### 一、系统设计概述

* ****系统架构图****：展示整个操作系统的设计架构，包括进程管理、处理器调度和请求页式存储管理等模块。

### 二、进程管理

* ****进程控制块（PCB）示例图****：展示一个或多个进程的控制块，包括进程名、状态、优先级等信息。
* ****进程状态转换图****：展示进程状态（就绪、运行、阻塞、完成）之间的转换。

### 三、处理器调度

* ****调度算法示意图****：展示抢占式动态优先级调度算法的工作流程。
* ****调度结果截图****：展示某个时间点的进程调度情况，包括进程的优先级和状态。

### 四、请求页式存储管理

* ****页表结构图****：展示页表的结构和内容，包括页号、物理块号和完成状态。
* ****页面置换示意图****：展示LRU页面置换算法的工作流程。
* ****内存分配截图****：展示某个时间点的内存页面分配情况。

### 五、进程运行

* ****进程执行流程图****：展示进程从创建到完成的整个执行流程。
* ****指令执行截图****：展示进程执行指令时的系统状态，包括页面访问和可能的缺页处理。

### 六、实验结果

* ****调度结果汇总图****：汇总展示所有进程的调度顺序和状态变化。
* ****内存管理结果图****：展示缺页中断次数、页面置换次数等内存管理相关的统计结果。
* ****性能指标图表****：如CPU利用率、平均等待时间和平均周转时间的图表。

目录

一、系统设计概述 1

二、进程管理 1

三、处理器调度 2

四、请求页式存储管理 2

五、进程运行 3

六、实验结果 4

七、结论 4

# 操作系统课程设计报告

## 一、系统设计概述

本系统设计模拟了一个简单的操作系统，涵盖了进程管理、处理器调度和请求页式存储管理等核心功能。系统的设计目标是提供一个基础的框架，用以演示操作系统在进程控制、任务调度和内存管理方面的基本原理和操作。

### 1. 进程管理

进程管理模块是系统的核心，它负责创建、调度和终止进程。在本设计中，我们定义了6个进程，每个进程具有独特的属性，包括进程名、到达时间、优先级、CPU运行所需时间、状态、阻塞时长、指令条数和页表号。进程状态可以是运行（W）、就绪（R）、阻塞（B）或完成（F），模拟了进程在操作系统中的生命周期。

### 2. 处理器调度

处理器调度模块负责决定哪个进程将获得CPU资源进行执行。本系统采用了抢占式动态优先级调度算法，优先执行优先级最高的进程。进程的优先级会根据其在系统中的行为动态调整，以模拟真实操作系统中的进程调度。

### 3. 请求页式存储管理

请求页式存储管理模块负责管理虚拟内存和物理内存之间的映射关系。每个进程都有自己的页表，系统通过模拟页表来跟踪进程的页面访问。当进程访问的页面不在物理内存中时，会发生缺页中断，系统将使用最近最少使用（LRU）算法进行页面置换，确保内存的有效利用。

### 4. 系统功能实现

系统设计要求至少包含5个进程，模拟执行读内存、写内存、输入和输出指令，并实现进程状态之间的转换。此外，系统还需模拟至少10个页面的访问，并产生5次以上的缺页中断以及3次以上的页面置换。通过这些功能的实现，系统能够展示操作系统在进程管理和内存管理方面的复杂性和效率。

## 二、进程管理

进程管理是操作系统中的一个关键功能，它涉及到进程的创建、执行、同步、通信以及终止等。在本课程设计中，进程管理模块的实现是模拟操作系统中进程的行为和调度策略的基础。

### 进程定义与属性

在本系统中，进程被定义为一个结构体（Task），包含以下属性：

* ****进程名（**process\_name**）****：标识每个进程的名称。
* ****到达时间（**arrive\_time**）****：进程到达并准备进入就绪队列的时间。
* ****优先级（**priority**）****：用于处理器调度的优先级值，影响进程被调度的顺序。
* ****CPU运行所需时间（**need\_time**）****：进程完成其任务所需的CPU时间。
* ****状态（**status**）****：表示进程的当前状态，可以是运行（W）、就绪（R）、阻塞（B）或完成（F）。
* ****阻塞时长（**block\_time**）****：如果进程进入阻塞状态，记录其阻塞的时间长度。
* ****指令条数（**instruction\_number**）****：进程需要执行的指令数量。
* ****页表号（**table\_number**）****：指向进程页表的索引，用于请求页式存储管理。

### 进程状态转换

进程在其生命周期中会经历不同的状态转换，包括：

* ****就绪（R）****：进程已到达并等待被调度执行。
* ****运行（W）****：进程正在使用CPU执行指令。
* ****阻塞（B）****：进程因等待某些事件（如I/O操作）而暂时无法执行。
* ****完成（F）****：进程已执行完毕，退出系统。

### 进程调度

本系统采用抢占式动态优先级调度算法，该算法考虑了进程的优先级和到达时间。在每个时间单位，系统会检查所有就绪状态的进程，并选择优先级最高的进程来执行。如果两个进程具有相同的优先级，则选择到达时间最早的进程。

### 进程控制块（PCB）

每个进程在系统中都有一个对应的进程控制块（PCB），存储进程的所有必要信息。PCB的设计使得系统能够有效地管理和调度进程，包括进程的创建、调度、状态转换和终止。

### 实现细节

在代码实现中，进程管理模块包括以下几个关键函数：

* **set\_task\_block\_time**：将进程状态设置为阻塞，并设置阻塞时间。
* **decrease\_block\_time**：每个时间单位减少阻塞时间，如果阻塞时间结束，则将进程状态设置为就绪。
* **update\_priority**：更新任务优先级，运行中的进程优先级降低，就绪中的进程优先级提高。
* **find\_highest\_priority\_task**：找到优先级最高的任务，如果当前运行的任务完成或阻塞，则选择新的最高优先级任务来执行。

通过这些机制，进程管理模块能够有效地模拟操作系统中的进程行为，为进一步的处理器调度和内存管理提供了基础。

## 三、处理器调度

处理器调度是操作系统中的一个核心功能，它决定了哪个进程将获得CPU时间片以执行。在本课程设计中，我们实现了一个抢占式动态优先级调度算法，以模拟操作系统中的处理器调度过程。

### 抢占式动态优先级调度算法

本系统采用的抢占式动态优先级调度算法基于以下原则：

* ****优先级规则****：进程根据其优先级进行调度，优先级高的进程将优先获得CPU资源。
* ****动态调整****：进程的优先级会根据其在系统中的行为动态调整。如果进程正在运行，其优先级会逐渐降低，以避免一个进程长时间占用CPU。如果进程处于就绪状态，其优先级会逐渐增加，以提高其被调度的机会。
* ****抢占****：如果一个更高优先级的进程变为就绪状态，当前运行的进程将被抢占，新的高优先级进程将获得CPU控制权。

### 实现细节

在代码实现中，处理器调度模块包括以下几个关键步骤：

****1.更新任务优先级（**update\_priority**）****：在每个时间单位，系统会更新所有进程的优先级。正在运行的进程优先级会降低，而处于就绪状态的进程优先级会增加。

****2.找到最高优先级任务（**find\_highest\_priority\_task**）****：系统会遍历所有就绪状态的进程，找到当前优先级最高的进程。如果存在优先级相同的进程，系统会选择到达时间最早的进程。

****3.任务状态转换****：当一个进程被选中执行时，其状态会从就绪（R）变为运行（W）。如果进程在执行过程中需要等待I/O操作或其他原因进入阻塞状态，其状态会从运行（W）变为阻塞（B）。

****4.处理阻塞和解除阻塞****：系统会跟踪每个进程的阻塞时间，并在适当的时候将进程状态从阻塞（B）转换为就绪（R），以便进程可以重新进入调度队列。

### 调度过程

在每个时间单位，处理器调度模块执行以下操作：

* 检查所有进程的到达时间和优先级，将新到达的进程加入到就绪队列。
* 更新所有进程的优先级，降低运行中进程的优先级，提高就绪进程的优先级。
* 找到当前优先级最高的进程，并检查其状态。如果进程处于就绪状态，则将其状态设置为运行，并执行；如果进程处于阻塞状态，则跳过该进程，继续寻找下一个最高优先级的进程。
* 如果当前运行的进程完成执行或进入阻塞状态，系统将选择新的最高优先级进程来执行。

### 结果展示

在模拟过程中，处理器调度模块会打印出每个时间单位的进程状态，包括进程的名称、优先级和状态。这为观察调度算法的行为提供了直观的视图，允许我们验证调度策略的有效性和公平性。

通过实现处理器调度模块，本系统能够模拟真实操作系统中的进程调度过程，展示了抢占式动态优先级调度算法在多进程环境中的管理能力。

## 四、请求页式存储管理

请求页式存储管理是操作系统中负责虚拟内存管理的关键部分，它允许系统通过将进程的虚拟地址空间分割成多个页，并只加载部分页到物理内存中来有效管理内存资源。在本课程设计中，我们模拟了请求页式存储管理的工作原理，包括页表的创建、缺页中断的处理以及页面置换算法的应用。

### 页表设计

每个进程都有自己的页表，该页表定义为一个结构体数组（PageTable），其中包含以下属性：

* ****页号（**page\_number**）****：标识虚拟内存中的页号。
* ****物理块号（**physical\_block\_number**）****：标识物理内存中的块号，用于存储对应页的数据。
* ****是否执行完（**ifFinsih**）****：标识该页中的指令是否已经执行完毕。

### 缺页中断处理

当进程尝试访问的页不在物理内存中时，会发生缺页中断。在本系统中，我们模拟了缺页中断的处理过程：

****1.检查页面是否已加载****：在进程执行指令时，系统会检查所需页面是否已经加载到物理内存中。

****2.缺页中断****：如果页面未加载，系统会触发缺页中断，并执行页面置换算法来选择一个页面进行置换。

****3.页面置换****：系统使用LRU（最近最少使用）算法选择最久未使用的页面，并用新请求的页面替换它。

### 页面置换算法

本系统采用了LRU算法作为页面置换策略，其工作原理如下：

* ****记录页面访问时间****：系统维护一个全局数组（last\_used），记录每个页面最后一次被访问的时间。
* ****选择最久未使用页面****：当需要置换页面时，系统会查找last\_used数组中值最小的页面，即最久未使用的页面。
* ****置换页面****：将选中的页面替换为新请求的页面，并更新last\_used数组以反映新页面的访问时间。

### 实现细节

在代码实现中，请求页式存储管理模块包括以下几个关键函数：

* **print\_page\_table**：打印指定进程的页表内容，展示页号、物理块号和执行状态。
* **simulate\_task\_execution**：模拟任务执行过程中的页面访问和缺页处理。
* **page\_replacement**：执行页面置换，使用LRU算法选择并替换最久未使用的页面。
* **print\_last\_used\_times**：打印当前内存中所有页面的最后使用时间，以展示LRU算法的执行情况。

### 结果展示

在模拟过程中，系统会打印出每次页面访问和置换的详细信息，包括发生缺页中断的页号、被置换的页面以及当前内存中所有页面的最后使用时间。这些信息为观察页面置换算法的行为提供了直观的视图，允许我们验证LRU算法的有效性和对系统性能的影响。

通过实现请求页式存储管理模块，本系统能够模拟真实操作系统中的内存管理过程，展示了请求页式存储管理在多任务环境中的内存调度能力。

## 五、进程运行

在本课程设计中，进程运行模拟了操作系统中进程从创建、执行到完成的整个生命周期。这一部分是理解操作系统如何管理和执行进程的关键环节。

### 进程执行模拟

进程运行模拟包括以下几个方面：

****1.进程创建****：在模拟开始时，所有进程根据其到达时间被创建并加入到等待队列中。

****2.进程调度****：进程根据抢占式动态优先级调度算法被选中执行。一旦进程开始执行，它将从就绪状态（R）变为运行状态（W）。

****3.指令执行****：进程执行其指令序列，这些指令包括读内存、写内存、输入和输出操作。每个指令的执行都可能涉及到页面的访问。

****4.页面访问和缺页处理****：当进程访问的页面不在物理内存中时，将触发缺页中断。系统将执行LRU算法进行页面置换，然后将新页面加载到物理内存中，以便进程可以继续执行。

****5.进程阻塞和唤醒****：如果进程在执行过程中需要等待I/O操作或其他资源，它将从运行状态变为阻塞状态（B）。一旦阻塞条件解除，进程将被唤醒并重新进入就绪队列等待下一次调度。

****6.进程完成****：进程执行完所有指令后，其状态将被标记为完成（F），并从CPU上移除。

### 进程状态跟踪

进程的每个状态变化都被跟踪和记录，以便于模拟和分析。系统维护一个进程控制块（PCB）数组，其中包含每个进程的详细信息和当前状态。

### 进程执行的关键函数

在代码实现中，进程运行模块包括以下几个关键函数：

* **simulate\_task\_execution**：模拟单个任务的执行过程，包括指令执行和页面访问。
* **page\_replacement**：当发生缺页中断时，调用此函数执行页面置换。
* **print\_task\_status**：打印当前所有进程的状态，包括进程名、状态和优先级。
* **decrease\_block\_time**：减少处于阻塞状态的进程的阻塞时间，并在阻塞时间结束后将其状态更改为就绪。

## 六、实验结果

在本课程设计的实验中，我们通过模拟一个简单的操作系统来验证进程管理、处理器调度和请求页式存储管理的功能。以下是实验的关键结果和观察：

### 进程调度结果

实验结果显示了进程的调度顺序和状态变化。通过动态优先级调度算法，系统能够有效地处理进程的优先级调整和抢占，确保CPU资源的合理分配。以下是一些具体的观察：

* ****优先级调整****：运行中的进程优先级逐渐降低，而就绪队列中的进程优先级逐渐增加，这有助于平衡进程的执行机会。
* ****抢占行为****：当更高优先级的进程变为就绪状态时，当前运行的进程被抢占，确保了紧急任务能够及时执行。
* ****阻塞和唤醒****：进程在等待I/O操作时进入阻塞状态，一旦条件满足，进程被唤醒并重新加入到就绪队列中。

### 内存管理结果

请求页式存储管理的实验结果展示了页面访问、缺页中断和页面置换的情况。以下是一些关键的观察：

* ****页面访问****：进程执行指令时访问的页面被记录，如果页面不在物理内存中，将触发缺页中断。
* ****缺页中断****：实验中发生了预定次数的缺页中断，系统通过LRU算法成功地进行了页面置换，将最久未使用的页面替换为新请求的页面。
* ****页面置换效率****：通过跟踪页面的最后使用时间，LRU算法能够有效地选择置换页面，减少了系统的页面置换次数。

### 进程执行和状态变化

实验中，每个进程的执行和状态变化都被详细记录。以下是一些具体的数据：

* ****进程执行时间****：每个进程的CPU执行时间被记录，显示了进程从就绪到完成的转变。
* ****状态转换次数****：进程在运行、就绪和阻塞状态之间的转换次数被统计，反映了进程的行为模式。
* ****完成时间****：每个进程的完成时间被记录，显示了进程从创建到完成的总时间。

## 七、结论

通过本课程设计的模拟实现，我们成功地展示了一个简单操作系统的核心功能，包括进程管理、处理器调度和请求页式存储管理。以下是对整个设计和实验的总结：

### 系统功能验证

****1.进程管理****：我们设计了一套完整的进程控制块（PCB），能够跟踪进程的名称、状态、优先级等关键信息，并模拟了进程的创建、执行、阻塞和完成的生命周期。

****2.处理器调度****：通过实现抢占式动态优先级调度算法，我们验证了进程调度的公平性和效率。进程优先级的动态调整和抢占机制确保了高优先级任务能够及时得到处理。

****3.请求页式存储管理****：我们模拟了页表的使用和页面置换过程，特别是LRU算法的应用，有效地管理了虚拟内存和物理内存之间的映射关系，处理了缺页中断和页面置换。

### 实验效果

****1.调度效率****：实验结果显示，动态优先级调度算法能够有效地响应进程的优先级变化，合理地分配CPU时间，提高了系统的响应速度和吞吐量。

****2.内存利用率****：通过LRU页面置换算法，系统能够高效地利用有限的物理内存资源，减少了因频繁缺页导致的性能损失。

****3.进程执行****：进程的执行和状态转换被准确地模拟和记录，展示了操作系统如何管理和调度多个并发进程。

### 教育意义

本课程设计不仅加深了对操作系统原理的理解，特别是进程管理和内存管理的复杂性，而且提供了一个实践平台，让学生能够通过实际编码和模拟来体验和学习这些概念。

### 未来工作

尽管本设计已经涵盖了操作系统的许多基本功能，但仍有进一步改进和扩展的空间。未来的工作可以包括：

****1.更复杂的调度算法****：引入如最短作业优先（SJF）或轮转（RR）等其他调度算法，以比较不同算法的性能和适用场景。

****2.内存管理优化****：探索其他页面置换算法，如CLOCK或OPT，以提高内存管理的效率。

****3.设备管理和文件系统****：扩展系统功能，模拟输入输出操作和文件系统的管理，以更全面地模拟操作系统的行为。

****4.性能评估****：通过更复杂的工作负载和性能评估指标，深入分析系统在不同条件下的表现。

综上所述，本课程设计的成功实施不仅验证了操作系统的基本理论，而且为操作系统的教学和研究提供了一个有价值的工具。