**实验三 处理机调度算法模拟程序设计**

**一、实验目的要求**

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。

**二、实验内容**

**1、设计简单的进程PCB 结构，完成进程信息记录；**

**2、设计一个有 N个进程并发执行的进程调度程序（以下四个至少选一个实现）**

1）编写并调试一个模拟的进程调度程序，采用“最高优先数优先”调度算法对五个进程进行调度。

“最高优先数优先”调度算法的基本思想是把CPU分配给就绪队列中优先数最高的进程。

静态优先数是在创建进程时确定的，并在整个进程运行期间不再改变。

动态优先数是指进程的优先数在创建进程时可以给定一个初始值，并且可以按一定原则修改优先数。例如：在进程获得一次CPU后就将其优先数减少1。或者进程等待的时间超过某一时限时增加其优先数的值，等等。

2）编写并调试一个模拟的进程调度程序，采用“轮转法”调度算法对五个进程进行调度。

轮转法可以是简单轮转法、可变时间片轮转法，或多队列轮转法。

简单轮转法的基本思想是：所有就绪进程按 FCFS排成一个队列，总是把处理机分配给队首的进程，各进程占用CPU的时间片相同。如果运行进程用完它的时间片后还未完成，就把它送回到就绪队列的末尾，把处理机重新分配给队首的进程。直至所有的进程运行完毕。

3）编写并调试一个模拟的进程调度程序，采用“短进程优先”调度算法对五个进程进行调度。

4）结合已学调度思想，**查阅科技文献资料**，设计并实现一个新的调度算法，实现对多个进程的系统调度模拟。

**3、完成设计的调度算法的测试**

针对特定系统进程情况进行算法的性能计算与对比分析。特定系统的进程（几乎同时到达）如下表所示，以其作为测试数据完成设计的调度算法的功能测试，并计算和输出各个进程的周转时间和带权周转时间及平均值，对比实现的多种调度算法的性能。

|  |  |  |
| --- | --- | --- |
| **进程名** | **处理时间** | **优先级** |
| A | 3 | 2 |
| B | 6 | 1 |
| C | 4 | 3 |
| D | 5 | 4 |
| E | 2 | 5 |

**4、撰写实验报告**

实验报告应包含所选择的算法描述、数据结构设计、算法流程、代码、实验运行结果与实验心得。

**三、实验参考1**

**1、参考的进程调度程序描述**

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的指定（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。

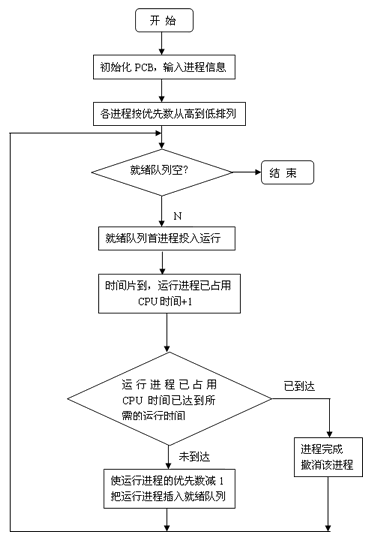
就绪进程获得CPU后都只能运行一个时间片。用已占用CPU时间加1来表示。

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。

**2、调度算法流程图**



**3、参考源代码**

#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;

int super;

int ntime;

int rtime;

struct pcb\* link;

}\*ready=NULL,\*p;

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 /\*向后移动指针\*/

{

first=first->link;

second=second->link;

}

}

/\* 插入进程优先数最低,则插入到队尾\*/

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

}

}

void input() /\* 建立进程控制块函数\*/

{

int i,num;

//clrscr(); /\*清屏\*/

//system("cls");

printf("\n 请输入进程数?");

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);

}

void 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");

}

void 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;

}

}

void 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);

}

void 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\*/

}

}

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();

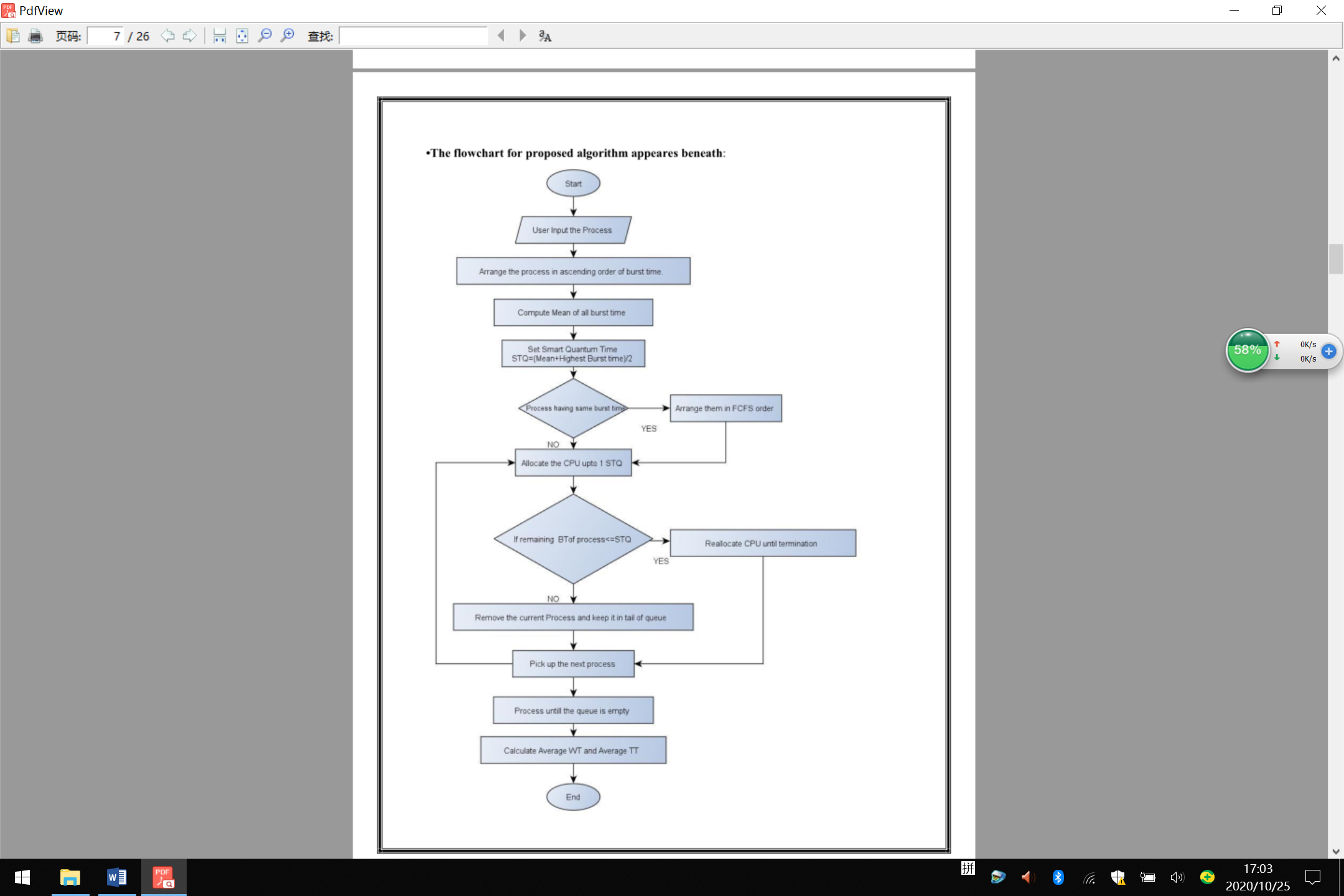
}

printf("\n\n 进程已经完成.\n");

ch=getchar();

}

**四、实验参考2**



该算法中会综合考虑所有进程的运行时间来设置时间片大小。首先计算所有进程运行时间的平均值Mean，然后一个时间片大小STQ=(Mean+进程的最大运行时间)/2。进程就绪队列按照进程的运行时间递增顺序组织，每次调队首进程运行一个STQ，如果剩余运行时间低于1个STQ，则为该进程重新分配CPU，让其运行完成，否则，将该进程插入就绪队列末尾，调下一个队首进程执行。依次轮流调度执行，直至就绪队列为空，计算所有进程的等待时间和周转时间平均值。

该算法以轮转调度为基础，考虑了作业的运行时间，时间片大小智能设置，尽量降低作业的平均等待时间和周转时间，故性能很好。