**《操作系统课程实验》**

**实验三 -- 单处理机进程调度**

1. **实验基本要求**
2. **实验按100分计（含基本分75 + 扩展分25）：现场演示（50分），提交实验报告（打印或手写，25分），提供设计文档和源代码（电子文档，25分）。**
3. **鼓励采用图形用户界面，有完善的I/O设计和容错处理者将有额外加分（最高20分）！**
4. **随机检查（询问）3处代码或设计方案，答错一处扣20分！若发现演示或设计文档和源代码雷同者0分处。**
5. **实验目的**
6. **加深进程概念理解，明确进程与程序区别。**
7. **理解操作系统中进程的组织、创建和调度等方法。**
8. **实验内容**

**编写程序完成单处理器系统的进程调度，要求采用时间片轮转法调度策略。具体内容：**

1. **确定PCB内容及其组织方式；**
2. **要求模拟进程空闲（新）、就绪、运行、阻塞和完成5个状态；**
3. **实现进程创建、进程调度、进程阻塞和进程唤醒4个原语；**
4. **编写主函数对整个系统进程测试。**
5. **提示**

**关键三点：**

1. **如何组织进程：**

* **确定PCB内容：标识信息、状态和运行时间与存储地址等信息、现场信息、管理信息**
* **PCB组织方式：相同状态的进程PCB构成一个队列（即有空闲、就绪、运行、阻塞和完成5个队列）**

1. **如何创建进程：**

* **申请PCB（从空闲队列）—> 申请资源—> 填写PCB—>挂就绪队列**

1. **如何实现处理机调度及进程状态切换：**

* **采用先来先服务（FCFS）调度策略实现进程调度；**
* **从就绪队列选择一个进程；摘取PCB，挂运行队列；修改状态等PCB内容； 保存现场、恢复现场；**
* **模拟运行--可以按照两种场景模拟进程运行：**

**（1）可以预先设置好各进程的运行时间、I/O时间、I/O发生的时刻等信息，之后操作系统控制进程运行，实现状态切换，直到全部进程完成。**

**（2）亦可以采用人工干预方式控制进程状态切换（运行时间已预先设置），比如输入“Esc”进入“阻塞”状态，输入“Enter” 则选择（新）进程运行（进程调度），当前进程回到就绪状态;输入“wakeup”,再选择阻塞进程，则被选中的阻塞进程回到就绪状态；输入“finished”，当前进程运行结束，回到完成状态；**

* **修改（剩余）运行时间。**

1. **测试输出**

* **手工输入信息建立几个进程（就绪队列）；**
* **进行进程调度。被调度上的进程（正在运行）输出：**

1. **退出CPU的进程名及其PCB内容。**
2. **被调度上进程相关信息：“This is Process ‘i’, I’am running in time-slice ‘j’”（i是被调度上的进程ID，j是该进程被第几次调度上--第j个时间片）。**
3. **被调度上的进程PCB内容。**

* **当进程状态切换时，输出：当前就绪队列、运行队列和阻塞队列中各进程的标识。**

1. **进一步扩展**

**采用多队列反馈式调度策略实现进程调度。**

您好，根据您的要求，我能够为您提供一些关于C语言程序实现时间片轮转法调度策略的建议，但是请您注意，这只是一些指导性的建议，具体的实现方式需要根据您的具体需求进行调整。

1、确定PCB内容及其组织方式：

进程控制块(Process Control Block, 简称PCB)是操作系统管理进程的数据结构，它保存了进程的基本信息和状态信息，包括进程的ID、状态、优先级、寄存器值、进程状态、时间片等。在实现时间片轮转法调度策略中，可以将PCB结构体定义为包含这些元素，如下所示：

typedef struct PCB { int pid; // 进程ID int cpu\_time; // 进程已经占用的CPU时间 int run\_time; // 进程需要运行的时间总长 int state; // 进程状态，0为新建、1为就绪、2为运行、3为阻塞、4为完成 ... } PCB;

2、要求模拟进程空闲（新）、就绪、运行、阻塞和完成5个状态：

在实现时间片轮转法调度策略过程中，需要实现五个进程状态，分别是新建、就绪、运行、阻塞和完成。其中新建是指当进程被创建后，其状态为新建状态，等待系统调度器分配CPU时间片；就绪是指当进程分配到CPU时间片后，等待系统调度器分配CPU执行时间；运行是指分配到CPU时间片后，进程处于执行状态；阻塞是指当前进程被外部信号或事件阻塞，不能继续执行；完成是指进程已经执行完毕。

可以使用switch语句来模拟五个状态的切换，具体实现可以像下面这样：

switch(pcb.state) { case 0: // 新建状态 printf("进程正在创建中...\n"); break; case 1: // 就绪状态 printf("进程已经准备就绪，等待运行...\n"); break; case 2: // 运行状态 printf("进程正在运行中...\n"); break; case 3: // 阻塞状态 printf("进程正在等待事件完成，不能继续执行...\n"); break; case 4: // 完成状态 printf("进程执行完毕，释放资源...\n"); break; default: printf("进程状态错误，请检查...\n"); }

3、实现进程创建、进程调度、进程阻塞和进程唤醒4个原语：

在实现时间片轮转调度算法时，需要实现进程创建、调度、阻塞和唤醒等操作，下面具体介绍：

进程创建：

进程创建指的是在系统中新建一个进程的过程，包括为进程分配PCB、为进程分配系统资源等。对于进程创建，可以定义一个函数来实现，如下所示：

PCB pcb\_create(int pid, int run\_time) { PCB pcb = { pid, 0, run\_time, 0 }; return pcb; }

进程调度：

时间片轮转调度算法能够平均分配所有进程的时间，防止某个进程过多占用CPU资源。在实现进程调度时，需要考虑到时间片，每个进程只能占用一定时间片的CPU处理时间，达到时间片后要将当前进程挂起，执行下一个进程。具体实现如下：

PCB scheduler(PCB queue[], int n, int curr) { queue[curr].cpu\_time += TIME\_SLICE; if(queue[curr].cpu\_time >= queue[curr].run\_time) { queue[curr].state = 4; // 进程执行完毕 return queue[curr]; } int next = (curr + 1) % n; // 计算下一个就绪进程 // 找到下一个就绪进程 while(next != curr) { if(queue[next].state == 1) { queue[next].state = 2; queue[curr].state = 1; return queue[next]; } else if(queue[next].state == 2) { next = (next + 1) % n; } } // 没有新进程，当前进程继续运行 return queue[curr]; }

进程阻塞：

进程阻塞是指当前进程因为某种原因（如等待某个信号、等待某个I/O操作完成等）暂时不能进行计算操作。在实现进程阻塞时，需要设置进程的状态为阻塞状态，并释放进程占用的I/O资源。具体实现如下：

void pcb\_block(PCB \*pcb) { pcb->state = 3; // 设置进程状态为阻塞状态 // 释放进程占用的I/O资源 ... }

进程唤醒：

进程唤醒是指当前进程解除阻塞后，重新被调度运行的过程。在实现进程唤醒时，需要设置进程的状态为就绪状态，并将其加入到就绪队列中，等待CPU调度。具体实现如下：

void pcb\_wakeup(PCB \*pcb, PCB queue[], int n) { pcb->state = 1; // 设置进程状态为就绪状态 // 将进程加入到就绪队列的最后 queue[n] = \*pcb; }

4、编写主函数对整个系统进程测试：

最后一步是编写一个测试程序，模拟CPU调度器执行进程，并输出相应的进程状态。具体步骤如下：

1. 创建进程测试数据，如下所示：

PCB pcb1 = pcb\_create(1, 20); PCB pcb2 = pcb\_create(2, 12); PCB pcb3 = pcb\_create(3, 5); PCB pcb4 = pcb\_create(4, 8); PCB pcb5 = pcb\_create(5, 30);

1. 将进程加入到就绪队列中：

PCB queue[] = { pcb1, pcb2, pcb3, pcb4, pcb5 }; // 就绪队列

1. 轮流调度五个进程：

int curr = 0; for(int i = 0; i < 5; i++) { PCB pcb = scheduler(queue, 5, curr); printf("当前进程 %d 的状态为：", pcb.pid); // 输出进程状态 ... }

通过以上步骤，可以简单地实现一个时间片轮转调度算法，并测试进程状态，实现进程的调度和管理。