第一章 概述

1. 操作系统基本特征：**并发**、**共享**、**虚拟**、**异步**
2. 操作系统是指**控制**和**管理**整个计算机系统的**硬件**和**软件**资源，并合理地**组织**和**调度**计算机的工作和资源的分配，以提供给用户和其它**软件**方便的**接口和环境**，它是计算机系统中**最基本的系统软件**。
3. 操作系统发展

   批处理系统（单道：一道作业 多道：多个程序）

   分时系统（时间片，人机交互性） **PC 系统**（个人计算机 Windows linux MacOS）

   并行系统（多处理机） **分布系统**（多台 PC）

   实时系统（某个时间内完成某些紧急任务）

   PS: 实时系统要求在严格时间内响应、处理事件，强调时间确定性；分时系统是多个用户分享 CPU 时间，响应时间以人机交互舒适为目标。
4. 单道批处理：内存中只有一个作业

   多道批处理：允许多个程序进入内存并在 CPU 交替运行
5. 操作系统的功能模块

   （1）**处理机**管理（进程管理 作业管理） （2）**存储器**管理

   （3）**文件**管理 （4）**设备**管理
6. CPU 运行模式 用户态 内核态

   （1）用户态->内核态 中断引起，硬件完成

   （2）特权指令 **IO 指令**等
7. 中断与异常

   （1）中断（外中断-硬件） IO 结束中断、时钟中断、缺页中断

   （2）异常（内中断） 非法操作码 地址越界 溢出 陷入 **访存缺页**

   （3）外中断可屏蔽，异常不可以，发生必须处理

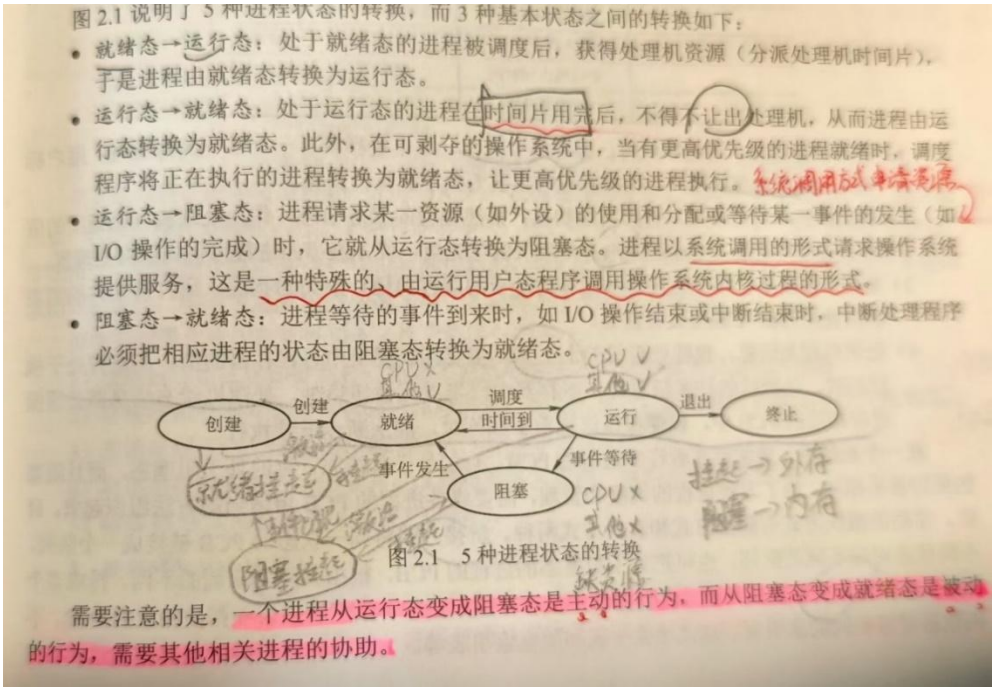
   （4）异常分为**故障**（指令）、**自陷**（用户态调内核态）、**终止**（CPU 硬件故障）

   （5）中断处理要保存 **PC 程序计数器**（硬件自动完成）、**程序状态字寄存器**（PSW）、**通用寄存器**
8. 并发和并行：并发是指多个事件在同一时间间隔内发生；并行是指多个事件在同一时刻发生。

9. 虚拟机：利用 CPU **调度**和**虚拟内存技术，**将物理硬件和 OS 内核统一看作为硬件
10. 操作系统的**软件体系结构**的发展：**单一结构，核心层次结构，微内核结构**
11. 操作系统内核架构
    （1） 宏内核：系统主要功能模块运行在核心态（主流操作系统），为用户程序提供高性能的系统服务
    （2） 微内核：最基本的功能留在内核，非核心的功能移动到用户空间（谷歌 Fuchsia 华为鸿蒙 OS）
    （3） 微内核结构是以**微内核为 OS 核心，以客户/服务器为基础，采用面向对象程序设计的特征，**是当今最有发展前途的 OS 结构。

第二章 进程

1. 引入进程的目的是更好地使**多道程序并发执行，**提高**资源利用率**和**系统吞吐量**。引入线程的目的是**减小**程序在并发执行时所付出的**时空开销，**提高操作系统的**并发性能**。
2. 进程实体（进程映像）：**程序段、相关数据段、PCB**（进程控制块）
3. 进程特征：动态、并发、独立、异步、交互、结构
4. 进程状态



5. PCB：进程控制块，描述进程的基本情况和运行状态，是进程唯一存在的标志。
6. 进程通信

（1）PV 操作：低级通信方式

（2）高级通信方式：**共享存储、消息传递、管道通信（生产者消费者方式）**

7. 线程与进程

（1）线程是独立调度的基本单位，进程是拥有资源的基本单位（线程仅有必不可少的小部分资源）

（2）线程的实现方式：用户级线程和内核级线程

连接方式：多对一、一对一、多对多

（3）同一进程内多线程可高效并发，进程间线程也可并发，并发粒度更细

（4）线程创建、切换仅需处理少量线程私有数据，开销小，通信（同进程内）无需跨进程边界，更高效

8. 调度层次：

（1）高级（作业）调度：按一定的原则从外存上处于后备状态的作业中挑选一个（或多个）作业

（2）中级（内存）调度：将那些暂时不能运行的进程调至外存等待，或将外存上那些已具备运行条件的就绪进程再重新调入内存，**提高内存利用率和系统吞吐量**

（3）低级（进程）调度：按照某种方法和策略从就绪队列中选取一个进程，将处理机分配给它

9. 调度算法：

（1）先来先服务调度算法（FCFS）：利于长作业

（2）短作业优先调度算法（SJF）：饥饿现象；平均等待时间、周转时间最小

（3）优先级调度算法

（4）时间片轮转调度算法：适用于**分时**系统

（5）多级队列调度算法：多个就绪队列，不同队列可以不同优先级

（6）多级反馈队列调度算法：**时间片轮转**和**优先级调度**算法的发展

（7）高响应比优先调度算法：响应比=1+等待时间/服务时间，解决饥饿

10. 上下文：每一时刻 CPU 寄存器和程序计数器的内容给

上下文切换时，将旧进程状态保存在其 PCB 中，加载新进程上下文

11. 引起进程调度的事件：进程创建、进程终止、I/O 请求、时间片耗尽、中断事件

12. 临界资源：一次仅运行一个进程使用的资源

    临界区：进程访问临界资源的代码段

13. 同步（直接制约关系）：协调多个线程/进程的执行顺序，确保某些操作必须按特定先后顺序发生

    互斥（间接制约关系）：保证同一时间只有一个线程/进程访问临界资源，防止数据冲突

14. 同步机制准则（临界区）：空闲让进，慢则等待，有限等待、让权等待

15. 临界区互斥方法

    （1）软件：

    单标志法：turn 表示临界区进程编号，违背空闲让进（若有一个不进）

    双标志法先检查：flag[i]，进程是否被 i 访问

    双标志法后检查：先设置自己标志再检查，可能会饥饿（同时设置）

    Peterson 算法：1 和 3 结合，但违背让权等待，空转（忙等待）

    （2）硬件：中断屏蔽、硬件指令（TS 和 Swap 指令），不能让权等待，饥饿

16. P/V 操作的含义

- **P(S)**（Wait 操作）：
  - 信号量 `S` 减 1。
  - 若 `S ≥ 0`，进程继续执行；
    若 `S < 0`，进程阻塞，进入等待队列。
- **V(S)**（Signal 操作）：
  - 信号量 `S` 加 1。
  - 若有进程在等待 `S`，则唤醒其中一个。

信号量是一种 **整数变量**，用于控制多个进程/线程对共享资源的访问，核心功能：

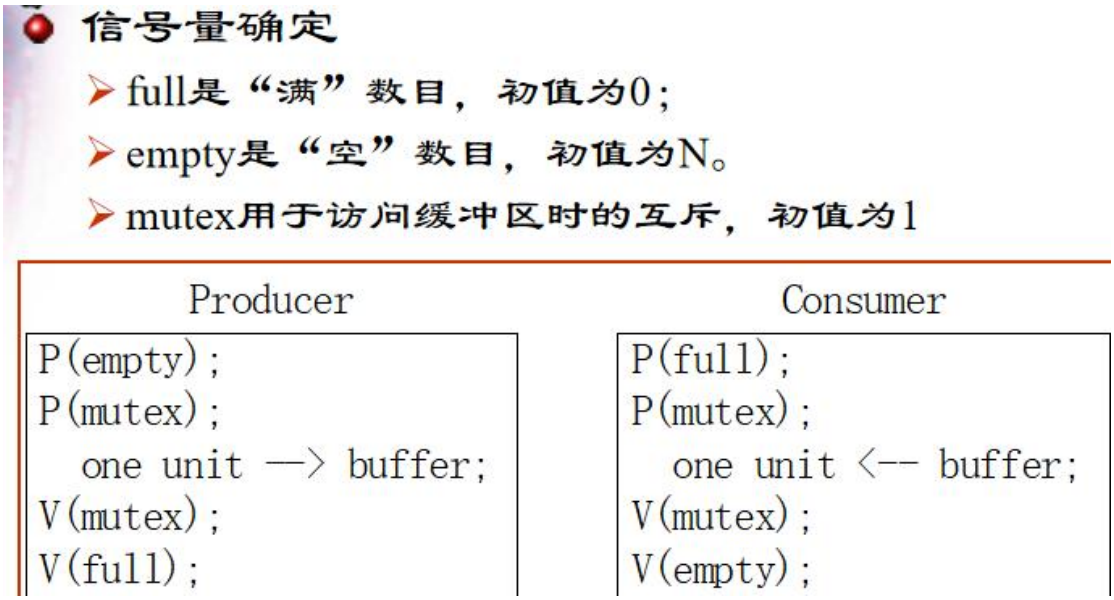
- **同步**：协调进程的执行顺序（如A进程必须等B进程完成后执行）。
- **互斥**：确保同一时间只有一个进程访问临界资源。

执行一次 P 操作时，信号量应该**减 1（或 -1、做减 1 操作）**；当其值为**小于 0（或 <0）**时，进程应该阻塞。

执行一次 V 操作时，信号量应该**加 1（或 +1、做加 1 操作）**；当其值为**小于等于 0（或 ≤0）**时，应该唤醒阻塞队列中的进程。

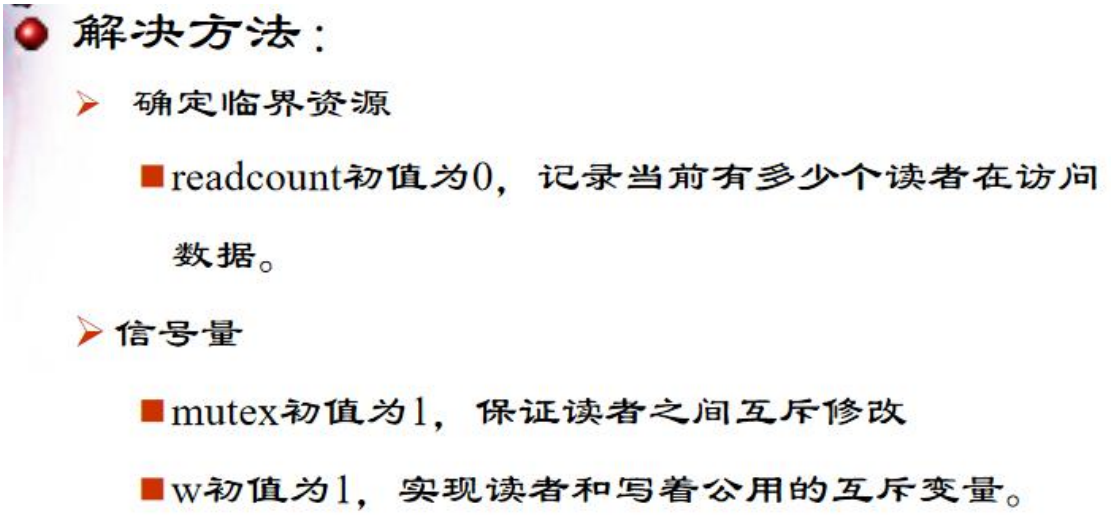
17. 通过信号量实现同步和互斥，实现让权等待

18. （1）生产者消费者问题



（2）读者写者

读者优先



读者：

```
while (true) {
P(mutex);
   readcount ++;
   if (readcount==1)
      P (w);
V(mutex);
  读
P(mutex);
   readcount --;
   if (readcount==0)
      V(w);
 V(mutex);
 };
```

写者：

```
      while (true) {


     P(w);
      写
     V(w);


      };
```

第一类读写问题

同济大学软件学院

写者优先

- w初值为1，读者和写者公用的互斥信号量；
- x初值为1，readcount读者数目的互斥信号量；
- y初值为1，wirtecount写者数目的互斥信号量；
- r,z初值为1，实现写者优先；

读者：

```
while (true) {
   P(z);
   P(r);
     P(x);
        readcount++;
        if(readcount==1) P(w);
    V(x);
   V(r);
   V(z);
     读
   P(x);
     readcount--;
     if(readcount==0) V(w);
   V(x);
   };
```

写者：

```
  while (true) {
    P(y);
      writecount++;
      if(writecount==1)  P(r);
    V(y);
    P(w);
        写
    V(w)
    P(y);
      writecount--;
      if(writecount==0)  V(r);
    V(y);
};
```

同济大学软件学院

（3）哲学家就餐问题

19. 死锁的四个必要条件：**互斥、非抢占、持有并等待、循环等待**
20. 克服死锁：死锁预防（破坏四个条件之一）、死锁避免（检查资源分配状态）、死锁检测和恢复（允许系统进入死锁状态，打破死锁状态）
21. 打破死锁状态的两个方法：终止进程方法、资源抢占方法、进程回退法

第三章 内存管理

1. 程序和数据装入：

   编译：将用户代码编译程若干个目标模块

   链接：将目标模块以及所需库函数链接在一起，形成完整装入模块。

   装入：将装入模块装入内存
2. 程序链接技术：静态链接、装入时动态链接、运行时动态链接
3. 程序装入技术：

   绝对装入技术：程序地址空间和内存地址空间是一一对应

   可重定位装入技术：静态重定位（装入过程）和动态重定位（执行过程）
4. 内存扩充技术：覆盖技术（发生在同一进程或作业内）和交换技术（发生在进程或作业之间，等待状态挂起到磁盘）

5. 分区（内存）保护：定位寄存器（基地址）和界限寄存器（限长）
6. 分区存储管理：固定分区（有内碎片无外碎片）

   动态分区（有外碎片）
7. **动态分区**策略

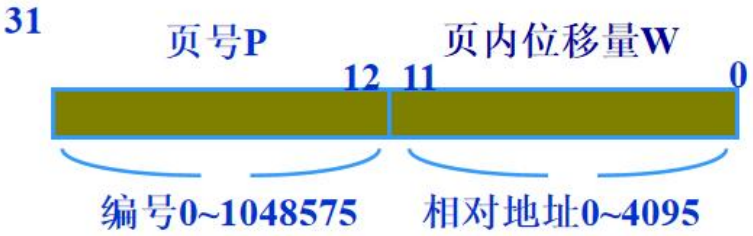
   （1）**最先适配算法**：从头查找，找到符合要求的第一个分区（低地址由很多很小的空闲分区，在高地址部分保存较大空闲区）

   （2）**循环最先适配算法**：从上次分配的分区起查找

   （3）**最佳适配算法**：找到第一个满足且最小的分区（较大的空闲分区可以被保留，但产生较多的外部碎片）

   （4）**最坏适配算法**：分区时取所有空闲区中最大的一块（导致没有大分区）
8. 页式存储管理

   （1）结构



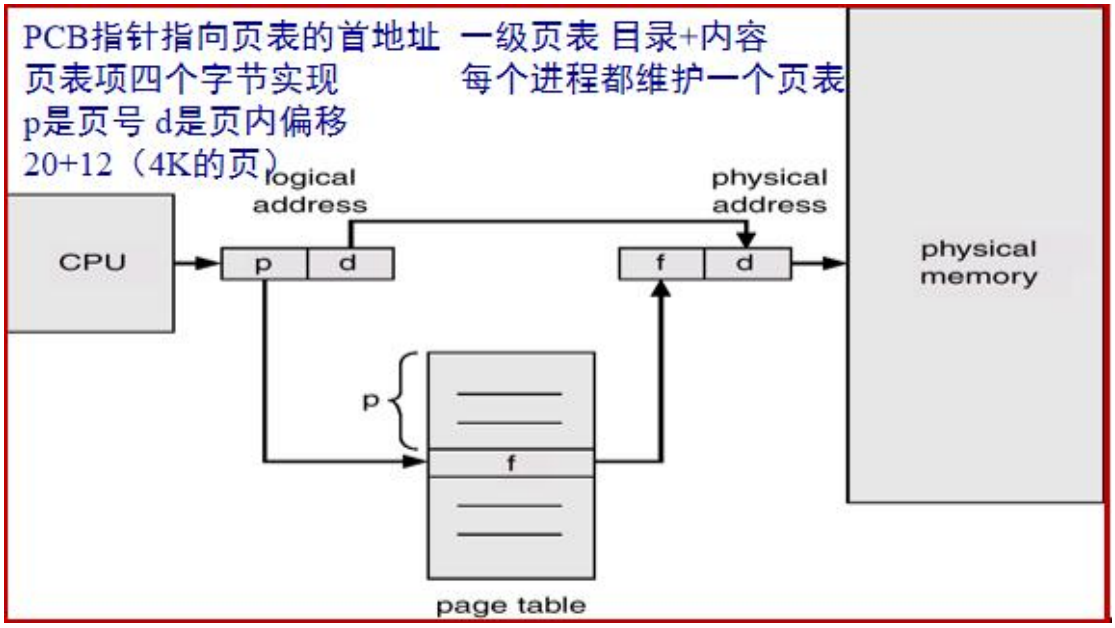
   2 的 32 4G（内存大小）

   2 的 12 4K（正常页大小--页内位移量）

   2 的 20 1M（页的个数--页号）

   （2）页表：每个进程建立一个，记录页面在内存中的物理块号，存放在内存中，**实现页号到物理块号的地址映射的数据结构**，页表项：4B（页号+块号）

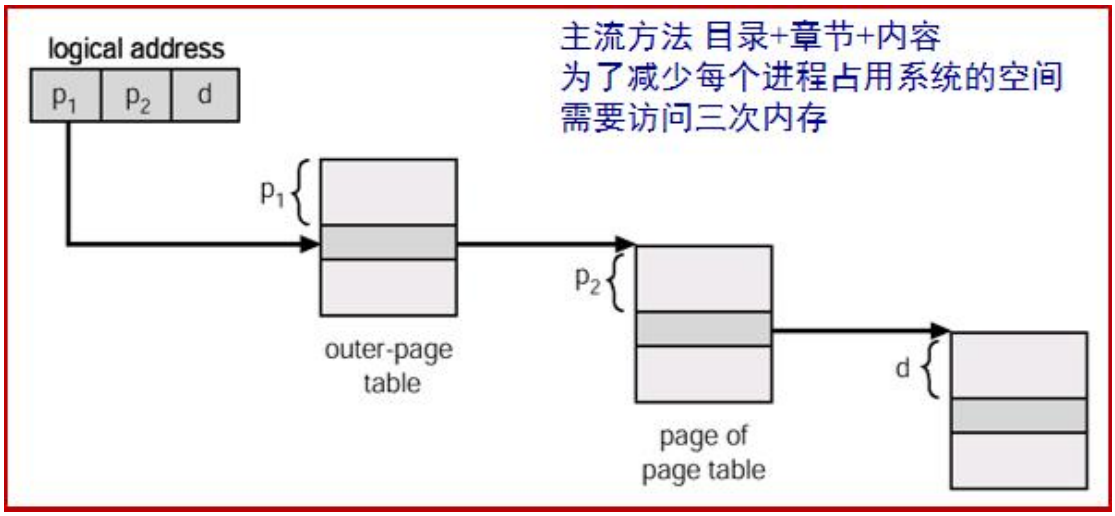
   （3）地址映射：



（4）解决了碎片问题，便于管理，但不易实现程序共享和动态链接

9. 快表 TLB（联想存储器）：是一种高速缓冲存储器，存储近期常用的页表项，加速虚拟地址到物理地址的转换

10. 二级页表（10+10+12）



（1）一级页表（目录）**有一个页面**，容纳 4KB/4B=1K 个页表项（2 的 10 次）

（2）二级页表表项是 2 的 10 次=1K 个

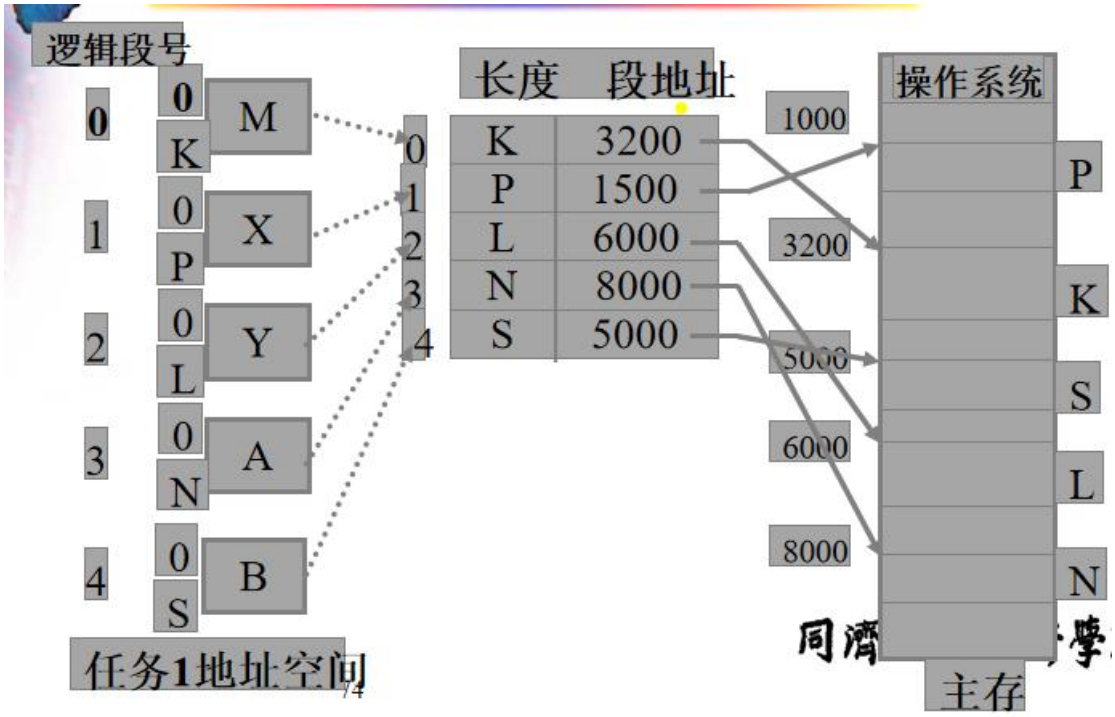
（3）二级页表在部分装入发挥作用，全部装入要读很多次内存

11. 段式存储管理：按程序自身的逻辑关系划分为若干个程序段，以段为单位分配内存，每一个段在内存中占据连续空间

（1）结构

段号	段首址	段长度
0	58K	20K
1	100K	110K
2	260K	140K

（2）



（3）便于动态申请内存、管理和使用统一化、便于共享、便于动态链接，但会产生碎片

12. 段式页式存储管理

（1）每个进程可有有一个段表，多个页表

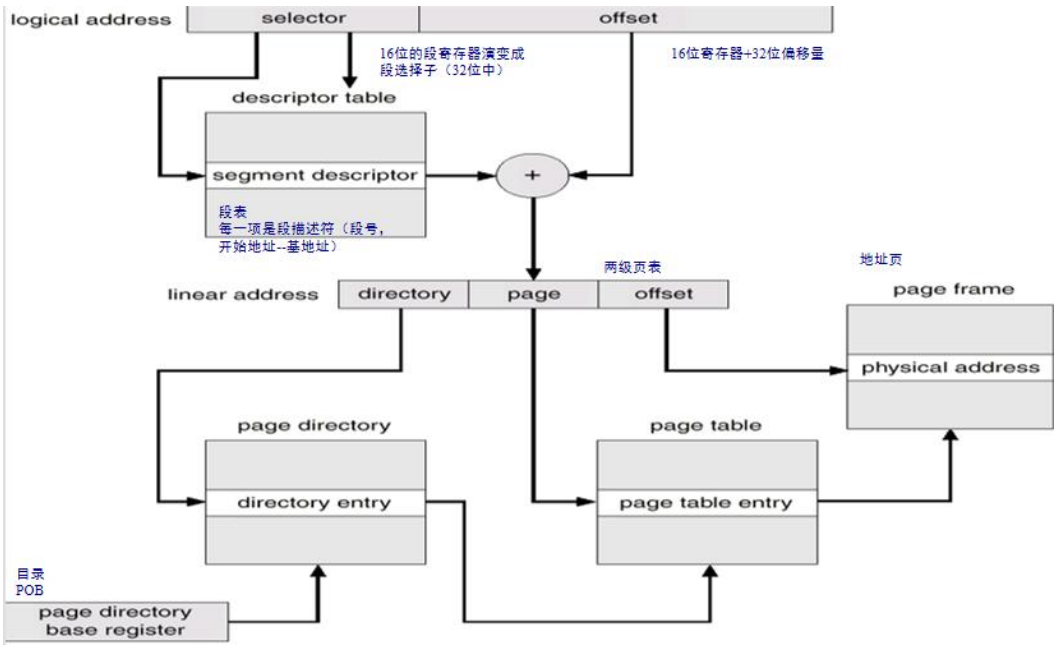
（2）

进程空间：段式划分

内存空间：页式管理

页为单位进行分配

（3）



13. **虚拟存储器**是指具有**请求调页功能**和**置换功能**，能从逻辑上对**内存**容量加以扩充的一种存储器系统（部分装入）。

    特征：**不连续性**（物理内存分配不连续）、**对换性、虚拟性、多次性**

    实现方式：请求分页、请求分段、请求段页式

14. 请求分页页表结构

页号	中断位	内存块号	外存地址	访问位	修改位

    中断位：是否调入内存

    访问位：最近多久被访问/次数

    修改位：调入内存后是否被修改过

15. 页面置换算法：缺页在两级都可能发生，因为是部分装入

    （1）**先进先出算法(FIFO)**：淘汰最早进入的页面，有 **Belady 异常**（分配块数增加反而缺页次数不减反增）

    （2）**最佳（OPT）算法**：最长时间没访问的页面替换（理想算法）

    （3）**最近最少使用算法（LRU)**：置换最长时间没有使用的页

    （4）**最不常用算法(LFU)**：到当前时间为止被访问次数最少的页面（计数器）

    （5）**轮转算法**：页被访问则置 user =1，寻找 use=0 的页面作为被置换页，并将指针经过的页修改为 user=0

16. （1）调入策略：

交换区：进程装入时，将全部页面复制到交换区，以后总是从交换区调入。

文件区：未被修改的页面，直接从文件区读入，被置换时不需调出；已被修改的页面，被置换时需调出到交换区，以后从交换区调入。

（2）调出策略：请求调出和预调出

17.（1）常驻集：虚拟页式管理中给进程分配的物理页面数目（因为不是所有页都调入主存），**有固定分配**和**可变分配**（按照缺页率动态调整）

置换范围：固定分配+局部置换，可变分配+局部置换，可变分配+全局置换（局部置换是指只能从分配的页面置换，全局可以从空闲的物理块置换）

（2）工作集：依据进程在过去一段时间内访问的页面来**调整常驻集大小**

第四章 文件管理

1. （1）**文件**是以计算机硬盘为载体存储在计算机上的信息集合。

（2）文件系统是一种操作系统提供的用于管理和组织文件的软件机制。

2. （1）文件的逻辑结构：数据的逻辑顺序、数据的类型和数据之间的关联关系；

（2）文件的物理结构：文件在存储介质上的实际存储方式

（3）物理结构：**顺序结构、链接结构、索引结构**

（4）无结构文件：**流式文件**（采用的文件逻辑结构）

有结构文件：**记录文件**

3. UNIX 文件系统采用多级索引结构(综合模式)：文件系统中维护一个索引节点表，每个索引节点（i 节点）对应一个文件。文件索引节点（inode）里设 13 个地址项：10 个直接地址，1 个一次间接地址，1 个二次间接地址，1 个三次间接地址

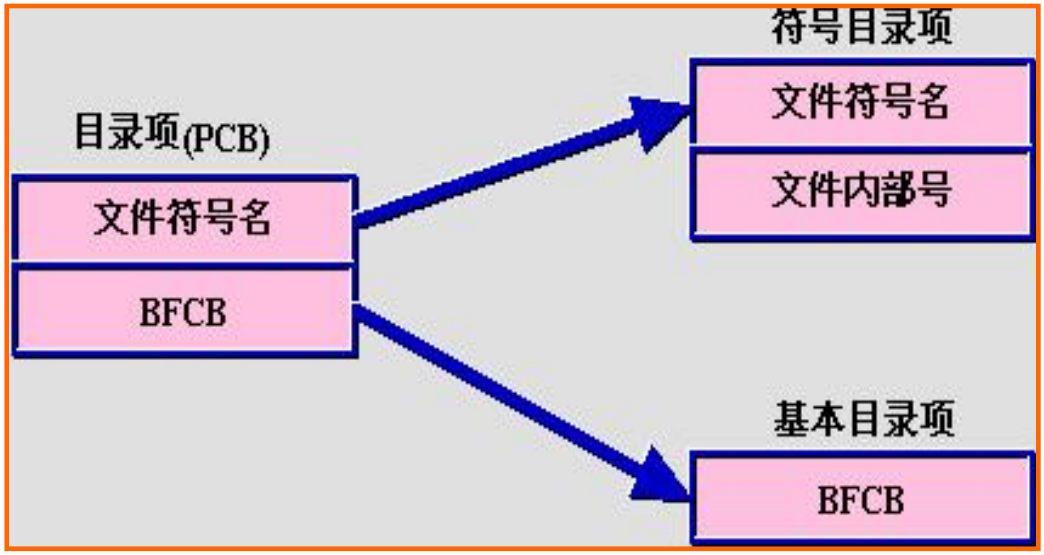
4. 磁盘空闲空间的管理方法：**空闲表法**（所有空闲块记录在一个表）、**空链表法**（所有空闲块链成一个链）、**位图法**（位示图，占内存）、**成组链接法**（每组的第一个空闲块记录了下一组空闲块的块数和块号，UNIX）

5. 目录结构：

（1）文件控制块（FCB）：存放了为管理文件所需的所有有关信息（文件属性）

（2）一级目录、二级目录、树型目录（多级目录）

（3）目录改进（加快目录检索）：把 FCB 分成两部分



举例：

- 一个FCB有48个字节，符号目录项占 8字节，文件名6字节，文件号2字节，基本目录项占 48-6=42字节（linux只考虑文件名，忽略文件号）。假设，物理块大小512字节，目录文件有128个目录项。
- 改进前：1个物理块含512/48=10个FCB，则该目录文件占13个物理块；
- 改进后：1个物理块含512/8=64个符号目录项或512/42=12个基本目录项，则该目录文件符号文件占2块，基本文件占11块。

- 查找一个文件的平均访盘次数

改进前：(1+13)/2=7次

改进后：(1+2)/2 +1 =2.5次

减少了访问硬盘的次数，提高了检索速度

6. 文件共享

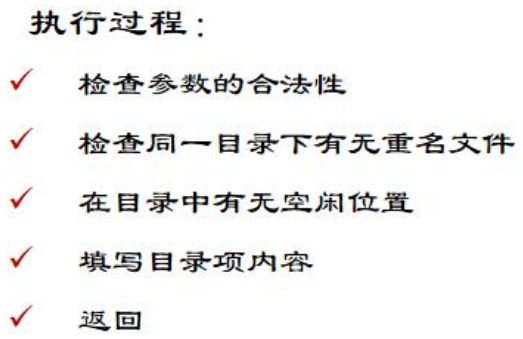
（1）系统文件表：系统打开文件表（整个系统一张）放在内存，用于保存已打开文件的 FCB 文件号、共享计数、修改标志

（2）用户文件表：每个进程一个，进程的 PCB 中，记录了用户打开文件表的位置

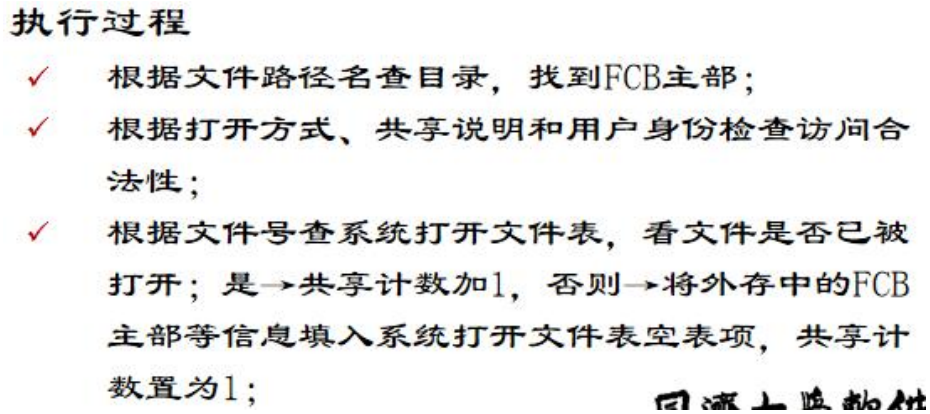
（3）**基于索引节点的硬链接**（拥有索引节点）和**基于符号链的软链接**（拥有路径名）

7. 安全性写方式：谨慎写、延迟写
8. 文件执行

（1）Create：实质是建立文件的 FCB，并建立必要的存储空间，分配空 FCB，根据提供的参数及需要填写有关内容，返回一个文件描述.



（2）Open()：文件使用前都要先打开，即把 FCB 送到内存。fd=open（文件路径名，打开方式）

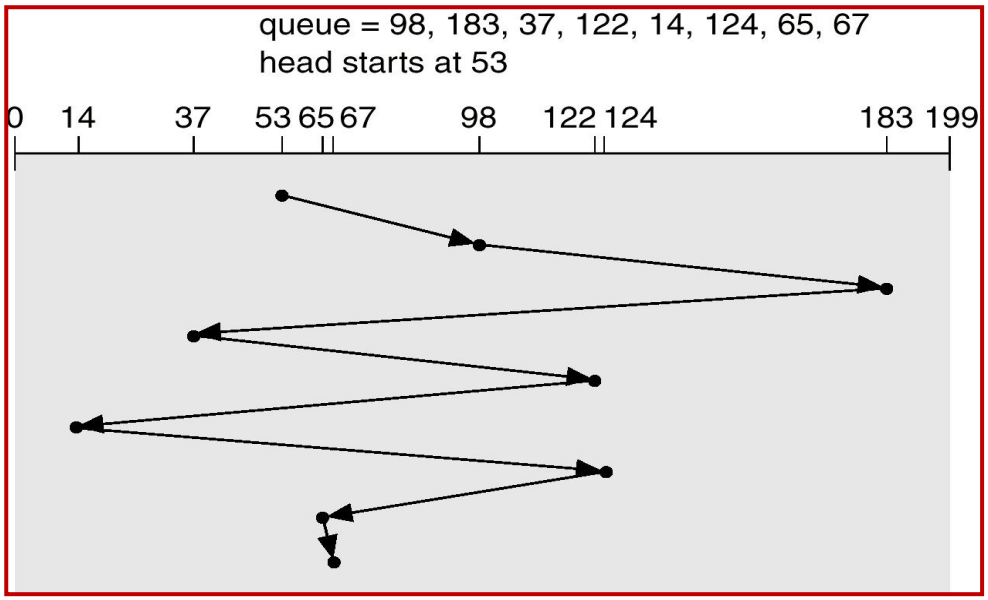


第五章 IO、磁盘

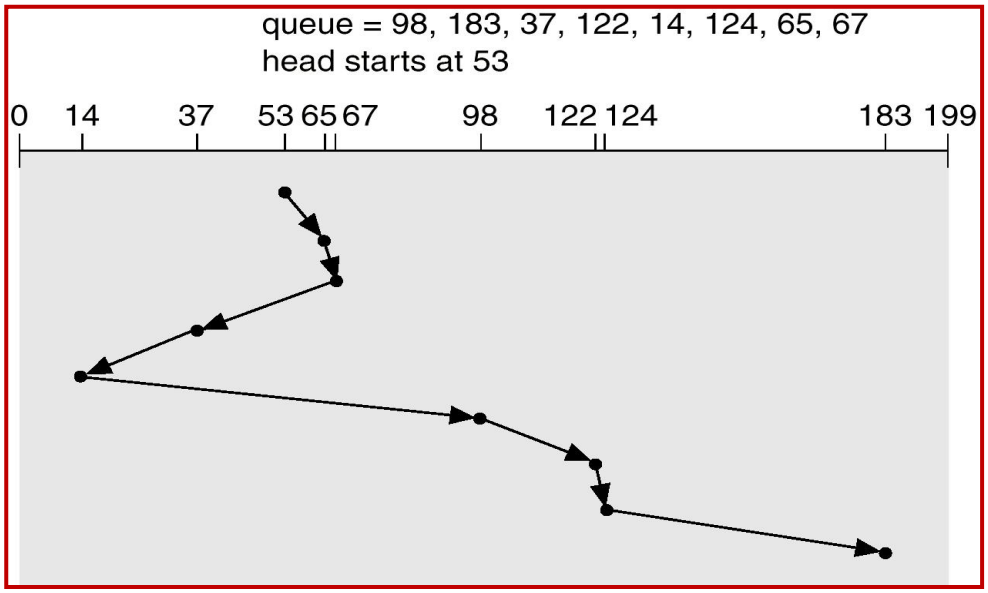
1. I/O 设备和 CPU 可并发执行

2. I/O 技术：可编程 I/O、中断、直接内存存取(DMA)、通道技术
3. 磁盘物理地址形式：磁头号（盘面号）、磁道号（柱面号-半径）、扇区号
4. 硬盘分类：固定头磁盘（每个磁道设置一个磁头）、移动头磁盘
5. 磁盘读取时间：**寻道时间、旋转延迟时间、数据传输时间**
6. 廉价磁盘冗余阵列 RAID：SFT-III 技术，技术特点： 并行交叉存取
7. 磁盘调度算法

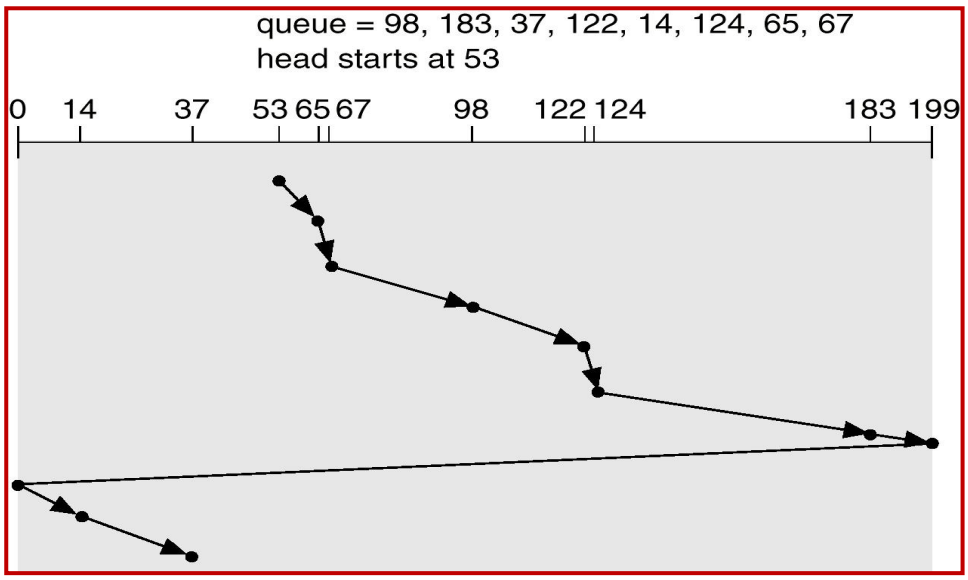
（1）先来先服务算法：根据访问磁盘的先后顺序调度



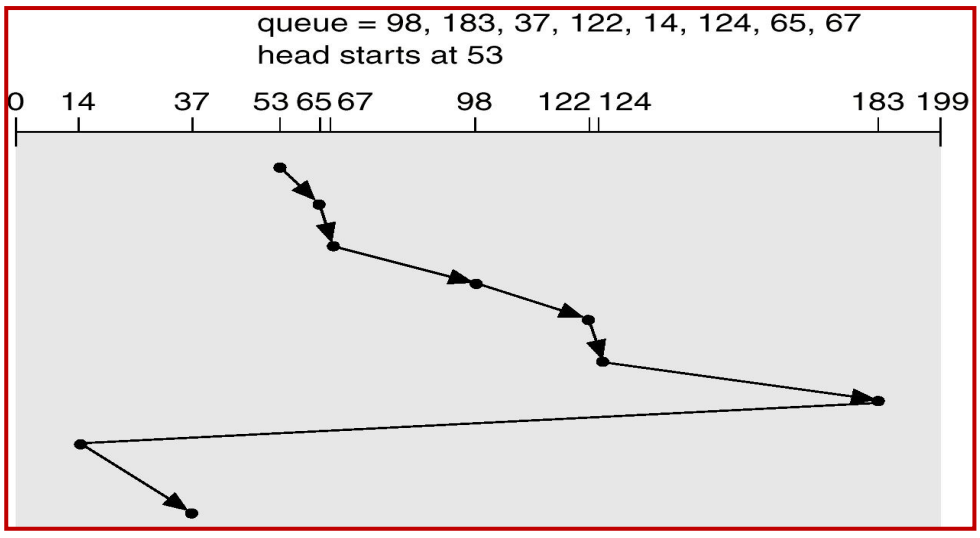
（2）最短寻道时间优先：处理当前磁头所在磁道距离最短的磁道（饥饿）



（3）扫描 SCAN 算法（电梯算法）：最短的基础上加上方向



（4）单向扫描算法：规定磁头单向移动（固定方向，例如从内到外），结束就快速返回起始段



参考题目与背诵

12. 假设*n*个进程共享一个互斥段，最多允许*m*个进程(*m*<*n*)同时进入该互斥段，则信号量的变化范围是（ ）。

○ - (n - m) ~ m

15. 解决“碎片”问题最好的存储管理方法是（ ）。

- **段式存储管理**：会产生段间碎片，即不同段之间因段长不匹配，空闲空间无法有效利用，不能很好解决碎片问题。
- **静态分区管理**：分区大小固定，作业大小不匹配时，易产生内部碎片（分区内未被利用空间）和外部碎片（分区间空闲空间无法分配），无法解决碎片问题。
- **页面存储管理**：以固定大小页面为单位分配内存，虽有内部碎片（最后一页可能未填满），但相比其他方式，可通过换页等机制，让内存分配更灵活，有效减少外部碎片，是解决碎片问题较好的方式，该选项正确。
- **可变分区管理**：动态划分分区，会产生大量外部碎片（如分配、回收后，空闲空间分散），难以高效利用内存，不能很好解决碎片问题。

综上，解决碎片问题最好的存储管理方法是页面存储管理，选“页面存储管理”。

15. 一个物理硬盘可分成多个逻辑分区，不同分区可采用不同文件系统。

○ True

操作系统定义

操作系统（OS，Operating System）是管理计算机硬件与软件资源的计算机程序，是计算机系统的核心，为用户和应用程序提供交互接口，协调、控制计算机各部件高效运作。

主要功能模块及逻辑功能

1. **进程管理（处理器管理）**：负责进程的创建、调度、撤销，通过进程调度算法（如分时、优先级调度），合理分配 CPU 时间，实现多进程并发执行，解决进程同步、互斥（如用信号量、锁机制），保障程序有序运行。
2. **存储管理**：管理内存空间，包括内存分配（为进程分配连续 / 非连续内存区，如分页、分段）、回收（释放进程结束后的内存），借助虚拟存储技术（如请求分页）扩充内存，通过地址映射（虚拟地址转物理地址）让程序正确访问内存，同时进行内存保护（防止进程越界访问）。
3. **文件管理**：管理外存（如磁盘）上的文件，提供文件的创建、删除、读写、查找等操作接口，维护文件目录（如树形目录），管理文件存储空间（分配、回收磁盘块），设置文件权限（控制用户访问），保障文件数据安全、有序存储与访问。
4. **设备管理**：管理计算机输入输出设备（如键盘、硬盘、打印机），实现设备的分配（将设备合理分配给进程）、控制（驱动设备完成 I/O 操作），采用缓冲技术（减少 CPU 与设备间速度差异）、spooling 技术（虚拟设备，提高设备利用率），让设备高效、协调工作。
5. **用户接口**：为用户提供与操作系统交互的界面，包括命令接口（如 Shell 命令行，用户输入指令操作计算机）、图形接口（GUI，通过窗口、图标交互）、程序接口（系统调用，供应用程序请求系统服务，如文件读写的系统调用），方便用户和程序使用操作系统功能。

1. 请求页式存储管理机制实现过程

1. **虚拟地址划分**：将虚拟地址拆分为**虚拟页号（VPN）**和**页内偏移（offset）**，页内偏移由页面大小决定（如页面大小 4KB 时，偏移占 12 位）。
2. **页表查询**：通过虚拟页号查询**页表**（记录虚拟页与物理页框映射关系），检查页表项的**有效位**：
   - 若有效位为 1（页在物理内存），结合物理页框号与页内偏移，拼接成**物理地址**，访问内存。
   - 若有效位为 0（页不在内存，缺页），触发**缺页中断**。
3. **缺页中断处理**：操作系统响应中断，从外存（如磁盘）中找到该页对应的内容，分配空闲物理页框，将页载入内存，更新页表项（填写物理页框号、置有效位为 1），然后重新执行触发缺页的指令，完成地址转换与内存访问。

2. 采用二级 / 三级页表的原因

核心解决**页表存储开销大**问题：

- **单级页表弊端**：32 位系统虚拟地址空间 4GB，若页面 4KB，需 100 万级页表项（4GB/4KB = 1M），页表需连续内存存储，占用大量空间（如 4 字节 / 项时，需 4MB）；64 位系统虚拟地址空间极大，单级页表存储开销会失控。
- **多级页表优化**：
  - 二级 / 三级页表将页表**分级存储**，仅在需要时加载对应级别的页表到内存，减少连续内存占用。例如二级页表，虚拟页号拆分为 “一级页号 + 二级页号”，一级页表项指向二级页表，仅用到的二级页表才会分配内存，大幅节省空间。
  - 适配 64 位系统大地址空间，通过分级灵活管理虚拟地址映射，平衡地址转换效率与内存开销，让页表存储更高效、可控。

什么是文件的物理结构？列举3个典型的文件物理结构，并说明其优缺点。

定义

文件物理结构指文件数据在存储设备（如磁盘）上的实际存储布局与组织方式，关乎文件在存储介质上的分配、访问等实现逻辑。

连续结构

为文件分配连续磁盘块，数据按序存于连续物理块。优点是顺序访问快，可直接计算物理地址，实现简单；缺点是文件扩展难，需连续空闲块，易产生磁盘碎片 。

链接结构

文件数据块分散，通过指针串联成链，目录项记首、尾块地址。优点是空间利用灵活，适配动态增长文件，实现简单；缺点是随机访问需遍历指针，效率低，指针损坏易致文件断裂 。

索引结构

为文件建索引表存逻辑块与物理块映射，目录项指向索引表位置，支持多级 / 混合索引。优点是随机访问高效，文件扩展灵活；缺点是需额外存索引表，空间开销大，多级索引访问稍慢 。

题目为：假设有三个进程 PA、PB 和 PC 合作解决文件打印问题：PA 将文件记录从磁盘读入主存的缓冲区 1，每执行一次读一个记录；PB 将缓冲区 1 的内容复制到缓冲区 2，每执行一次复制一个记录；PC 将缓冲区 2 的内容打印出来，每执行一次打印一个记录。缓冲区的大小等于一个记录大小。

1. 请定义相关信号量，并确定其初值；
2. 请用 P、V 操作来保证文件的正确打印。

信号量	初值	作用
`empty1`	`1`	缓冲区1是否为空（初始可写入）。
`full1`	`0`	缓冲区1是否已满（初始无数据可读）。
`empty2`	`1`	缓冲区2是否为空（初始可写入）。
`full2`	`0`	缓冲区2是否已满（初始无数据可打印）。

```
semaphore empty1 = 1, full1 = 0;  // 缓冲区1的信号量
semaphore empty2 = 1, full2 = 0;  // 缓冲区2的信号量

// 进程PA：读取磁盘到缓冲区1
void PA() {
    while (有记录未读) {
        从磁盘读取一个记录;
        P(empty1);          // 等待缓冲区1为空
        将记录写入缓冲区1;
        V(full1);           // 标记缓冲区1已满
    }
}

// 进程PB：复制缓冲区1到缓冲区2
void PB() {
    while (有记录未处理) {
        P(full1);           // 等待缓冲区1有数据
        P(empty2);          // 等待缓冲区2为空
        从缓冲区1复制记录到缓冲区2;
        V(empty1);          // 释放缓冲区1（可重新写入）
        V(full2);           // 标记缓冲区2已满
    }
}

// 进程PC：打印缓冲区2内容
void PC() {
    while (有记录未打印) {
        P(full2);           // 等待缓冲区2有数据
        从缓冲区2读取记录并打印;
        V(empty2);          // 释放缓冲区2（可重新写入）
    }
}
```

2. 某虚拟存储器的用户空间共有32个页面，每页1K，主存16K。假定某个时刻系统为该用户的第0、1、2、3页分配的物理块号分别为5、10、4、7。

a）请写出十六进制逻辑地址05AC所对应的物理地址。

b）能直接实现十六机制逻辑地址103C对应的物理地址转换么？如果能，请给出其对应的物理地址；如果不能，请描述后续转换过程。

答:由题目所给出条件可知,该系统的逻辑地址有 15 位,其中高 5 位为页号,低 10 位为页内地址;物理地址有 14 位,其中高 4 位为块号,低 10 位为块内地址。另外,由于题目中给出的逻辑地址是 16 进制数,故可先将其转换成二进制数以直接获得页号和页内地址,再完成地址的转换.

1. 页号+页内偏移

2. 不在内存发生**缺页中断**：从外存中找到该页对应的内容，然后在内存中寻找空闲物理块（主存 16 个物理块，还有 12 个空闲块可用），将找到的页装入空闲物理块，接着更新页表，把该页对应的物理块号记录到页表中页号为 4 的条目里。完成这些操作后，重新执行地址转换指令，再次尝试将逻辑地址 $103C$ 转换为物理地址 。

处理机管理复习

操作系统是通过 PCB 来控制和管理进程，用户进程可从 PCB 中读出与本身运行状态相关的信息。(**错，用户进程没有访问自身 PCB 权限**)

1. 不管是批处理、分时还是其他基本类型操作系统，**进程调度**是基础，用于调度进程占用 CPU
2. 批处理系统要从作业队列选作业调入内存，需**作业调度**
3. **分时系统**为平衡负载、提高内存利用率，除进程调度外，常引入**中级调度（调出内存挂起）**换入换出进程
4. 多处理机系统要协调多个处理器调度，需**多处理机调度**

系统产生**死锁**是指（**若干进程等待被其他进程占用而又不能被释放的资源**），产生死锁的基本原因是（**资源不足**）和（**进程在使用资源的时候推进速度和顺序不当)**，产生死锁的四个必要条件是（**互斥、非抢占、循环等待、持有并等待**）。

1. 为什么要引入**挂起**状态？

提示：**中级调度**

因为**内存空间有限**，当进程过多时，需要把一些进程移动到外存上，这些被转移到外存上的进程的状态就是挂起状态，这个调度方式就叫中级调度。当内存中一些进程运行结束被删除后，再将外设中的进程转移回主存。这样的操作大大**提高了内存的利用率和系统的吞吐量**。

2. 试说明进程三种基本状态以及各个状态之间的转换情况。

（1）**就绪**状态：进程已获得**除 CPU 外的所有必要资源**，只等待 CPU 时的状态。一个系统会将多个处于就绪状态的进程排成一个**就绪队列**。

（2）**执行**状态：进程已**获 CPU**，正在**执行**。单处理机系统中，处于执行状态的进程只一个；

多处理机系统中，有多个处于执行状态的进程。

（3）**等待**状态：正在执行的进程**由于某种原因而暂时无法继续执行**，便放弃处理机而处于暂停状态，即进程执行受阻。

状态切换：

(1) 就绪→执行

处于就绪状态的进程，当进程调度程序为之**分配了 cpu 资源**后，该进程便由就绪状态转变成执行状态。

(2) 执行→就绪

处于执行状态的进程在其执行过程中，**被优先级高的进程抢占**(或**分配的时间片用完**等，不同种类操作系统有不同原因)，于是进程从执行状态转变成就绪状态。

(3) 执行→等待

正在执行的进程因**等待外部资源等事件**而无法继续执行时，便从执行状态变成等待状态。

(4) 等待→就绪

处于等待状态的进程，若其**等待的事件已经发生**，于是进程由等待状态转变为就绪状态。

3. 说明引起**进程创建**的事件

用户登录：系统为用户创建一个进程，并插入就绪队列

作业调度

提供服务：系统为用户请求创建一个进程

应用请求：用户程序自己创建进程

4. 进程运行时，存在哪**两种制约**关系？举例说明

一、**直接制约——进程同步**

多个相互合作的进程在一些关键点上可能需要**互相等待或互相交换消息**。

二、**间接制约——进程互斥**

当一个进程正在使用**某独占型资源**时，其他希望使用该资源的进程必须等待。

5.用信号量解决“独木桥”问题：同一方向的行人可连续过桥，当某一方向有人过桥时，另一个方向的行人必须等待；当某一方向无人过桥时，另一方向的行人可以过桥。

```
# 全局变量
mutex = 1  # 方向互斥
countA = 0  # A 方向人数
countB = 0  # B 方向人数
semA = 1    # A 方向许可
semB = 1    # B 方向许可

# A 方向行人过桥
def A_person():
    P(semA)          # 申请 A 方向过桥许可
    P(mutex)         # 互斥访问方向计数器
    if countA == 0:  # 若自己是 A 方向第一个人
        P(semB)      # 禁止 B 方向过桥
    countA += 1      # A 方向人数+1
    V(mutex)         # 释放方向计数器互斥锁
    V(semA)          # 释放 A 方向许可

    # 过桥动作（省略具体逻辑）
    print("A 方向行人过桥")

    P(mutex)         # 互斥访问方向计数器
    countA -= 1      # A 方向人数-1
    if countA == 0:  # 若自己是 A 方向最后一个人
        V(semB)      # 允许 B 方向过桥
    V(mutex)         # 释放方向计数器互斥锁
```

```
# B 方向行人过桥
def B_person():
    P(semB)          # 申请 B 方向过桥许可
    P(mutex)         # 互斥访问方向计数器
    if countB == 0:  # 若自己是 B 方向第一个人
        P(semA)      # 禁止 A 方向过桥
    countB += 1      # B 方向人数+1
    V(mutex)         # 释放方向计数器互斥锁
    V(semB)          # 释放 B 方向许可

    # 过桥动作（省略具体逻辑）
    print("B 方向行人过桥")

    P(mutex)         # 互斥访问方向计数器
    countB -= 1      # B 方向人数-1
    if countB == 0:  # 若自己是 B 方向最后一个人
        V(semA)      # 允许 A 方向过桥
    V(mutex)         # 释放方向计数器互斥锁
```

6.设有三个进程 A、B、C，其中 A 与 B 构成一对生产者与消费者（A 为生产者，B 为消费者），共享一个由 n 个缓冲块组成的缓冲池；B 与 C 也构成一对生产者与消费者（此时 B 为生产者，C 为消费者），共享另一个由 m 个缓冲块组成的缓冲池。用 P、V 操作描述它们之间的同步关系。

```
# 全局信号量
empty1 = n    # 缓冲池1空位数
full1 = 0     # 缓冲池1数据数
mutex1 = 1    # 缓冲池1互斥锁
empty2 = m    # 缓冲池2空位数
full2 = 0     # 缓冲池2数据数
mutex2 = 1    # 缓冲池2互斥锁
```

```
# 进程A：生产者，向缓冲池1写入数据
def process_A():
    while True:
        data = 生产数据()
        P(empty1)        # 申请缓冲池1空位
        P(mutex1)        # 互斥访问缓冲池1
        将data放入缓冲池1
        V(mutex1)        # 释放缓冲池1互斥锁
        V(full1)         # 缓冲池1数据数+1
```

```
# 进程C：消费者，从缓冲池2读取数据
def process_C():
    while True:
        P(full2)         # 申请缓冲池2数据
        P(mutex2)        # 互斥访问缓冲池2
        data = 从缓冲池2取出数据
        V(mutex2)        # 释放缓冲池2互斥锁
        V(empty2)        # 缓冲池2空位+1
        消费数据(data)
```

```
# 进程B：消费者（从缓冲池1读）+ 生产者（向缓冲池2写）
def process_B():
    while True:
        P(full1)         # 申请缓冲池1数据
        P(mutex1)        # 互斥访问缓冲池1
        data = 从缓冲池1取出数据
        V(mutex1)        # 释放缓冲池1互斥锁
        V(empty1)        # 缓冲池1空位+1

        P(empty2)        # 申请缓冲池2空位
        P(mutex2)        # 互斥访问缓冲池2
        将data放入缓冲池2
        V(mutex2)        # 释放缓冲池2互斥锁
        V(full2)         # 缓冲池2数据数+1
```

内存管理复习

1. 静态重定位：在作业装入过程中完成

- **定义**：指在程序装入内存时，由装入程序一次性将程序中的逻辑地址转换为物理地址，且转换后地址在程序运行期间不再改变。

2. 动态重定位：在作业执行过程中完成

- **定义**：指程序在内存中执行时，由硬件地址转换机制（如 MMU，内存管理单元）实时将逻辑地址转换为物理地址。

1. 一般情况下，一个计算机系统中，虚拟存储器的最大容量是由 **CPU 的地址结构（可寻址范围）**确定的，其实际容量是由**内存和硬盘交换区容量之和**确定的 。

2. 在请求调页系统中，内存分配有两种策略（常驻集）：**固定分配**和**可变分配**，固定分配的

缺点是可能导致频繁地出现缺页中断而造成 cpu 利用率下降。

3. 在段页式存储管理中，在段页式存储管理中，用**分段**方法来管理逻辑存储空间，用**分页**方法来管理物理存储空间。

4. 简单分页和分段存储管理方式过程，以及使用的数据结构是什么样子的？

分页：分页存储管理将进程的**逻辑地址空间分成若干个页**，并为各页加以编号，从 0 开始。在为进程分配内存时，以块为单位，将进程中的若干个页分别装入到多个可以不相邻接的物理块中。其中页的大小必须相同。

数据结构：

进程页表：系统为每个进程单独建立的页表，页表给出逻辑页号和具体内存块号相应的关系。页表放在**内存**，属于进程的现场信息。

分段：以段为单位进行分配，每一个段在内存中占连续空间，但各段之间能够不相邻。**按照程序自身的逻辑关系划分为若干个段**，每一个段都有一个段名（在低级语言中，程序员使用段名来编程），每段从 0 开始编址。其中段的大小可以不同。

数据结构：

进程段表：记录了段号、段首地址、段长度。每个进程都有一个段表，放在**内存**，属于进程的现场信息。

5. 说明请求分页系统中的缺页中断过程？

首先中断进程，保护该进程的现场，然后**请求页面调度**。

检查这个进程的内部表，以确定该引用时有效还是无效的访问

如果引用无效，则终止进程。如果引用有效但尚未调入的页面，那么就开始准备从磁盘中调页。

通过某种页面置换算法(FIFO、LRU 等)，来找出应当调出的页面。

在通过某种调入策略(请求调页或预调页)来将页面调入内存中。

当磁盘读取完成时，修改进程的内部表和页表，以表示该页现在处于内存中。

重新启动被中断的进程。

6. 某系统采用页式存储管理方法，其逻辑空间 32 页，每页是 2k，拥有物理空间为 1M。

（1）写出逻辑地址的格式

（2）如果不考虑访问权限等，进程的页表有多少项？每项至少有多少位？

（3）如果物理地址空间减少一半，页表结构应如何改变?

(1) 逻辑地址格式

- **页号位数**：逻辑空间共 32 页，$32 = 2^5$，因此页号占 **5 位**。
- **页内偏移位数**：每页大小 $2\text{KB} = 2^{11}$ 字节，因此页内偏移占 **11 位**。
- **逻辑地址总位数**：$5 + 11 = 16$ 位，格式为 **[5位页号][11位页内偏移]**。

(2) 页表结构

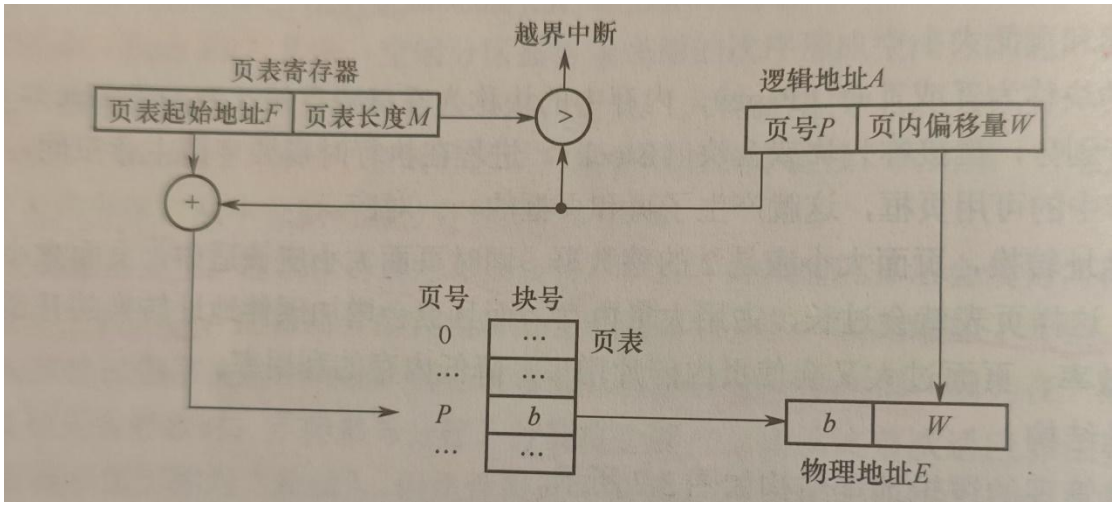
- **页表项数量**：逻辑空间有 32 页，因此页表最多需要 **32 项**（每个逻辑页对应一项）。
- **页表项位数**：
  - 物理空间 $1\text{MB} = 2^{20}$ 字节，每页 $2\text{KB} = 2^{11}$ 字节，则物理块数为 $2^{20}/2^{11} = 2^9$ 块。
  - 物理块号需 **9 位**（$2^9 = 512$ 块），因此页表项至少 **9 位**（存储物理块号）。

(3) 物理空间减半后的页表结构变化

- **物理空间调整**：物理空间减少一半至 $0.5\text{MB} = 2^{19}$ 字节，物理块数变为 $2^{19}/2^{11} = 2^8$ 块。
- **页表项数量**：逻辑页数量不变（仍为 32 页），因此页表项仍为 **32 项**。
- **页表项位数**：物理块号需 **8 位**（$2^8 = 256$ 块），因此页表项位数从 **9 位**减少至 **8 位**。

逻辑地址：页号+页内偏移

页表项：逻辑页与物理块的映射



3．在一个请求分页存储管理系统中，一个进程的页面走向为 4、3、2、1、4、3、5、4、3、2、1、5，当分配给该进程的物理页面分别为 3、4 时，采用下述页面淘汰算法时的缺页次数（假设开始执行时主存中没有页面），并比较所得结果。

(1)最佳置换法(OPT)　　(2)先进先出法(FIFO)

物理页面为 3

（1）

4	3	2	1	4	3	5	4	3	2	1	5
4	3	2	1			5			2	1	
	4	3	3			3			5	5	
		4	4			4			4	4	

共缺页 7 次

（2）

4	3	2	1	4	3	5	4	3	2	1	5
4	3	2	1	4	3	5			2	1	
	4	3	2	1	4	3			5	2	
		4	3	2	1	4			3	5	

共缺页 9 次

物理页面为 4

(1)OPT

4	3	2	1	4	3	5	4	3	2	1	5
4	3	2	1			5				1	
	4	3	2			2				5	
		4	3			3				3	
			4			4				4	

共中断 6 次

(2)FIFO

4	3	2	1	4	3	5	4	3	2	1	5
4	3	2	1			5	4	3	2	1	5
	4	3	2			1	5	4	3	2	1
		4	3			2	1	5	4	3	2
			4			3	2	1	5	4	3

共中断 10 次

文件系统复习

1. 文件系统最基本的目标是**按名存取**，它主要是通过**目录管理**功能实现的，文件系统所追求的最重要的目标是**提高对文件的存取速度**
2. 对文件空闲空间管理，MS-DOS 采用的是**位示图**，UNIX 采用的是**成组链接法**
3. 在树型目录结构中，用户对某个文件的首次访问通常都采用**文件路径名**，文件被打开后，对文件的访问通常采用**用户文件的描述符**

首次访问文件：采用文件路径名（2）

原因：

首次访问文件时，文件尚未被打开，系统需要通过完整的**路径名**（如/home/user/documents/file.txt）来定位文件在目录树中的位置。路径名从根目录或用户当前目录出发，逐层解析目录项，最终找到目标文件的索引结点（Inode）。

文件打开后：采用用户文件的描述符（4）

原因：

文件被打开后，系统会为其分配一个**文件描述符**（File Descriptor，如整数 1、2、3 等），作为该文件在当前进程中的唯一标识。此时，进程无需再通过路径名解析，直接使用文件描述符快速访问文件内容。

4. 在 create（）过程中，如果没有检索到指定文件的索引结点，此时属于**创建文件**，检索到指定文件的索引结点，此时若允许写，则此时属于**重写文件**，否则属于**出错**。
5. 什么是索引文件？说明混合索引分配方式？

索引文件是指当记录为可变长度时，通常为之**建立一张索引表**，并为每个记录设置一个表项构成的文件。通常将索引非顺序文件简称为索引文件。

混合索引分配方式是指将多种索引分配方式结合而成的分配方式。常见的是采用直接地址和一级索引联合的分配方式，或两级索引分配方式，甚至三级索引分配方式。如将每个文件索引表分为 13 个索引项，每项 2 个字节。最前面 10 项直接登记存放文件信息的物理块号（直接寻址）。如果文件大于 10 块，则利用第 11 项指向一个物理块，该块中最多可放 256 个文件物理块的块号(一次间接寻址)。对于更大的文件还可利用第 12 和第 13 项作为二次和三次间接寻址。

1. 假定盘块大小为 1KB，硬盘大小为 500MB，采用 FAT 显式链接分配方式时，其 FAT 表需占用多少存储空间？

答：由题目可可知道，该硬盘共有 500K 个盘块，则 FAT 表共有 500K 项；要表示这 500K 个盘块，每个 FAT 表项至少需要 19 位。通常在实际中，FAT 表项的长度取作半个字节的整数倍，所以这里可以取每个 FAT 表项为 20 位。这样，FAT 表需要的存储空间为：2.5 字节×500K=1250KB.

3. 某个文件系统中，每个盘块为 512 字节，文件控制块占 64 字节，其中文件名占 8 个字节。如果索引节点编号占 2 个字节，对一个存放在磁盘上的 256 个目录项的目录，试比较引入索引节点前后，为找到其中一个文件的 FCB，平均启动磁盘的次数。

答：

（1）引入索引前：

每块存放的 FCB 个数为 512/64=8，故共需 256/8=32 个盘块，

平均启动磁盘的次数为(1+32)/2 = 16.5 次

（2）引入索引后：

每块存放的符号目录项个数为 512/(8+2)=51，故共需 256/51=6 块

平均启动磁盘的次数为(1+6)/2+1 =4.5 次