# 操作系统模拟实验

实验一：进程调度实验

【实验目的】

1. 理解在单处理器环境下的进程调度及状态转换的基本原理和关键点。
2. 编程实现时间片轮转调度、短进程优先调度和多级反馈队列调度算法。
3. 分析并比较不同调度算法的效率和公平性。

【实验要求】

1. 模拟每一个进程，需要包含到达时间，运行时间，优先级等属性。
2. 每一种调度算法都需要实现一个模拟函数，对进程结束时间进行仿真模拟。多级反馈队列调度算法至少包含三个队列，每个队列的时间片大小不同。
3. 对给定的一组进程，当采用不同的调度算法时，需仿真模拟这组进程的状态转换过程。进程的状态至少有就绪、运行和阻塞三种状态。
4. 当这组进程调度执行完成时，根据仿真模拟的结果，计算平均周转时间/平均带权周转时间，平均等待时间、平均响应时间、利用率等性能指标及公平性指标。
5. 能够提供合适的测试数据集，对每一种算法进行测试，并分析其结果。

【涉及的基本原理】

进程调度是指操作系统中关于处理系统进程运行顺序和方式的机制。在设计进程调度程序时，需要了解以下基本原理：

1. 进程的概念：进程是程序的一次执行过程，是系统进行资源分配和调度的基本单位。它由程序代码和相关数据、堆栈、PCB(Process Control Block, 进程控制块)等部分组成。
2. 进程的状态：进程最基本的三种状态是：就绪状态（Ready）、执行状态（Running）、和等待状态（Waiting）。在进程的生命周期中，会反复在这几种状态之间转换。
3. 调度器：负责根据特定策略，决定哪个进程占有CPU。调度器分为长程调度器（作业调度器）、短程调度器（CPU调度器）、中程调度器。本模拟实验仅涉及短程调度。
4. 进程调度策略：调度策略主要有先来先服务（FCFS）、最短作业优先（SJF）、优先级调度、轮转法（RR）、多级反馈队列等。这些调度策略均有其特定的场合和使用环境。
5. 多级反馈队列调度算法是一种兼顾了高响应比优先、短作业优先和时间片轮转等算法优点的算法。它将就绪队列分为若干个队列，每个队列的优先级不同；同时，各队列可以采用不同的调度算法，以达到最大程度地提高系统的吞吐量、响应时间以及带宽等性能指标。
6. 上下文切换：当CPU从一个进程切换到另一个进程时，需要保存当前进程的状态，并加载新的进程状态，这个过程被称为上下文切换，它会产生一定的开销。
7. 并发性：操作系统可以让多个进程“同时”运行，实际上是通过迅速地在不同进程间进行切换，为每个进程分配一段时间来执行，从而实现宏观上的并发。

【实验提示】

1. 在模拟实验中，进程的到达时间、运行时间以及进程的特性（CPU繁忙型、I/O繁忙型）通常是通过人工指定或使用随机数生成方法产生的。人工指定的方法更适用于测试环境，因为你可以控制所有的变量，使测试结果可预测并被理解。随机数生成的方法更能够创建更真实的运行环境，但在使用随机数生成方法确定进程到达时间和运行时间时，需要注意一些细节。例如，进程到达时间必须是递增的；运行时间应确保在合理范围内，不能太长也不能为0。同时，如果用随机方法生成的测试数据，务必记住保存这批数据或者保存随机种子，以确保实验结果的可重复性。
2. 在编程过程中，需要注意进程控制块PCB结构的选择，以便适应于不同算法的需求。
3. 在分析调度效果时，需要理解在不同的进程运行情况下（比如到达时间，服务时间，进程的数量等），不同的调度算法可能会有完全不同的表现。在进行平均周转时间等性能指标的计算时，也需要注意公平性的考量。
4. 在设计进程和调度算法时，要考虑清楚测试用例是否可以覆盖多种可能的情况，以便于发现并处理边缘情况。有时甚至要创建一些额外的测试用例去覆盖边缘情况，比如所有进程几乎同时到达，或者所有进程运行时间都一样等。

【验收标准】

1. 基本功能的实现：能够正确模拟时间片轮转调度、短进程优先调度和多级反馈队列调度算法，并且模拟的结果正确。
2. 选择合理的进程数据集进行展示：测试数据集能够为每种调度算法提供足够的测试覆盖。
3. 能够选择合适的性能评估指标和公平性指标来分析和评价各种调度算法。
4. 系统的稳定性和错误处理：程序在运行过程中是否能稳定执行，是否能有效地处理错误的情况。
5. 用户友好度：输出信息要能够清晰反映调度过程中进程的状态变化。
6. 代码质量：高质量的代码表现在清晰的组织结构，良好的命名规范，适当的注释等。良好的代码质量可以提高代码的可读性和可维护性。
7. 实验报告：一个好的实验报告应该详细描述算法的设计与实现的关键点、充分展示测试情况，记录遇到的问题及解决方案等，并对不同的调度算法进行性能和公平性分析，并得出你的结论。

实验二：虚拟页式存储管理系统

【实验目的】

1. 理解虚拟内存和物理内存之间的关系及转化，在什么情况下会发生缺页中断。
2. 理解和掌握各种页面置换算法的工作原理，包括先进先出（FIFO）、最近最少使用（LRU）、时钟置换（CLOCK）和随机置换算法，并通过模拟实验观察它们的执行过程。
3. 通过比较不同页面置换算法下系统的性能，理解不同算法的优缺点以及在何种情况下更适用。
4. 学习如何评估系统性能。通过实验结果，理解如何通过缺页中断次数、平均访问时间等指标去评估系统性能，并能合理地解释实验结果。
5. 从实验中体验和理解内存访问的局部性原理，提高自己对操作系统虚拟内存管理的理解。

【实验要求】

1. 设计并实现一个模拟的简单虚拟页式存储管理系统。在该系统中，应能模拟以下功能：创建进程的页表，为进程页面分配内存块，实现逻辑地址向物理地址的转换。当发生缺页时，通过缺页中断处理调入页面，如果需要，可以进行页面置换。
2. 制定一套模拟数据，包括要访问的页面、需执行的操作等，并按照虚拟页式存储管理的原则进行处理，展示每一步的执行结果。
3. 能够模拟出页面置换的情况，例如FIFO、LRU、CLOCK和随机置换等算法，比较并分析每种算法下系统的性能。

【涉及的基本原理】

1. 虚拟内存技术：这是一种使得计算机系统看上去具有比实际RAM更大的内存的技巧。它使用了硬盘空间作为对RAM的扩充并将之作为应用程序的虚拟存储器。
2. 页（Page）：操作系统管理内存的一个单位。操作系统会把逻辑地址空间分割为一系列连续的、固定大小的区域，这些区域就称为页。
3. 帧（Frame）：操作系统管理对应实际内存的一个单位，与页的大小一致。
4. 页表：虚拟页面和物理内存帧之间的映射存放在页表中。为了提高地址转换的效率，操作系统通常会采取一些策略来优化页表，如多级页表、倒置页表、TLB（Translation Lookaside Buffer）等。
5. 地址映射：操作系统会维护一个页表，记录页号和帧号的映射关系。当CPU发出一个虚拟地址时，操作系统会通过页表将虚拟地址转换为实际的物理地址。
6. 页面置换：由于内存有限，当分配给一个进程的所有页面都在使用中且需要加载新的页面时，操作系统必须选择一个页面将其移出内存以空出空间给新的页面，这个决定过程就是页面置换。常见的页面置换算法包括最近最久未使用（LRU）、最佳置换算法（OPT）、先进先出（FIFO）、时钟置换（CLOCK）等。
7. 缺页中断：当程序引用的页不在内存中时，会产生缺页中断，操作系统需要从磁盘中将页调入内存，并更新页表。

【实验提示】

1. 设计进程类，包含该进程的页表，页表中应包含页号和对应的物理块号等。
2. 设计内存类，模拟内存中的物理块，能够根据进程请求分配和回收物理块。
3. 设计地址转换函数，输入逻辑地址，输出物理地址。
4. 实现页面置换算法，对内存中的页面进行动态调整。
5. 缺页中断通常是因为进程尝试访问其虚拟内存空间中一个不存在于物理内存（RAM）中的页面时触发的，如果分配给该进程的物理内存已满，则需要进行页面置换。
6. 为了展示该内存管理系统，需设计合理的模拟数据，当运行内存管理程序时，展示每一步的内存分配、地址转换和页面置换情况。
7. 在模拟不同页面置换算法时，请选择合适的测试用例来体现内存访问的局部性原理，比如循环和递归时会访问重复的页面集，而访问数组或其它大型数据结构时会访问一大块连续的内存页面，这些情况在测试时需要考虑。
8. 分析并比较不同页面置换算法下系统的性能，需要定义一些性能指标，比如缺页中断次数、平均访问时间等。

【验收标准】

1. 基本功能的实现：能正确执行虚拟地址向物理地址的转换，正确实现了四种页面置换算法，包括LRU（最近最久未使用）、FIFO（先入先出）、CLOCK（时钟算法）和随机置换算法，能够在内存空间不足时，正确选择被替换的页面。
2. 选择合理的测试数据进行展示：能够设计合理的模拟数据，展示虚拟内存管理过程，能够通过模拟数据触发缺页置换过程，而不是直接指定并执行页面置换程序。
3. 系统的稳定性和错误处理：程序在运行过程中是否能稳定执行，是否能有效地处理错误的情况（如页面错误、地址越界错误）。
4. 用户友好度：系统应当对用户友好，易于使用。具体表现在接口设计的合理性，输出信息的清晰性等。
5. 代码质量：高质量的代码表现在清晰的组织结构，良好的命名规范，适当的注释等。良好的代码质量可以提高代码的可读性和可维护性。
6. 实验报告：一个好的实验报告应该详细描述系统的设计、实现的功能及使用方法，测试情况，以及遇到的问题、解决方式等。在实验报告中需要包含采用不同页面置换算法时系统性能的分析比较，需要采用合理的性能指标。

实验三：文件系统

【实验目的】

1. 理解文件系统的主要组成和工作原理。
2. 学习并掌握文件的存储方式，文件的创建、读写、删除等操作的实现机制。
3. 提高对文件系统内部组织和管理的认识及熟悉实际系统使用上的细节。

【实验要求】

1. 设计并实现一个可以支持树状目录的文件系统。
2. 能够执行基本的文件操作，如创建、打开、关闭、读取、写入、删除文件，创建、删除和列目录等。
3. 通过终端接口与用户交互，实现对模拟文件系统的操作。
4. 不能使用与文件操作相关的系统调用来完成该模拟文件系统的设计，需要基于自定义的文件、目录等数据结构来实现对文件、目录的管理。

【涉及的基本原理】

1. 文件系统的基本构造和功能：文件系统由文件和目录组成，带有访问权限和管理信息，实现了对底层存储设备的抽象。文件系统是操作系统用于存储、组织、共享和访问数据的一种机制。
2. 文件的存储方式：包括连续存储方式、链接存储方式和索引存储方式。
3. 文件和目录：文件系统将所有的信息存储在文件中。每个文件都有一个唯一的名字，并放置在目录中。目录本身也是文件，它包含一组链接，每个链接将一个文件名与一个文件相关联。
4. 权限和特权：文件系统为每个文件和目录都定义了一组权限，如读（r）、写（w）、执行（x）等。还可以定义谁可以使用这些权限，如文件所有者，所在组的成员，或者其他人。
5. 空间管理：文件系统管理存储设备上的可用空间。当文件被创建或扩展时，文件系统需要找到可用的空间。当文件被删除或缩小时，其空间需要被回收。
6. 文件系统的层次结构：文件系统通常使用一个树形结构来组织文件和目录。树的根是一个特殊的目录，称为根目录。每个目录可以包含文件和其他目录。
7. 硬链接和软链接：硬链接是另一个指向同一个文件的引用，而软链接（或符号链接）则是指向另一个链接的引用。使用链接可以方便地共享和组织文件。
8. 一致性和完整性：为了保证在系统崩溃或断电情况下数据的一致性和完整性，文件系统需要在修改数据时使用某种形式的原子操作或者日志。当操作失败时，可以回到操作前的状态。

【实验提示】

1. 首先确定你的文件系统的基本结构和组织方式，需要思考，文件采用什么存储方式才能更有效地既支持大文件，又支持小文件，空闲空间如何组织和管理，目录如何组织才能支持多级目录。设计合理的文件和目录的数据结构。
2. 实现文件系统的初始化函数，可以设定一个大小固定的 Block 来模拟磁盘空间，初始化时所有 Block 为空闲。
3. 编写创建文件的函数，该函数需能够为新文件分配 Block，更新文件的元数据metadata，将新文件添加到其所在目录下。
4. 编写读取和写入文件的函数，这两个函数需要通过文件的 metadata 找到文件在 Block 上的位置。在写入或删除文件时，需要考虑 Block 空间的分配和回收。
5. 编写删除文件和目录的函数，需要释放文件对应的 Block 并从所在目录中删除该文件。
6. 编写用户交互接口，可以通过终端命令来执行对文件的创建、打开、关闭、读取、写入、删除文件操作以及对目录的创建、删除、列目录等操作。在实现用户接口时，需要对输入命令进行充分的验证和错误处理。

【验收标准】

1. 功能完整性：一个好的模拟文件系统应该实现操作系统中文件系统管理的相关基础功能，这些包括文件的创建、打开、关闭、读取、写入、删除，目录的创建、删除，以及权限管理等。

2. 文件系统的组织和管理：模拟文件系统应刻画出现实文件系统中的数据组织方式。例如，如何组织文件存放、如何管理空间分配、如何划分目录等，都是评价的重要考量。不能使用操作系统提供的文件、目录相关系统调用来完成对文件和目录的操作。

3. 代码质量：代码质量是评判所设计文件系统的一项重要标准。好的代码应该简洁，清晰，易于利用和理解。应该有适当的注释，良好的命名规范，以及恰当的异常处理。

4. 错误处理能力：模拟文件系统应具备良好的错误处理能力。例如，对于非法输入应给出明确的错误提示，处于故障状态时应提供合理的恢复方案。

5. 用户界面：虽然这是一个模拟的系统，但是一个直观，简单，易用的用户界面也是评价一个文件系统好坏的一个标准。系统应该提供清晰的指导，易于理解的命令和明确的反馈。

6. 实验报告：一个好的实验报告应该详细描述系统的设计、实现的功能及使用方法，测试情况，以及遇到的问题、解决方式等。