课程名称 计算机操作系统

实验成绩

指导教师 曹勇



实 验 报 告

院系 信息工程学院

班级 物联网工程

学号 1601141019

姓名 晏沈威

日期 2018.10.10 地点：505

**­** 实验一、进程调度试验

[目的要求]

用高级语言编写和调试一个进程调度程序，以加深对进程的概念及进程调度算法的理解．

[准备知识]

一、基本概念

1、进程的概念；

2、进程的状态和进程控制块；

3、进程调度算法；

二、进程调度

1、进程的状态

2、进程的结构——PCB 进程都是由一系列操作(动作)所组成，通过这些操作来完成其任务。因此，不同的进程，其内部操作也不相同。在操作系统中，描述一个进程除了需要程序和私有数据之外，最主要的是需要一个与动态过程相联系的数据结构，该数据结构用来描述进程的外部特性(名字、状态等)以及与其它进程的联系(通信关系)等信息，该数据结构称为进控制块(PCB，Process Control Block)。进程控制块PCB与进程一一对应，PCB中记录了系统所需的全部信息、用于描述进程情况所需的全部信息和控制进程运行所需的全部信息。

因此，系统可以通过进程的PCB来对进程进行管理。

[试验内容]

设计一个有N个进程共行的进程调度程序。进程调度算法：采用最高优先数优先的调度算法（即把处理机分配给优先数最高的进程）和先来先服务算法。每个进程有一个进程控制块（PCB）表示。进程控制块可以包含如下信息：进程名、优先数、到达时间、需要运行时间、已用CPU时间、进程状态等等。 进程的优先数及需要的运行时间可以事先人为地指定（也可以由随机数产生）。进程的到达时间为进程输入的时间。

进程的运行时间以时间片为单位进行计算。

每个进程的状

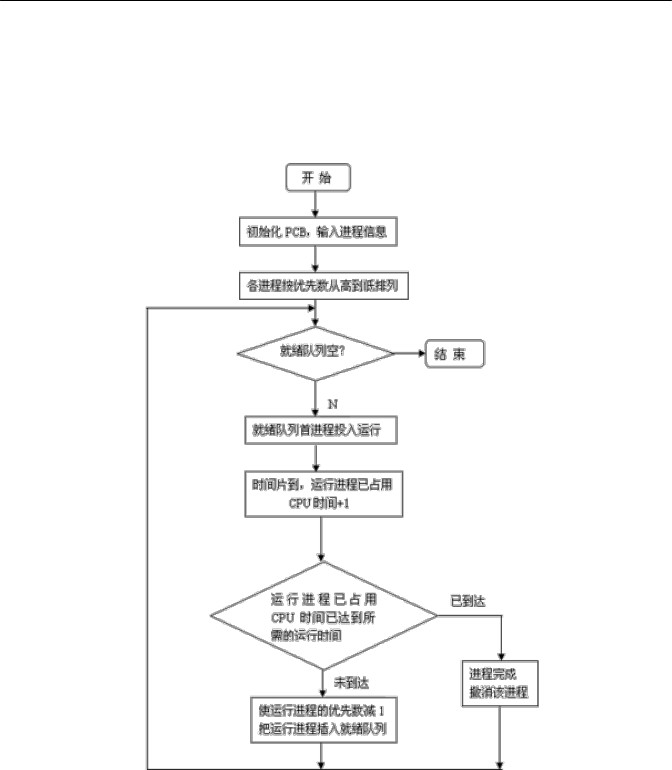
态可以是就绪

W（Wait）、运行R（Run）、或完成F（Finish）三种状态之一。就绪进程获得CPU后都只能运行一个时间片。用已占用CPU时间加1来表示。如果运行一个时间片后，进程的已占用CPU时间已达到所需要的运行时间，则撤消该进程，如果运行一运行就绪阻塞进程因某事件（如等待I/O完成）变成阻塞状态某事件被解除（I/O完成）时间片已用完进程调度程序把处理机分配给进程

计算机操作系统教程（第三版）

个时间片后进程的已占用

CPU时间还未达所需要的运行时间，也就是进程还需要继续运行，此时应将进程的优先数减1（即降低一级），然后把它插入就绪队列等待CPU。每进行一次调度程序都打印一次运行进程、就绪队列、以及各个进程的PCB，以便进行检查。重复以上过程，直到所要进程都完成为止。调度算法的流程图如下:



进程调度源程序如下：

#include "stdio.h"

#include <stdlib.h>

#include <conio.h>

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

#define NULL 0

struct pcb { /\* 定义进程控制块PCB \*/

char name[10];

char state; /\*状态\*/

int super; /\*优先级\*/

int ntime;

int rtime;

struct pcb\* link; /\*结构体pcb的指针类型\*/

}\*ready=NULL,\*p;/\*指向当前用行的pcb\*/

typedef struct pcb PCB;

void sort() /\* 建立对进程进行优先级排列函数\*/

{

PCB \*first, \*second;

int insert=0;

if((ready==NULL)||((p->super)>(ready->super))) /\*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->super)>(second->super)) /\*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;

}

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

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

printf("\n");

}

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;

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

while(pr!=NULL)

{

disp(pr);

pr=pr->link;

}

}

destroy() /\*建立进程撤消函数(进程运行结束,撤消进程)\*/

{

printf("\n 进程[%s] 已完成.\n",p->name);

free(p);

}

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->super)--;

p->state='w';

sort(); /\*调用sort函数\*/

}

}

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 进程已经完成.\n");

ch=getchar();

}

运行结果：

