暨南大学本科实验报告专用纸

课程名称 操作系统原理 成绩评定

实验项目名称 使用动态优先权的进程调度算法的模拟 指导教师 赵阔

实验项目编号 0806002904 实验项目类型 设计型 实验地点

学生姓名 陈旭天 学号 2021100733

学院 智能科学与工程学院 系 人工智能 专业

实验时间 2023 年 4 月 20 日 下 午～ 4 月 20 日 下 午 温度 ℃湿度

暨南大学本科实验报告专用纸(附页)

一、实验目的和要求

通过动态优先权算法的模拟加深进程概念和进程调度过程的理解，并学习撰写规范的科学研究报告。

1．对 N 个进程采用动态优先权算法的进程调度；

2．每个用来标识进程的进程控制块 PCB 用结构描述，包括以下字段：进程标识数 ID，进程优先数 PRIORITY，进程以占用的 CPU 时间 CPUTIME，进程还需占用的 CPU 时间 ALLTIME，进程状态 STATE 等。

3．优先数改变的原则：进程在就绪队列中呆一个时间片，优先数增加 1，进程每运行一个时间片优先数减 3。

4．设置调度前的初始状态。

5．将每个时间片内的进程情况显示出来

二、实验原理和主要内容

三、C 程序源码

1. 运行结果：

```
root@LAPTOP-40J2BMTQ:~/OS2023/dynamic_prior# vim dyproir.c
root@LAPTOP-40J2BMTQ:~/OS2023/dynamic_prior# gcc dyproir.c && ./a.out
请输入进程个数：
10
请输入第1个进程的ID号：
23
请输入第2个进程的ID号：
24
请输入第3个进程的ID号：
25
请输入第4个进程的ID号：
26
请输入第5个进程的ID号：
27
请输入第6个进程的ID号：
28
请输入第7个进程的ID号：
29
请输入第8个进程的ID号：
0
请输入第9个进程的ID号：
1
请输入第10个进程的ID号：
2
            动态优先级调度算法
------------------------------------------------
ID      prior   runtime state
23      4       7       1
24      8       6       1
25      4       6       1
26      7       3       1
27      10      2       1
28      3       8       1
29      1       10      1
0       4       7       1
1       1       7       1
2       3       7       1
------------------------------------------------
```

2. 实验源码：

```
//按优先数调度算法实现处理器调度的程序
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```
#include <time.h>

//进程结构体
struct Jc{
	int ID; //进程 id
	int prior; //进程优先数
	int runtime; //运行时间
	int state; //进程状态  1 -  就绪态    0 -  结束态
};

//显示函数,便于查看进程每一步的调度情况
void    show(int num,struct Jc Que[]){
	printf("-----------------------------------------------\n");
	printf("ID\tprior\truntime\tstate\n");
	for(int i=0;i<num;i++){
		printf("%d\t%d\t%d\t%d\n",Que[i].ID,Que[i].prior,Que[i].runtime,Que[i].state);
	}
	printf("-----------------------------------------------\n");
}

//查找函数，查找队列中优先数最大的进程
int max(int num,struct Jc Que[]){
	int key=0; //记录优先数最高的进程的在队列中的位置，默认为第 1 个进程优先数最高

	for(int i=1;i<num;i++){                                  //遍历进程队列寻找优先数最大的进程，
并记录在 max 中
		//printf("%d 的 state    %d\n",i,Que[i].state);
		if( Que[i].state == 2){                          //state = 2    表示正在运行，此时不需要查
找
		//printf("max 函数中判断运行的 if    %d 的 state    %d\n",i,Que[i].state);
		return -1;
		}
		if( Que[key].prior < Que[i].prior && Que[i].state == 1){
			key = i;
		}
	}

	//printf("max 函数中得到结果  %d\n",key);
	return key;
}

int main(){
	int num;
	printf("请输入进程个数：\n");                  //用户输入进程个数
```

```
    scanf("%d",&num);
    struct Jc Que[num];              //定义进程控制块队列
    for(int i=0;i<num;i++){                                    //初始化队列
        printf("请输入第%d 个进程的 ID 号：\n",i+1);
        scanf("%d",&Que[i].ID);
        Que[i].prior = rand()%10+1;
        Que[i].runtime = rand()%10+1;
        Que[i].state = 1;
    }
    getchar();   //设置间隔，接收到回车时结束输入
    int sum=0;   //记录总运行时间
    for(int i=0;i<num;i++){                  //遍历进程队列计算总运行时间
        sum +=Que[i].runtime;
        //printf("%d\n",sum);
    }
    printf("                  动态优先级调度算法                        \n");
    show(num,Que); //显示最初进程控制块队列
    getchar();   //设置间隔，接收到回车时开始调度算法
    for(int j=0;j<sum;j++){
        int key = max(num,Que);
        if(key!=-1){ //查找到处于就绪态的最大优先级的进程时，设置其运行
            Que[key].state = 2;
            //printf("%d 的 state    %d\n",key,Que[key].state);
        }
        //printf("j       %d\n",j);
        for(int i=0;i<num;i++){
            //printf("i     %d\n",i);
            if(Que[i].state == 2){ //将运行进程的优先级数和运行时间-1，表示已经完成一
次对该进程的调度
                Que[i].prior -=1;
                Que[i].runtime -=1;
                if(Que[i].runtime == 0) //完成一次调度后，若该进程还需运行时间为 0 则
设置为结束态，不为 0 则设置为就绪态
                Que[i].state = 0;
                else
                Que[i].state = 1;
                show(num,Que); //显示完成这次调度后进程控制块队列信息
                getchar();   //设置间隔，接收到回车时进入下一次调度
            }
        }
    }
}
```

四、实验总结

动态优先调度算法在每次调用进程的时候，都会重新给每一个进程进行更新优先值然后排序，这符合在高并发领域中程序不断动态变化的要求