

第零章 复习大纲须知

1. 在朋辈助学（2023 年春）第六期答疑会上说要给大家讲一期复习总结，后来因为期末考试周一再耽误，同时考虑到只是过一遍知识点不够好。于是在老师给的复习大纲的基础上融合了往年题，助力大家复习操作系统。
2. 本大纲中知识点为黑色字体标注，往年考试试题为红色和蓝色标注，蓝色标注为已解答或者参考类似解答可得到结果的题目，红色标注为暂时无法得到解答的题目（受制于时间关系和作者自身实力），期待各位同学的解答！
3. 版权须知：本复习材料仅供山东大学软件学院内部传阅，不可上传公共网络或者用于商业目的。原版 word 文件请联系作者获取（联系方式参考 9）。
4. 通过分析往年出题规律，可以发现操作系统科目的高频考点较为集中。考虑到各位同学复习时间的限制，建议优先复习如下内容（建议仅供参考，不知道会不会被背刺 QAQ）：
 - A. 第一章：硬中断，软中断，中断的作用；计算机操作系统的发展历史，多道程序设计，分时操作系统，实时操作系统；并发和并行的区别；用户态和核心态；特权指令，访管指令，陷入指令、广义指令等
 - B. 第二章：系统调用；操作系统的结构（简单结构、分层方法、微内核、模块、虚拟机）
 - C. 第三章：进程和线程的定义；进程状态转换图（重要，应知必会）；调度程序（短期，中期，长期调度）
 - D. 第四章：进程和线程的定义；用户级线程和内核级线程的定义和比较
 - E. 第五章：调度算法 [FCFS、SJF、抢占式 SJF、优先级、RR、最高响应比、多级队列]（非常重要，应知必会）；抢占式调度和非抢占式调度
 - F. 第六章：临界区和信号量的定义；同步和互斥的定义；信号量如何处理同步互斥问题 [哲学家就餐问题，信号量问题的求解]（非常重要，应知必会）
 - G. 第七章：死锁定义；产生的条件；解决方案（死锁避免、死锁预防）；银行家算法（重要）；资源分配图
 - H. 第八章：逻辑地址和物理地址绑定；内存分配方式；分页管理、分页机制、单级页表和多级页表及其设计方案、快表机制（非常重要，应知必会）
 - I. 第九章：局部性原理与缺页中断；掌握几种替换页选择方法：最佳算法、先进先出、最近最久未使用，能给出页面替换序列和次数（非常重要，应知必会）；系统颠簸[现象描述、产生原因、解决办法]（非常重要，应知必会）
 - J. 第十章：文件系统的定义及其基本概念；文件共享；文件基本操作 [打开文件]（重要）
 - K. 第十一章：FCB 定义以及 FCB 与文件之间的关系；分配方法 [连续分配、链接分配、

索引分配，各种分配方法的优缺点】(重要)；空闲空间管理(重要)

- L. 第十二章：RAID；磁盘调度算法[FCFS、SSTF、SCAN、C-SCAN、LOOK、C-LOOK]（非常重要，应知必会）
- M. 第十三章：I/O 控制方式中，轮询方式，中断方式，DMA 方式；两种方式接口：阻塞（同步）、非阻塞（异步）
- 5. 以上知识点的重要性依次为(非常重要, 应知必会)、(重要)、黑体加粗、不加粗。
- 6. 信号量机制设计的题目较难，大家可以优先学习与调度算法相关的知识点，对于所有的算法需要知道其算法原理和算法执行流程(可能需要阐述算法流程)。
- 7. 据 2020 级考试经验回顾，考试可能出现每题要写的文字很多的情况，【举个例子：什么是死锁，死锁的产生条件；比较死锁的各类解决方案及其优缺点？这种题，需要作答的东西非常多，需要大家对相应的知识点较为熟悉】，请大家合理安排作答时间。
- 8. 最后附录 1 给出朋辈助学 2023 年春操作系统 1-6 期讲义和答疑视频链接，期待大家取得优异的成绩！
- 9. 欢迎大家指出本复习大纲存在的错误和提供对于题目的解答！联系方式
QQ：2215058009/weixin:wuwenlincccc (请备注: 年级-姓名)。

附录：朋辈助学历次答疑讲义和答疑视频

答疑讲义：

见压缩包文件夹【朋辈助学 2023spring】

答疑视频：

1. 2021 级本科生学业朋辈导师辅导答疑会

(第一期)

录制回放：

主题: 线上答疑分享交流会 0426

日期: 2023-04-26 18:47:26

录制文件：

<https://meeting.tencent.com/v2/cloud-record/share?id=d3741f82-448f-4a35-a935-6b18eb670d6c&from=3>

访问密码：Ezwm

2. 2021 级本科生学业朋辈导师辅导答疑会

(第二期)

录制回放：

主题：线上答疑分享交流会 0506

日期：2023-05-06 18:56:19

录制文件：

<https://meeting.tencent.com/v2/cloud-record/share?id=56273cf8-eada-4c21-8c1e-387e51952744&from=3>

访问密码：gg4e

3. 2021 级本科生学业朋辈导师辅导答疑会

(第三期)

主题：线上答疑分享交流会 0510

日期：2023-05-10 18:44:35

录制文件：

<https://meeting.tencent.com/v2/cloud-record/share?id=f43909e1-a999-460f-b4d9-9f0ab61578f8&from=3>

访问密码：7BhN

4. 2021 级本科生学业朋辈导师辅导答疑会

(第四期)

录制回放：

主题：线上答疑分享交流会 0517

日期：2023-05-17 18:52:11

录制文件：

<https://meeting.tencent.com/v2/cloud-record/share?id=e3ce2403-eb1e-45ae-a1cd-2dd20a633575&from=3>

访问密码：cKBx

5. 2021 级本科生学业朋辈导师辅导答疑会

(第五期)

录制回放：

主题：线上答疑分享交流会 0524

日期: 2023-05-24 18:52:57

录制文件:

<https://meeting.tencent.com/v2/cloud-record/share?id=bbbaedf2-6bbc-45ae-94ae-087f2364b092&from=3>

访问密码: j6F9

6. 2021 级本科生学业朋辈导师辅导答疑会

(第六期)

录制回放:

主题: 线上答疑分享交流会 0531

日期: 2023-05-31 18:55:03

录制文件:

<https://meeting.tencent.com/v2/cloud-record/share?id=418d69d8-5d58-4090-8bdff-e603b94019e7&from=3>

访问密码: q9pO

第一章 绪论

1. 计算机系统四个组成部分

计算机硬件、操作系统、系统程序、用户。

2. 操作系统概念

操作系统是一组控制和管理计算机硬件和软件资源、合理地对各类作业进行调度，以及方便用户的程序集合。

3. 操作系统的目标

执行用户程序，更容易地解决用户问题。

使计算机系统使用方便。

有效地使用计算机硬件。

4. 两个视角

用户视角、系统视角

5. 硬中断 (interrupt)

interrupt 是一种在系统内硬件设备产生，又称外部中断（相对于 CPU）。

6. 软中断 (trap)

Trap 是指令产生的中断，又称内部中断（相对于 CPU）。

A. 请结合操作系统对资源和程序的管理和分配，阐述中断的作用？【2021】

中断的作用 中断函数的作用 -CSDN博客

B. 我们一直强调操作系统要充分利用硬件资源，有时操作系统也会违背这个原则，请举例并说明理由。【2012】

New Bing回答：仅供参考【操作系统的任务是控制和管理计算机系统的硬件和软件资源，以提供给用户和其他软件方便的接口和环境。操作系统需要充分利用硬件资源，但有时也会违背这个原则。例如，当操作系统需要保护计算机系统的安全时，它可能会限制对某些硬件资源的访问，以防止恶意软件或用户对系统造成损害。此外，当操作系统需要保证某些应用程序的稳定性时，它可能会限制对某些硬件资源的访问，以避免应用程序崩溃或出现其他问题。因此，在某些情况下，操作系统可能会违背充分利用硬件资源的原则，以保证计算机系统的安全性和稳定性。】

7. 存储结构

主存：CPU 可以直接访问的大型存储介质。

辅存：非易失性大存储容量的主存储器的扩展。（又称二级存储）

磁盘：最常用的辅存。磁盘表面逻辑上划分为磁道，再细分为扇区。磁盘控制器决定设备和计算机之间的逻辑交互。

8. DMA（直接内存访问）

用于高速 I/O 设备，能够以接近内存速度传送信息。

设备控制器无需 CPU 干预直接将数据块从缓冲存储器（设备控制器中的存储）直接传输到主存，每个块只产生一个中断，而不是每个字节的一个中断（块设备）。

9. 计算机系统

单处理器系统、多处理器系统、集群系统

10. 多处理器系统（并行系统、紧耦合系统）

定义：多处理器系统有多个紧密通信的 CPU，它们共享计算机总线，有时还有时钟、内存和外设等。

A. 什么是IPC？名词解释？【2020】

[科普 通俗易懂解释处理器IPC的含义 - 知乎 \(zhihu.com\)](#)

优点：增加吞吐量、规模经济、增加可靠性。

分类：

非对称多处理：每个处理器都有各自特定的任务。一个主处理器控制系统，其他处理器或者向主处理器要任务或做预先定义的任务。

对称多处理：每个处理器都要完成操作系统中的所有任务。所有处理器对等，处理器之间没有主-从关系。

11. 集群系统

定义：与多处理器系统一样，集群系统将多个 CPU 集中起来完成计算任务。然而，集群系统与多处理器系统不同，它是由两个或多个独立的系统耦合起来的。集群计算机共享存储并通过局域网络连接或更快的内部连接。

12. 操作系统结构

操作系统最重要的一点是要有多道程序处理能力。多道程序设计通过组织作业（编译或数据）使 CPU 总有一个作业在执行，从而提高了 CPU 的利用率。

13. 操作系统的三种基本类型

Batch systems (批处理系统)

Time-sharing systems (分时系统)

Real time systems (实时系统)

A. 什么是并发 (Concurrent) ? 【2012】

[并发与并行的区别 \(超级通俗易懂 CSDN博客\)](#)

14. 批处理系统

工作方式：

用户将作业交给系统操作员，系统操作员将许多用户的作业组成一批作业(jobs)之后输入到计算机中，在系统中形成一个自动转接的连续的作业流，系统自动、依次执行每个作业。最后由操作员将作业结果交给用户。操作系统：自动将控制从一个任务转到下一个任务。

分类：单道批处理系统、多道批处理系统

批处理操作系统优点：作业流程自动化、效率高、吞吐量高。

批处理操纵系统缺点：无交互手段、调试程序困难。

15. 分时系统- 交互式计算

A. 什么是 Time-Sharing Operating System? 名词解释? 【2020, 2022, 2012 改】

TimeSharing:

- 手工操作：手动输入输出，人机矛盾明显
- 批处理系统（单道批处理/多道批处理系统）：解决系统利用率低的问题，多道批处理系统时期形成了早期的操作系统
- 分时操作系统：人机交互
 - TimeSharing：所谓分时技术，是指把处理器的运行时间分成很短的时间片，按时间片轮流把处理器分配给各联机作业使用。若某个作业在分配给它的时间片内不能完成其计算，则该作业暂时停止运行，把处理器让给其他作业使用，等待下一轮再继续运行。由于计算机速度很快，作业运行轮转得也很快，因此给每个用户的感觉就像是自己独占一台计算机。
 - vs：分时操作系统：分时操作系统是指多个用户通过终端同时共享一台主机，这些终端连接在主机上，用户可以同时与主机进行交互操作而互不干扰。因此，实现分时系统最关键的问题是如何使用户能与自己的作业进行交互，即当用户在自己的终端上键入命令时，系统应能及时接收并及时处理该命令，再将结果返回用户。分时系统也是支持多道程序设计的系统，但它不同于多道批处理系统。多道批处理是实现作业自动控制而无须人工干预的系统，而分时系统是实现人机交互的系统，这使得分时系统具有与批处理系统不同的特征。
- 实时操作系统：处理紧急任务，在限制时间内完成某项任务
- 网络/分布式Windows Server/个人计算机操作系统

Multiprogramming:

- 即多道程序设计，指在内存中同时存放若干个作业，并使它们同时运行的一种程序设计技术。在单处理机环境下，仅在宏观上这些作业在同时运行，而在微观上它们是在交替执行。即每一时刻只有一个作业在执行，其余作业或处于阻塞状态，或处于就绪状态。

分时系统（或多任务）是多道程序设计的延伸。

共享需要一种交互计算机系统，它能提供用户与系统之间的直接通信。响应时间短（通常小于一秒钟）。

由于每个动作或命令都较短，每个用户只需少量 CPU 时间，用户之间切换时间短，所以用户会感觉整个系统为自己所用。

16. 实时系统

定义：实时操作系统是保证在一定时间限制内完成特定功能的操作系统。

分类：

硬实时系统：硬实时要求在规定的时间内必须完成操作，这是在操作系统设计时保证的（不按时间约束完成意味着失败）。（又称强实时系统）

软实时系统：软实时则只要按照任务的优先级，尽可能快地完成操作即可，（不按时完成意味着“不好”）。（又称准实时系统、若实时系统）。

A. Multiprogramming: 【2019】

即多道程序设计，指在内存中同时存放若干个作业，并使它们同时运行的一种程序设计技术。在单处理机环境下，仅在宏观上这些作业在同时运行，而在微观上它们是在交替执行。即每一时刻只有一个作业在执行，其余作业或处于阻塞状态，或处于就绪状态。

17. 操作系统的双重模式操作

为了区分操作系统代码和用户定义代码的执行，至少需要两种独立的操作模式：用户模式（用户态）、监督程序模式（管理模式、系统模式、特权模式、核心态）。操作系统保护能力实现的必要条件。

将能引起损害的机器指令作为特权指令。用户模式下想要执行特权指令，硬件不会执行，会认为是非法指令，并以陷阱的形式通知操作系统。

A. 什么是Privileged Instruction? 名词解释？【2020】

特权指令、访管指令、陷入指令、广义指令- CSDN博客

本章应掌握主要内容：

1. 计算机系统硬件结构
2. 操作系统定义
3. 应用、操作系统、硬件的层次关系
4. 多道批处理、分时系统、实时系统的定义及意义。
5. 用户态与核心态（管态、监督模式、系统态）的理解。

第二章 操作系统结构

1. 操作系统服务

用户界面（一种是命令行界面；另一种是批界面，最为常用的是图形用户界面）、程序执行、I/O 操作、文件系统操作、通信、错误检测、资源分配、统计、保护和安全。

2. 操作系统的用户界面：

命令解释程序（CLI）、图形用户界面（GUI）

命令解释程序主要作用：获取并执行用户指定的下一条命令。

系统调用（System Call）

操作系统内核提供一系列预定功能，通过一组称为系统调用的接口呈现给编程人员，系统调用把应用程序的请求传给内核，系统调用相应的内核函数完成所需的处理，将处理结果返回给应用程序。

A. 系统调用：【2019, 2018, 2015, 2012】

运行在使用者空间的程序向操作系统内核请求需要更高权限运行的服务。系统调用提供了用户程序与操作系统之间的接口。大多数系统交互式操作需求在内核态执行。或：操作系统中一组用于实现各种功能的子程序，用户在应用程序中可以通过系统调用命令调用它们

3. 向操作系统传递参数的三种方法：

通过寄存器来传递参数。

若参数数量比寄存器多，参数通常存在内存的块和表中，并将块的地址通过寄存器来传递。

参数也可以通过程序放在或压入堆栈中，并通过操作系统弹出。

4. 系统程序分类：

文件管理、状态信息、文件修改、程序语言支持、程序装入和执行、通信。

5. 操作系统设计和实现：

设计目标需求：用户目标和系统目标

机制和策略：机制决定如何做，策略决定做什么

6. 操作系统结构

简单结构、分层方法、微内核、模块、虚拟机

1. 简单结构

MS-DOS、原始的 UNIX 操作系统

2. 分层方法

定义：操作系统分成若干层（级）。最底层（层 0）为硬件，最高层（层 N）为用户接口。（理想方法，困难在于如何划分层）

3. 微内核

微内核方法将所有非基本部分从内核中移走，并将它们实现为系统或用户程序，这样得到了更小的内核。

微内核的主要功能是使客户程序和运行在用户空间的各种服务之间进行通信。

4. 模块：

大多数现代操作系统按模块方式实现内核

采用面向对象的方法

每个核心组件是分开的

每部分与已知接口的其他部分通信

可以是动态加载方式，每部分根据需要加载到内核

总之，类似于层，但更灵活。

5. 虚拟机

虚拟机（VirtualMachine）指通过软件模拟的具有完整硬件系统功能的、运行在一

个完全隔离环境中的“完整”计算机系统。

- A. 什么是虚拟机？【2012改】
- B. 什么是Microkernels？【2012改】
- C. 请解释使用Time Sharing 和Multiprogramming的意义，请比较Layered Approach， MicroKernal， Virtual Machine的特点和区别。【2022】

Layered Approach: 分层设计方法

- 分层法是将操作系统分为若干层，最底层（层0）为硬件，最高层（层N）为用户接口，每层只能调用



MicroKernel: 微内核

- 微内核构架，是指将内核中最基本的功能保留在内核，而将那些不需要在核心态执行的功能移到用户态执行，从而降低内核的设计复杂性。那些移出内核的操作系统代码根据分层的原则被划分成若干服务程序，它们的执行相互独立，交互则都借助于微内核进行通信。

Virtual Machine:

1/16

- 第一种是“虚拟主机”的概念，第二种是“运行环境”的概念。这两种概念是不同的。

“虚拟机”在作“虚拟主机”讲的时候，指的是操作系统内安装另一个操作系统。你如在Windows里，你可以装一个VMWare，然后在VMWare里安装一个Ubuntu Linux。

“虚拟机”在作“运行环境”讲的时候，指的是操作系统被安装一个支持其他软件包运行的软件。最明显的例子就是“Java虚拟机”

操作系统结构			
	特性、思想	优点	缺点
分层结构	内核分多层，每层可单向调用更低一层提供的接口	<ul style="list-style-type: none"> ① 1. 便于调试和验证，自底向上逐层调试验证 2. 易扩充和易维护，各层之间调用接口清晰固定 	<ul style="list-style-type: none"> 1. 仅可调用相邻低层，难以合理定义各层的边界 ② 2. 效率低，不可跨层调用，系统调用执行时间长
模块化	<p>将内核划分为多个模块，各模块之间相互协作。</p> <p>内核 = 主模块+可加载内核模块 ● 主模块：只负责核心功能，如进程调度、内存管理 可加载内核模块：可以动态加载新模块到内核，而无需重新编译整个内核</p>	<ul style="list-style-type: none"> 1. 模块间逻辑清晰易于维护，确定模块间接口后即可多模块同时开发 2. 支持动态加载新的内核模块（如：安装设备驱动程序、安装新的文件系统模块到内核），增强 OS适应性 ③ 3. 任何模块都可以直接调用其他模块，无需采用消息传递进行通信，效率高 	<ul style="list-style-type: none"> 1. 模块间的接口定义未必合理、实用 2. 模块间相互依赖，更难调试和验证
宏内核（大内核）	所有的系统功能都放在内核里（大内核结构的OS通常也采用了“模块化”的设计思想）	<ul style="list-style-type: none"> ④ 1. 性能高，内核内部各种功能都可以直接相互调用 	<ul style="list-style-type: none"> ⑤ 1. 内核庞大功能复杂，难以维护 2. 大内核中某个功能模块出错，就可能导致整个系统崩溃
微内核	只把中断、原语、进程通信等最核心的功能放入内核。进程管理、文件管理、设备管理等功能以用户进程的形式运行在用户态	<ul style="list-style-type: none"> ⑥ 1. 内核小功能少、易于维护，内核可靠性高 2. 内核外的某个功能模块出错不会导致整个系统崩溃 	<ul style="list-style-type: none"> ⑦ 1. 性能低，需要频繁的切换用户态/核心态。用户态下的各功能模块不可以直接相互调用，只能通过内核的“消息传递”来间接通信 2. 用户态下的各功能模块不可以直接相互调用，只能通过内核的“消息传递”来间接通信
外核 (exokernel)	内核负责进程调度、进程通信等功能，外核负责为用户进程分配未经抽象的硬件资源，且由外核负责保证资源使用安全	<ul style="list-style-type: none"> ⑧ 1. 外核可直接给用户进程分配“不虚拟、不抽象”的硬件资源，使用户进程可以更灵活的使用硬件资源 ⑨ 2. 减少了虚拟硬件资源的“映射层”，提升效率 	<ul style="list-style-type: none"> 1. 降低了系统的一致性 2. 使系统变得更复杂

D. 虚拟机：【2019, 2017】

第一种是“虚拟主机”的概念，第二种是“运行环境”的概念。这两种概念是不同的。虚拟机在作“虚拟主机”讲的时候，指的是操作系统内安装另一个操作系统。你如在Windows里，你可以装一个VMWare，然后在VMWare里安装一个Ubuntu Linux。虚拟机在做“运行环境”讲的时候，指的是操作系统被安装一个支持其他软件包运行的软件。最明显的例子就是“Java虚拟机”。或者解释为：通过软件模拟的、具有完整硬件系统功能的、运行在一个完全隔离环境中的完整计算机系统。编程用到的java虚拟机有自己完善的硬件架构，比如处理器、堆栈、寄存器等，还具有相应的指令系统。Java虚拟机屏蔽了与具体平台相关的操作信息

本章应掌握主要内容：

1. 操作系统组成部分
2. 系统调用、系统调用参数传递方式、系统调用分类
3. 操作系统内核结构（单核+模块、分层结构、微内核、虚拟机、模块化结构、混合结构）不同结构的基本思想和优缺点。
4. 本章概念总结

稍微有点综合题目的意思：

A. 操作系统需要考虑哪些调度，这些调度目标和方法：【2018】

结合进程调度，内存调度，磁盘调度三者作答即可。

[操作系统中的调度算法](#) [操作系统调度算法](#) -CSDN博客

第三章 进程

1. 进程概念

进程是执行中的程序。进程还包括当前活动信息：通过程序计数器的值和处理器寄存器的内容等来表示。另外，进程内存影像包括进程堆栈段（包括临时数据，如函数参数、返回地址和局部变量）和数据段（包括全局变量），包括堆，是在进程运行期间动态分配的内存。

进程与程序是截然不同的两个概念

一个程序可以对应多个进程 (One program can be several processes)

一个进程也可以由多个程序段共同完成一项任务（接力）

进程五个特征（而程序不具备）

动态性：是进程最基本的特征，表现在进程由创建而产生，由调度而执行，因得不到资源而暂停执行，由撤销而消亡。

并发性：多个进程实体同存于内存中，能在一段时间内同时执行。

独立性：进程实体是一个能独立运行的基本单位，同时也是系统中独立获得资源和独立调度的基本单位。

异步性：指进程按各自独立的、不可预知的速度向前推进；或者说，进程按异步方式运行。

结构特征：从结构上看，进程实体由程序段、数据段以及进程控制块组成。三部分也称为进程影像。

进程状态

新建 (new)：进程正在被创建。

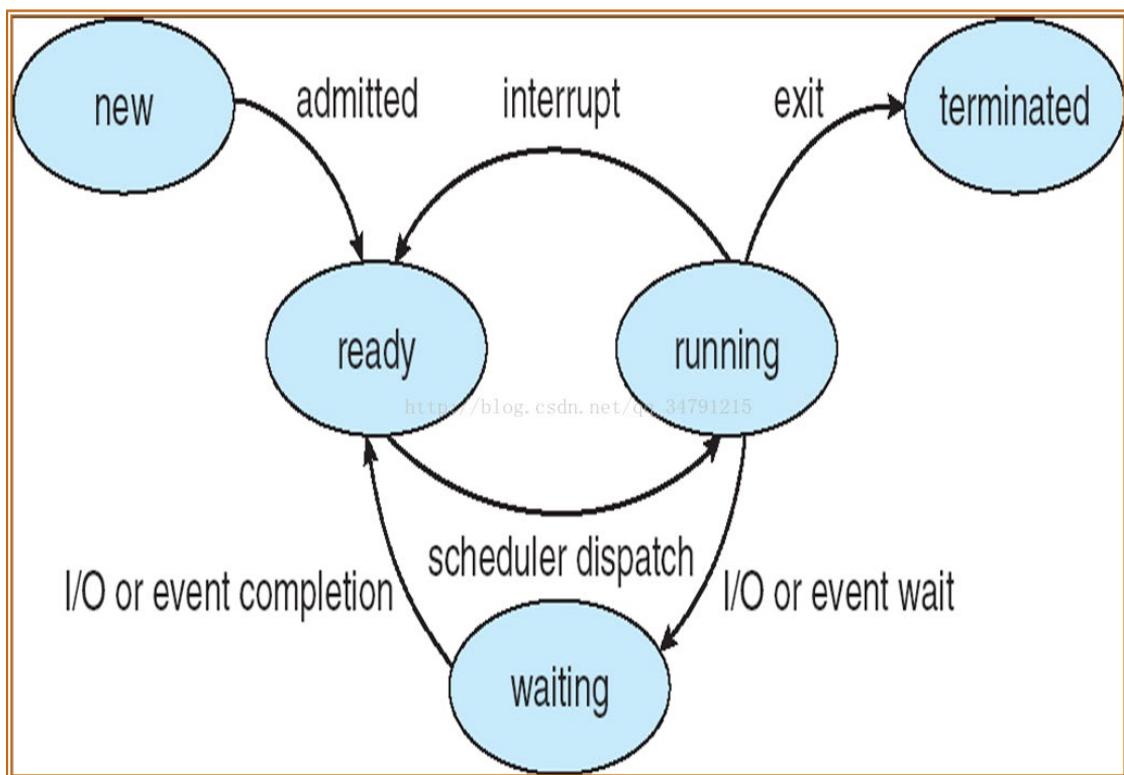
运行 (running)：指令正在被执行。

等待 (waiting)：进程等待某个时间的发生（如 I/O 完成或收到信号）。

就绪 (ready)：进程等待分配处理器。

终止 (terminated)：进程完成执行。

进程状态图 “”



- A. 请解释进程的状态转化图，并解释每种状态发生的变化原因；请解释在一个线程 blocking 时，是否能够运行相应的进程，为什么？【2022】
- B. 画出进程的状态图（new、ready、running、wait、terminated），并标注转化条件。【2015】
- C. 画出进程NEW、READY、RUNNING、WAITING、TERMINATED的状态图，并说明状态之间变换的原因。【2020】

2. 画出进程 NEW、READY、RUNNING、WAITING、TERMINATED 的状态图，并说明状态之间变换的原因；
请解释在一个线程 blocking 时，是否能够运行相应的进程，为什么？(2022)



- 状态转换图如上图所示：
- 创建态->就绪态：进程正在被创建，尚未转到就绪态。创建进程需要多个步骤：首先申请一个空白 PCB，并向 PCB 中填写用于控制和管理进程的信息；然后为该进程分配运行时所必须的资源；最后把该进程转入就绪态并插入就绪队列。但是，如果进程所需的资源尚不能得到满足，如内存不足，则创建工作尚未完成，进程此时所处的状态称为创建态。
- 运行态->结束态：进程正从系统中消失，可能是进程正常结束或其他原因退出运行。进程需要结束运行时，系统首先将该进程置为结束态，然后进一步处理资源释放和回收等工作。
- 就绪态->运行态：处于就绪态的进程被调度后，获得处理器资源（分派处理器时间片），于是进程由就绪态转换为运行态。
- 运行态->就绪态：处于运行态的进程在时间片用完后，不得不让出处理器，从而进程由运行态转换为就绪态。此外，在可剥夺的操作系统中，当有更高优先级的进程就绪时，调度程序将正在执行的进程转换为就绪态，让更高优先级的进程执行。
- 运行态->阻塞态：进程请求某一资源（如外设）的使用和分配或等待某一事件的发生（如 I/O 操作的完成）时，它就从运行态转换为阻塞态。进程以系统调用的形式请求操作系统提供服务，这是一种特殊的、由运行用户态程序调用操作系统内核过程的形式。
- 阻塞态->就绪态：进程等待的事件到来时，如 I/O 操作结束或中断结束时，中断处理程序必须把相应进程的状态由阻塞态转换为就绪态。

当一个线程 Blocking 时，能否运行相应的进程：

有些系统使用组合方式的多线程实现。在组合实现方式中，内核支持多个内核级线程的建立、调度和管理，同时允许用户程序建立、调度和管理用户级线程。一些内核级线程对应多个用户级线程，这是用户级线程通过时分多路复用内核级线程实现的。同一进程中的多个线程可以同时在多处理器上并行执行，且在阻塞一个线程时不需要将整个进程阻塞，所以组合方式能结合 KLT（内核级线程）和 ULT（用户级线程）的优点，并且克服各自的不足。

进程控制块

每个进程在操作系统内对应一进程控制块 (PCB, process control block)。

概念：

系统为了管理进程设置的一个专门的数据结构，用它来记录进程的外部特征，描述进程的运动变化过程。

要点：

系统利用 PCB 来控制和管理进程，所以 PCB 是系统感知进程存在的唯一标志

进程与 PCB 是一一对应的

通常进程队列是进程所对的 PCB 队列

操作系统通过 PCB 来感知进程的存在

PCB 表：

系统把所有 PCB 组织在一起，并把它们放在内存的固定区域，就构成了 PCB 表。PCB 表的大小决定了系统中最多可同时存在的进程个数，称为系统的并发度。

结构：

链接结构：同一状态进程的 PCB 组成一个链表，不同状态的进程对应多个不同的链表。（就绪链表、阻塞链表……）

索引结构：对具有相同状态的进程，分别设置各自的 PCB 索引表，表明 PCB 在 PCB 表中的地址。

...

2. 进程调度

多道程序设计的目的是无论何时都有进程在运行，从而使 CPU 利用率达到最大化。

分时系统的目的是在进程之间快速切换 CPU 以便用户在程序运行时能与其进行交互。

为达到此目的，进程调度选择一个可用的进程到 CPU 上执行。单处理器系统从不会有超过一个进程在运行。如果有多个进程，那么余下的则需要等待 CPU 空闲并重新调度。

3. 调度程序

长期调度程序（或作业调度程序）

短期调度层序（或 CPU 调度程序）

这两个调度程序的主要差别是它们执行的频率。

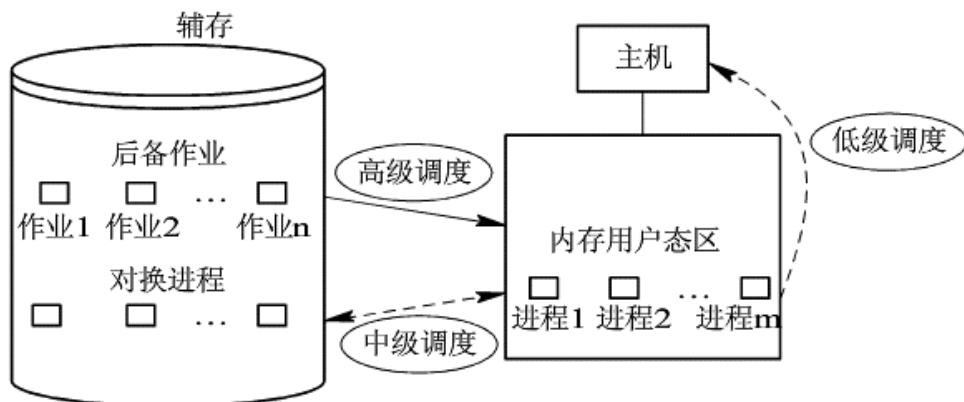
短期调度程序必须频繁为 CPU 选择新进程。

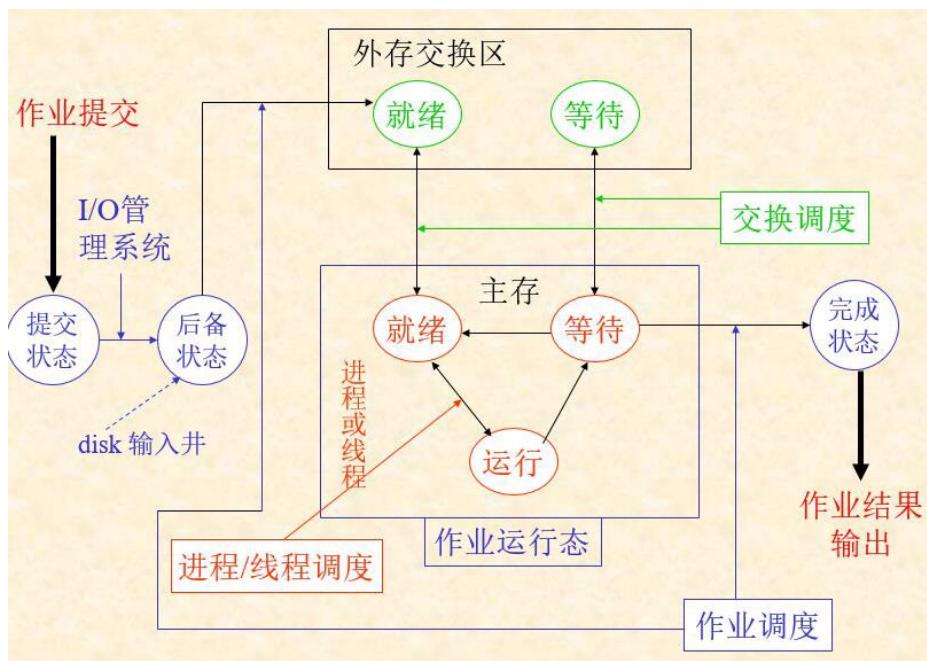
长期调度程序控制多道程序设计的程度（内存中的进程数量）。

中期调度程序（又名交换）核心思想：调度程序根据一定策略临时把内存中的某个进程换出到外存或者把外存中的进程换入内存的过程。（一定程度上降低长期调度的设计或策略难度）。

A. 对换技术(swapping): 【2018, 2012】

把内存中处于等待状态（或者被剥夺 CPU 执行权力的进程）从内存移到外存，腾出内存空间，这一过程叫换出。再把外存中处于后备队列且已经准备好竞争 CPU 的进程调入内存，这一过程叫做换入，中级调度（也称 CPU 调度）采用的就是这种方式来扩大虚拟内存的





- A. 解释可执行程序从外存调入后，运行，结束的全过程，并具体描述这当中发生了什么事？【2021】

答：见下文，三级调度。

短期调度执行的频率最高。短期调度在内存作业中选择就绪执行的作业，并为他们分配CPU。中期调度作为一种中等程度的调度程序，尤其被用于分时系统，一个交换方案的实施，将部分运行程序移出内存，之后，从中断处继续执行，从而改善进程组合或者因内存要求的改变引起了可用内存的过度使用而需要释放内存。长期调度确定哪些作业调入内存以执行。它们主要的不同之处是它们的执行的频率。短期调度必须经常调用一个新进程，由于在系统中，长期调度处理移动的作业时，并不频繁被调用，可能在进程离开系统时才被唤起。

4. 上下文切换

将CPU切换到另一个进程需要保存当前进程的状态并恢复另一个进程的状态，这一任务称为上下文切换。

5. 进程操作

进程标识符 (pid)

通过 fork() 系统调用，可创建新进程。

新进程通过复制原来的地址空间形成。

子进程返回 0，父进程返回大于 0，错误返回 -1.

6. Fork() 的功能

内核为子进程做一个父进程的上下文的拷贝(复制父进程的 PCB 作为子进程的 PCB)，子进程与父进程共享子进程创建之前父进程所有的资源，父进程和子进程在不同的地址空间上运行。

7. Fork() 的几个要点

父子进程具有独立的内存空间。

父子进程资源的共享与分离：父进程中在 fork 之前创建的变量——先继承，后分离，子进程继承了父进程的私有变量，作为自己的私有变量；Fork 之后各自创建的变量——完全分离。

子进程继承了父进程的所有资源，其中包括父进程的这些私有变量，但继承以后互相不能访问。

进程间通信有两种基本模式

共享内存：允许以最快的速度进行方便的通信，在计算机中可以达到内存的速度，等等。

消息传递：对于交换较少数量的数据很有用，因为不需要避免冲突，对于计算机间的通信，消息传递也比共享内存更易于实现。

A. 写出下面程序的输出结果，并解释这样输出的原因。【2020】

```
int a = 0;
int main()
{
    int pid = fork();
    if (pid == 0)
    {
        a = 2;
        print("child leaving/n");
    }
    else
    {
        wait(NULL);
        print("a=%d/n". a);
    }
}
```

参考这篇文章理解：[进程 第二天 \(fork 函数&子进程与父进程&守护进程\) -CSDN 博客](#)

B. 给出一段代码，父进程创建子进程，子进程创建两个线程，分析3个输出语句应

该输出的值并说明原因。【2017】

4. 某进程创建了一个子进程，子进程中又创建了两个线程，程序的大致结构如下：

```
void *Runner1(void *param);
void *Runner2(void *param);
Int main(int argc;char *argv[])
{
    pthread_t tid1,tid2;
    pthread_attr_t attr1,attr2;
    int value=5;
    int pid=fork();
    If(pid=0)
```

```
{
    pthread_attr_init(&attr1);
    pthread_create(&tid1,&attr1,Runner1,NULL);
    pthread_join(tid1,NULL);
    printf("\nCHILD:Runner1:value=%d\n",value); //输出 1

    pthread_attr_init(&attr1);
    pthread_create(&tid2,&attr2,Runner2,NULL);
    pthread_join(tid2,NULL);
    printf("\nCHILD:Runner2:value=%d\n",value); //输出 2
}
else
if(pid>0)
{
    wait(NULL);
    [rintf("parent:value=%d\n",value); //输出 3
}
void *Runnre1(void *param)
{
    value=value+3;
    Pthread_exit(0);
}
Void *Runner2(void *param)
{
    value =value+3;
    Pthread_exit(0);
}
```

请问上述代码执行时，三条输出语句（输出 1、输出 2、输出 3）分别输出的结果是什么，说明你的理由。

参考答案：略；请参考 A 题

第四章 线程

1. 引入线程的原因：进程时空开销大、通信代价大、不能很好的利用多处理器系统、不适合并行计算和分布计算的要求。

2. 线程的定义：

进程中的一个实体、CPU 调度和分派的基本单位、与同进程内的其它线程共享进程所拥有的资源：线程必须在某个进程内执行，它所需的其它资源，如代码段、数据段、打开的文件和信号等，都由它所属进程拥有。

只单独拥有运行所必须的资源：如 PC、寄存器、栈

3. 线程的优点：并发程度高、响应度高；易于调度，开销小；资源共享；多处理器体系结构的利用。

4. 进程和线程的区别

一个进程可以有多个线程，但至少有一个线程；而一个线程只能在一个进程的地址空间内活动。

每当创建一个进程时，至少要同时为该进程创建一个线程，否则该进程无法被调度执行。

地址空间和其他资源（如打开文件）：进程间相互独立，同一进程的各线程间共享——某进程内的线程在其他进程不可见。

通信：进程间通信采用 IPC，线程间可以直接读写进程数据段（如全局变量）来进行通信；所有线程可共享进程的主存，不需要特殊的通信机制。

调度：线程上下文切换比进程上下文切换要快得多。

OS 中引入进程目的：使多个程序并发执行，以便改善资源使用率和提高系统效率。

OS 中引入线程目的：减少程序并发执行时所付出的时空开销，并发性更好。

A. 给出一段来自维基百科thead safty的英文，问进程、线程概念，以及为什么没有process safty（= =我把第二段翻译了一边，凑了2点）【2019】

1. 进程：程可以定义为“可与其他程序并发执行的程序J在一个数据集合上的运行过程”。进程具有动态性、并发性、独立性、异步性和结构特征

2. 线程：在引入线程的操作系统中，线程是进程中的一个实体，是被操作系统独立调度和分派的基本单位。线程自己基本上不拥有资源，

3. 只拥有在运行中必不可少的资源，如程序计数器、一组寄存器和栈。但线程可与同属于一个进程的所有进程共享进程所拥有的全部资源。

4. 一个线程可以创建和撤销另一个线程。同一进程中的线程可以并发执行

B. 进程和线程概念特点？【2018】

C. 什么是线程？【2013】

D. 什么是Thread？【2015】

又称轻量级进程，是CPU执行的基本单元，由线程ID，PC，寄存器集合和堆栈组成，线程自己不拥有系统资源，只拥有运行时必不可少的一点资源。一个进程包含一个或多个线程，这些线程可以共享进程的资源。线程也有就绪、阻塞和运行三种基本状态。

E. 写出一种使用Java语言创建线程的方法（用Java程序描述）。【2012】

E题：[java中创建线程的三种方法以及区别 - Bruce-Lee - 博客园 \(cnblogs.com\)](#)

5. 用户级线程

用户线程的维护由应用进程通过线程库来完成；

线程库：应用进程利用线程库提供创建、同步、调度和管理线程的函数来控制用户线程，无需内核支持。

特点：

内核不了解用户线程的存在；

用户线程切换不需要内核特权；

速度快。线程的创建和调度由应用软件内部进行，无需用户态/核心态切换，所以速度特别快。

优点：

线程切换不调用核心；

调度是应用程序特定的：可以选择最好的算法；

ULT 可运行在任何操作系统上（只需要线程库）。

6. 内核线程

有关线程的所有管理工作都在 OS 内核完成，应用程序部分没有线程管理的代码，只有一个到内核线程的 API

线程切换由内核完成；

内核维护进程和线程的上下文信息；

优点：

一个线程发起系统调用而阻塞，不会影响其它线程的运行，内核可以继续调度调度同一个进程中的另一个线程；

内核可以将同一进程中的多个线程调度到多个处理器上。

缺点：

由于有内核的参与，需要额外开销；

线程之间的切换需要内核的模式切换。

A. 用户级线程和内核级线程是什么？相对的各自有什么优点？【2016】

1. 用户级线程由应用程序管理，操作系统内核感知不到用户线程的存在
2. 在内核级线程中，管理工作由内核完成，应用程序没有进行线程管理的代码，只有到内核线程的编程接口。
3. 应用程序可以使用线程库设计成多线程程序，提高程序执行效率
4. 内核级线程相较于进程来说，创建的开销较小，且线程之间容易切换、方便通信

1. 用户级线程和内核级线程是什么？相对的各自有什么优点？[2016]父进程创建子进程和主程序调用子程序有何不同？[进程管理]

- 用户级线程和内核级线程是什么？相对的各自有什么优点？
- 用户级线程由应用程序管理，操作系统内核感知不到用户线程的存在，在内核级线程中，管理工作由内核完成，应用程序没有进行线程管理的代码，只有到内核线程的编程接口。
- 应用程序可以使用线程库设计成多线程程序，提高程序执行效率，内核级线程相较于进程来说，创建的开销较小，且线程之间容易切换、方便通信

- 父进程创建子进程后，父进程与子进程同时执行（并发）。主程序调用子程序后，主程序暂停在调用点，子程序开始执行，直到子程序返回，主程序才开始执行。

B. 什么是User Thread？名词解释？【2020】

C. 用户级线程、内核级线程：【2019】

1. 在引入线程的操作系统中，线程是进程中的一个实体，是被操作系统独立调度和分派的基本单位。
2. 线程自己基本上不拥有资源，只拥有在运行中必不可少的资源，如程序计数器、一组寄存器和栈。
3. 但线程可与同属于一个进程的所有进程共享进程所拥有的全部资源。一个线程可以创建和撤销另一个线程。同一进程中的线程可以并发执行。

7. 多线程模型

多对一模型：将多个用户级线程映射到一个内核线程。

一对一模型：将每个用户线程映射到一个内核线程。一个线程阻塞时，另一个线程可以继续执行。

多对多模型：可以创建任意多的必要用户线程，且相应内核线程能在多处理器系统中并发执行；

一个线程阻塞时，另一个线程可以继续执行。

8. 线程池：

在进程建立时就创建若干线程，将这些线程放在一个“池”中等待工作。当服务器接收到一个请求时，就唤醒池中一个线程，并将要处理的请求传递给它；一旦线程完成了任务，它会返回到池中再等待其它的工作；如果池中没有可用的线程，服务器就会一直等待，指导有空闲线程为止。

优点：用现有线程处理请求通常比等待创建新线程快；线程池限定了任何时候可存在线程的数量。

第五章 CPU 调度

1. 调度类型

1) 长期(长程、高级)调度：从外存的后备队列中选择一个或者多个作业调入内存，

并为它们创建进程，分配必要的资源。创建→就绪/挂起；创建→就绪。

2) 中期(中程、中级)调度：将进程的部分或全部加载到内存中，提高内存利用率。
进程状态变化(通过执行挂起和激活操作)：就绪/挂起<→就绪；阻塞/挂起<→阻塞。

3) 短期(短程、低级、CPU)调度：选择哪个进程在处理机上执行，执行最频繁)。
进程状态：就绪<→运行。

2. 抢占/非抢占、

1) 非抢占(非剥夺、协作)调度：就绪进程不可以从运行进程手中抢占 CPU。一旦进程处于运行状态，它就不断执行直到终止或者为等待 I/O 或请求某些操作系统服务而阻塞自己，才把 CPU 让给别人。

2) 抢占(剥夺)调度：就绪进程可以从运行进程手中抢占 CPU。允许调度程序根据某种策略中止当前运行进程的执行，将其转移到就绪状态，并选择另一个进程投入运行。

A. 操作系统如何防止一个用户进程无限制地使用CPU，而保证它对CPU的控制？

【2012】

解答：从CPU调度算法，尤其是抢占式调度算法方面作答即可。

B. 抢占：【2017】

当一个进程正在处理机上面执行时，若有一个更为紧迫的进程申请使用CPU，则立刻暂停当前进程的执行，保存当前程序执行状态后将CPU的使用权交给更为紧迫的进程

C. preemptive scheduling：【2019】

当一个进程正在执行时，调度程序基于某种原则，剥夺已分配给该进程的处理机，将它分配给其他进程并使之执行。剥夺的原则有：

(1) 优先权原则；(2) 短进程优先原则；(3) 时间片原则等

3. 调度程序的功能

- 1) 保存现场：记录放弃 CPU 的进程 A 的现场信息(如 PC，通用寄存器的内容等)。
- 2) 选择进程：当进程出让 CPU 或调度程序剥夺执行状态进程占用的 CPU 时，选择适当的进程 B 分派 CPU。
- 3) 完成上下文切换：用户态执行进程 A 代码，进入 OS 内核(通过时钟中断或系统调用)；保存进程 A 的上下文，载入进程 B 的上下文(CPU 寄存器和一些表格的当前指针)；用户态执行进程 B 代码。

4. 调度准则

面向用户的准则：响应时间(min)；周转时间(结束时间-进入系统时间)(min)；优先级。

面向系统的准则吞吐量(max)；CPU 利用率(max)；公平；资源的平衡使用；系统开销(min)。

5. 调度算法

FCFS、SJF、抢占式 SJF、优先级、RR、最高响应比。给出进程到达机描述，掌握各种算法的调度顺序及个指标计算。

首先熟悉算法流程：[CPU 调度算法——FCFS 算法/SJF 算法/优先级调度算法/RR 算法](#)
[-CSDN 博客](#)

A. FCFS 在 CPU 调度、磁盘调度、页置换中分别的优缺点？【2013】

参考：熟悉 CPU 调度算法，磁盘调度算法，页面置换算法作答即可。

New Bing 的回答：

【FCFS 调度算法按照进程请求 CPU 的先后顺序进行调度，即先到先服务。这种算法比较公平，但是它通常并不提供最快的服务。在磁盘调度中，FCFS 算法根据进程请求访问磁盘的先后顺序进行调度，这是一种最简单的调度算法。该算法的优点是具有公平性。如果只有少量进程需要访问，且大部分请求都是访问簇聚的文件扇区，则有望达到较好的性能；但如果大量进程竞争使用磁盘，那么这种算法在性能上往往接近于随机调度。在页置换中，FCFS 算法也是一种最简单的置换算法。当需要置换一页时，选择最先进入内存的页面进行置换。该算法的优点是实现简单，缺点是没有考虑页面是否常用等因素，因此可能会出现“抖动”现象】

B. 进程调度，SJF 的抢占，非抢占，RR (time=2) 算法？【2018】

参考：熟悉 SJF，RR，非抢占式算法的计算过程。

C. 解释多级队列调度和多级反馈队列调度算法的不同点，分析算法的优点在什么地方？【2021】

1. [进程管理]解释多级队列调度和多级反馈队列调度算法的不同点，分析算法的优点在什么地方(比较多级队列和多级反馈队列的算法思想，并分析对比好处与坏处).

- 多级队列调度算法：前述的各种调度算法，由于系统中仅设置一个进程的就绪队列，即调度算法是固定且单一的，无法满足系统中不同用户对进程调度策略的不同要求。在多处理机系统中，这种单一调度策略实现机制的缺点更为突出，多级队列调度算法能在一定程度上弥补这一缺点。该算法在系统中设置多个就绪队列，将不同类型或性质的进程固定分配到不同的就绪队列。每个队列可实施不同的调度算法，因此，系统针对不同用户进程的需求，很容易提供多种调度策略。同一队列中的进程可以设置不同的优先级，不同的队列本身也可以设置不同的优先级。在多处理机系统中，可以很方便为每个处理机设置一个单独的就绪队列，每个处理机可实施各自不同的调度策略，这样就能根据用户需求将多个线程分配到一个或多个处理机上运行。

- 多级反馈队列调度算法的实现思想如下：

- 1) 设置多个就绪队列，并为每个队列赋予不同的优先级。第1级队列的优先级最高，第2级队列的优先级次之，其余队列的优先级逐个降低。

- 2) 赋予各个队列的进程运行时间片的大小各不相同。在优先级越高的队列中，每个进程的时间片就越小。例如，第*i+1*级队列的时间片要比第*i*级队列的时间片长1倍。

- 3) 每个队列都采用FCFS算法。当新进程进入内存后，首先将它放入第1级队列的末尾，按FCFS原则等待调度。当轮到该进程执行时，如它能在该时间片内完成，便可撤离系统。若它在一个时间片结束时尚未完成，调度程序将其转入第2级队列的末尾等待调度；若它在第2级队列中运行一个时间片后仍未完成，再将它放入第3级队列...，依此类推。当进程最后被降到第*n*级队列后，在第*n*级队列中便采用时间片轮转方式运行。

- 4) 按队列优先级调度。仅当第1级队列为空时，才调度第2级队列中的进程运行；仅当第1~*i-1*级队列均为空时，才会调度第*i*级队列中的进程运行。若处理机正在执行第*i*级队列中的某进程时，又有新进程进入任一优先级较高的队列，此时须立即把正在运行的进程放回到第*i*级队列的末尾，而把处理机分配给新到的高优先级进程。

多级反馈队列的优势有以下几点：

- 1) 终端型作业用户：短作业优先。

- 2) 短批处理作业用户：周转时间较短。

- 3) 长批处理作业用户：经过前面几个队列得到部分执行，不会长期得不到处理。

D. 给定下面的作业顺序：

进程号	到达时间	运行时间
P1	0.0	7.0
P2	1.0	4.0
P3	2.0	1.0
P4	3.0	4.0

请使用最短作业优先算法画出甘特图，计算平均等待时间；

请使用最短剩余时间优先算法画出甘特图，计算平均等待时间。【2022】

4. 给定下面的作业顺序：

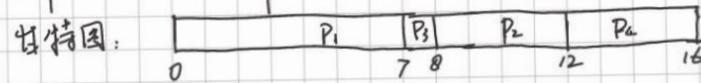
进程号	到达时间	运行时间
P1	0.0	7.0
P2	1.0	4.0
P3	2.0	1.0
P4	3.0	4.0

(1) 请使用最短作业优先算法画出甘特图，计算平均等待时间；

(2) 请使用最短剩余时间优先算法画出甘特图，计算平均等待时间。[2022]

JF算法：

4. SJF算法(不是抢占式算法)



等待时间：P1: 0

$$P_2: 12 - 1 = 11$$

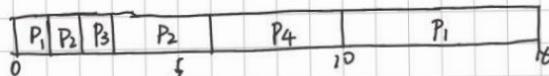
$$P_3: 8 - (2+1) = 5$$

$$P_4: 12 - 3 = 9$$

平均: $\frac{11}{4}$

最短剩余时间优先算法：

甘特图(抢占式)



对于 P1: 等待: $10 - 1 = 9$

$$P_2: \text{等待: } 6 - 5 = 1 \quad \text{平均: } \frac{13}{4}$$

$$P_3: \text{等待: } 0$$

$$P_4: \text{等待: } 10 - (3+4) = 3.$$

E. 根据进程的到达和执行时间，画出相应算法的甘特图，并求出平均等待时间。

进程	到达时间	执行时间
P1	0	3
P2	1	2
P3	2	3

进程	到达时间	执行时间
P4	3	6

(1) 抢占式最短作业优先调度

(2) 轮转法调度 (时间片大小为 2)。【2020】 解答过程类似 D 题。

F. 甘特图、平均等待时间

1. 甘特图法观察图像用CPU利用时间/总工作时间=CPU的利用率；
2. 在甘特图中如果各个占用线是连续的则此进程在运行过程中无等待现象，否则就有等待现象，即等待现象发上在间断处。
 - 1) 抢占最短作业
 - 2) 轮转 (时间片=5) 【2019】

参考解答：熟悉甘特图画法，熟悉抢占式算法，时间片轮转算法工作原理。

G. 给出一组进程，使用抢占式SJF调度，写出进程的调度顺序并计算平均等待时间？

【2017】

3. 设有四个进程，到达就绪队列时间及执行时间如下，若采用剥夺式(抢先式)最短作业优先调度，geichu8 各进程的调度次序及平均等待时间，给出计算过程。

进程到达及执行时间：

进程	到达就绪队列时间	执行时间
P1	0	6
P2	1	8
P3	2	3
P4	3	12

H. 【2016】年试题

8. 定义进程的响应比为（等待时间+运行时间）/（运行时间）。高响应比进程调度算法就是将响应度高的进程先进入运行状态。现在有 4 个进程，如下图，求高响应比进程调度下的平均周转时间。

进程	到达时间	运行时间
J1	8.0	2.0
J2	8.6	0.6
J3	8.8	0.2
J4	9.0	0.5

I. 以下是四个进程的到达时间和运行时间。

	到达时间	运行时间
P1	0	12
P2	1	8
P3	2	3
P4	3	6

分别画出 FIFO 和 SJF 调度的甘特图，并计算平均等待时间。【2015】

参考：熟悉 FIFO 算法，SJF 算法，甘特图。

第六章 进程同步

1. 进程同步（直接要求同步）：指系统中一些进程需要相互合作，共同完成一项任务。具体说，一个进程运行到某一点时要求另一伙伴进程为它提供消息，在未获得消息之前，该进程处于等待状态，获得消息后被唤醒进入就绪态。

2. 互斥（间接要求同步）：由于各进程要求共享资源，而有些资源需要互斥使用，因此各进程间竞争使用这些资源，进程的这种关系为进程的互斥。

3. 一些相关概念：

互斥：指多个进程不能同时使用同一个资源；

死锁：指多个进程互不相让，都得不到足够的资源；

饥饿：指一个进程一直得不到资源（其他进程可能轮流占用资源）；

临界资源：系统中某些资源一次只允许一个进程使用，称这样的资源为临界资源或互斥资源或共享变量；

临界区：进程中访问临界资源的一段代码。

A. 临界区及其解决方案条件的解释？【2018, 2013】

[操作系统-实现临界区互斥的基本办法 -CSDN博客](#)

B. 解释临界区需要满足的条件，解释信号量，并阐述信号量如何实现（满足）了这些条件？【2021】

临界区问题的解决方案要满足三条要求

互斥：如果Pi在临界区执行，则其他进程不能在其中执行

同步：如果没有进程在临界区执行，并且有进程要进入临界区，那么只有那些不在剩余区的进程可以参加选择，并且这种选择不能无限推迟

有限等待：从一个进程要求进入临界区到这个请求被允许，其他进程不能无休止地进入其临界区

互斥也称间接制约关系。当一个进程进入临界区使用临界资源时，另一个进程必须等待，当占用临界资源的进程退出临界区后，另一进程才允许去访问此临界资源。

例如，在仅有一台打印机的系统中，有两个进程 A 和进程 B，若进程 A 需要打印时，系统已将打印机分配给进程 B，则进程 A 必须阻塞。一旦进程 B 将打印机释放，系统便将进程 A 唤醒，并将其由阻塞态变为就绪态。

为禁止两个进程同时进入临界区，同步机制应遵循以下准则：

- 1) 空闲让进。临界区空闲时，可以允许一个请求进入临界区的进程立即进入临界区。
- 2) 忙则等待。当已有进程进入临界区时，其他试图进入临界区的进程必须等待。
- 3) 有限等待。对请求访问的进程，应保证能在有限时间内进入临界区。
- 4) 让权等待。当进程不能进入临界区时，应立即释放处理器，防止进程忙等待。

3. 利用信号量实现同步

信号量机制能用于解决进程间的各种同步问题。设 S 为实现进程 P₁, P₂同步的公共信号量，初值为 0。进程 P₂中的语句 y 要使用进程 P₁中语句 x 的运行结果，所以只有当语句 x 执行完成之后语句 y 才可以执行。其实现进程同步的算法如下：

```

semaphore S=0;           //初始化信号量
P1 () {
    x;                  //语句 x
    V(S);               //告诉进程 P2, 语句 x 已经完成
    ...
}
P2 () {
    ...
    P(S);              //检查语句 x 是否运行完成
    y;                  //检查无误，运行 y 语句
    ...
}

```

若 P₂先执行到 P(S)时，S 为 0，执行 P 操作会把进程 P₂阻塞，并放入阻塞队列；当进程 P₁

中的 x 执行完后，执行 V 操作，把 P₂ 从阻塞队列中放回就绪队列，当 P₂ 得到处理机时，就得以继续执行。

4. 利用信号量实现进程互斥

信号量机制也能很方便地解决进程互斥问题。设 S 为实现进程 P₁, P₂ 互斥的信号量，由于每次只允许一个进程进入临界区，所以 S 的初值应为 1(即可用资源数为 1)。只需把临界区置于 P(S) 和 V(S) 之间，即可实现两个进程对临界资源的互斥访问。其算法如下：

```

semaphore S=1;           // 初始化信号量
P1() {
    ...
    P(S);           // 准备开始访问临界资源，加锁
    进程 P1 的临界区;
    V(S);           // 访问结束，解锁
    ...
}
P2() {
    ...
    P(S);           // 准备开始访问临界资源，加锁
    进程 P2 的临界区;
    V(S);           // 访问结束，解锁
    ...
}

```

当没有进程在临界区时，任意一个进程要进入临界区，就要执行 P 操作，把 S 的值减为 0，然后进入临界区；当有进程存在于临界区时，S 的值为 0，再有进程要进入临界区，执行 P 操作时将会被阻塞，直至在临界区中的进程退出，这样便实现了临界区的互斥。

互斥是不同进程对同一信号量进行 P, V 操作实现的，一个进程成功对信号量执行了 P 操作后进入临界区，并在退出临界区后，由该进程本身对该信号量执行 V 操作，表示当前没有进程进入临界区，可以让其他进程进入。

下面简单总结一下 PV 操作在同步互斥中的应用：在同步问题中，若某个行为要用到某种资源，则在这个行为前面 P 这种资源一下；若某个行为会提供某种资源，则在这个行为后面 V 这种资源一下。在互斥问题中，P, V 操作要紧夹使用互斥资源的那个行为，中间不能有其他冗余代码。



C. 什么是Critical Section? 名词解释？【2020】

进程中访问临界资源的一段代码。

D. Race Condition: 【2019】

计算机运行过程中，并发、无序、大量的进程在使用有限、独占、不可抢占的资源，由于进程无限，资源有限，产生矛盾，这种矛盾称为竞争（Race）由于两个或者多个进程竞争使用不能被同时访问的资源，使得这些进程有可能因为时间上推进的先后原因而出现问题，这叫做竞争条件（Race Condition）

竞争条件分为两类：

Mutex (互斥)：两个或多个进程彼此之间没有内在的制约关系，但是由于要抢占使用某个临界资源（不能被多个进程同时使用的资源，如打印机，变量）而产

生制约关系。

Synchronization (同步)：两个或多个进程彼此之间存在内在的制约关系（前一个进程执行完，其他的进程才能执行），如严格轮转法。

E. 什么是Race Condition? 【2015】

首先了解竞争的概念，计算机运行过程中，大量、并发、无序的进程使用有限、独占、不可抢占的资源，由于进程无限、资源有限所产生的这种矛盾叫做竞争。

竞争条件则指两个或多个进程竞争使用不能被同时访问的资源，使得这些进程可能因为时间上推进的先后原因而出现问题，这叫做竞争条件。

不同事务在并发执行过程中产生冲突

4. 使用临界区应遵循的准则：

空闲（有空）让进：当无进程在临界区时，任何有权使用临界区的进程可进入；

互斥（无空等待）：不允许两个以上的进程同时进入临界区；

多中择一：当没有进程在临界区，而同时有多个进程要求进入临界区，只能让其中之一进入临界区，其他进程必须等待；

有限等待：任何进入临界区的要求应在有限的时间内得到满足；

让权等待：处于等待状态的进程应放弃占用 CPU；

平等竞争：任何进程无权停止其它进程的运行进程之间相对运行速度无硬性规定。

5. 互斥的实现-硬件方法

(1) 中断禁用（关中断， Interrupt Disabling）

如果进程访问临界资源时(执行临界区代码)不被中断，就可以利用它来 保证互斥地访问。

过程：

关中断原语；

临界区

开中断原语

其余部分

(2) 专门的机器指令：设计一些机器指令，用于保证两个动作的原子性，如在一个指令周期中实现测试和修改。

6. 信号量

A. 什么是Semaphore? 名词解释？【2020, 2018】

信号量：一种功能较强的通信机制，可用来解决互斥和同步问题，只能被两个标准的原语操作所访问即wait(S), signal(S)，也可写作p(S), v(S)

B. 描述信号量的数据结构，给出信号量方式中P操作和V操作的过程描述。【2012】

信号量机制是一种功能较强的机制，可用来解决互斥与同步问题，它只能被两个标准的原语 wait(S) 和 signal(S) 访问，也可记为“P 操作”和“V 操作”。

原语是指完成某种功能且不被分割、不被中断执行的操作序列，通常可由硬件来实现。例如，前述的 Test-and-Set 和 Swap 指令就是由硬件实现的原子操作。原语功能的不被中断执行特性在单处理机上可由软件通过屏蔽中断方法实现。原语之所以不能被中断执行，是因为原语对变量的操作过程若被打断，可能会去运行另一个对同一变量的操作过程，从而出现临界段问题。

1. 整型信号量

整型信号量被定义为一个用于表示资源数目的整型量 S，wait 和 signal 操作可描述为

```
wait(S) {
    while (S<=0);
    S=S-1;
}
signal(S) {
    S=S+1;
}
```

在整型信号量机制中的 wait 操作，只要信号量 $S \leq 0$ ，就会不断地测试。因此，该机制并未遵循“让权等待”的准则，而是使进程处于“忙等”的状态。

C. 管程：【2019】

一种程序结构，结构内的多个子程序（对象或模块）形成的工作线程互斥访问共享资源。这些共享资源一般是硬件设备或一群变量。管程实现了在一个时间点，最多只有一个线程在执行管程的某个子程序。管程提供了一种机制，线程可以临时放弃互斥访问，等待某些条件得到满足后，重新获得执行权恢复它的互斥访问。

初始化指定一个非负整数值，表示空闲资源总数（又称为“资源信号量”）

若为非负值：表示当前的空闲资源数 ($s.\text{count} \geq 0$ 可用的资源数 :)

若为负值：其绝对值表示当前等待临界区的进程数 ($|s.\text{count}|$ 为等待的进程数)

操作：操作系统对信号量只能通过初始化和两个标准的原语来访问。对信号量的操作只有三种

原子操作：

初始化：通常将信号量的值初始化为非负整数。

P 操作 (wait 操作)：使信号量的值减 1 (申请一个单位的资源 ($s.\text{count}--$))

如果使信号量的值变成负数，则执行 P 操作的进程被阻塞 (当 $s.\text{count} < 0$ 时，资源已分配完毕，进程自己阻塞在 S 的队列上——让权等待)

V 操作 (signal 操作)：使信号量的值加 1 (释放一个单位资源 ($s.\text{count}++$))

如果信号量的值不是正数，则使一个因执行 v 操作被阻塞的进程解除阻塞 (若 $s.\text{count} \leq 0$ ，则唤醒一个等待进程)。

A. 什么是哲学家就餐问题？利用信号量写出解决哲学家就餐问题的同步程序。

【2012】

[五个哲学家就餐问题 信号量解决5位哲学家就餐问题 -CSDN博客](#)

B. 信号量问题。假设操场上共有22个名额，有两个体育活动A和B，规定当在操场上

的人数不大于22时可以参与活动，否则需要等待。如果A的人数比B的人数多5人以上，参加A活动的需要等待；同理，如果B的人数比A的人数多5人以上，参加B活动的需要等待。参加A和B活动的可随时退出。根据“参与A”“退出A”“参与B”“退出B”和相应的信号量写出伪代码。【2017】

3. 有一个活动场地最多可容纳 22 名同学参加活动，其中一部分同学参与打篮球活动，不妨设为活动 A，另一部分同学参与大羽毛球活动，不妨设为活动 B。规定如下：

- (1) 若活动场地中同学人数已经超过 22 人，则申请进入活动场地的同学等待。
- (2) 参与 A、B 类活动的同学人数之差不能超过 5 人，即若参与活动 A 的人数比参与活动 B 的人数多 5 人，则申请参与活动 A 的同学等待；若参与过的 B 的人数比参与 A 的人数多 5 人，则申请参与活动 B 的同学等待。
- (3) 参与 A、B 活动的同学可以随时离开。

请用“参与 A”、“参与 B”、“离开 A”、“离开 B”及信号量机制描述同学进入活动场地的过程。

解答：参考C题

C. 朋辈助学讲义试题

4. [进程管理]在一个仓库中可以存放A和B两种产品，要求：

- ①每次只能存入一种产品。
- ②A产品数量-B产品数量 < M。
- ③B产品数量-A产品数量< N。

其中，M,N是正整数，试用 P操作、V操作描述产品 A 与产品 B 的入库过程。

[解答] 使用信号量mutex控制两个进程互斥访问临界资源（仓库），使用同步信号量Sa和Sb(分别代表产品A与B的还可容纳的数量差，以及产品B与A的还可容纳的数量差) 满足条件2和条件3。代码如下：

```

Semaphore Sa=M-1,Sb=N-1;
Semaphore mutex=1;
//访问仓库的互斥信号量
process A(){
    while(1){
        P(Sa);
        P(mutex);
        A产品入库;
        V(mutex);
        V(Sb);
    }
}
process B(){
    while(1){
        P(Sb);
        P(mutex);
        B产品入库;
        V(mutex);
        V(Sa);
    }
}

```

D. 朋辈助学讲义试题

2. 设自行车生产线上有一个箱子，其中有 N 个位置 ($N \geq 3$)，每个位置可存放一个车架或一个车轮，又设有 3 名工人，其活动分别为：

1 工人1活动:	2 do{ 3 加工一个车架;	4 车架放入箱中;
5 }while(1)	do{ 5 箱中取一个车架;	6 箱中取二个车轮;
	7 组装为一台车;	8 }while(1)

试分别用信号量与 PV 操作实现三名工人的合作，要求解中不含死锁。[进程互斥同步]

- 用信号量与 PV 操作实现三名工人的合作。首先不考虑死锁问题，工人 1 与工人 3、工人 2 与工人 3 构成生产者与消费者关系，这两对生产/消费关系通过共同的缓冲区相联系。从资源的角度来看，箱子中的空位置相当于工人 1 和工人 2 的资源，而车架和车轮相当于工人 3 的资源。
- 分析上述解法易见，当工人 1 推进速度较快时，箱中空位置可能完全被车架占满或只留有一个存放车轮的位置，此时工人 3 同时取 2 个车轮将无法得到，而工人 2 又无法将新加工的车轮放入箱中；当工人 2 推进速度较快时，箱中空位置可能完全被车轮占满，而此时工人 3 取车架将无法得到，而工人 1 又无法将新加工的车架放入箱中。上述两种情况都意味着死锁。为防止死锁的发生，箱中车架的数量不可超过 $N-2$ ，车轮的数量不可超过 $N-1$ ，这些限制可以用两个信号量来表达。具体解答如下：

```

semaphore empty=N;           //空位置
semaphore wheel=0;           //车轮
semaphore frame=0;           //车架
semaphore s1=N-2;             //车架最大数
semaphore s2=N-1;             //车轮最大数

工人 1 活动:
do{
    加工一个车架;
    P(s1);                  //检查车架数是否达到最大值
    P(empty);                //检查是否有空位
    车架放入箱中;
    V(frame);                //车架数加 1
}while(1);

工人 2 活动:
do{
    加工一个车轮;
    P(s2);                  //检查车轮数是否达到最大值
    P(empty);                //检查是否有空位
    车轮放入箱中;
    V(wheel);                //车轮数加 1
}while(1);

工人 3 活动:
do{
    P(frame);                //检查是否有车架
    箱中取一车架;
    V(empty);                //空位数加 1
    V(s1);                   //可装入车架数加 1
    P(wheel);                //检查是否有一个车轮
    P(wheel);                //检查是否有另一个车轮
    箱中取二车轮;
    V(empty);                //取走一个车轮，空位数加 1
    V(empty);                //取走另一个车轮，空位数加 1
    V(s2);                   //可装入车轮数加 1
    V(s2);                   //可装入车轮数再加 1
    组装为一台车;
}while(1);

```

```

V(empty);                  //空位数加 1
V(s1);                     //可装入车架数加 1
P(wheel);                  //检查是否有一个车轮
P(wheel);                  //检查是否有另一个车轮
箱中取二车轮;
V(empty);                  //取走一个车轮，空位数加 1
V(empty);                  //取走另一个车轮，空位数加 1
V(s2);                     //可装入车轮数加 1
V(s2);                     //可装入车轮数再加 1
组装为一台车;
}while(1);

```

E. 见下图，仓库改成无限，去掉要求 1。【2018】

例题：有一个仓库，可以存放 A 与 B 两种产品，仓库的存储空间足够大，但要求：

- ①每次只能存入一种产品（A 或 B）；
- ② $-N < A$ 产品数量 - B 产品数量 $< M$ ；

其中，N 和 M 是正整数。

试用“存放 A”和“存放 B”和 wait、signal 描述产品 A 与产品 B 的入库过程。

解析：每次存放 A 以前都要检查 $A-B$ 是否已经超过 $M-1$ 了，存放 A 结束以后告知 $B-A$ 这个数需要减一，也就是如果 B 原本已经到达上限，那么现在就又可以继续放 B 了。B 的存放同理。现在将 $A-B$ 和 $B-A$ 抽象为两个信号量。

semaphore

mutex=1

sa=M-1

sb=N-1;

Process Input_A:

while (true){

Get a product A;

wait(sa);

wait(mutex);

// put product A into the

// depository;

signal(mutex);

signal(sb); }

}

Process Input_B :

while (1) {

Get a product B;

wait(sb);

wait(mutex);

// put product B into the

// depository;

signal(mutex);

signal(sa);

}

https://blog.csdn.net/Jemary_

分

F. (信号量机制问题) 现在有三个人小华，小明，小亮，运送疫情防控物资口罩，没人运送两百个口罩，他们在骑行运送的过程中需要经过一座桥(七里桥)，小华需要最先过桥，小明需要最后过桥，同时只能有一个人过桥，其中一个人过桥之后不需要等待后面的人，如果在自己前面的人还没过桥，自己则需要等待前面的人过桥之后才能过桥。到达目的地后，三个人分别在白板上更新物资数量，请根据上述描述，利用信号量机制，wait() 和 signal() 原语，藐视每个人所工作的全部流程。(题目做了相应转述，大意不变) 【2022】

3. 现有5个操作A、B、C、D和E,操作C必须在A和B完成后执行, 操作E必须在C和D完成后执行, 请使用信号量的wait()、signal()操作(P、V操作)描述上述操作之间的同步关系, 并说明所用信号量及其初值。[2020统考真题]

3-1: (信号量机制问题) 现在有三个人小华, 小明, 小亮, 运送疫情防控物资口罩, 每人运送两百个口罩, 他们在骑行运送的过程中需要经过一座桥(七里桥), 小华需要最先过桥, 小明需要最后过桥, 同时只能有一个人过桥, 其中一个人过桥之后不需要等待后面的人, 如果在自己前面的人还没过桥, 自己则需要等待前面的人过桥之后才能过桥。到达目的地后, 三个人分别在白板上更新物资数量, 请根据上述描述, 利用信号量机制, wait()和signal()原语, 描述每个人所工作的全部流程。(题目做了相应转述, 大意不变)(2022)

明晰一下什么是互斥和同步: (思想参考拓扑排序?)

- 互斥: 某一资源同时只允许一个访问者对其进行访问, 具有性和排它性。但互斥无法限制访问者对资源的访问顺序, 即访问是无序的。
- 同步: 互斥的基础上, 通过其它机制实现访问者对资源的有序访问。在大多数情况下, 同步已经实现了互斥。

本题不涉及对于临界资源的访问, 是单纯的同步问题, 对访问顺序做出的要求。

- 本题要求实现操作的先后顺序, 没有互斥关系, 是一个简单的同步问题。本题虽然有5个操作, 但是只有4个同步关系, 因此分别设置信号量SAC、SBC、SCE和SDE对应4个同步关系。

```
1 | Semaphore SAC = 0;
2 | //控制A和C的执行顺序
3 | Semaphore SBC = 0;
4 | //控制B和C的执行顺序
5 | Semaphore SCE = 0;
6 | //控制C和E的执行顺序
7 | Semaphore SDE = 0;
8 | //控制D和E的执行顺序
```

- 5个操作可以描述为如下方式：

```

1 | Begin
2 | A() {
3 |   完成动作A;
4 |   V(SAC); //实现A、C之间的同步关系
5 |
6 |
7 | B() {
8 |   完成动作B;
9 |   V(SBC); // 实现B、C之间的同步关系
10|
11| C() {
12|   // C必须在A、B都完成后才能完成
13|   P(SAC);
14|   P(SBC);
15|   完成动作C;
16|   V(SCE); //实现C、E之间的同步关系
17|
18| D() {
19|   完成动作D;
20|   V(SDE);
21|   //实现D、E之间的同步关系
22|
23| E() {
24|   //E必须在完成C、D之后执行
25|   P(SCE);
26|   P(SDE);

```

4/16

```

27 |   完成动作E;
28 |
29 | End

```

G. 甲乙丙三人疫情结束后一起聚餐，吃完饭后，三人一起看电影。用伪代码写出三人一起行动的过程，并且要防止死锁的发生。【2020】

解答：

H. 信号量：两个人，一人拿黑子，一人拿白子，差不超过M，一次只能一只手伸入棋盘【2019】

另一个回忆版：临界区互斥的题，比较简单

围棋问题：黑子与白子混在一起，利用两个进程分开。一个进程拣白子，另一个进程拣黑子。要求：

(1) 每次只能有一个捡子

(2) 白子的数量M，黑子的数量N，要求 $M-N>X, N-M>X$ (白子不能比黑子多多少个，黑子不能比白子多多少个)。【2019】

例题：有一个仓库，可以存放 A 与 B 两种产品，仓库的存储空间足够大，但要求：

- ①每次只能存入一种产品（A 或 B）；
- ② $N \geq A$ 产品数量 - B 产品数量 $\leq M$ ；

其中，N 和 M 是正整数。

试用“存放 A”和“存放 B”和 wait、signal 描述产品 A 与产品 B 的入库过程。

解析：每次存放 A 以前都要检查 A-B 是否已经超过 M-1 了，存放 A 结束以后告知 B-A 这个数需要减一，也就是如果 B 原本已经到达上限，那么现在就又可以继续放 B 了。B 的存放同理。现在将 A-B 和 B-A 抽象为两个信号量。

```

semaphore
mutex=1
sa=M-1
sb=N-1;
Process Input _A:
while (true){
    Get a product A;
    wait(sa);
    wait(mutex);
    // put product A into the
    // depository;
    signal(mutex);
    signal(sb); }
Process Input _B :
while (1) {
    Get a product B;
    wait(sb);
    wait(mutex);
    // put product B into the
    // depository;
    signal(mutex);
    signal(sa); }
https://blog.csdn.net/Jemary\_
```

- I. 从前有座山，山上有座庙，山下有口井。庙里小和尚需要挑水。有人舞担，有人拿桶，有人诵挑水秘诀。挑水时，三个和尚必须一人持担，一人拿桶、一人诵挑水秘诀(同时进行)后方能挑水。每个和尚都是先喜欢诵诀，其次持担、其次持桶。

请写出信号量和相关伪代码。【2015】

解答：

- J. 信号量问题：一个收银员，10张凳子。顾客来了之后，如果有空位则取号入座，收银员空闲则叫号处理，写伪代码？【2013】

解答：把座位看作临界资源，同步关系为：入座处理和叫号处理。

- K. 有一个食堂，最多容许200人同时就餐。食堂仅有一个门，该门可以同时进出三个人。试分析就餐者之间的同步关系，并用 P、V 操作描述之。【2012】

解答：需要注意食堂为临界资源，互斥访问，进入3个人和出去三个人需要通过同步信号量控制即可。

需要掌握基本概念以及用信号量等机制解决同步问题。

第七章 死锁

1. 定义：一组进程中，每个进程都无限等待被该组进程中另一进程所占有的资源，因而永远无法得到资源，这种现象称为进程死锁，这一组进程就称为死锁进程。

- 2、产生原因：

资源不足导致的资源竞争：多个进程所共享的资源不足，引起它们对资源的竞争而

产生死锁。

并发执行的顺序不当。进程运行过程中，请求和释放资源的顺序不当，而导致进程死锁。如 P, V

操作的顺序不当。

3、四个必要条件：

互斥条件：指进程对所分配到的资源进行排它性使用，即在一段时间内某资源只能由一个进程占有。如果此时还有其它进程申请该资源，则它只能阻塞，直至占有该资源的进程释放。

占有且等待（请求和保持条件）：进程已经保持了至少一个资源，但又提出了新的资源要求，而该资源又已被其它进程占有，此时请求进程阻塞，但又对已经获得的其它资源保持不放。

非抢占（非剥夺）条件：进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放。

循环等待条件：在发生死锁时，必然存在一个进程-资源的封闭的环形链。即进程集合 {P0, P1, P2, …, Pn} 中的 P0 正在等待一个 P1 占用的资源；P1 正在等待 P2 占用的资源，…，Pn 正在等待已被 P0 占用的资源。

4、处理方法：

预防死锁：通过限制如何申请资源的方法来确保至少有一个条件不成立。

避免死锁：根据有关进程申请资源和使用资源的额外信息，确定对于一个申请，进程是否应该等待。

检测死锁和恢复：通过算法来检测并恢复。

忽视此问题：认为死锁不可能在系统内发生。如 Unix 采用这种方法。

5、死锁避免：不需象死锁预防那样，事先采取限制措施破坏产生死锁的必要条件；在资源的动态分配过程中，采用某种策略防止系统进入不安全状态，从而避免发生死锁。

定义：在系统运行过程中，对进程发出的每一个系统能够满足的资源申请进行动态检查，并根据检查结果决定是否分配资源，若分配后系统可能发生死锁，则不予分配，否则予以分配。

A. 什么是安全状态？【2013】

安全状态：系统能按某种顺序，如<P1, P2, „, Pn>，为每个进程分配所需资源，直到最大需求，使每个进程都可顺序完成，称系统处于安全状态。

只要系统处于安全状态，必定不会进入死锁状态；死锁状态是不安全状态。

不安全状态不一定是死锁状态（不安全状态可能导致死锁）。

B. 简述阻塞、饥饿、死锁的区别。【2015】

死锁，活锁，饥饿，阻塞，无锁-CSDN博客

C. 什么是死锁？产生死锁的条件是什么？请给出死锁的解决方案并说明各解决方案的优缺点？【2022】

解答：定义略，参考知识点3：四个必要条件；处理方法请看知识点4.

[死锁和产生死锁的四个必要条件以及如何避免和预防死锁_产生死锁的必要条件-CSDN 博客](#)

D. 什么是死锁？【2016, 2012】

指在一个进程集合中，进程处于等待状态且等待的事件永远不会发生

E. 简述死锁的避免、死锁预防并比较区别（6分）【2015】

死锁的预防是系统预先确定一些资源分配策略，破坏产生死锁的四个条件（互斥条件，占用和等待条件，不剥夺条件，循环等待条件），进程按规定申请资源，系统按预先规定的策略进行分配，从而防止死锁的发生。

而死锁的避免是当进程提出资源申请时系统测试资源分配，仅当能确保系统安全时才把资源分配给进程，使系统一直处于安全状态之中，从而避免死锁。

6、安全状态：系统能按某种顺序，如 $\langle P_1, P_2, \dots, P_n \rangle$ ，为每个进程分配所需资源，直到最大需求，使每个进程都可顺序完成，称系统处于安全状态。

只要系统处于安全状态，必定不会进入死锁状态；死锁状态是不安全状态。

不安全状态不一定是死锁状态（不安全状态可能导致死锁）。

7、银行家算法：

$Available[j]$: 尚未分配的资源 j 的数量；

$Max[i, j]$ ($Claim[I, j]$): 进程 i 对资源 j 的最大需求量；

$Allocation[i, j]$: 进程 i 获得的资源 j 的数量；

$Need[i, j]$: 进程 i 尚需的资源 j 的数量。

[操作系统--银行家算法详解 -CSDN 博客](#)

A. 朋辈助学试题

5. 假设具有5个进程的进程集合 $P=\{P_0, P_1, P_2, P_3, P_4\}$,系统中有三类资源A,B,C,假设在某时刻有如下状态:

	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	0	3	0	0	4	1	4	0
P_1	1	0	0	1	7	5			
P_2	1	3	5	2	3	5			
P_3	0	0	2	0	6	4			
P_4	0	0	1	0	6	5			

当前系统是否处于安全状态? 若系统中的可利用资源Available为(0,6,2), 系统是否安全? 若系统处在安全状态, 请给出安全序列; 若系统处在非安全状态, 简要说明原因。[银行家算法]

银行家算法: 死锁避免策略, 防止系统进入不安全状态

- 根据Need矩阵可知, 初始Work等于Available为(1,4,0), 可以满足进程 P_2 的需求; 进程 P_2 结束后释放资源, Work为(2,7,5), 可以满足 P_0, P_1, P_3 和 P_4 中任一进程的需求, 所以系统不会出现死锁, 当前处于安全状态。 P_2, P_0, P_1, P_3, P_4 (P_2 在首位, 余下任意)

$$\text{Need} = \text{Max} - \text{Allocation} = \begin{bmatrix} 0 & 0 & 4 \\ 1 & 7 & 5 \\ 2 & 3 & 5 \\ 0 & 6 & 4 \\ 0 & 6 & 5 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 3 \\ 1 & 0 & 0 \\ 1 & 3 & 5 \\ 0 & 0 & 2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 7 & 5 \\ 1 & 0 & 0 \\ 0 & 6 & 2 \\ 0 & 6 & 4 \end{bmatrix}$$

- 若初始Work=Available为(0,6,2), 可满足进程 P_0, P_3 的需求; 这两个进程结束后释放资源, Work为(0,6,7), 仅可满足进程 P_4 的需求: P_4 结束后释放资源, Work为(0,6,8), 此时不能满足余下任一进程的需求, 系统出现死锁, 因此当前系统处在非安全状态。

$P_0 -> P_3 -> P_4 -> ?$

- 注意: 在银行家算法中, 实际计算分析系统安全状态时, 并不需要逐个进程进行。如本题中, 在(1)的情况下, 当计算至进程 P_2 结束并释放资源时, 系统当前空闲资源可满足余下任一进程的最大需求量, 这时已经不需要考虑进程的执行顺序。系统分配任意一个进程所需的最大需求资源, 在其执行结束释放资源后, 系统当前空闲资源会增加, 所以余下的进程仍然可以满足最大需求量。因此, 在这里可以直接判断系统处于安全状态。在(2)的情况下, 系统当前可满足进程 P_0, P_3 的需求, 所以可以直接让系统推进到 P_0, P_3 执行完并释放资源后的情形, 这时系统出现死锁, 由于此时是系统空闲资源所能达到的最大值, 所以按照其他方式推进, 系统必然还是出现死锁。因此, 在计算过程中, 将每步中可满足需求的进程作为一个集合, 同时执行并释放资源, 可以简化银行家算法的计算。

B. 题目给出了Available, Request, Allocation矩阵, 判断当前状态下是否存在死锁, 如果存在, 是哪些进程死锁。类似课本上的这个图【2021】

解答: 会计算即可

	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0	0	0	0	0	0
P_1	2	0	0	2	0	2			
P_2	3	0	3	0	0	0			
P_3	2	1	1	1	0	0			
P_4	0	0	2	0	0	ps:2	log.csdn.net/weixin_46841376		

C. 是否死锁, 哪些死锁【2019】

	allocation	need	available
p1	0, 1, 0	0, 1, 0	0, 0, 0
p2	2, 0, 0	1, 0, 0	
p3	0, 0, 3	0, 0, 0	
p4	2, 1, 1	2, 1, 0	
p5	0, 0, 2	0, 0, 2	

D. 使用死锁检测算法判断是否有死锁？如果有死锁，那么哪些进程死锁？【2020】

L. 进程	Allocation	Request	Available
	A B C	A B C	A B C
P0	2 0 0	忘了	0 0 0
P1	忘了	忘了	
P2	1 0 0	0 0 0	
P3	忘了	0 1 0	
P4	忘了	忘了	

8、死锁恢复方法：

进程终止：终止所有的死锁进程—OS 中常用方法；一次只终止一个进程直到取消死锁循环为止—基于某种最小代价原则。

资源抢占：逐步从进程中强占资源给其它进程使用，直到死锁环被打破为止。

9、资源分配图：

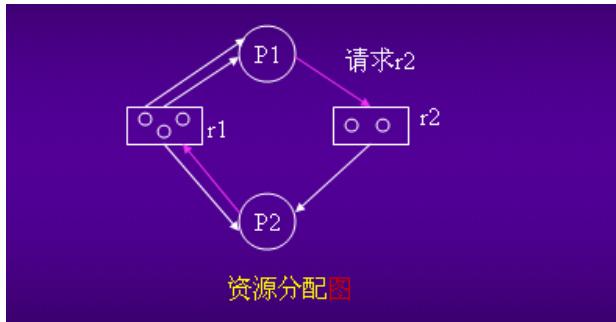
(1) 表示方法：

圆圈代表进程，方块代表一类资源。

方框中的点：由于一种类型的资源可能有多个，可用方框中的一个点代表一类资源中的一个资源。

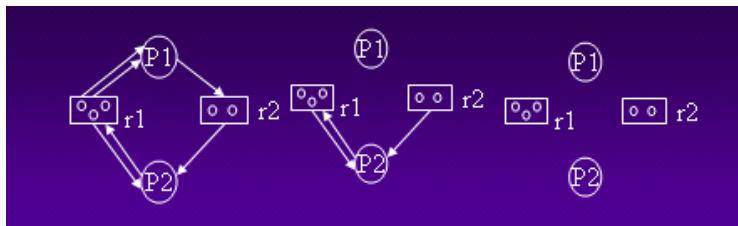
分配边：从资源节点（圆圈）到进程节点（方块）的有向弧表示资源已经分配给进程；
 请求边：从进程到资源的有向弧表示进程当前正处于阻塞状态，等待资源变为可用。

(2) 有向图形成环路则形成死锁。



(3) 化简方法：假设某个 RAG 中存在一个进程 P_i ，此刻 P_i 是非封锁进程，那么可以进行如下化简：当 P_i 有请求边时，首先将其请求边变成分配边（即满足 P_i 的资源请求），而一旦 P_i 的所有资源请求都得到满足， P_i 就能在有限的时间内运行结束，并释放其所占用的全部资源，此时 P_i 只有分配边，删去这些分配边（实际上相当于消去了 P_i 的所有请求边和分配边），使 P_i 成为孤立结点。

（反复进行）



系统中某个时刻 S 为死锁状态的充要条件是 S 时刻系统的资源分配图是不可完全简化的。

在经过一系列的简化后，若能消去图中的所有边，使所有的进程都成为孤立结点，则称该图是可完全简化的；反之的是不可完全简化的。

A. 给出类似于课本6.5.3节的两个进程，问是否会发生死锁，如果会发生死锁，修改代码并说明原因？【2017】

- 有两个并发进程 P_1 和 P_2 ，他们都要使用临界资源 A 和 B，为了实现对着两种资源的互斥访问，定义了两个信号量 mutexA 和 mutexB 分别对应资源 A 和资源 B，信号量的定义及进程 P_1 和 P_2 的程序如下。

Semaphore mutexA=1,mutexB=1;

Process P1	Process P2
<pre>{ wait(mutexA); wait(mutexB); 使用资源 A 和 B; signal(mutexA); signal(mutexB); }</pre>	<pre>{ wait(mutexA); wait(mutexB); 使用资源 A 和 B; signal(mutexB); signal(mutexA); }</pre>

请回答以下问题：

- 上述两个并发进程 P_1 和 P_2 会不会存在死锁的可能？说明理由。
- 如果可能产生死锁，请修改上述程序，使得不会产生死锁，说明你的修改依据。

6.5.3 死锁与饥饿

具有等待队列的信号量的实现可能导致这样的情况：两个或多个进程无限地等待一个事件，而该事件只能由这些等待进程之一来产生。这里的事件是 signal() 操作的执行。当出现这样的状态时，这些进程就称为死锁（deadlocked）。

为了说明，考虑一个由两个进程 P_0 和 P_1 组成的系统，每个都访问共享信号量 S 和 Q，这两个信号量的初值均为 1：

```

 $P_0$            $P_1$ 
wait(S);    wait(Q);
wait(Q);    wait(S);
⋮           ⋮
signal(S);  signal(Q);
signal(Q);  signal(S);

```

假设 P_0 执行 wait(S)，接着 P_1 执行 wait(Q)。当 P_0 执行 wait(Q) 时，它必须等待，直到 P_1 执行 signal(Q)。类似地，当 P_1 执行 wait(S)，它必须等待，直到 P_0 执行 signal(S)。由于这两个操作都不能执行，那么 P_0 和 P_1 就死锁了。

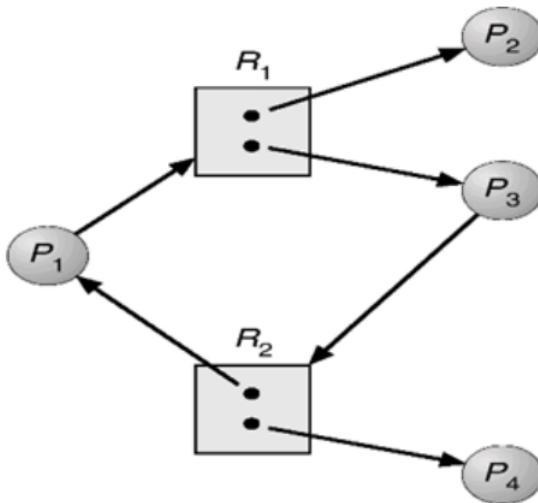
说一组进程处于死锁状态，即组内的每个进程都等待一个事件，而该事件只可能由组内的另一个进程产生。这里主要关心的事件是资源获取和释放（resource acquisition and release）。然而，如第 7 章所述，其他类型的事件也能导致死锁。在第 7 章，将讨论各种机制以处理死锁问题。

与死锁相关的另一个问题是无限期阻塞（indefinite blocking）或饥饿（starvation），即进程在信号量内无限期等待。如果对与信号量相关的链表按 LIFO 顺序来增加和移动进程，那么可能会发生无限期阻塞。

C①找出进程

B. 【2012】年试题

2. [进程管理]系统中现有四个进程和两类资源，进程和资源的关系如下图。试分析目前系统中是否存在死锁进程，并说明理由。



解答：由图可知， R_1 资源和 R_2 资源已经全部分配完毕，首先考虑 P_2 ， P_2 不请求分配资源，进程 P_2 结束之后，会释放已经申请的资源，此时 P_1 申请的 R_1 资源可以得到满足，因此进程 P_1 运行结束之后释放与其相连的边， P_3 申请资源 R_2 也可以得到满足，运行结束之后会释放与其相连的边，最后 P_4 进程运行结束之后也会释放与其相连的边，因此所有进程都可以运行完毕，故没有死锁进程。

C. 【2016】年试题

4. 两个进程 T1 和 T2 并发执行，共享变量 x，初值为 1，T1 使 $x+1$ ，
T2 使 $x-1$ ，过程如下。

问两个进程结束后 x 有多少种可能取值？有哪些方法使结果唯一？

选取一种方法修改下面的程序，保证两进程结束后结果唯一。

T1	T2	
Load R1,x	Load R2,x	将 x 取到寄存器中

Inc R1	Dec R2	
Store x,R1	Store x,R2	将寄存器的值放回 x

参考D题即可。

D. 朋辈助学试题

3. 有两个并发进程，对于如下这段程序的运行，进程是否会死锁，是否会“饥饿”？为什么？[进程死锁]

```

1 int x,y,z,t,u
2 P1(){
3     while(1){
4         x = 1;
5         y = 0;
6         if x>=1 then y=y+1;
7         z = y;
8     }
9 }
10
11 P2(){
12     while(1){
13         x = 0;
14         t = 0;
15         if x<=1 then t=t+2;
16         u = t;
17     }
18 }
```

本题中两个进程不能正确地工作，运行结果的可能性详见下面的说明。

- | | |
|------------------------|------------------------|
| 1. x = 1; | 5. x = 0; |
| 2. y = 0; | 6. t = 0 |
| 3. If x>=1 then y=y+1; | 7. if x<=1 then t=t+2; |
| 4. z=y; | 8. u=t; |

不确定的原因是由于使用了公共变量 x，考查程序中与变量 x 有关的语句共四处，执行的顺序是 1→2→3→4→5→6→7→8 时，结果是 y=1, z = 1, t = 2, u = 2, x = 0；并发执行过程是 1→2 →5→6→3→4→7→8 时，结果是 y = 0, z = 0, t = 2, u = 2, x = 0；执行的顺序是 5→6→7→8 →1→2→3→4 时，结果是 y = 1, z = 1, t = 2, u = 2, x = 1；执行的顺序是 5→6→1→ 2→7→8→3→4 时，结果是 y = 1, z = 1, t = 2, u = 2, x = 1。可见结果有多种可能性。

很明显，无论执行顺序如何，x 的结果只能是 0 或 1，因此语句 7 的条件一定成立，即 t = u = 2 的结果是一定的；而 y = z 必定成立，只可能有 0, 1 两种情况，又不可能出现 x = 1, y = z = 0 的情况，所以总共只有 3 种结果（答案中的 3 种）。

第八章 内存管理

1、地址重定位：将逻辑地址转变为物理地址的过程。

(1) 静态地址重定位：在目标程序装入内存时，由装入程序对目标程序中的指令和数据的地址进行修改，即把程序的逻辑地址都改成实际的物理内存地址。当用户程序被装入内存时，一次性实现逻辑地址到物理地址的转换，以后不再转换。程序的存储空间只能是连续的一片区域不能再移动。

(2) 动态地址重定位：在程序运行过程中要访问内存数据时再进行地址变换，即在逐条指令执行时完成地址映射。OS 可以将一个程序分散存放于不连续的内存空间，可以

移动程序。

A. 逻辑地址和物理地址绑定的时间有几种，优缺点？【2018】

2、覆盖与交换

(1) 覆盖：

原理：在任何时候只在内存中保留所需的指令和数据；当需要其它指令时，它们会装入到刚刚不再需要的指令所占用的内存空间。

特点：覆盖不需要 OS 提供特殊的 support，但程序员必须适当地设计和编写覆盖结构。

(2) 交换：

原理：暂停执行内存中的进程，将整个进程的地址空间保存到外存的交换区中（换出），而将外存中由阻塞变为就绪的进程的地址空间读入到内存中，并将该进程送到就绪队列（换入）。交换单位为整个进程的地址空间。

与覆盖的比较：与覆盖技术相比，交换技术不要求用户给出程序段之间的逻辑覆盖结构。交换发生在进程或作业之间，而覆盖发生在同一进程或作业内。此外，覆盖只能覆盖那些与覆盖段无关的程序段。

3、连续内存分配—固定分区：

把内存分为一些大小相等或不等的分区(partition)，每个应用进程占用一个或几个分区。操作系统占用其中一个分区。分区的划分原则一般由系统操作员或操作系统决定，分区一旦划分结束，在整个执行过程中每个分区的长度和内存的总分区个数将保持不变。

特点：适用于多道程序系统和分时系统、支持多个程序并发执行、难以进行内存分区的共享。

问题：可能存在内碎片和外碎片。

分区大小相等和分区大小不相等两种情况。

4、连续内存分配—可变分区：

内存不是预先划分好的，而是当进程装入时，根据进程的需求和内存空间的使用情况来决定是否分配。若有足够的空间，则按需要分割一部分分区给该进程；否则，令其等待主存空间。

评价：没有内碎片；在开始时是很好的，但最后，导致在存储器中出现很多空洞—外部碎片；

解决方法：压缩。

压缩 (compaction)：OS 不时地移动进程，将它们放在一起，并且使所有空闲空间连成一片。

压缩非常费时，浪费了处理器的时间。压缩需要动态重定位的能力，即必须能够将程序从主存的一块区域移动到另一块区域，而不会使程序中的存储器访问无效。

分配算法：首次适配法（分区按起始地址递增的顺序排列）、最佳匹配法（按空闲区大小从小到大的顺序排列）、最差匹配法（按空闲区大小从大到小的顺序排列）、临近匹配法（到最后分区时再回到开头）。

5、紧缩 (compaction)：移动内存内容，以便所有空闲空间合并成一整块。

条件：允许进行动态重定位可以首先移动程序和数据，然后再根据新基址的值来改变基地址寄存器。

实现：一种简单的方法是将所有进程移动到内存的一端，而将所有的空闲区（孔）移动内存的另一端，以生成一个大的空闲块。

开销：开销大。

6、分页管理

1) 分页机制概述：

进程的物理地址空间可以是非连续的；只要物理内存可用，就为进程分配物理内存，这样避免外部碎片，避免了大小不一的内存块问题，将物理内存划分为称为帧的固定大小的块(帧，页框)，大小是 2 的幂，介于 512 字节和 16 MB 之间

将逻辑内存划分为大小相同的块，称为页(页)，跟踪所有可用帧，要运行大小为 N 页的程序，需要找到 N 个可用帧并加载程序；设置页面表(页表) 将逻辑地址转换为物理地址；备份存储同样拆分为页面。仍有很多内部碎片。

A. 逻辑地址映射到物理地址（页表）计算题 【2018】

B. 画图说明在分页内存管理中是如何分享页面的，并说明分享代码与分享数据需要注意什么问题？【2017】

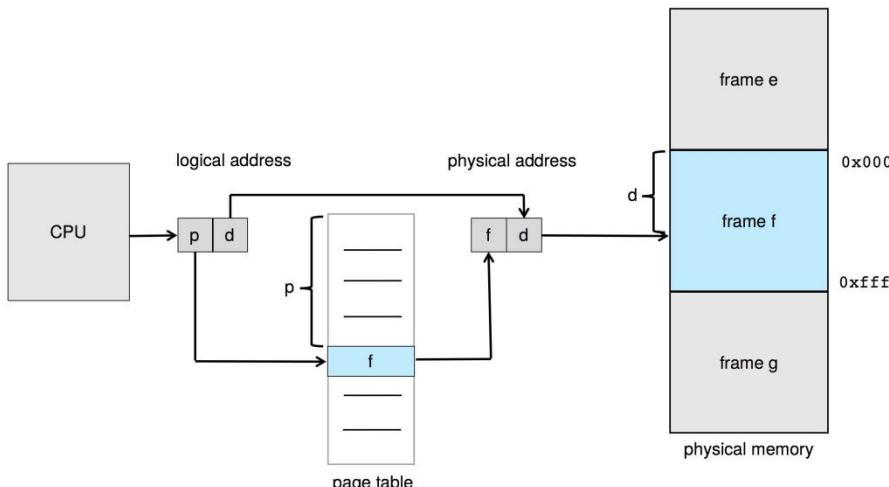
写时复制：[有关 COW \(CopyOnWrite\) 的一切 - 知乎 \(zhihu.com\)](#)

C. 画出分页内存管理方案的过程图，描述流程、过程中硬件或软件所起的作用？【2016】

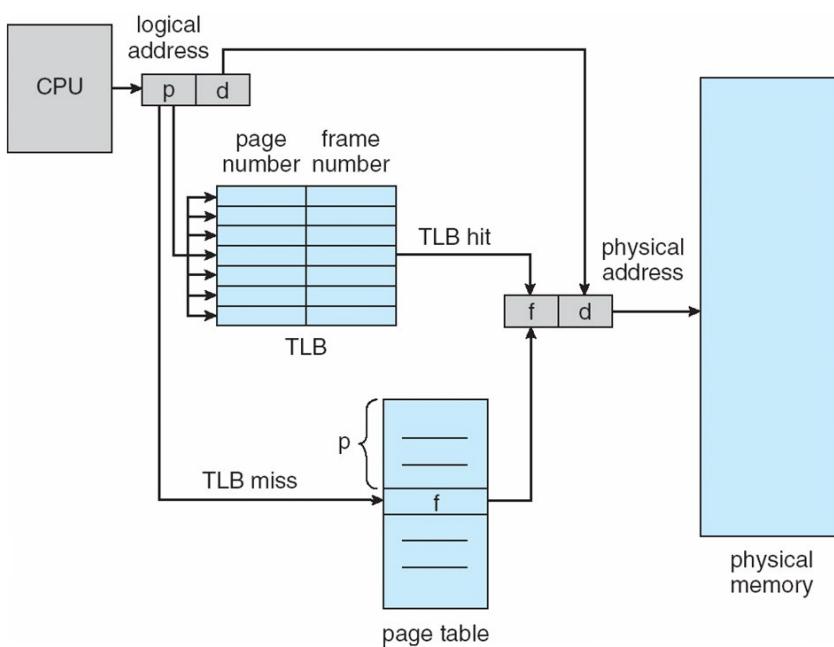
参考：[内存管理\(三\)——内存分页 - 知乎 \(zhihu.com\)](#)

D. 分页式文件系统，页表的主要数据结构，要使用这个分页结构需要哪些硬件支持？
【2013】

2) 单级页表



访问内存变为两次，效率下降，采用 TLB 缓存提高效率：



A. 什么是TLB? 名词解释? 【2020, 2016】

全称 Translation lookaside Buffer 转换检测缓冲区是一个内存管理单元，用于提高虚拟地址到逻辑地址的转换速度。TLB 是一个小小的、虚拟寻址的缓存，每一行都保存由单个 PTE (Page Table Entry 即页表项) 所组成的块。如果没有 TLB，每次取数据都要两次访问内存，即查页表获得物理地址和取数据。

TLB 快表，用来存放当前访问的若干页表项，与此对应，

B. 一个进程的页表如右图，页的大小为1024字节，为执行指令 MOV AX, [2560]，计算逻辑地址2560（十进制）对应的物理地址。如果指令中的地址为4219（十进制）时，情况会怎样？【2012】

页号	块号
0	20
1	30
2	18
3	59

解答：参考 C 题

C. 带有快表 (TLB) 的分页 (单级页表) 系统中。快表内容如下，一页大小为2048字节

(1) 在执行MOV AX [2560], MOV BX [8196] 指令时，请由逻辑地址[2560], [8196]计算出物理地址。(5分)

(2) 设快表命中概率为90%，快表查询时间为5ns，内存访问时间为25ns，求有效内存访问时间。(5分) 【2021】

页号	块号

页号	块号
0	7
1	30
2	11
3	56

解：① [2560]，由于 $2560 \% 2048 = 512$ ，页号为1。

则 块号为30，物理地址为 $30 \times 2048 + 512$

② [8196]，由于 $8196 \% 2048 = 4$ ，页号为4 块内偏移量为4。
此时快表中没有，需要查询页表中页号4对应的块号
进而计算出物理地址。

(2) 访存时间。快表命中访存1次

未命中访存2次

$$\text{故 } (5\text{ns} + 25\text{ns}) \times 90\% + (5\text{ns} + 25\text{ns} + 25\text{ns}) \times 10\% \\ = 30 \times 0.9 + 55 \times 0.1 = 27 + 5.5 = 32.5\text{ns}.$$

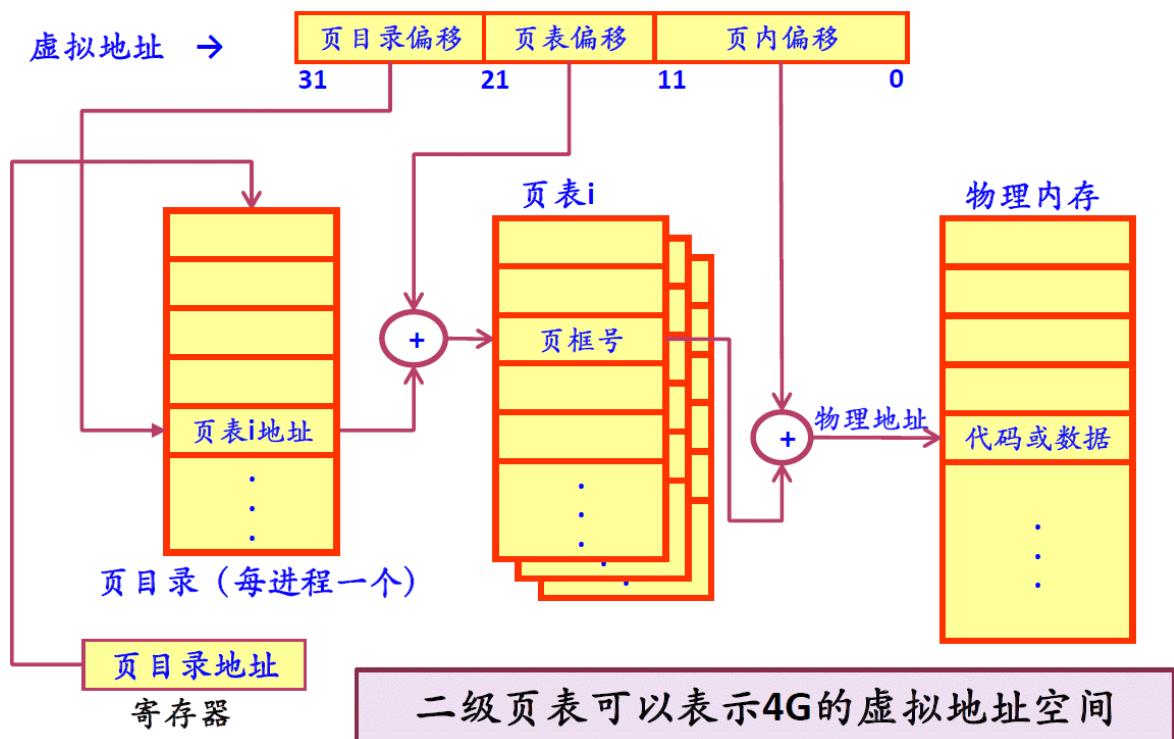
加权平均。快表命中。一次访存。 $5\text{ns} + 25\text{ns} = 30\text{ns}$

未 ... 二次访存。 $5\text{ns} + 25\text{ns} + 25\text{ns} = 55\text{ns}$

3) 多级页表（二级以上）

随着内存容量的扩大，页表变大，因页表要求连续内存，造成内存管理变得困难，采用多级页表，使得内层页表不必要连续。

二级页表示意如下：



需要根据要求给出页表设计方案。

- A. 操作系统采用分段式存储管理方式，其中每个页面大小为64Byte，操作系统的物理地址和逻辑地址的寻址空间为64KB，由16位二进制位组成。采用两级页表的分页管理方式，包括一级页表和二级页表。
1. 请问上述分配方式下一级页表和二级页表的逻辑地址的bit？一级页表和二级页表的大小以及页表占用的存储空间大小？
 2. 请画出一级页表和二级页表逻辑地址结构图(简单解释一下一级页表和二级页表项里面有啥)
 3. 为加快操作系统的对内存的访问速度，引入TLB机制，请问TLB是如何加快操作系统对内存的访问速度的？【2022】

(3) 为加快操作系统的对内存的访问速度，引入 TLB 机制，请问 TLB 是如何加快操作系统对内存的访问速度的？

① 传统.

- 由介绍的地址变换过程可知，若页表全部放在内存中，则存取一个数据或一条指令至少要访问两次内存：第一次是访问页表，确定所存取的数据或指令的物理地址；第二次是根据该地址存取数据或指令。显然，这种方法比通常执行指令的速度慢了一半。为此，在地址变换机构中增设一个具有并行查找能力的高速缓冲存储器——快表，又称相联存储器（TLB），用来存放当前访问的若干页表项，以加速地址变换的过程。与此对应，主存中的页表常称为慢表。具有快表的地址变换机构如图 3.10 所示。

CPU \leftrightarrow Cache \leftrightarrow 内存

↓ 局部性原理 Cache. ↓ 越界中断 TLB.

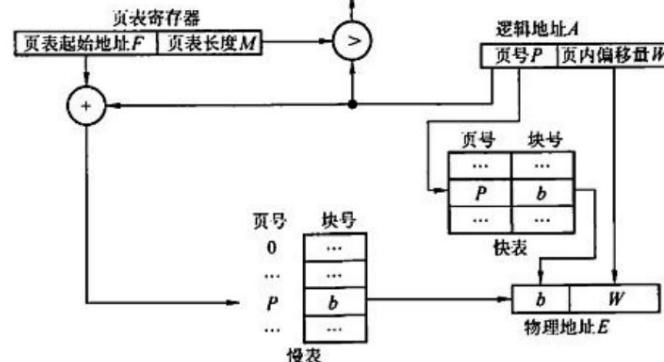


图 3.10 具有快表的地址变换机构

- 在具有快表的分页机制中，地址的转换过程如下：
- ① CPU 给出逻辑地址后，由硬件进行地址转换，将页号送入高速缓存寄存器，并将此页号与快表中的所有页号进行比较。
- ② 若找到匹配的页号，说明所要访问的页表项在快表中，则直接从中取出该页对应的页框号，与页内偏移量拼接形成物理地址。这样，存取数据仅一次访存便可实现。
- ③ 若未找到匹配的页号，则需要访问主存中的页表，读出页表项后，应同时将其存入快表，以便后面可能的再次访问。若快表已满，则须按特定的算法淘汰一个旧页表项。
- 注意：有些处理机设计为快表和慢表同时查找，若在快表中查找成功则终止慢表的查找。
- 一般快表的命中率可达 90% 以上，这样分页带来的速度损失就可降低至 10% 以下。快表的有效性基于著名的局部性原理，后面讲解虚拟内存时将会具体讨论它。

②

↓ 大部分时间.

↓ 周期 .

B. 某磁盘逻辑地址 32 位，页大小 16K，页表项大小 4B

- 采用多层页表结构，该采用几层页表？页偏移多少比特？画出地址分配。
- 对逻辑地址 54321（10 进制），简述求实际地址的过程（忽略缺页中断）
- CPU 和操作系统在分页中各自承担了那些工作，简要说明。【2015】

解答：参考 A 题

第九章 虚拟内存管理

1、虚拟内存及其作用

虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续的可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。这样可以在内存中同时加载更多的进程，提高资源的利用率，尤其是 CPU 的利用率，目前，大多数操作系统都使用了虚拟内存技术。

2、局部性原理

程序在执行的一段较短的时期，所执行的指令地址、相应操作数据地址，分别局限

于一定区域内，这样不必要把整个程序加载到内存。

3、虚拟内存情况下页表的改变：增加标志 P：表示该页是否在内存中，如果在，对应页条目中包含了对应的物理帧号；M：修改位，表示该页内容在上次装入后修改过；还有其他一些用于保护和共享的位。

4、缺页中断：在地址映射过程中，在页表中发现所要的页不在内存，则产生缺页中断，缺页中断处理程序会申请物理帧，并把相应页调入，并修改页表项。然后恢复指令执行。

A. 什么是Demand Paging（缺页中断）？【2012改】

B. 什么是缺页率？【2012改】

2、缺页率

定义：

表示“缺页次数/内存访问次数”（比率）或“缺页的平均时间间隔的倒数”。

5、Beladay 异常：有些情况下，缺页数会随着所分配的物理帧数的增加而增加。

6、掌握几种替换页选择方法：最佳算法、先进先出、最近最久未使用，能给出页面替换序列和次数。

熟悉各类算法：[一文看懂页面置换算法 - 知乎 \(zhihu.com\)](#)

A. FCFS在CPU调度、磁盘调度、页置换中分别的优缺点？【2013】

B. 某请求分页式存储管理系统，接收一个共7页的作业。作业运行时的页面走向如下：1、5、2、1、3、2、4、7、2、4。假定系统为该作业分配了3块内存空间，内存页块初始均为空，假设FIFO算法以队列，LRU算法以堆栈作为辅助结构，请填表并计算：采用先进先出（FIFO）页面淘汰算法时，会产生多少次缺页中断？缺页率是多少？【2015】

C. FIFO和LRU的页面置换算法哪个更好，为什么？【2015】

D. 页替换（FIFO，LRU，OPT）？计算题【2018】

E. 局部置换，FIFO置换算法，写出物理地址顺序。【2019】

2帧，现在指向第一条命令，已分配A、B

命令	地址(十六进制)	
1	F0, F1	
2	EA、EB、EF	
3	F8	
4	EA、EE	

5	EF、 BA	
---	--------	--

- F. 阐述在页面置换中增强型二次机会算法的原理，解释系统抖动发生的原因，以及解决方法（10分）【2021】

解答： [【OS】虚存管理—页面置换算法_增强二次机会算法 CSDN博客](#)

- G. 当前页表如下。页大小为1024字节，该程序分配2个帧，页号0先装入内存。采用先进先出和局部置换策略，现在访问逻辑地址为3000的字节，问在这个过程中发生了什么主要事件并写出置换后的页表。【2020】

页号	帧号	Valid/Invalid
0	130	Valid
1	570	Valid
2	-1	Invalid
3	-1	Invalid

解：访问3000的字节，首先计算页号为2，偏移量为952。
 帧号0先装入内存，由表可知，页号1也装入内存中。
 此时程序分配的2个帧均已分配。（缺页中断）
 选择页号为0的帧换出内存，并且修改页表。
 插入页号为2。

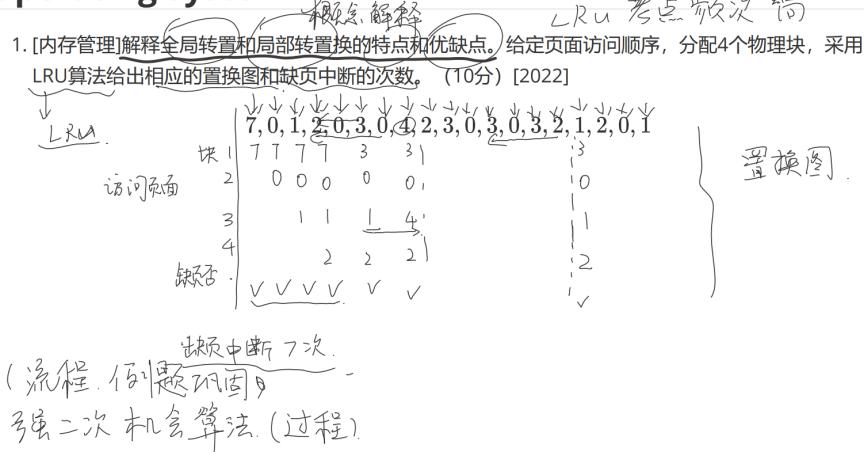
- H. 请解释全局替换和局部替换的特点和优缺点？给定页面访问顺序，采用LRU算法给出相应的置换图和缺页中断的次数。

访问页面	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
物理块 1	7	7	7	2		2	2	4	4	4	0			0	0			7	7	7
物理块 2		0	0	0		3	3	3	2	2				1	1			1	0	0
物理块 3			1	1		1	0	0	0	3	3			3	2			2	2	1
缺页否	✓	✓	✓	✓		✓	✓	✓	✓	✓				✓	✓			✓	✓	✓

图 3.23 FIFO 置换算法时的置换图

序列是 19 个数，和这个类似，原题中给该进程分配 4 个物理块，使用 LRU 算法，给出缺页中断次数，和画出上述置换图。

Operating System



[解答]

- 在请求分页系统中，可采取两种内存分配策略，即固定和可变分配策略。在进行置换时，也可采取两种策略，即全局置换和局部置换。于是可组合出下面三种适用的策略。
 - (1) 固定分配局部置换
 - 为每个进程分配一定数目的物理块，在进程运行期间都不改变。所谓局部置换，是指如果进程在运行中发生缺页，则只能从分配给该进程在内存的页面中选出一页换出，然后再调入一页，以保证分配给该进程的内存空间不变。实现这种策略时，难以确定应为每个进程分配的物理块数目：太少会频繁出现缺页中断，太多又会降低CPU和其他资源的利用率。
 - (2) 可变分配全局置换
 - 先为每个进程分配一定数目的物理块，在进程运行期间可根据情况适当地增加或减少。所谓全局置换，是指如果进程在运行中发生缺页，系统从空闲物理块队列中取出一块分配给该进程，并将所缺页调入。这种方法比固定分配局部置换更加灵活，可以动态增加进程的物理块，但也存在弊端，如它会盲目地给进程增加物理块，从而导致系统多道程序的并发能力下降。
 - (3) 可变分配局部置换
 - 为每个进程分配一定数目的物理块，当某进程发生缺页时，只允许从该进程在内存的页面中选出一页换出，因此不会影响其他进程的运行。若进程在运行中频繁地发生缺页中断，则系统再为该进程分配若干物理块，直至该进程的缺页率趋于适当程度；反之，若进程在运行中的缺页率特别低，则可适当减少分配给该进程的物理块，但不能引起其缺页率的明显增加。这种方法在保证进程不会过多地调页的同时，也保持了系统的多道程序并发能力。当然它需要更复杂的实现，也需要更大的开销，但对比频繁地换入/换出所浪费的计算机资源，这种牺牲是值得的。

7、系统颠簸。(现象描述、产生原因、解决办法)。

- A. 请解释系统产生颠簸的原因，请问如何有效地消除或减轻颠簸现象的发生？请给出有效的解决方法和评价解决方案的优点？【2022】

- B. 什么是Working Directioy? 名词解释? 【2020, 2013】
- C. 颠簸产生的原因解决方案? 【2018】
- D. 什么是颠簸, 利用工作集合模型限制颠簸的原理【2017】
- E. 什么是颠簸(抖动 thrashing)? 说明采用工作集模型预防系统抖动的思想与过程。【2017】
- F. 什么是颠簸? 系统提供哪几种方法避免颠簸? 如果发生颠簸该怎么处理? 【2016】

程序执行过程中频繁发生缺页异常。

系统可以利用虚拟内存技术保留尽可能多的进程在内存中, 还可以选择合适的页面置换算法。

处理方式有: 修改页面置换算法、正确地选择工作集的大小、降低多道程序设计的程度、挂起该进程

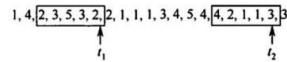
- G. 什么是抖动? 解决抖动问题有哪些方法? 【2012】

4. [内存管理]描述系统颠簸产生的原因, 以及如何有效地消除或减轻颠簸现象的发生? 请给出有效的解决方法和评价解决方案的优点? (10分) [2022]

- 在页面置换过程中, 一种最糟糕的情形是, 刚刚换出的页面马上又要换入主存, 刚刚换入的页面马上又要换出主存, 这种频繁的页面调度行为称为抖动或颠簸。
- 系统发生抖动的根本原因是, 系统中同时运行的进程太多, 由此分配给每个进程的物理块太少, 不能满足进程正常运行的基本要求, 致使每个进程在运行时频繁地出现缺页, 必须请求系统将所缺页面调入内存。这会使得在系统中排队等待页面调入/调出的进程数目增加。显然, 对磁盘的有效访问时间也随之急剧增加, 造成每个进程的大部分时间都用于页面的换入/换出, 而几乎不能再去做任何有效的工作, 进而导致发生处理机的利用率急剧下降并趋于零的情况。
- 抖动是进程运行时出现的严重问题, 必须采取相应的措施解决它。由于抖动的发生与系统为进程分配物理块的多少有关, 于是又提出了关于进程工作集的概念。

2. 工作集

工作集是指在某段时间间隔内, 进程要访问的页面集合。基于局部性原理, 可以用最近访问过的页面来确定工作集。一般来说, 工作集 W 可由时间 t 和工作集窗口大小 Δ 来确定。例如, 某进程对页面的访问次序如下:



假设系统为该进程设定的工作集窗口大小 Δ 为 5, 则在 t_1 时刻, 进程的工作集为 {2, 3, 5}, 在 t_2 时刻, 进程的工作集为 {1, 2, 3, 4}。

实际应用中, 工作集窗口会设置得很大, 即对于局部性好的程序, 工作集大小一般会比工作集窗口 Δ 小很多。工作集反映了进程在接下来的一段时间内很有可能会频繁访问的页面集合。因此, 若分配给进程的物理块小于工作集大小, 则该进程就很有可能频繁缺页, 所以为了防止这种抖动现象, 一般来说分配给进程的物理块数(即驻留集大小)要大于工作集大小。

工作集模型的原理是, 让操作系统跟踪每个进程的工作集, 并为进程分配大于其工作集的物理块。落在工作集内的页面需要调入驻留集中, 而落在工作集外的页面可从驻留集中换出。若还有空闲物理块, 则可再调一个进程到内存。若所有进程的工作集之和超过了可用物理块总数, 则操作系统会暂停一个进程, 将其页面调出并将物理块分配给其他进程, 防止出现抖动现象。

- 8、请求段式管理
9、虚拟段页式管理。

第十章 文件系统

A. 文件系统(file system): 【2019】

包含若干文件以及其属性说明、对文件进行操纵和管理的软件，以及系统向用户提供的使用文件的接口等的集合。文件系统是操作系统的一个重要组成部分。

1、基本概念

- 1) 文件：文件是命名的数据流、连续的逻辑地址空间。
- 2) 文件的几种类型：数据文件、程序文件、目录文件...
- 3) 文件属性：名字、标识符、类型、位置、大小、权限、时间戳等描述文件的元数据。
- 4) 目录：用于组织文件的文件，每个条目对应一个目录或普通文件。

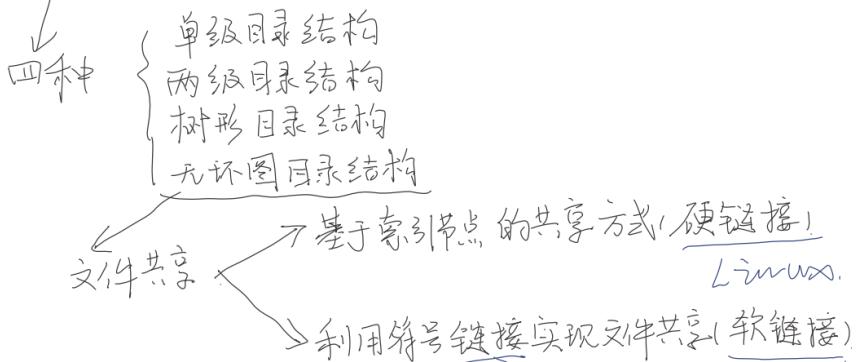
A. 什么是文件目录？名词解释？【2016】

用户通过文件的目录执行相关操作，例如创建和删除一个文件分别对应在目录添加和删除一个目录项。文件的目录结构由单极、两级、多级和无环图目录结构等

5) 文件共享：同步锁（强制锁、建议锁）

A. 在实现目录管理的文件系统中，描述实现文件共享的几种解决方法和设计思想。（10分）【2021】

2. [文件管理]结合目录结构的文件系统，提出设计方案，实现文件系统内文件的共享。（在实现目录管理的文件系统中，描述实现文件共享的几种解决方法和设计思想。）



4.2.5 文件共享

文件共享使多个用户共享同一个文件，系统中只需保留该文件的一个副本。若系统不能提供共享功能，则每个需要该文件的用户都要有各自的副本，会造成对存储空间的极大浪费。

现代常用的两种文件共享方法如下。

1. 基于索引结点的共享方式（硬链接）

在树形结构的目录中，当有两个或多个用户要共享一个子目录或文件时，必须将共享文件或子目录链接到两个或多个用户的目录中，才能方便地找到该文件，如图 4.15 所示。

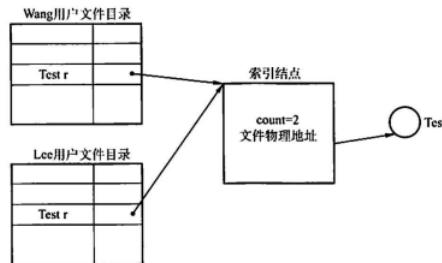


图 4.15 基于索引结点的共享方式

在这种共享方式中，诸如文件的物理地址及其他文件属性等信息，不再放在目录项中，而放在索引结点中。在文件目录中只设置文件名及指向相应索引结点的指针。在索引结点中还应有一个链接计数 count，用于表示链接到本索引结点（即文件）上的用户目录项的数目。当 count = 2 时，表示有两个用户目录项链接到本文件上，或者说有两个用户共享此文件。

用户 A 创建一个新文件时，他便是该文件的所有者，此时将 count 置为 1。用户 B 要共享此文件时，在 B 的目录中增加一个目录项，并设置一个指针指向该文件的索引结点。此时，文件主仍然是用户 A，count = 2。如果用户 A 不再需要此文件，能否直接将其删除呢？答案是否定的。因为若删除了该文件，也必然删除了该文件的索引结点，这样便会使用户 B 的指针悬空，而 B 可能正在此文件上执行写操作，此时将因此半途而废。因此用户 A 不能删除此文件，只是将该文件的 count 减 1，然后删除自己目录中的相应目录项。用户 B 仍可以使用该文件。当 count = 0 时，表示没有用户使用该文件，才会删除该文件。如图 4.16 给出了用户 B 链接到文件上的前、后情况。

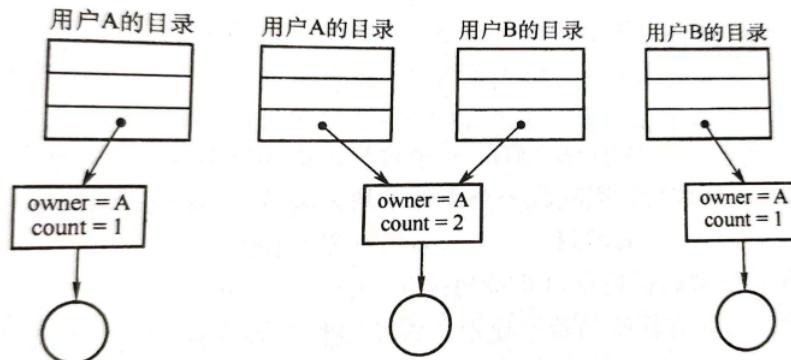


图 4.16 文件共享中的链接计数

2. 利用符号链实现文件共享（软链接）

为使用户 B 能共享用户 A 的一个文件 F，可以由系统创建一个 LINK 类型的新文件，也取名为 F，并将该文件写入用户 B 的目录中，以实现用户 B 的目录与文件 F 的链接。在新文件中只包含被链接文件 F 的路径名。当用户 B 要访问被链接的文件 F 且正要读 LINK 类新文件时，操作系统查到要读的文件是 LINK 类型，则根据该文件中的路径名去找到文件 F，然后对它进行读，从而实现用户 B 对文件 F 的共享。这样的链接方法被称为符号链接。

在利用符号链方式实现文件共享时，只有文件主才拥有指向其索引结点的指针。而共享该文件的其他用户只有该文件的路径名，并不拥有指向其索引结点的指针。这样，也就不会发生在文件主删除一共享文件后留下一悬空指针的情况。当文件主把一个共享文件删除后，若其他用户又试图通过符号链去访问它时，则会访问失败，于是将符号链删除，此时不会产生任何影响。

在符号链的共享方式中，当其他用户读共享文件时，系统根据文件路径名逐个查找目录，直至找到该文件的索引结点。因此，每次访问共享文件时，都可能要多次地读盘。使得访问文件的开销甚大，且增加了启动磁盘的频率。此外，符号链的索引结点也要耗费一定的磁盘空间。

利用符号链实现网络文件共享时，只需提供该文件所在机器的网络地址及文件路径名。

硬链接和软链接都是文件系统中的静态共享方法，在文件系统中还存在着另外的共享需求，即两个进程同时对同一个文件进行操作，这样的共享称为动态共享。

可以这样说：文件共享，“软”“硬”兼施。硬链接就是多个指针指向一个索引结点，保证只要还有一个指针指向索引结点，索引结点就不能删除；软链接就是把到达共享文件的路径记录下来，当要访问文件时，根据路径寻找文件。可见，硬链接的查找速度要比软链接的快。

6) 文件保护：权限。

7) 虚拟文件系统

2、目录的结构

1) 单层：存在命名问题、分组问题。

2) 两层：目录名、不同用户可用同文件名（部分解决应用命名问题）、搜索快，不支持分组。

3) 树结构：解决命名、分组问题，搜索快。

4) 无环图：在树结构基础上可以用别名，更方便共享。

A. 请解释在给定文件地址的条件下，使用 `open()` 系统调用的过程中在系统中增加文件描述符的全部执行过程（操作系统发生了什么样的变化和发生这种变化的流程），请问操作系统是如何实现多个用户对同一个文件的访问和操作的？【2022】

4. [文件管理]请解释在给定文件地址的条件下，使用 open() 系统调用的过程中第一次用打开文件，操作系统从传入路径数据到返回描述符的内部工作流程。另外，操作系统是如何支持多进程打开操作同一文件。[2022]

[解答]

- 当用户对一个文件实施操作时，每次都要从检索目录开始。为了避免多次重复地检索目录，大多数操作系统要求，在文件使用之前通过系统调用open被显式地打开。操作系统维护一个包含所有打开文件信息的表（打开文件表）。所谓“打开”，是指调用open根据文件名搜索目录，将指明文件的属性（包括该文件在外存上的物理位置），从外存复制到内存打开文件表的一个表目中，并将该表目的编号（也称索引）返回给用户。当用户再次向系统发出文件操作请求时，可通过索引在打开文件表中查到文件信息，从而节省再次搜索目录的开销。当文件不再使用时，可利用系统调用close关闭它，操作系统将会从打开文件表中删除这一条目。
- 在多个不同进程可以同时打开文件的操作系统中，通常采用两级表：每个进程表和整个系统表。每个进程表根据它打开的所有文件，表中存储的是进程对文件的使用信息。系统打开文件表包含文件相关信息，如文件在磁盘的位置、访问日期和大小。一旦有进程打开了一个文件，系统表就包含该文件的条目。当另一个进程执行调用open时，只不过是在其进程打开表中增加一个条目，并指向系统表的相应条目。通常，系统打开文件表为每个文件关联一个打开计数器（Open Count），以记录多少进程打开了该文件。每个关闭操作close使count递减，当打开计数器为0时，表示该文件不再被使用，并且可从系统打开文件表中删除相应条目。
- 文件名不必是打开文件表的一部分，因为一旦完成对FCB在磁盘上的定位，系统就不再使用文件名。对于访问打开文件表的索引，UNIX称之为文件描述符，而Windows称之为文件句柄。因此，只要文件未被关闭，所有文件操作就通过打开文件表来进行。

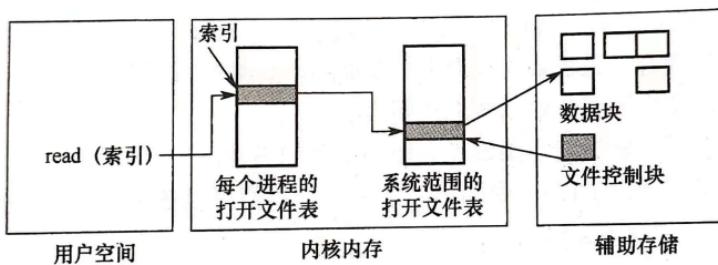


图 4.3 内存中文件的系统结构

- 每个打开文件都具有如下关联信息：
- 文件指针。系统跟踪上次的读写位置作为当前文件位置的指针，这种指针对打开文件的某个进程来说是唯一的，因此必须与磁盘文件属性分开保存。
- 文件打开计数。计数器跟踪当前文件打开和关闭的数量。因为多个进程可能打开同一个文件，所以系统在删除打开文件条目之前，必须等待最后一个进程关闭文件。
- 文件磁盘位置。大多数文件操作要求系统修改文件数据。查找磁盘上的文件所需的信息保存在内存中，以便系统不必为每个操作都从磁盘上读取该信息。
- 访问权限。每个进程打开文件都需要有一个访问模式(创建、只读、读写、添加等)。该信息保存在进程的打开文件表中，以便操作系统能够允许或拒绝后续的I/O请求。

B. 为什么访问文件需要先open()？以及该操作之后，操作系统产生了什么结构？【2020】

解答：参考A题

掌握以上概念

第十一章 文件系统实现

1、文件系统结构

- 应用程序：用户程序进行文件系统操作
- 逻辑文件系统：管理文件系统元数据(文件系统结构，FCB)，为应用提供 API(应用程序接口)

A. 什么是FCB？名词解释？【2020】

B. 什么是文件控制块？【2013】

C. 什么是API？名词解释？【2016】

全称Application Programming Interface, 即应用程序编程接口， API是一些预先定义的函数，目的用来提供开发人员访问一组软件或硬件的能力，并且无需知道内部实现细节

D. 用户给出文件名，文件系统如何找到它的FCB?【2017】

解答：操作系统之文件管理：3、文件目录(文件控制块FCB) -CSDN博客

E. FCB 包含什么内容？FCB 是集中存储比较好还是与每个相关的文件一起存储比较好？解释原因。（集中存储）【2016】

4. 在文件系统的管理中，当用户访问某文件时，需要给出要访问的文件名，系统根据该文件对应的文件控制块 FCB 确定该文件在磁盘上的物理位置，然后对文件内容进行存取。

- (1) 请说明文件系统如何依据用户给出的文件名找到该文件对应的 FCB；
- (2) 在连续、链式、索引三种文件的磁盘块组织方式中，分别说明 FCB 如何给出文件在磁盘上的物理位置。

- 文件组织模块：转换逻辑块地址到物理块地址；空闲空间管理
- 基本文件系统：向设备驱动程序发出通用的命令，读写物理数据块；管理文件系统的缓冲区及 cache
- 控制：设备驱动程序及中断处理程序，输入输出操作控制，发出的指令为特定设备的格式
- 设备：提供永久存储媒介，数据块。

A. 缓冲区的作用，使用Cache的意义？【2018】

理解：缓冲区(buffer)与缓存(cache) 的概念理解 - 知乎

B. 什么是设备驱动程序？名词解释？【2020, 2018, 2015】

是一个允许高级 (High level) 计算机软件 (computer software) 与硬件 (hardware) 交互的程序，这种程序创建了一个硬件与硬件，或硬件 与软件沟通的接口，经由主板上的总线 (bus) 或其它沟通子系统 (subsystem) 与硬件形成连接的机制，这样的机制使得硬件设备 (device) 上的数据交换 成为可能。

或者：硬件直接相关，负责实现系统对设备发出的操作指令、驱动I/O设备工作的驱动程序。每一类设备配置一个设备驱动程序，是I/O进程和设备控制器之间的通信程序，封装了设备的具体差别，将接受到的抽象I/O请求转换为具体请求后发送给设备控制器

C. block device: 【2019】

是磁盘指对数据的存取是以数据块为单位的设备，典型的块设备是磁盘。

D. FAT文件系统：【2019】

1. FAT(File Allocation Table, 文件分配表)文件系统是 Windows操作系统所使用的一种文件系统，他的发展过程经历了FAT12,FAT16,FAT32三个阶段。

2. FAT文件系统用 “簇” 作为数据单元。一个“簇”由一连串的扇区组成，簇所含的扇区数必须是 2的整数次幂 。簇的最大值是64个扇区，即32kb。本文中一簇是4kb。所有簇从2开始进行编号，每个簇都有一个自己的编号。

3. 用户文件和目录都存储在簇中。

4. 文件系统分配磁盘按照簇进行分配，因此一个文件即使只有1kb，那么也会被分配4kb的空间。

E. FAT 工作原理, 在链接分配中引入FAT后的优点? 【2018】

2、文件系统实现

1) 文件系统对应结构

- 引导控制块
- 超级块
- 目录结构
- 文件控制块

2) 内存中的结构

- 安装表
- 系统范围打开文件表
- 单个进程打开文件表

3、目录实现

目录项目存储的线性表、Hash 表。

4、分配方法: 连续分配、链接分配、索引分配, 各种分配方法的优缺点。

实际实现一般用混合机制。

A. 一级目录下, 文件写入不可更改, 可不断新建, 1) 问连续、链接、索引哪个比较符合这个文件系统, 为啥? 其FCB中需要存储什么之类的; 2) FCB连续存储还是跟着文件存储比较好, 为啥? 【2013】

B. 组合索引, 块长512B, 块号2B, 直接块10T, 一级、二级索引块1T, 求最大文件大小? 【2015】

C. 【2016】年试题

6. 文件目录为一级目录, 文件能一次性写入, 且写入后无修改, 但是可以创建多个新文件。

(1) 为文件分配磁盘空间的方法有连续、链接、索引, 用哪种方法(连续)比较好? 解释原因(连续的优点)。FCB 包含什么内容?

(2) FCB 是集中存储比较好还是与每个相关的文件一起存储比较好? 解释原因。(集中存储)

D. 磁盘空间分配的三种方式-连续分配、链接分配、索引分配的优缺点?

【2019】

E. 简述磁盘分配的三种方式（continuous、linked、indexed）和各自优缺点（6分）【2015】

1. 连续：连续分配要求每个文件在磁盘上占有一组连续的块
2. 链接：链接分配采用离散分配的方式，消除了外部碎片。具体又分为隐式链接和显式链接，隐式链接指文件的每个磁盘块除了最后一个，在其末尾都有指向下一个磁盘块的指针，显式链接指把链接文件各物理块的指针，从每个磁盘块的末尾提出来，显式地存放在内存中的一张链接表中，即分块分配表(FAT)中
3. 索引：为每一个文件分配一个索引块，索引块的第*i*号条目对应文件的第*i*块

表 4-2 文件三种分配方式的比较

	访问第 n 个记录	优 点	缺 点
顺序分配	需访问磁盘 1 次	顺序存取时速度快，当文件是定长时可以根据文件起始地址及记录长度进行随机访问	文件存储要求连续的存储空间，会产生碎片，也不利于文件的动态扩充
链接分配	需访问磁盘 n 次	可以解决外存的碎片问题，提高了外存空间的利用率，动态增长较方便	只能按照文件的指针链顺序访问，查找效率低，指针信息存放消耗外存空间
索引分配	m 级需访问磁盘 $m+1$ 次	可以随机访问，易于文件的增删	索引表增加存储空间的开销，索引表的查找策略对文件系统效率影响较大

F. 写出三种磁盘分配方法下FCB是如何找到文件的物理地址的？【2017】

4. 在文件系统的管理中，当用户访问某文件时，需要给出要访问的文件名，系统根据该文件对应的文件控制块 FCB 确定该文件在磁盘上的物理位置，然后对文件内容进行存取。
 - (1) 请说明文件系统如何依据用户给出的文件名找到该文件对应的 FCB；
 - (2) 在连续、链式、索引三种文件的磁盘块组织方式中，分别说明 FCB 如何给出文件在磁盘上的物理位置。

5、空闲空间管理：位图、链表、块组链表、第一空闲块+计数。

- A. 请描述操作系统对于空闲的磁盘块的管理方式，请问如何有效地分配和回收相应的磁盘块？【2022】
- B. 以磁盘外存空间为例，设计高效的空闲块分配、回收算法，给出设计思想，操作方法，数据结构。（10分）【2021】

3. [文件管理]以磁盘外存空间为例，设计高效的空闲块分配、回收算法，给出设计思想，操作方法，数据结构。 (10分)

- 位图

- 位示图是利用二进制的一位来表示磁盘中一个盘块的使用情况，磁盘上所有的盘块都有一个二进制位与之对应。当其值为“0”时，表示对应的盘块空闲；为“1”时，表示已分配。这样，一个 $m \times n$ 位组成的位示图就可用来表示 $m \times n$ 个盘块的使用情况：

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

- 盘块的分配：

- 1)顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位。
- 2)将找到的一个或一组二进制位，转换成与之对应的盘块号。若找到的其值为“0”的二进制位位于位示图的第*i*行、第*j*列，则其相应的盘块号应按下式计算(*n*为每行位数)：

$$b = n(i - 1) + j.$$

- 盘块的回收：

- 1)将回收盘块的盘块号转换成位示图中的行号和列号。转换公式为：

$$i = (b - 1) \text{DIV} n + 1$$

$$j = (b - 1) \text{MOD} n + 1$$

- 2)修改位示图，令 $\text{map}[i, j] = 0$.

空闲表法和空闲链表法都不适用于大型文件系统，因为这会使空闲表或空闲链表太大。

- 链表

- 将所有空闲盘区拉成一条空闲链。根据构成链所用基本元素的不同，分为两种形式：
- 1)空闲盘块链。将磁盘上的所有空闲空间以盘块为单位拉成一条链。当用户因创建文件而请求分配存储空间时，系统从链首开始，依次摘下适当数目的空闲盘块分配给用户。当用户因删除文件而释放存储空间时，系统将回收的盘块依次插入空闲盘块链的末尾。这种方法的优点是分配和回收一个盘块的过程非常简单，但在为一个文件分配盘块时可能要重复操作多次，效率较低。又因它是以盘块为单位的，空闲盘块链会很长。
- 2)空闲盘区链。将磁盘上的所有空闲盘区（每个盘区可包含若干个盘块）拉成一条链。每个盘区除含有用于指示下一个空闲盘区的指针外，还应有能指明本盘区大小（盘块数）的信息。分配盘区的方法与内存的动态分区分配类似，通常采用首次适应算法。在回收盘区时，同样也要将回收区与相邻接的空闲盘区合并。这种方法的优缺点刚好与第一种方法的相反，即分配与回收的过程比较复杂，但效率通常较高，且空闲盘区链较短。

- 块组链表

对空闲链表的一个改进是将*n*个空闲块的地址存在第一个空闲块中。这些块中的前*n*-1个确实为空，而最后一块包含另外*n*个空闲块的地址，如此继续。大量空闲块的地址可以很快地找到，这一点有别于标准链表方法。

- 第一空闲块+计数

另外一种方法是利用这样一个事实：通常，有多个连续块需要同时分配或释放，尤其是在使用连续分配和采用簇时更是如此。因此，不是记录个空闲块的地址，而是可以记录第一块的地址和紧跟第一块的连续的空间块的数量。这样，空闲空间表的每个条目包括磁盘地址和数量。虽然每个条目会比原来需要更多空间，但是表的总长度会更短，这是因为连续块的数量常常大于1。

6、效率与性能：提高性能的方法（缓冲/缓存），包括目录缓存、预读取、后台写，内存磁盘等。

A. 具体解释 bad-section mapping、prefetching、buffer、caching 的概念和用途？【2016】

B. 虚拟文件系统：【2019】

虚拟文件系统(VFS)是一种用于网络环境的分布式文件系统，允许使用和操作系统不同的实验接口，他将文件系统通用操作和具体实现分开。

或者解释为：虚拟文件系统可以理解为一个中间件。以Linux举例，linux 支持ext, ext1, ext2等文件系统，这些文件系统对外的接口不统一，虚拟文件系统就是在这些系统之上的一层，它封装了底层系统的不一致性，对外提供统一的接口。

第十二章 大规模存储

1、大规模存储设备

磁盘、固态盘、可移动磁盘、光盘、磁带、存储阵列、网络存储、云存储等。

A. RAID(Redundant Arrays of Independent Disks, 独立磁盘冗余阵列): 【2019, 2018, 2017, 2015】

解释1：其基本思想就是把多个相对便宜的硬盘组合起来，成为一个硬盘阵列组，使性能达到甚至超过一个价格昂贵、容量巨大的硬盘。根据选择的版本不同，RAID 比单颗硬盘有以下一个或多个方面的好处：增强数据集成度，增强容错功能，增加处理量或容量。另外，磁盘阵列对于计算机来说，看起来就像一个单独的硬盘或逻辑存储单元。简单来说，RAID 把多个硬盘组合成为一个逻辑扇区，因此，操作系统只会把它当作一个硬盘。RAID 常被用在服务器计算机上，并且常使用完全相同的硬盘作为组合。

解释2：独立磁盘冗余阵列，简称磁盘阵列。把一个或多个独立的磁盘块按照某种方式组合起来形成一个磁盘组，从而提供比单个磁盘更高的存储性能和数据备份技术。通过引入冗余来提高数据可靠性，存储在正常情况下不需要的信息，以便在数据故障时修复数据。通过并行来提高性能，将数据按位或按块拆分到多张磁盘，从而达到并行读取数据，提高效率。

解释3：独立磁盘冗余阵列，简称磁盘阵列，是把一块或者多块独立的物理磁盘按照不同的方式组合起来形成一个硬盘组，从而提供比单个硬盘更高的存储性能和额外的数据备份等功能。通过引入冗余来提高数据可靠性，存储正常情况下不需要的数据，以便在数据故障时修复数据。

2、磁盘调度

寻道延迟 (Seek Time)、Rotate Latency。

FCFS、SSTF、SCAN、C-SCAN、LOOK、C-LOOK

A. FCFS在CPU调度、磁盘调度、页置换中分别的优缺点？【2013】

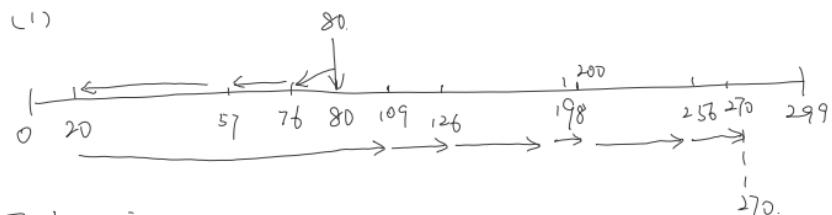
B. 磁盘柱面编号为0~299，当前磁头位于编号为80的柱面，有以下任务请求同时到达，主码编号为 20、198、256、76、57、126、270、109、200。求以下两种磁盘调度算法的寻道距离。（10分）

(1) 最短时间优先算法 (SSTF)

(2) LOOK (当前磁头向柱面编号增大的方向移动) 【2021】

3. [文件管理]磁盘柱面编号为0~299，当前磁头位于编号为80的柱面，有以下任务请求同时到达，主码编号为 20、198、256、~~76~~ 57、126、270、109、200。求以下两种磁盘调度算法的寻道距离。（10分）

- (1) 最短时间优先算法 (SSTF)
- (2) LOOK (当前磁头向柱面编号增大的方向移动)



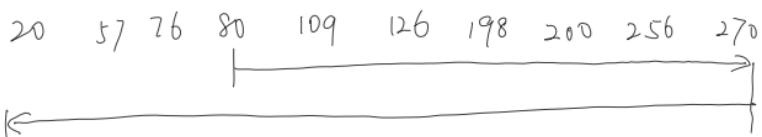
$$\text{寻道距离: } 4 + (176 - 57) + (57 - 20)$$

$$+ (109 - 20) + (126 - 109) + (198 - 126) + (200 - 198)$$

$$+ (256 - 200) + (270 - 256)$$

$$= 80 - 20 + 270 - 20 = 60 + 250 = 310.$$

(2) Look 算法不到达磁盘端点。



$$\text{故寻道距离为 } 270 - 80 + 270 - 20$$

$$= 190 + 250 = 440$$

C. 假设一个可移动磁头的磁盘具有200个磁道，其编号为0~199，当前它刚刚结束了125道的服务请求，正在处理149道的服务请求，假设系统当前磁盘请求序列为：88, 147, 95, 177, 94, 150, 102, 175, 138。试问对以下的磁盘调度算法而言，满足以上请求序列，磁头将如何移动？并计算

总的磁道移动数。SCAN算法 总的磁道移动数为：

$$1+25+2+30+9+36+7+1+6=117$$

150、175、177、199、147、138、102、95、94、88

算不算199？？？？ 【2015】

- D. 某磁盘有5000个磁道，从0-4999。当前磁头的位置为143，前一个请求的位置为144。给出当前的请求队列 83、252、143、…、1774（题中共给了9个），分别用下面的算法求出寻道路径长度。【2020】

(1) SSTF

(2) SCAN

- E. 磁盘调度算法：已完成 144，目前指向 143，队列有 13、54、83、941、1050、1550、1701 间距离。【2019】

为了减少对文件的访问时间，应采用一种最佳的磁盘调度算法，以使各进程对磁盘的平均访问时间最少。由于在访问磁盘时主要是寻道时间。因此，磁盘调度的目标是使磁盘的平均寻道时间最少

- 1) FIFO: 先来先服务算法。这种算法的思想比较容易理解。假设当前磁道在某一位置，依次处理服务队列里的每一个磁道，这样做的优点是处理起来比较简单，但缺点是磁头移动的距离和平均移动距离会很大。
- 2) SSTF: 最短寻道时间算法。这种算法的本质是利用贪心算法来实现，假设当前磁道在某一位置，接下来处理的是距离当前磁道最近的磁道号，处理完成之后再处理离这个磁道号最近的磁道号，直到所有的磁道号都服务完了程序结束。这样做的优点是性能会优于FIFO算法，但是会产生距离。当前磁道较远的磁道号长期得不到服务，也就是“饥饿”现象，因为要求访问的服务的序列号是动态产生的，即各个应用程序可能不断地提出访问不同的磁道号的请求
- 3) SCAN: 电梯调度算法。先按照一个方向(比如从外向内扫描)，扫描的过程中依次访问要求服务的序列。当扫描到最里层的一个服务序列时反向扫描，这里要注意，假设最里层为0号磁道，最里面的一个要求服务的序列是5号，访问完5号之后，就反向了，不需要再往里扫。结合电梯过程更好理解，在电梯往下接人的时候，明知道最下面一层是没有人的，它是不会再往下走的
- 4) CSCAN: 循环扫描算法。CSCAN算法，循环扫描算法，来看一下上一种算法，有什么问题。仔细一看，我们会发现，在扫描到最里面的要求服务的序列时，接着会反向，在接下来的很大一部分时间里，应该是

没有要求服务的磁道号的，因为之前已经访问过了。什么意思，就是说从初始磁道号到最里层的一个磁道号之间的所有序列都已经访问过了，所以SCAN会增加等待的时间。为了解决这样的情况，CSCAN算法的思想是，访问完最里面一个要求服务的序列之后，立即回到最外层欲访问磁道。也就是始终保持一个方向。故也称之为单向扫描调度算法。从最里面的一个磁道立即回到最外层欲访问的磁道，这步的距离是两者磁道号差的绝对值。

- 5) 各种算法应用举例：假设当前磁头在67号，要求访问的磁道号顺序为98, 25, 63, 97, 56, 51, 55, 55, 6（电脑随机产生的，设定最外层磁道号为100号）FIFO算法的服务序列是：98, 25, 63, 97, 56, 51, 55, 55, 6，磁头移动的总距离distance =

$$(98-67)+(98-25)+(63-25)+(97-63)+(97-56)+(56-51)+(55-51)+(55-55)+(55-6)$$
- 6) SSTF算法的服务序列是：63, 56, 55, 55, 51, 25, 6, 97, 98，磁头移动的总距离distance =

$$(67-63)+(63-56)+(56-55)+(55-55)+(55-51)+(51-25)+(25-6)+(97-6)+(98-97)$$
- 7) SCAN算法的服务序列是：63, 56, 55, 55, 51, 25, 6, 97, 98//先下楼，再上楼，磁头移动的总距离distance =

$$(67-63)+(63-56)+(56-55)+(55-55)+(55-51)+(51-25)+(25-6)+(97-6)+(98-97)$$
- 8) CSCAN算法的服务序列是：63, 56, 55, 55, 51, 25, 6, 98, 97, distance =

$$(67-63)+(63-56)+(56-55)+(55-55)+(55-51)+(51-25)+(25-6)+|6-98|+(98-97)$$

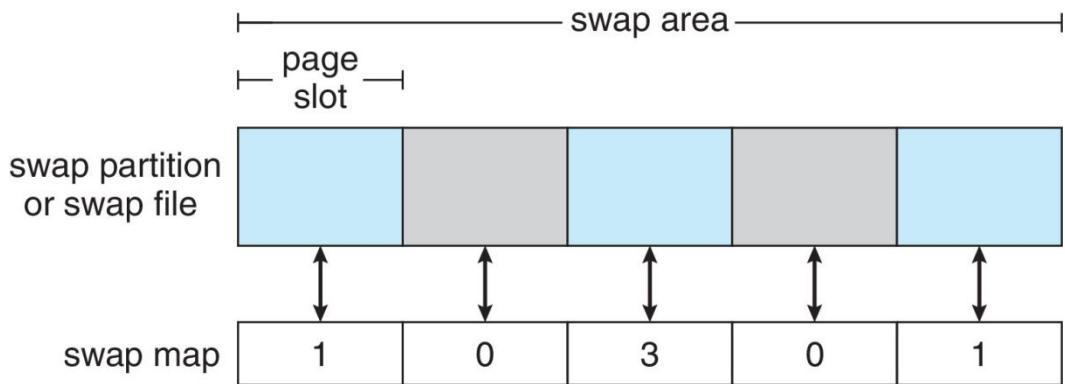
3、磁盘管理

低级（物理）格式化、分区、逻辑格式化。

A. 什么是逻辑格式化？名词解释？【2016】

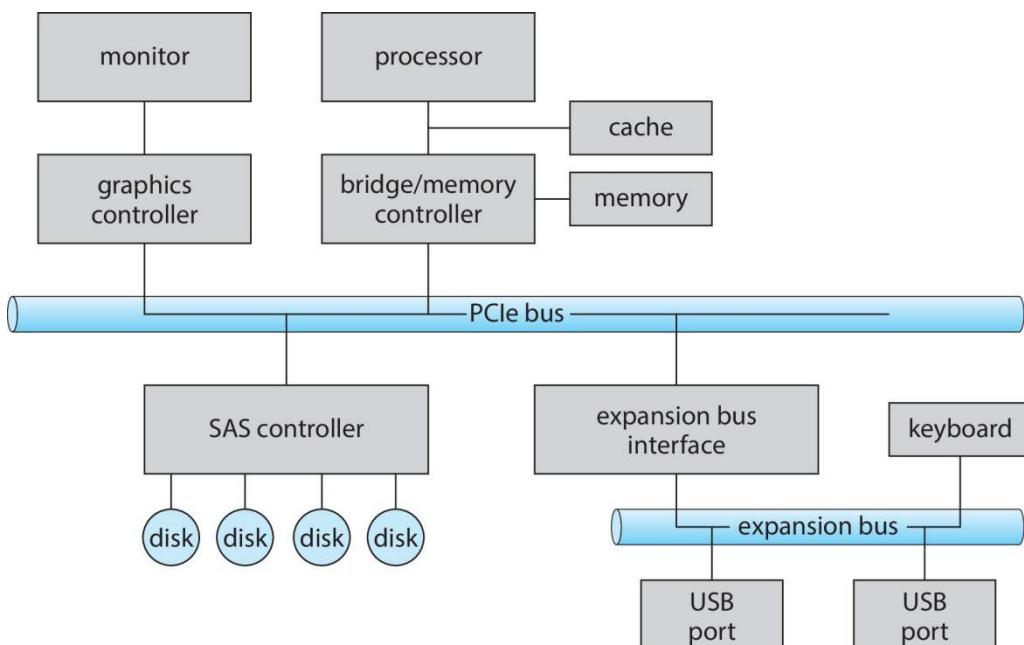
磁盘经过低级格式化被划分为多个扇区，进一步在这些山区上建立文件系统就是逻辑格式化

4、交换空间管理



第十三章 I/O 系统

1、I/O 硬件



2、I/O 方式

轮询、中断、DMA。

- A. I/O 查询方式和特点？【2018】
- B. 请简述在I/O控制方式中，轮询方式，中断方式，DMA方式的工作过程和各自的优缺点？【2022】

1. [I/O管理] I/O方式中，简述轮询、中断、DMA方式的机制和各自优缺点，并给出一个外部空闲空间分配和回收的方案。 (10分) 【2022】

- 轮询：计算机从外部设备读取的每个字，CPU需要对外设状态进行循环检查，直到确定该字已经在I/O控制器的数据寄存器中。在程序直接控制方式中，由于CPU的高速性和I/O设备的低速性，致使CPU的绝大部分时间都处于等待I/O设备完成数据I/O的循环测试中，造成了CPU资源的极大浪费。在该方式中，CPU之所以要不断地测试I/O设备的状态，就是因为在CPU中未采用中断机构，使I/O设备无法向CPU报告它已完成了字符的输入操作。程序直接控制方式虽然简单且易于实现，但其缺点也显而易见，由于CPU和I/O设备只能串行工作，导致CPU的利用率相当低。
- 中断：中断驱动方式的思想是，允许I/O设备主动打断CPU的运行并请求服务，从而“解放”CPU，使得其向I/O控制器发送读命令后可以继续做其他有用的工作。我们从I/O控制器和CPU两个角度分别来看中断驱动方式的工作过程。从I/O控制器的角度来看，I/O控制器从CPU接收一个读命令，然后从外部设备读数据。一旦数据读入I/O控制器的数据寄存器，便通过控制线给CPU发出中断信号，表示数据已准备好，然后等待CPU请求该数据。I/O控制器收到CPU发出的取数据请求后，将数据放到数据总线上，传到CPU的寄存器中。至此，本次I/O操作完成，I/O控制器又可开始下一次I/O操作。从CPU的角度来看，CPU发出读命令，然后保存当前运行程序的上下文（现场，包括程序计数器及处理机寄存器），转去执行其他程序。在每个指令周期的末尾，CPU检查中断。当有来自I/O控制器的中断时，CPU保存当前正在运行程序的上下文，转去执行中断处理程序以处理该中断。这时，CPU从I/O控制器读一个字的数据传送到寄存器，并存入主存。接着，CPU恢复发出I/O命令的程序（或其他程序）的上下文，然后继续运行。中断驱动方式比程序直接控制方式有效，但由于数据中的每个字在存储器与I/O控制器之间的传输都必须经过CPU，这就导致了中断驱动方式仍然会消耗较多的CPU时间。
- DMA：在中断驱动方式中，I/O设备与内存之间的数据交换必须要经过CPU中的寄存器，所以速度还是受限，而DMA（直接存储器存取）方式的基本思想是在I/O设备和内存之间开辟直接的数据交换通路，彻底“解放”CPU。DMA方式与中断方式的主要区别是，中断方式在每个数据需要传输时中断CPU，而DMA方式则是在所要求传送的一批数据全部传送结束时才中断CPU；此外，中断方式的数据传送是在中断处理时由CPU控制完成的，而DMA方式则是在DMA控制器的控制下完成的。
- 分配和回收的方案见题目3.

3、IO 应用接口

1) 两种方式接口：阻塞（同步）、非阻塞（异步）。

- A. 非阻塞和阻塞IO是什么，主要有什么不同，分别用在哪里？【2013】
- B. 访存操作可能会导致IO的进行，某进程读写文件时可能并没有IO设备执行，为什么？（8分）【2015】

【仅供参考】 读写文件首先是CPU读写内存，然后内存中的数据与设备进行IO交互。所以读写文件时在CPU和内存交互时并没有IO设备参与。

2. [I/O管理] 非阻塞和阻塞I/O是什么，主要有什么不同，分别用在哪里。

操作系统的I/O接口还涉及两种模式：阻塞和非阻塞。

阻塞I/O是指当用户进程调用I/O操作时，进程就被阻塞，需要等待I/O操作完成，进程才被唤醒继续执行。

非阻塞I/O是指用户进程调用I/O操作时，不阻塞该进程，该I/O调用返回一个错误返回值，通常，进程需要通过轮询的方式来查询I/O操作是否完成。比如：打印机的输入和输出。

大多数操作系统提供的I/O接口都是采用阻塞I/O。等待I/O操作完成，将数据读入内存等等

C. 阻塞IO: 【2019】

操作系统内核对于I/O只有两种方式：阻塞和非阻塞。调用阻塞I/O时，应用程序需要等待I/O完成才返回结果，阻塞I/O的一个特点是调用之后一定要等到系统内核层面完成所有操作后，调用才结束。

调用非阻塞I/O跟阻塞I/O的差别为调用之后立即返回，返回后，CPU的时间片可以用来处理其他事务，此时性能是提升的。但是非阻塞I/O的问题是：由于完整的I/O没有完成，立即返回的并不是业务层期望的数据，而仅仅是当前调用的状态。为了获取完整的数据，应用程序需要重复调用I/O操作来确认是否完成。这种重复调用判断操作是否完成的技术叫做轮询。

轮询技术主要有以下四种：

1、read:最原始、性能最低，通过重复调用来检查I/O的状态来完成完整数据的读取。在得到数据前，CPU一直好用在等待上。

2、select:改进了read，通过对文件描述符上的事件状态来进行判断。有一个较弱的限制为由于它采用一个1024长度的数组来存储状态，所以它最多可以同时检查1024个文件描述符。

3、poll:对select改进，采用链表方式避免数组长度的限制，其次它能避免不需要的检查。担当文件描述符较多时，它的性能还是低下。

4、epoll:Linux下效率最高的I/O事件通知机制，在进入轮询的时候如果没有检查到I/O事件，将会进行休眠，直到事件发生将它唤醒。它是真实利用了事件通知、执行回调的方式，而不是遍历查询，所以不会浪费CPU，执行效率较高。

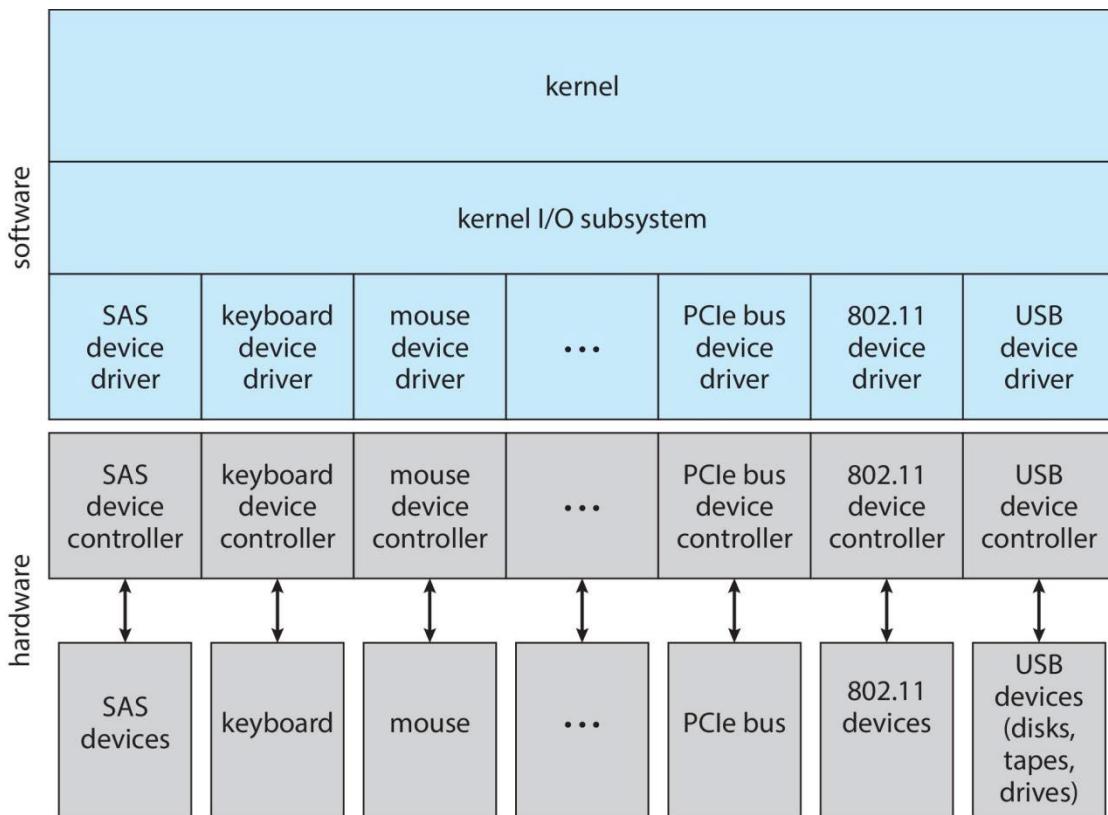
5、kequeue:与epoll类似，仅在FreeBSD系统下存在。

2) 不同设备类型的抽象

A. Spooling: 【2019, 2017】

即同时联机外围操作，又称脱机操作。在多道程序环境下，可利用多道程序中的一道程序，来模拟脱机的输入输出功能。即在联机条件下，将数据从输入设备传送到磁盘，或从磁盘传送到输出设备。

或者解释为：即假脱机I/O技术，本质上就是对I/O操作进行批处理，是一种以空间换时间的技术。优点有提高了I/O速率、将独占设备改造为共享设备、实现了虚拟设备的功能。



第十四章 保护

注意：这部分不知道有无归为考点？2020 级未归为考点，2021 级请询问任课教师！

- 外设、CPU、内存，操作系统对并行的保护措施？【2019】
- 结合你所学的计算机知识，简述保护的概念（从硬件执行、文件管理、存储管理、设备管理、进程同步等方面回答）【2017】
- 计算机系统为保证操作系统及应用程序的可靠真确执行，采取了很多保护措施，阐述你所了解的相关技术。（提示：从硬件的执行模式、进程对临界区的并发访问、存储管理、文件管理、设备管理等内容涉及的保护技术进行阐述）
- 结合你所学的计算机知识，简述保护的概念。（卷子上提示：从硬件层面、文件管理、设备管理、存储管理、进程同步等方面）（10分）【2015】
- 什么是保护域（domain of protection）？【2012】

E. 什么是 protection domain? 【2012改】

【2012】的填空题 不做要求，感兴趣可以填一填。

1. _____ 系统能保证紧急任务能被按时完成。
2. 线程分两种，分别是_____ 和_____。
3. 直接控制外部设备的核心模块称为_____。
4. 如果系统中所有作业是同时到达的，则使作业平均周转时间最短的作业调度算法是_____。
5. 磁盘空闲空间的管理方法主要有_____ 和_____。
6. 将一台独占设备改造成共享设备的一种行之有效的方法是_____。
7. 同步机制应该满足的条件是让权等待，_____，_____ 和_____。
8. 采用_____ 可以缓和 CPU 和外部设备速度不一致的矛盾。
9. 操作系统是_____ 和_____ 之间的一层系统软件。
10. 链接分配方法的一个重要的变种是_____。这个简单而有效的磁盘管理方法被用在 MS-DOS 和 OS/2 中。
11. 对计算机系统中的各类资源提供通用的分配和控制功能是_____ 的重要功能。
12. 一段时间内只允许一个进程访问的资源称为_____。
13. I/O 控制方式主要包括_____，_____，_____ 以及通道方式。