间等

CPU每次调度的过程

1. 理解下面基于动态优先数的进程调度算法（提供示例代码），并思考如何改为固定优先数并编程实现。
2. 优先数大者优先，且优先数每运行一个时间单位降低一级（即优先数=优先数-1）
3. 进程的优先数及需要的运行时间事先人为地指定，以1个CPU时间单位进行计算
4. 进程状态：W（就绪态）、R（运行态）、F（完成态）
5. 输出：每进行一次调度程序都输出一次运行进程和就绪队列中的所有进程信息。



#include "stdio.h"

#include <stdlib.h>

#define getpch(type) (type\*)malloc(sizeof(type))

struct pcb { /\* 定义进程控制块PCB \*/

char name[10]; //进程名

char state; //进程状态："W"-就绪态，"R"-运行态

int nice; //进程优先级

int ntime; //需要运行时间

int rtime; //已经运行的时间

struct pcb\* link;

}\*ready=NULL,\*p;

typedef struct pcb PCB;

char sort() /\* 建立对进程进行优先级排列函数，优先数大者优先\*/

{

PCB \*first, \*second;

int insert=0;

if((ready==NULL)||((p->nice)>(ready->nice)))/\*优先级最大者,插入队首\*/

{

p->link=ready;

ready=p;

}

else /\* 进程比较优先级,插入适当的位置中\*/

{

first=ready;

second=first->link;

while(second!=NULL)

{

if((p->nice)>(second->nice)) /\*若插入进程比当前进程优先数大,\*/

{ /\*插入到当前进程前面\*/

p->link=second;

first->link=p;

second=NULL;

insert=1;

}

else /\* 插入进程优先数最低,则插入到队尾\*/

{

first=first->link;

second=second->link;

}

}

if(insert==0) first->link=p;

}

}

char input() /\* 建立进程控制块函数\*/

{

int i,num;

printf("\n 请输入被调度的进程数目：");

scanf("%d",&num);

for(i=0;i<num;i++)

{

printf("\n 进程号No.%d:",i);

p=getpch(PCB);

printf("\n 输入进程名:");

scanf("%s",p->name);

printf(" 输入进程优先数:");

scanf("%d",&p->nice);

printf(" 输入进程运行时间:");

scanf("%d",&p->ntime);

printf("\n");

p->rtime=0;

p->state='W';

p->link=NULL;

sort(); /\* 调用sort函数\*/

}

}

int space()

{

int l=0; PCB\* pr=ready;

while(pr!=NULL)

{

l++;

pr=pr->link;

}

return(l);

}

char disp(PCB \* pr) /\*建立进程显示函数,用于显示当前进程\*/

{

printf("\n qname \t state \t nice \tndtime\truntime \n");

printf("%s\t",pr->name);

printf("%c\t",pr->state);

printf("%d\t",pr->nice);

printf("%d\t",pr->ntime);

printf("%d\t",pr->rtime);

printf("\n");

}

char check() /\* 建立进程查看函数 \*/

{

PCB\* pr;

printf("\n \*\*\*\* 当前正在运行的进程是:%s",p->name); /\*显示当前运行进程\*/

disp(p);

pr=ready;

if (pr!=NULL)

printf("\n \*\*\*\*当前就绪队列状态为:"); /\*显示就绪队列状态\*/

else

printf("\n \*\*\*\*当前就绪队列状态为: 空\n"); /\*显示就绪队列状态为空\*/

while(pr!=NULL)

{

disp(pr);

pr=pr->link;

}

}

char destroy() /\*建立进程撤消函数(进程运行结束,撤消进程)\*/

{

printf(" 进程 [%s] 已完成.\n",p->name);

free(p);

}

char running() /\* 建立进程就绪函数(进程运行时间到,置就绪状态\*/

{

(p->rtime)++;

if(p->rtime==p->ntime)

destroy(); /\* 调用destroy函数\*/

else

{

(p->nice)--;

p->state='W';

sort(); /\*调用sort函数\*/

}

}

void main() /\*主函数\*/

{

int len,h=0;

char ch;

input();

len=space();

while((len!=0)&&(ready!=NULL))

{

ch=getchar();

h++;

printf("\n The execute number:%d \n",h);

p=ready;

ready=p->link;

p->link=NULL;

p->state='R';

check();

running();

printf("\n按任一键继续......");

ch=getchar();

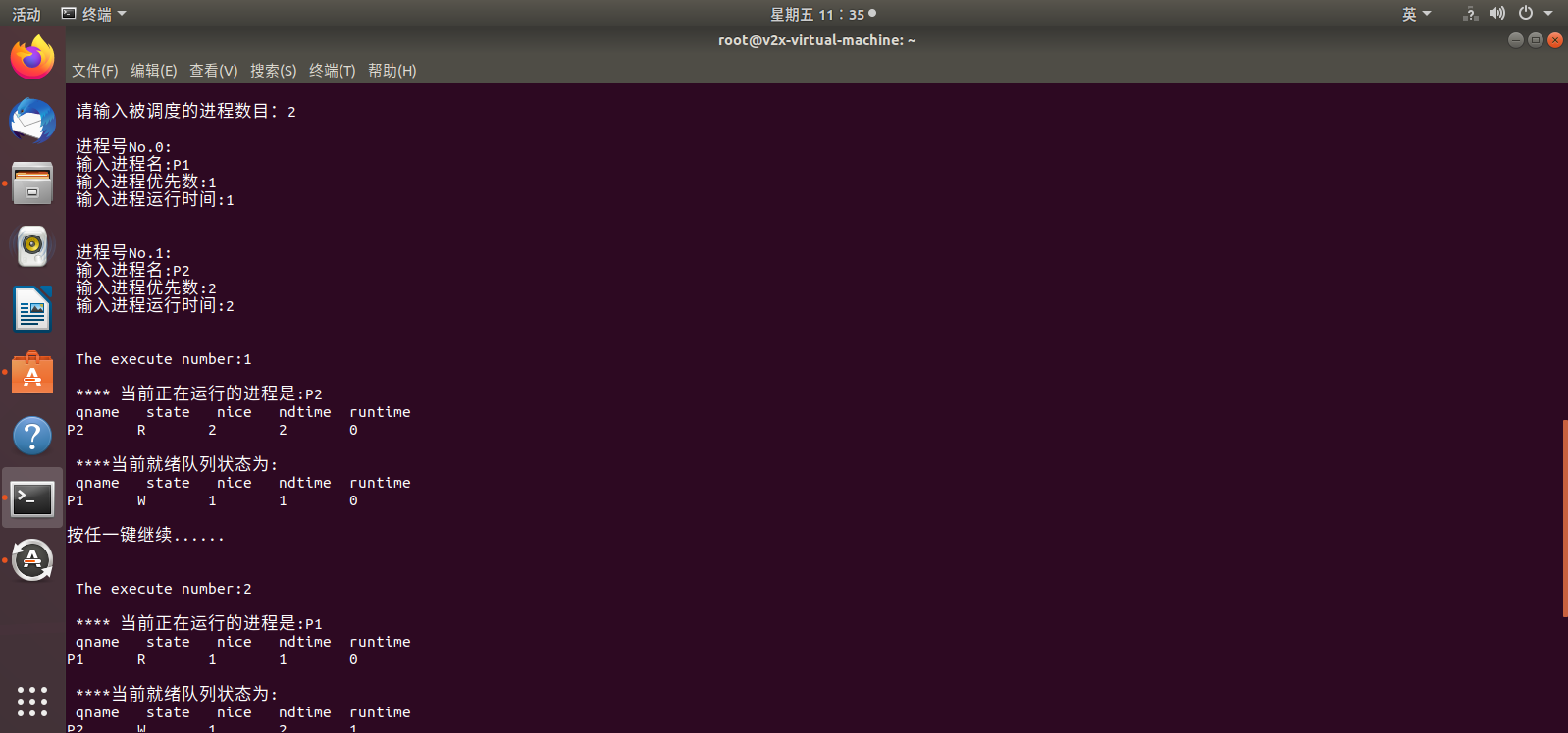
}

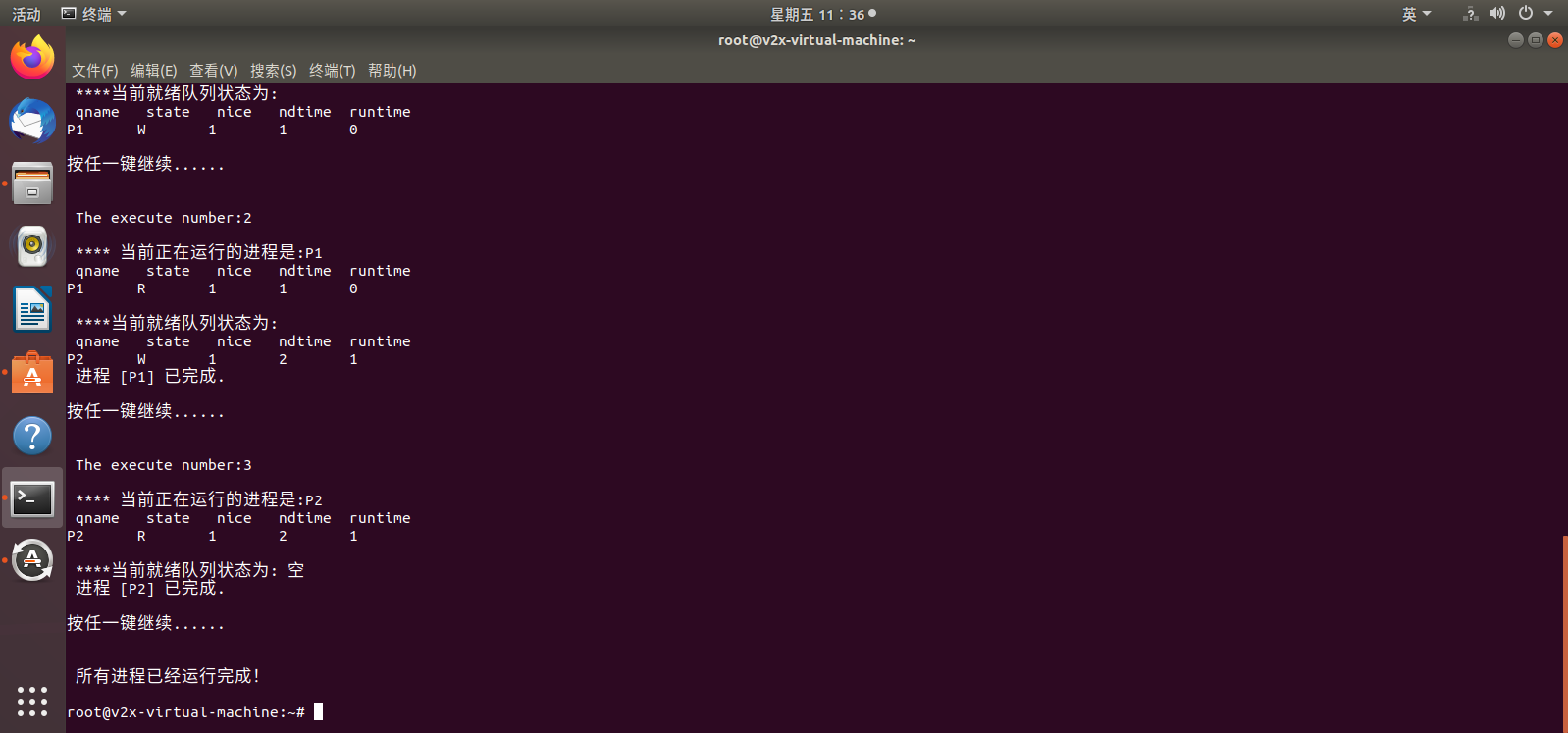
printf("\n\n 所有进程已经运行完成！\n");

ch=getchar();

}

运行输出结果演示如下：





1. 实验习题
2. 编写C程序模拟实现单处理机系统中的进程调度算法，实现对多个进程的调度模拟，要求采用常见进程调度算法（先来先服务、短作业优先、时间片轮转、优先级调度、最高响应比优先等算法）选择其中一个进行模拟调度。
3. 实验报告

将调试运行成功的程序和写好的实验报告一起压缩打包，以实验X-学号-姓名.rar这样的形式命名，并上传提交。