**操作系统** 实 验 预 习 报 告

专业 **软件工程**  班级 **软件171**  姓名 **韩玉娟**  学号 **2017021054**

|  |  |
| --- | --- |
| 实验名称 | 实验1 单处理机系统进程调度 |
| 实验目的：  在多道程序或多任务系统中，系统同时处于就绪态的进程有若干个，为使系统中的各进程能有条不紊地运行，必须选择某种调度策略，以选择一进程占用处理机。要求学生设计一个模拟单处理机调度算法，以巩固和加深对进程概念和进程调度算法的理解。 | |
| 实验环境：  计算机一台、Dev C++ | |
| 实验内容及方法：  注：内部字号均用小四宋体，英文用times new roman字体，汉字行距1.25，英文可固定值15磅  编写程序完成单处理机系统中的进程调度，要求采用最高优先级优先调度算法。  具体内容包括：  1、确定进程控制块的内容和组织方式；  2、完成进程创建原语和进程调度原语；  3、编写主函数对所做的工作进行测试。 | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| （1）相关知识及其分析  进程调度算法：采用最高优先级优先的调度算法（即把处理机分配给优先级最高的进程）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。进程的优先数（表示优先级别）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，如果运行一个时间片后进程的已占用CPU时间还未到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，直到所有进程都完成为止。 | | | | | |
| 参考资料：   1. 汤小丹主编.操作系统（第4版）,西安电子科技大学出版社. 2. 课程平台学习资料和网络学习资料. | | | | | |
| 实验预习报告评分标准 | | | | | |
| 评分项目 | 满分 | 得分 | 评分项目 | 满分 | 得分 |
| 实验项目 | 1 | 1 | 内容与方法 | 2 | 2 |
| 参考资料 | 0.5 | 0.5 | 书写规范 | 0.5 | 0.5 |
| 总分 | 4 | | 指导教师签字：杜晓昕 2020年 6月10 日 | | |

齐齐哈尔大学计算机与控制工程学院

实 验 报 告

**课程名称： 操作系统（实验）**

**实验名称： 实验1 单处理机系统进程调度**

**专业班级 ：­­­­­­ 软件171**

**姓 名：­ 韩玉娟**

**学 号： 2017021054**

**指导教师： 杜晓昕**

**实验地点： 家 + 钉钉课堂 + 课程平台**

**完成日期： 2020.06.12**

实 验 报 告

|  |  |
| --- | --- |
| 实验  名称 | 实验1 单处理机系统进程调度 |
| 实验  目的  及要  求 | 在多道程序或多任务系统中，系统同时处于就绪态的进程有若干个，为使系统中的各进程能有条不紊地运行，必须选择某种调度策略，以选择一进程占用处理机。要求学生设计一个模拟单处理机调度算法，以巩固和加深对进程概念和进程调度算法的理解。 |
| 实验  环境 | （列出本次实验所使用的平台和相关软、硬件）  计算机一台、Dev C++ |
| 实  验  内  容 | 编写程序完成单处理机系统中的进程调度，要求采用最高优先级优先调度算法。  具体内容包括：  1、确定进程控制块的内容和组织方式；  2、完成进程创建原语和进程调度原语；  3、编写主函数对所做的工作进行测试。  （1）相关知识及其分析  进程调度算法：采用最高优先级优先的调度算法（即把处理机分配给优先级最高的进程）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。进程的优先数（表示优先级别）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。 |

|  |  |
| --- | --- |
| 实  验  原  理  及  实  验  步  骤 | （用适当的形式表达本次实验的设计思想及实现步骤，如算法、流程图、原理图、程序代码清单等）  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，如果运行一个时间片后进程的已占用CPU时间还未到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，直到所有进程都完成为止。    图 1 进程调度算法流程图 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 调  试  过  程  及  实  验  结  果 | （详细记录在调试过程中出现的问题及解决方法，记录实验执行结果）  IMG_256IMG_256IMG_256 | | | | | |
| 总  结 | （对实验结果进行分析，问题回答，实验心得体会及改进意见）  通过本次实验，我了解到进程调度算法：采用最高优先级优先的调度算法（即把处理机分配给优先级最高的进程）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。 | | | | | |
| 实验报告评分标准 | | | | | | |
| 评分项目 | | 满分 | 得分 | 评分项目 | 满分 | 得分 |
| 预习与实验准备 | | 4 | 4 | 实验过程 | 4 | 4 |
| 书写规范程度 | | 2 | 2 | 实验完成情况 | 4 | 4 |
| 实验态度 | | 2 | 2 | 总结 | 4 | 4 |
| 总分 | | 20 | | 指导教师签字：杜晓昕 2020 年 6 月 15 日 | | |
| 附录 | | （源程序清单等） | | | | |

附录：（程序清单等）如果程序代码很多，可以只挑选每个程序的一段重要代码放在下面，并做说明。

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <windows.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;

int super;

int ntime;

int rtime;

struct pcb\* link;

}\*ready=NULL,\*p;

typedef struct pcb PCB;

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;

}

}

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

}

}

//说明：上述程序首先定义进程控制块结构，然后对每个进程的优先级赋初值，完成优先级的排序过成，把优先级大的进程插入就绪队首，等待CPU调度。该实验完成了优先级大者优先调度的原则，对进程调度过程加深理解和掌握

input() /\* 建立进程控制块函数\*/

{

int i,num;

//clrscr(); /\*清屏\*/

system("CLS");

printf("\n 请输入进程号:");

scanf("%d",&num);

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

{

printf("\n 进程号No.%d:\n",i);

p=getpch(PCB);

printf("\n 输入进程名:");

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

printf("\n 输入进程优先数:");

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

printf("\n 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);

}

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 super \t ndtime \t runtime \n");

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

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

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

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();

}





