第 1 章　操作系统概论

1　早期操作系统设计的主要目标是什么?

- 方便性：方便用户使用计算机。用户通过操作系统来使用计算机。
- 有效性：使计算机系统能高效可靠地运转，提高系统资源的利用率。
- 便于操作系统的设计、实现和维护。

2　操作系统是资源管理程序，它管理系统中的什么资源?

处理机管理、存储器管理、设备管理、文件管理

3　为什么要引入多道程序系统?它有什么特点?

- 引入多道程序设计技术的根本目的是提高 CPU 的利用率，充分发挥系统设备的并行性。这包括程序之间、CPU 与设备之间、设备与设备之间的并行操作。
- 特点：主存有多道，宏观上并行，微观上串行

4　叙述操作系统的基本功能。

- 处理机管理：解决处理机如何调度的问题：FCFS、优先级、时间片轮转
- 存储器管理：存储分配、存储保护、主存扩充。
- 设备管理：分配设备，控制设备传输数据。
- 文件管理：将程序、数据、操作系统软件等组织成文件，存放在磁盘或磁带上，方便用户访问。

5　批处理系统、分时系统和实时系统各有什么特点?各适合应用于哪些方面?

- 批处理系统：适合计算量大、自动化程度高的成熟作业
  - ✧ 优点: 系统吞吐量大，资源利用率高;
  - ✧ 缺点: 用户与作业无法交互，作业平均周转时间较长。
- 分时系统：适用于短小作业

- ✧ 同时性：若干用户同时使用一台计算机。
- ✧ 独立性：每个用户占有一台终端，独立操作，感觉不到别的用户存在。
- ✧ 交互性：用户可通过终端与系统进行人机对话。
- ✧ 及时性：用户的请求能在较短时间内得到响应。

◆ 实时系统：适用于需要计算机能对随机发生的外部事件做出及时的响应和处理的作业

- ✧ 实时性。其响应时间由被控制对象所能承受的延迟来确定。
- ✧ 可靠性。要具有容错能力，可采用双工机制：一台主机；一台后备机。
- ✧ 确定性。是指系统按照固定的、预先确定的时间执行指定的操作。其可确定性取决于系统响应中断的速度和处理能力。

6 操作系统的特性?

◆ 并发性：并发是指系统中存在着若干个逻辑上相互独立的程序，它们都已被启动执行，都还没有执行完，并竞争系统资源。

◆ 共享性：是指系统中的资源可供系统中多个并发执行的进程共同使用。如打印机、磁带机、磁盘等。支持系统并发性的物质基础是资源共享

◆ 虚拟性：把共享资源的一个物理实体变为若干个逻辑上的对应物。如，CPU 的分时共享；虚拟存储器技术。

◆ 异步性（随机性）：有限的资源共享使并发进程之间产生相互制约关系。各个进程何时执行、何时暂停、以怎样的速度向前推进、什么时候完成等都是不可预知的。

7 衡量 OS 的性能指标有哪些？什么是吞吐量、响应时间和周转时间?

◆ 响应时间：从用户发出请求到系统开始响应所需的时间

◆ 资源利用率：指在给定时间内，系统中某一资源（如 CPU、存储器、外部设备等）实际使用时间所占比率。

- 吞吐量(Throughput)：指单位时间内系统所处理的信息量。它通常是用每小时或每天所处理的作业个数来度量。
- 周转时间：指从作业进入系统到作业退出系统所用的时间。而平均周转时间是指系统运行的几个作业周转时间的平均值。

8 什么是嵌入式系统?

以实际应用为中心、以计算机技术为基础、软硬件可裁剪的专用计算机系统。

9 什么是对称多处理？它有什么好处?

- 对称多处理(SMP)：操作系统和用户程序可安排在任何一个处理机上运行，各处理机共享主存和各种 I/O 设备。
- 增加了系统的吞吐率：多个作业可以分配在任何一个处理机上执行。
- 增加了系统的可靠性。一个处理机的失效，这是性能的降低，不会影响整个系统。

10 为了实现系统保护，CPU 通常有哪两种工作状态？各种状态下分别执行什么程序？什么时候发生状态转换？状态转换由谁实现的?

- CPU 通常有核心态、用户态两种工作状态。
- 核心态执行操作系统内核程序，用户态执行用户自编程序和系统外层的应用程序。
- 通过系统调用可以实现状态的转换，具体是通过中断和异常来实现。
- 用户态到核心态的转换由**硬件**完成；核心态到用户态的转换由**内核程序**执行后完成。

11 什么是系统调用？什么是特权指令？特权指令执行时，CPU 处于哪种工作态?

- 系统调用就是内核向用户提供的，用来运行系统内核子程序的接口。

- 特权指令是指关系系统全局的指令，如存取和操作 CPU 状态，启动各种外部设备，设备时钟时间，关中断，清主存，修改存储器管理寄存器，改变用户方式到核心方式和停机指令等。
- 特权指令执行时，CPU 处于核心态。

12 操作系统通常向用户提供哪几种类型的接口?其主要作用是什么?

- 操作接口：命令语言、窗口界面
- 编程接口：系统调用

第 2-3 章 进程管理

1 程序顺序执行的特点

- 封闭性：程序在运行时独占全机资源，因此，这些资源的状态只能由该程序决定和改变，不受外界因素影响。
- 可再现性：只要初始条件相同，无论程序连续运行，还是断断续续地运行，程序的执行结果不变。

2 何谓进程，进程由哪些部分组成？试述进程的四大特性（动态性、独立性、并发性、结构性）及进程和程序的区别。

- 进程是可以和其他程序并行执行的程序关于某个数据集合的一次执行过程。
- 组成：程序、数据、进程控制块。其包含的程序可以有一个或多个；数据可能隐含包括一两个堆栈，主要用来保存函数调用和系统调用时要传递的参数、返回地址和一些临时变量等；进程控制块是操作系统对进程进行管理控制使用的一个结构，用来记录进程的属性信息。
- 四大特性：

- ✧ 动态性。进程是程序的一次执行过程，是临时的，有生命期的。
- ✧ 独立性。进程是系统进行资源分配和调度的一个独立单位。
- ✧ 并发性。多个进程可在处理机上交替执行。
- ✧ 结构性。系统为每个进程建立一个进程控制块。

◆ 进程与程序的区别：

- ✧ 进程是动态的，程序是静态的。程序是有序代码的集合，进程是程序的执行，没有程序就没有进程。通常，进程不可以在计算机之间迁移，而程序可以复制。
- ✧ 进程是暂时的，程序是永久的。
- ✧ 进程包括程序、数据和进程控制块。
- ✧ 通过多次执行，一个程序可对应多个进程；通过调用关系，一个进程可包括多个程序。进程可创建其他进程，而程序不能形成新的程序。

3 进程控制块的作用是什么?它主要包括哪几部分内容?

◆ 作用：进程存在的唯一标识

◆ 组成：

- ◆ 进程标识数：用于唯一地标识一个进程，通常是一个整数。
- ◆ 外部标识符，由用户使用。如：send 进程、print 进程等。
- ◆ 进程的状态、调度、存储器管理信息：是调度进程所必需的信息，包括进程状态、优先级、程序在主存地址、在外存的地址等。
- ◆ 进程使用的资源信息：分配给进程的 I/O 设备、正在打开的文件等。
- ◆ CPU 现场保护区：保存进程运行的现场信息。包括：程序计数器(PC)、程序状态字、通用寄存器、堆栈指针等。
- ◆ 记帐信息：包括使用 CPU 时间量、帐号等。

- 进程之间的家族关系：类 UNIX 系统，进程之间存在着家族关系，父/子进程。
- 进程的链接指针：链接相同状态的进程。

4 进程的基本状态，试举出使进程状态发生变化的事件并描绘它的状态转换图。

- 基本状态：
  - 运行态(running)：进程正在 CPU 上运行。单 CPU 系统一次只有一个运行进程；多 CPU 系统可能有多个运行进程。
  - 阻塞态(blocked)：又称等待态。当进程因等待某个条件发生而不能运行时所处的状态。等待 I/O 完成，等待一个消息
  - 就绪态(ready)：已获得除 CPU 之外的全部资源，只要再获得 CPU，就可执行。
- 状态转换图

5 什么是原语？什么是进程控制？

- 原语是不能被中断的。
- 进程控制是指系统使用一些具有特定功能的程序段来创建、撤消进程，以及完成进程各状态之间的转换。

6 进程调度的功能、方式、时机、算法。作业调度，交换调度。作业的周转时间和作业的带权周转时间？

- 进程调度的功能：
  - 记录系统中各进程的执行状况
  - 管理进程控制块，将进程的状态变化及资源需求情况及时地记录到 PCB 中。
  - 选择就绪进程占有 CPU
  - 进行进程上下文的切换

- ✧ 将正在执行进程的上下文保存在该进程的 PCB 中，将刚选中进程的运行现场恢复起来，以便执行。

◆ 进程调度的方式：

- ✧ 非抢先方式(非剥夺方式)：某一进程占用 CPU,直到运行完或不能运行为止，其间不被剥夺。用在批处理系统。主要优点：简单、系统开销小。
- ✧ 抢先方式(剥夺方式)：允许调度程序基于某种策略（优先级、时间片等）剥夺现行进程的 CPU 给其它进程。用在分时系统、实时系统。

◆ 进程调度的时机：

- ✧ 现行进程完成或错误终止；
- ✧ 提出 I/O 请求，等待 I/O 完成时；
- ✧ 在分时系统，按照时间片轮转，分给进程的时间片用完时；
- ✧ 优先级调度，有更高优先级进程就绪；
- ✧ 进程执行了某种原语操作，如阻塞原语和唤醒原语，都可能引起进程调度。

◆ 进程调度的算法：

- ✧ 先来先服务(FCFS)
  - ■ 简单，节省机器时间
  - ■ 平均周转时间延长
- ✧ 最短作业优先(SJF)
  - ■ 对短作业有利，作业的平均周转时间最佳
  - ■ 长作业没有机会运行，出现饥饿现象
- ✧ 响应比高者优先(HRN)
  - ■ 响应比 = （作业等待时间+作业估计运行时间）/作业估计运行时间

        - 有限运行短作业和等待时间足够长的长作业
        - 算法比较复杂
    - 优先级调度法
    - 轮转法(Round Robin)
    - 多级反馈队列轮转法
- 作业调度：高级调度。多道批处理系统。多个用户作业以成批的形式提交到外存，形成后备作业队列。被作业调度选中进内存，就处于运行态。
- 交换调度：中级调度。将主存就绪或主存阻塞等暂不具备运行条件的进程换出到外存交换区；或将外存交换区中的已具备运行条件的进程换入主存。交换调度可以控制进程对主存的使用。

7 线程的定义，线程与进程的比较。系统对线程的支持。（用户级线程、核心级线程、两级组合）

- 线程：是进程内的一个可执行实体，是处理机调度的基本单位。
- 线程与进程的比较：
    - 拥有的资源
        - 进程拥有一个独立的地址空间，用来存放若干代码段和数据段。若干打开文件，以及至少一个线程。
        - 一个进程内的多线程共享该进程的所有资源，线程自己拥有很少资源。
    - 调度
        - 进程调度需进行进程上下文的切换，开销大。
        - 同一进程内的线程切换，仅把线程拥有的一小部分资源变换了即可，效率高。同一进程内的线程切换比进程切换快得多。

- 并发性
  - 引入线程后，使得系统的并发执行程度更高。 进程之间、进程内的多线程之间可并发执行。
- 安全性
  - 同一进程的多线程共享进程的所有资源，一个线程可以改变另一个线程的数据，而多进程实现则不会产生此问题。共享方便。

◆ 系统对线程的支持：

- 用户级线程
  - 有关线程的所有管理工作都由用户进程通过调用用户态运行的线程库完成。自己设计线程调度算法。
  - 内核以进程为单位进行调度。一个线程阻塞，其依附的进程也阻塞。
  - 多线程对应核心级一个进程。
- 核心级线程
  - 有关线程的管理工作都由内核完成。应用程序通过系统调用来创建或撤销线程。
  - 一个线程的阻塞，不影响其他线程的执行。
  - Windows、Linux、多处理机系统
- 两级组合
  - 既支持用户级线程，也支持核心级线程。
  - 用户级多个线程对应核心级多个线程。
  - 当内核了解到一个线程阻塞后，通知运行时系统，重新调度其他线程。

8 并发执行的进程在系统中通常表现为几种关系?各是在什么情况下发生的?

◆ 互斥关系

资源共享。间接制约。

- 同步关系

  合作完成任务。直接制约。

- 前序关系

  由于进程之间的互斥同步关系，使得进程之间具有了前序关系，这些关系决定了各个进程创建和终止的时间。

9 什么叫临界资源?什么叫临界区?对临界区的使用应符合的四个准则（互斥使用、让权等待、有空让进、有限等待)。

- 临界资源：就是一次仅允许一个进程使用的系统中共享资源。
- 临界区(critical section)：就是并发进程访问临界资源的那段必须互斥执行的程序。
- 四个准则：
  - 互斥使用：不能同时有两个进程在临界区内执行
  - 让权等待：等待进入临界区的进程，应释放处理机后阻塞等待
  - 有空让进：在临界区外运行的进程不可阻止其他进程进入临界区
  - 有限等待：不应使要进入临界区的进程无限期等待在临界区之外

10 解决进程之间互斥的办法：开、关中断，加锁、开锁（又叫测试与设置，通常由一条机器指令完成)，软件方法，信号量与 P、V 操作。

- 软件实现方法
  - 共享变量
  - 信号量与 P、V 操作：克服了加锁操作的忙等待现象，提高了系统的效率。
- 硬件实现方法
  - 开关中断：最简单的方法。在进程刚进入临界区后，立即禁止所有中断；在进程要离

开之前再打开中断。因为 CPU 只有在发生时钟中断或其它中断时才会进行进程切换。简单，但限制了处理机交叉执行程序的能力

- ✧ 加锁、开锁：test&set，通常由一条机器指令完成

11 若信号量 S 表示某一类资源，则对 S 执行 P、V 操作的直观含意是什么？当进程对信号量 S 执行 P、V 操作时，S 的值发生变化，当 S>0、S=0、和 S<0 时，其物理意义是什么?

- ◆ 调用 P(mutex)申请临界资源。调用 V(mutex)释放临界资源。
- ◆ S>0 时表示有 S 个资源可用；

  S=0 时表示无立即可用资源；

  S<0 时表示前面有|S|个进程正在等待使用资源。

12 在用 P/V 操作实现进程通信时，应根据什么原则对信号量赋初值?

- ◆ 设置一个互斥信号量 mutex，初值为 1，表示该临界资源空闲。信号量的取值范围：+1～ -(n-1)。
  - ✧ 调用 P(mutex)申请临界资源。执行 P 操作时，将信号量 s.value-1，若 s.value>=0，则执行 P 操作的进程继续执行；若 s.value<0，则执行 P 操作的进程变为阻塞状态，并排到与该信号量有关的 list 所指队列中等待。
  - ✧ 调用 V(mutex)释放临界资源。执行 V 操作时，将信号量 s.value+1，若 s.value<=0，则执行 V 操作的进程从与该信号量有关的 list 所指队列中释放一个进程，使它由阻塞变为就绪状态，之后执行 V 操作进程继续执行；若 s.value>0，则执行 P 操作的进程继续前进。

13 经典的 IPC 问题。

- ◆ **生产者和消费者问题：**

①桌上有一空盘，一次只允许放一只水果。爸爸可向盘中放苹果，也可向盘中放桔子，儿子专等吃盘中的桔子，女儿专等吃盘中的苹果。请用 P、V 操作实现爸爸、儿子、女儿三个并发进程的同步。初始状态：同步信号量 s1=1，表示盘子为空。放苹果发同步信号 s2，放桔子发同步信号 s3。

Father：
while(1) {
p(s1);
if(放入的是苹果)v(s2);
else v(s3);
}
	Daughter：
	while(1) {
	p(s2);
	从盘中取出苹果；
	v(s1);
	}
		Son：
		while(1) {
		p(s3);
		从盘中取出桔子；
		v(s1);
		}

②看病。

Labora：
while(化验未完成) {
p(s1);有无化验单，无则等
v(s2);送出结果
}
	Diagnosis：
	while(看病未完成) {
	看病;
	V(s1);送单
	p(s2);等结果
	}

◆ **读者和写者问题：**

写互斥信号量 wmutex：实现读写互斥和写写互斥地访问共享文件，初值为 1。

计数器 readcount：记录同时读的读者数，初值为 0。

读互斥信号量 rmutex：使读者互斥地访问共享变量 readcount，初值为 1。

int rmutex=1,wmutex=1,readcount=0;
Reader:
P(rmutex); //互斥访问 readcount
if readcount=0 then P(wmutex);
readcount++;
V(rmutex);
读文件;
P(rmutex);
readcount= readcount-1;
if readcount=0 then V(wmutex);
V(rmutex);
	Writer: …
	P(wmutex);
	写文件;
	V(wmutex);
	…

◆ **理发师问题:**

有一个理发师、一把理发椅和 n 把供等候理发的顾客坐的椅子。如果没有顾客，则理发师坐在椅子上睡觉，当有一个顾客到来时，必须唤醒理发师，请求理发；如果理发师正在理发，又有顾客到来时，只要有空椅子，他就坐下来等待，如果没有空椅子，他就离开。请为理发师和顾客各编写一段程序来描述他们的同步问题。

设两个信号量：(1)用 s1 制约理发师，初值为 0，表示有 0 个顾客；(2)用 s2 制约顾客，表示可用椅子数，初值为 n。

顾客: P(s2);申请椅子 V(s1);发信号 坐椅子等理发;	理发师: P(s1);查是否有顾客 给一名顾客理发; V(s2);让顾客离开

◆ **哲学家进餐问题**:

```
int fork[0]=fork[1]=…=fork[4]=1;
第 i 个哲学家所执行的程序:
do{
        P(mutex);
        P(fork[i]);
        P(fork[(i+1)mod5]);
        V(mutex);
         吃饭
        V(fork[i]);
        V(fork[(i+1)mod5]);
    } while(1);
```

◆ 3-17 读者优先 int rmutex=1, rwmutex=1, readcount=0

Reader:	Writer:

Begin:
P(z);读写进程在 z 上排队
P(rmutex);
If(readcount==0)
P(rwmutex);第一个读者申请读文件的使用权
End if
++readcount;
	Begin
	P(z);读写进程在此排队
	P(rwmutex);申请写文件
	Write data;
	V(rwmutex);释放文件的使用权
	V(z);
	…
	End

V(rmutex);释放对读者计数的使用权
V(z);无写者时，多个读者可同时
read data;
P(rmutex);读完退出时，申请对读者计数减 1
--readcount;
If(readcount==0)
V(rwmutex);最后一个读者释放文件使用权
End if
V(rmutex);释放对读者计数的使用权
…
end	

◆ 写者优先

Reader:来一个写进程通过 p(x)禁止后续进程读
While(1){
P(z);其他读进程在 z 上排队
P(x);一个读进程与一个写进程为 X 竞争
P(rmutex)
++readcount;
If(readcount==1)
	Writer:
	While(1){
	P(wmutex);写进程互斥
	++writecount;
	If(writecount==1)
	P(x);
	V(wmutex);
	P(rwmutex);其他进程在 rwmutex 上排

P(rwmutex)
V(rmutex);
V(x)
V(z);
read data;
P(rmutex);
--readcount;
If(readcount==0)
V(rwmutex);
V(rmutex);
}
	队
	Write data;
	V(rwmutex);
	P(wmutex);
	--writeclunt;
	If(writecount==0)
	V(x);
	V(wmutex);
	}

14 进程高级通信有哪些实现机制?

- 消息缓冲通信
  - 消息缓冲池
  - 信箱
- 管道通信
- 共享存储区

15 产生死锁的原因

- 系统资源配置不足
- 系统的各并发进程请求资源的随机性
- 各进程在系统中异步向前推进，造成进程推进顺序的非法性

16 死锁产生的必要条件及解决死锁的方法

◆ 必要条件：

- 互斥条件。独占性的资源。
- 保持和等待条件。进程因请求资源而阻塞时，对已经获得的资源保持不放。
- 不剥夺条件。已分配给进程的资源不能被剥夺，只能由进程自己释放。
- 循环等待条件。存在一个进程循环链，链中每个进程都在等待链中的下一个进程所占用的资源。

◆ 解决方法：

- 鸵鸟算法。忽略死锁。
- 死锁的预防。通过破坏产生死锁的四个必要条件中的一个或几个，来防止发生死锁。
- 死锁的避免。是在资源的动态分配过程中，用某种方法去防止系统进入不安全状态，从而避免发生死锁。
- 死锁的检测和恢复。允许死锁发生，通过设置检测机构，及时检测出死锁的发生，然后采取适当措施清除死锁。

◆ 产生死锁的根本原因：是对独占资源的共享，并发执行进程的同步关系不当。

17 理解银行家算法的实质。能够利用银行家算法避免死锁。

第 4 章 存储器管理

1 存储器管理的功能。名字空间、地址空间、存储空间、逻辑地址、物理地址。

◆ 存储器管理的功能：

- ✧ 存储器分配：解决多道程序或多进程共享主存的问题。
- ✧ 地址转换或重定位：研究各种地址变换方法及相应的地址变换机构。

- ✧ 存储器保护：防止故障程序破坏 OS 和其它信息。
- ✧ 存储器扩充：采用多级存储技术实现虚拟存储器及所用的各种管理算法。
- ✧ 存储器共享：并发执行的进程如何共享主存中的程序和数据。

◆ 名字空间：源程序中的各种符号名的集合所限定的空间。

◆ 逻辑地址空间：程序限定的空间。

◆ 物理地址空间：物理存储区中全部物理单元的集合所限定的空间。

◆ 存储空间：由字或字节组成的一个大的阵列，每一个字或字节都有它自己的编号地址。一个程序只有从地址空间装入到存储空间后才能运行。

2 什么是地址重定位?分为哪两种?各是依据什么和什么时候实现的?试比较它们的优缺点。

◆ 地址重定位：把程序地址空间的逻辑地址转换为存储空间的物理地址。分为静态重定位、动态重定位。

- ✧ 静态重定位：在进程执行前，由装入程序把用户程序中的指令和数据的逻辑地址全部转换成存储空间的物理地址。只需经重定位直接装入程序即可使用。
  - ■ 特点：
    - ● 无硬件变换机构；
    - ● 为每个程序分配一个连续的存储区；
    - ● 在程序执行期间不能移动，主存利用率低；
    - ● 难以做到程序和数据的共享；
    - ● 用于单道批处理系统。
- ✧ 动态重定位：装入程序把程序和数据原样装入到已分配的存储区中。程序运行时，把该存储区的起始地址送入重定位寄存器。需硬件地址转换机构（即重定位寄存器）。

- 特点：
  - 主存利用充分。
  - 可移动用户程序。移动后，只需修改重定位寄存器。
  - 程序不必占有连续的存储空间。
  - 便于多用户共享存储器中的同一程序和数据。
  - 用于多道批处理系统、分时系统。
  - 需额外的硬件支持，增加了系统成本。

3 内存划分为两大部分：用户空间和操作系统空间。存储器管理是针对用户空间进行管理的。

4 存储保护的目的是什么?对各种存储管理方案实现存储保护时，硬件和软件各需做什么工作?

- 目的：
  - 防止地址越界：进程运行时产生的所有存储器访问地址都要进行检查，确保只访问为该进程分配的存储区域。
  - 正确地进行存取：对所访问的存储空间的操作方式（读、写、执行）进行检查，以防止由于误操作，使其数据的完整性受到破坏。
- 工作：
  - 静态：上、下界寄存器
  - 动态：基址+限长寄存器
  - MMU

5 试述可变式分区管理空闲区的方法及存储区的保护方式。覆盖与交换有什么特点?

- 管理可变式分区的数据结构：分区说明表、空闲区链表。
- 分配算法：
  - ✧ 首次适应法(first fit)
  - ✧ 最佳适应法(best fit)
  - ✧ 最坏适应法(worst fit)
- 固定式分区：设置上、下界寄存器来实现存储器保护

  可变式分区：设置基址+限长寄存器来实现存储器保护
- 覆盖特点：打破了必须将一个进程的全部信息装入主存后才能运行的限制。在逻辑上扩充了主存。小主存可运行大进程。
- 交换特点：打破了一个程序一旦进入主存，便一直运行到结束的限制。

6 页表的作用是什么?简述页式管理的地址变换过程。能利用页表实现逻辑地址转换成物理地址。管理内存的数据结构有哪些?

- 页表的作用：记录逻辑页与主存块的映射关系。
- 页式管理的地址变换过程：

①把该进程的页表始址和页表长度放入 CPU 的控制寄存器中

②将程序计数器内容的页号部分与控制寄存器中的页表长度相比较，若页号 p 小于页表长度时转③，否则产生地址越界，终止程序运行。

③将程序计数器中的页号与控制寄存器的页表始址相加，得到该访问操作所在页号在页表中的入口地址。这里的加是根据页表项占用的字节数决定的。

④用该地址去访问页表，获得该页对应的主存块号

⑤把主存块号与程序计数器中的页内地址相拼接，从而得到该操作对应主存的物理地址

- 记录存储空间使用情况的两种数据结构：

- ✧ 存储分块表
- ✧ 位示图

7 什么是页式存储器的内零头?它与页的大小有什么关系?可变式分区管理产生什么样的零头（碎片）？

- ◆ 零头（碎片）：在分区存储管理中，每道程序要求占用主存的一个连续的存储区域，因而会产生许多碎片。
- ◆ 页尺寸大：页的内部碎片大（进程的代码段、数据段、堆栈段的最后一页都有内部碎片）
- ◆ 可变式分区：

8 段式存储器管理与页式管理的主要区别是什么?

- ◆ 段是由用户划分的；页是为了方便管理由硬件划分的，对用户是透明的。
- ◆ 页的大小固定；段的大小不固定。
- ◆ 段式用二维地址空间；页式用一维地址空间。
- ◆ 段允许动态扩充，便于存储保护和信息共享。
- ◆ 段可能产生主存碎片；页消除了碎片。
- ◆ 段式管理便于实现动态链接，页式管理只能进行静态链接。
- ◆ 段与页一样，实现地址变换开销大，表格多。

9 什么是虚拟存储器。虚拟存储器的容量能大于主存容量加辅存容量之和吗?

- ◆ 虚拟存储器是系统为了满足应用对存储器容量的巨大需求而构造的一个非常大的地址空间。
- ◆ 虚拟器的容量可以大于主存加辅存容量。

10 实现请求页式管理，需要对页表进行修改，一般要增加状态位、修改位、访问位。试说明它们的作用。

- 有效位：用来指示某页是否在主存
- 修改位：指示该页调入主存后是否被修改过。“1”表示修改过，“0”表示未修改过。
- 访问位：指示该页最近是否被访问过，“1”表示最近访问过，“0”表示最近未访问

11 产生缺页中断时，系统应做哪些工作?

- 操作系统处理缺页中断，寻找一个空闲的页框。
- 若有空闲页，则把从磁盘读入信息装入该页框。
- 若无空闲页，则按某种算法选择一个已在内存的页面，暂时调出内存。
- 若修改过还要写磁盘。调入需要的页。之后要修改相应的页表和内存分配表。
- 恢复现场，重新执行被中断的指令。

12 会利用 FIFO、LRU、OPT 以及时钟页面置换算法描述页面置换过程，计算产生的缺页率。Belady 异常。

- 页面置换算法：
  - 最佳置换算法（OPT 算法）：

    选择以后不再访问的页或经很长时间之后才可能访问的页进行淘汰
  - 先进先出置换算法(FIFO)（可能出现抖动）：

    当淘汰一页时，选择在主存驻留时间最长的那一页。
  - 最近最少使用的页面置换算法(LRU)(不会出现 Belady 异常)：

    根据局部性原理，淘汰那些在最近一段时间里最少使用的一页。
  - 时钟页面置换算法
- Belady 异常：

  采用 FIFO 算法，当为进程分配的主存块多时，有时产生的缺页中断次数反而增多。

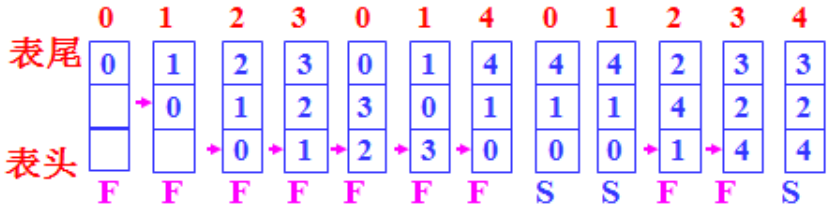
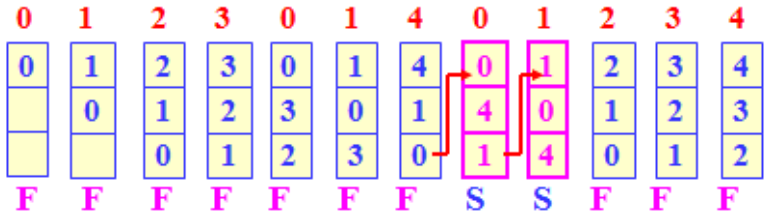


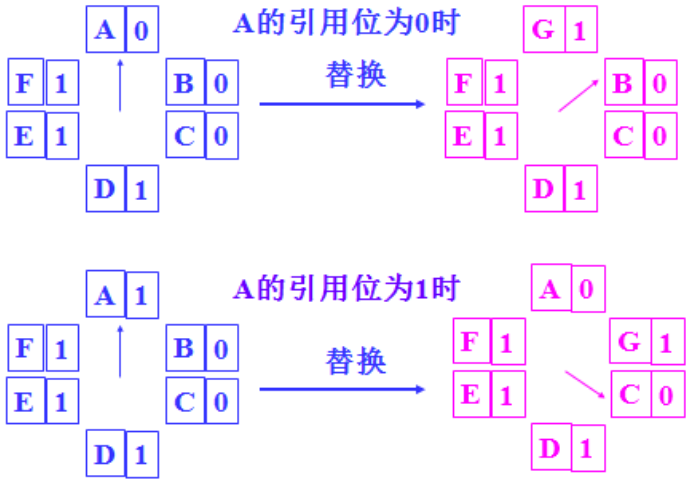
图4.21 FIFO（3个内存块时，缺页9次）

LRU 页面置换算法。根据局部性原理，淘汰那些在最近一段时间里最少使用的一页。



LRU（3个内存块时，缺页10次）

时钟页面置换算法：将进程所访问的页放在一个像时钟一样的循环链中。链中的节点数就是为之分配的主存块数。



138

13 什么是程序的局部性原理？什么叫系统抖动？工作集模型如何防止系统抖动？

◆ 程序的局部性原理：

✧ 时间局部性：程序中往往含有许多循环，在一段时间内会重复执行该部分。

✧ 空间局部性：程序中含有许多分支，在一次执行中，只有满足条件的代码运行，不满

足条件的代码不运行。即使顺序执行程序，程序的地址域在短时间内变化不大。在进程运行过程中，用到哪部分程序或数据再由系统自动装入。

- 抖动：刚被淘汰的页面马上又要用，因而又要把它调入。调入不久再被淘汰，淘汰不久再次装入。如此频繁地调入调出，降低了系统的处理效率。
- 进程的工作集：在一段时间内，进程集中在一组子程序或循环中执行，导致所有的存储器访问局限于进程地址空间的一个固定的子集。可以大大减少进程产生缺页中断的次数，甚至在一段时间内不产生缺页中断，从而大大提高了系统的运行效率。

14　多级页表的概念，多级页表中页表建立的时机。写时复制技术的概念。

- 多级页表的概念：页表在内存不必连续存放
- 多级页表中页表的建立时机：页表的建立不再是在进程装入主存时，而是推迟到要访问页时，才为包含该页的页表分配空间和建立页表页。
- 写时复制技术：若没有进程向共享主存页写时，两个进程就共享之。若有进程要写某页，系统就把此页复制到主存的另一个页框中，并更新该进程的页表，使之指向此复制的页框，且设置该页为可读/写。

15　页的共享问题。需要一个专门数据结构来记录进程间共享页。

- 页的共享可以避免在主存中同时有多个相同页的副本。节约主存。
- 通常，只读页（如程序文本）可以共享，可读/写的数据页不可共享。
- 需要一个专门的数据结构来记录共享页。把共享页锁在内存，且在页表中增加引用计数项，仅当其引用计数为 0 时，才允许调出或释放盘空间。

第 5 章　文件系统

1　什么是文件和文件系统？文件系统的主要功能。UNIX 系统如何对文件进行分

类？它有什么好处？

- 文件：是存储在外部存储器上的具有符号名的相关信息的集合。
- 文件系统：OS 中管理文件的软件机构。包括管理文件所需的数据结构、相应的管理软件和被管理的文件。
- 文件系统的主要功能：
  - 管理文件存储器。记录空间使用情况，分配空间，调整或回收空间。
  - 实现按名存取。利用目录结构快速定位文件。
  - 应具有灵活多样的文件结构和存取方法，便于用户存储和加工处理信息。
  - 提供一套使用方便、简单的操作命令。
  - 保证文件信息的安全性。
  - 便于文件的共享。
- UNIX 的文件分类：
  - 普通文件：通常的文件。
  - 目录文件：由文件目录构成的一类用来维护文件系统结构的文件。对其处理同普通文件。
  - 特别文件：输入设备和输出设备（字符型特别文件），输入/输出型设备（字符块特别文件），管道文件。
- 文件分类的目的：便于系统对不同类型的文件进行不同的管理，以实现文件的保护和共享。

2 文件目录的作用是什么？文件目录项通常包含哪些内容？文件控制块。

- 文件目录的作用：使用户实现按名存取文件，是文件系统的主要目的，实现的主要结构是文件目录。

- 文件目录是指记录文件的名字及其存放物理地址的一张映射表，表中包括了许多文件控制块。
- 文件控制块：包含了文件的说明信息和管理控制信息。

3 文件的逻辑结构有几种形式？文件的存取方法?

- 逻辑结构的形式：
  - 无结构的字节流式文件。由无结构的先后到达的相关字节组成，其文件长度就是所包含的字节个数。
  - 有结构的记录式文件。分为定长记录式文件和变长记录式文件。
- 文件的存取方法：
  - 顺序存取：按照文件信息的逻辑顺序依次存取。是在前一次存取的基础上进行的。
  - 直接存取（随机存取）：基于文件的磁盘模型，磁盘允许对任意文件块进行随机读和写。
  - 对记录式文件而言。根据记录的编号来直接存取文件中的任意一个记录。
  - 对字节流文件而言。根据系统调用命令把读/写指针调整到欲读/写位置上，然后读/写指定字节数的信息。

4 文件的物理结构有哪几种?对于不同的结构，文件系统是如何进行管理的?

- 连续文件（顺序文件）
  - 支持顺序存取和随机存取
  - 文件内容连续存放。
  - 优点：简单；支持顺序存取和随机存取；存取速度快。只要访问一次文件的管理信息，就可方便地存取到任一记录。
  - 缺点：不灵活；容易产生碎片。连续结构适合存储长度不变的文件系统。

- 链接文件
  - 只能按文件的指针链顺序存取
  - 不要求文件内容连续存放。把文件所占用的物理块用链接指针链接起来。
  - 优点：可以解决外存的碎片问题，提高了外存空间的利用率；允许文件动态增长。
  - 缺点：只能按文件的指针链顺序存取，查找效率较低。

为了克服链接结构文件的缺点，可以把指针字从文件的各物理块中取出，放在一个表中，并将此表叫盘文件映射表。MS-DOS 就使用这种方式分配和管理磁盘空间，并将该表叫做文件分配表。利用文件分配表，不但能方便地实现顺序存取，而且也很容易实现随机存取。但缺点是运行时整个表必须在主存，主存消耗大。

- 索引文件
  - 随机、顺序存取
  - 为每个文件建立一张索引表。用索引表记录文件内容的存放地址，即记录文件的逻辑块号和对应的物理块号之间的关系。
  - 实例：Linux/Unix
- 索引顺序文件
  - 实例：Windows

5 DOS 文件卷的结构，DOS 系统的文件物理结构是什么？书 110-111

- DOS 文件卷的结构：
  - 引导扇区
  - 文件分配表
  - 根目录区
  - 文件数据区

- DOS 系统的文件物理结构：链接文件

6 了解记录的组块和分解。

- 组块：把多个逻辑记录存放在一个物理块中的工作叫做记录的组块。
- 分解：从一个物理块中将一个逻辑记录分离出来的工作叫做记录的分解。

7 文件存储空间的管理方法有几种?它们各是如何实现文件存储空间的分配和回收的?

- 空白文件目录

  适合于文件的静态分配（连续文件的分配）

- 空闲块链表
  - 空闲块链：把所有空闲块连接成一个链表。
    - 优点：简单。适合文件动态分配。
    - 缺点：工作效率低。分配和回收多个盘块时要多次访问磁盘才能完成。
  - 空闲块成组链表（Unix 使用）
    - 利用盘空闲块来管理盘上的空闲块，每个磁盘块记录尽可能多的空闲块而成一组。各组之间也用链指针链接在一起。
    - 便于空闲块的分配与回收。
    - 适合连续文件、链接文件和索引文件的存储分配。
- 位映像表(bit map)或位示图

  每一个二进制位对应一个物理盘块。为 1 时表示块已分配，为 0 时空闲。

  优点：易找到一个或几个连续的空闲块。其尺寸固定，可以保存在主存，便于分配和回收。

8 建立多级目录有哪些好处?文件的重名和共享问题是如何得到解决的?

- 多级目录的好处：
  - 层次结构清晰，便于管理和保护；
  - 有利于文件分类；
  - 解决重名问题；
  - 提高文件检索速度；
  - 能够控制存取权限。
- 解决共享问题：
  - 符号链接文件：创建一个新目录项，其中存有指向另一个文件或目录的绝对路径名。
  - 硬链接：在共享目录项中简单地重复被共享文件的信息，因此两个目录项完全相同。

9 文件系统中，常用的文件操作命令有哪些？它们的具体功能是什么?打开和关闭文件命令的目的是什么?

- 创建(Create)文件：在指定设备上为指定路径名的文件建立一个目录项，并设置文件的有关属性。
- 删除(Delete)文件：根据文件的路径名找到指定的目录项，回收其占用的各个物理块，再将该目录项置为空。
- 打开(Open)文件：建立进程和文件之间的联系
  - 根据文件路径名找到目录项，进而找到 FCB。FCB 进内存，再创建文件对象。
  - 系统打开文件表是由文件对象组成的链表
  - 文件对象记录了文件读/写位置指针、文件方法、共享该文件的进程数等信息。
  - 将文件的目录项复制到主存一个专门区域，返回文件在该区域的索引。建立进程与文件的联系。
  - 目的：避免多次重复地检索文件目录。系统维护了一个系统当前打开文件表。当读/

写文件时，通过这个表的索引找到文件的主存目录项。不需要重复地对磁盘进行检索。

- 关闭(Close)文件：
  - ✧ 释放文件在主存中的目录项，切断用户与文件的联系。
  - ✧ 若该目录项被修改过，则写回辅存。
  - ✧ 若文件作过某些修改，则写回辅存。
- 读(Read)文件：命令中必须指出要读的数据个数，以及存放数据的主存地址。根据文件所在设备、文件类型的不同，系统设置不同的读命令。
- 写(Write)文件：命令中必须指出要写的数据个数，以及存放数据的主存地址，将主存中的数据写到指定的文件中。
- 追加(Append)文件：限制了写文件的形式，将数据追加到文件尾。
- 随机存取(Seek)文件：重新定位文件的读/写位置指针。
- 得到文件属性(Get Attributes)：进程在执行时常常需要了解文件的属性。在 UNIX 系统中，一个软件开发项目通常由多个源文件组成，make 程序用来管理这些软件开发项目。当 make 被调用时，它检查所有源文件和目标文件的修改时间，并且编排出需要重新编译的文件数。
- 设置文件属性(Set Attributes)：修改文件的一些属性，以适应用户的要求。
- 重命名(Rename)文件：重新命名一个已经存在的文件。

10 存取控制表 ACL 的概念。

- 为存取控制矩阵中的每一列建立一张存取控制表(ACL)，用一有序对(域，权集)表示。
- 文件的 ACL 记录在 FCB 中。

11 理解内存映射文件（memory mapped file）的过程。

- 将文件映射到进程地址空间的一个区域，返回起始虚地址，仅当需要对文件存取时，才传输实际的数据。
- 使用与请求页式虚存管理相同的存取机制。访问的页不在主存时，产生缺页中断读入主存。
- OS 提供映射文件的系统调用。

第 6 章　设备管理

1　I/O 设备通常大致可分为哪两大类？各自传输的信息单位有什么特点？

- 字符设备：人机交互设备。是以字符为单位发送和接收数据的，通信速度比较慢。键盘和显示器、鼠标、扫描仪、打印机、绘图仪等。
- 块设备：外部存储器。以块为单位传输数据。常见块尺寸：512B～32KB。如磁盘、磁带、光盘等。
- 网络通信设备：主要用于远程设备的通信。传输速度比字符设备快，比块设备慢。如网卡、调制解调器等。
- 时钟：按预先规定好的时间间隔产生中断。

2　常用的四种数据传输控制方式。

- 程序查询方式（polling）：CPU 直接向设备控制器发出启动的 I/O 命令，一直循环查询设备是否完成输入输出。该方式的工作过程非常简单，但 CPU 执行指令的速度高出 I/O 设备几个数量级，所以在循环测试中浪费了大量的 CPU 处理时间。CPU 与设备串行操作。
- 中断方式：CPU 一旦启动设备成功，CPU 转去执行另一个程序。当设备完成时，向 CPU 提出中断请求，CPU 执行完当前一条指令，就响应中断，转去执行中断处理程序。CPU 与设备并行操作。

- 直接存储器访问(DMA)方式：允许 DMA 控制器接管地址总线的控制权，直接控制设备与主存之间的数据交换。仅在开始和结束时需要 CPU 干预。
- 通道控制方式：与 DMA 方式相比，通道所需的 CPU 干预更少，且可以做到一个通道控制多台设备，进一步减轻了 CPU 的负担。通道是一种专用的 I/O 处理机。通道有自己的指令系统，若干条通道命令连接成通道程序。
- 比较：
  - 中断方式是在数据缓冲满之后，发中断要求 CPU 进行处理，而 DMA 方式则是在所要求转送的数据块全部结束时，要求 CPU 处理。这就大大减少了 CPU 进行中断处理的次数。
  - 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。

3 根据设备的使用方式，设备被分为几种类型？何为虚拟设备？它是通过什么技术实现的?

- 独占设备：临界资源，如打印机。
- 共享设备：多个进程可交叉访问。如磁盘。
- 虚拟设备：是指设备本身是独占设备，而经过虚拟技术处理，可以把它改造成共享设备，供多个进程同时使用。通过 Spooling 技术实现。

4 按照设备管理的层次结构，I/O 软件划分为几层？各层主要实现哪些功能?

① I/O 硬件

② 中断处理程序

- 进程在启动一个 I/O 操作后阻塞起来，I/O 操作完成，控制器产生一个中断。
- CPU 响应中断，执行中断处理程序。

✧ 检查设备状态。

- 若正常完成，就唤醒等待的进程。然后检查是否还有待处理的 I/O 请求若有就启动。
- 若传输出错，再发启动命令重新传输；或向上层报告“设备错误”的信息。

✧ 返回被中断的进程，或转进程调度。

③ 设备驱动程序

✧ 设备初始化。

✧ 启动设备传输数据。

✧ 中断处理。

④ 独立于设备的软件

✧ 设备命名

✧ 设备保护

✧ 提供与设备无关的块尺寸

✧ 缓冲技术

✧ 设备分配

✧ 出错处理

⑤ 用户层的 I/O 接口

大部分 I/O 软件都包含在操作系统中，有一小部分是由与用户程序连接在一起的库函数构成的。这些函数通常只是将系统调用时所需要的参数放在合适的位置，由系统调用实现真正的操作。

5 何为设备的独立性?

设备独立性，是用户在编写使用设备的程序时，通过系统提供的一组统一的操作接口，这些接口与具体的设备无关。这样，无论系统设备如何改变，用户的程序不受影响。

6 什么是 SPOOLING 技术？以输出为例，说明它的实现原理。(SPOOLING 技术是以空间换时间)

- SPOOLING 技术用来实现虚拟设备，实际是一种缓冲技术。
- 当进程要求输出打印时，系统并不为它分配打印机设备，而是分配可共享设备（如硬盘）的一部分空间，使每个进程的输出被缓冲到一个独立的磁盘文件上。当进程完成执行时，再将相应的被缓冲文件进行排队，准备打印输出。之后，SPOOLING 技术一次一个地将排队的待打印的文件复制给打印机进行实际的打印。
- SPOOLING 技术又叫缓输出技术，提高了独占设备的利用率，也减少了进程等待设备进行数据传输的时间，缩短了用户进程的周转时间。

7 一个特定磁盘上的信息如何进行编址？盘面号、磁道号和扇区号（或柱面号、磁头号和扇区号)。

8 要将磁盘上一个块的信息传输到主存需要系统花费哪些时间?

寻道时间、旋转延迟时间、读/写传输时间

9 常用的磁盘调度算法

- 先来先服务（FCFS）

  最简单、易实现、公平合理

- 最短寻道时间优先

  在将磁头移向下一请求磁道时，总是选择移动距离最小的磁道。

- 扫描法（SCAN, C_SCAN, LOOK, C_LOOK)。

设某单面磁盘的旋转速度为每分钟 6 000 转，每个磁道有 100 个扇区，相邻磁道间的平均移动时间为 1ms。若在某时刻，磁头位于 100 号磁道处，并沿着磁道号增大的方向移动，磁道号的请求队列为 50、90、30、120，对请求队列中的每个磁道需读取 1 个随机分布的扇区，则读完这 4 个扇区总共需要多少时间？需要给出计算过程。

解：循环查询 C-LOOK 算法。被请求的磁道号顺序为 100、120、30、50、90，因此，寻道需要移动的磁道数为：20+90+20+40=170。寻道用去的总时间为：

(20 + 90 + 20 + 40)×1ms = 170ms

磁盘每分钟 6000 转，转一圈的时间为 0.01s，通过一个扇区的时间为 0.0001s。

总共要随机读取四个扇区，用去的时间为：(0.01×0.5 + 0.0001)×4 = 0.0204s = 20.4ms

所以 170ms + 20.4ms = 190.4ms。

第 7 章　Linux 进程管理

1　进程控制块，其中与进程管理、存储器管理和文件管理有关的一些字段，线程组标识符。

- task_struct 进程描述符
- thread_info 当前进程基本信息
- mm_struct 指向进程所拥有的虚拟内存描述符
- thread_struct 保存进程的硬件上下文
- signal_struct 所接收的信号
- pid 进程标识符
- tgid 线程组标识符

2　与进程创建有关的函数：fork( )、vfork( )、clone( )。

- 创建子进程函数 fork()：创建成功之后，子进程采用写时复制技术读共享父进程的全部地址空间，仅当父或子要写一个页时，才为其复制一个私有的页的副本。
- 创建轻量级进程函数 clone()：实现对多线程应用程序的支持。共享进程在内核的很多数据结构，如页表、打开文件表等等。
- vfork()系统调用：创建的子进程能共享父进程的地址空间，为了防止父进程重写子进程需

要的数据，先阻塞父进程的执行，直到子进程退出或执行了一个新的程序为止。

3 理解进程切换的过程。涉及到页目录表、核心栈、硬件上下文。

- 进程切换只发生在核心态。在发生进程切换之前，用户态进程使用的所有寄存器值都已被保存在进程的核心栈中。
- 进程硬件上下文存放在进程描述符的 thread_struct thread 中。该结构包含的字段涉及到大部分 CPU 寄存器，但像 eax、ebx 等通用寄存器的值仍被保留在核心栈中。
- 进程切换步骤：

  ① 切换页目录表以安装一个新的地址空间；

  ② 切换核心栈和硬件上下文。由 schedule()函数完成进程切换。

4 进程调度方式。进程调度时机。

- Linux 系统采用可抢先式的动态优先级调度方式。无论进程处于用户态还是核心态运行，都可能被抢占 CPU。
- 调度类型
  - 先进先出的实时进程
  - 时间片轮转的实时进程
  - 普通的分时进程：基本时间片、动态优先级
- 通过系统调用可改变普通进程的静态优先级。
- 进程调度时机：
  - 出现了更高优先级的实时进程。
  - 进程执行了阻塞操作而进入睡眠状态。
  - 进程停止运行或被杀死。
  - 进程调用自愿放弃处理机。

- ✧ 在基于时间片轮转的实时进程调度过程中，进程用完了自己的时间片。

5 Linux 有很多内核线程，了解 0 号进程和 1 号进程的作用。

- ◆ 0 号进程是一个内核线程，使用静态分配的数据结构，其进程描述符存放在 init_task 变量中。0 号进程是所有进程的祖先进程，又叫 idle 进程或叫做 swapper 进程。每个 CPU 都有一个 0 号进程。
- ◆ 1 号进程是由 0 号进程创建的内核线程 init，负责完成内核的初始化工作。在系统关闭之前，init 进程一直存在，它负责创建和监控所有用户进程。

第 8 章 Linux 存储器管理

1 进程地址空间的划分？管理进程私有地址空间的数据结构？链接虚拟内存区域的单链表和红黑树。指向映射文件对象的指针字段？指向进程页目录表的指针字段?

- ◆ 进程地址空间的划分：Linux 把地址空间分成两部分。进程的私有空间是前 3G，进程的公有空间是后 1G 的内核虚空间。
- ◆ 管理私有地址空间的数据结构：
  - ✧ 虚拟内存描述符 mm_struct。
  - ✧ 虚拟内存区域描述符 vm_area_struct。
- ◆ 进程地址空间的各个虚拟内存区域是采用单链表和红黑树相组合的方式进行管理的。
  - ✧ 当进程需要的虚拟内存区域数较少时，使用链表来管理虚拟内存区域；较多时，采用红黑树来管理。
  - ✧ 红黑树是一棵排好序的平衡二叉树。必须满足四条规则：①树中的每个节点或为红或为黑；②树的根节点必须为黑；③红节点的孩子必须为黑；④从一个节点到后代

诸叶子节点的每条路经，都包含相同数量的黑节点，在统计黑节点个数时，空指针也算作黑节点。这四条规则保证：具有 n 个节点的红黑树，其高度至多为 2*log(n+1)。

- 指向映射文件对象的指针：vm_area_struct 中的 vm_file
- 指向进程页目录表的指针：mm_struct 中的 pgd

2 Linux 堆的管理：malloc( )，free( )。

- malloc(size)：请求 size 个字节的动态内存。
- free(addr)：释放内存。

3 管理物理内存页框的数据结构？内存管理区 zone 结构，伙伴系统？分区页框分配器分配页框的过程。

- 页框描述符为 struct page。所有页框描述符存放在 mem_map 数组中。
- Linux 把内存节点划分为 3 个管理区 zone。
  - ZONE_DMA：包含低于 16MB 的常规内存页框。用于对老式的基于 ISA 设备的 DMA 支持。
  - ZONE_NORMAL：包含高于 16MB 且低于 896MB 的常规内存页框。
  - ZONE_HIGHMEM：包含从 896MB 开始的高端物理页框。内核不能直接访问这部分页框。在 64 位体系结构上，该区总是空的。
- 伙伴系统：管理连续的空闲内存页框，以解决外碎片问题。所谓外碎片，是夹杂在已分配页框中间的那些连续的小的空闲页框。伙伴算法把空闲页框组织成 11 个链表，以页框为单位，适合于对大块内存的分配请求。
- 分区页框分配器分配页框的过程：
  - 假设要请求一个具有 8 个连续页框的块。

- 先在 8 个连续页框块的链表中检查是否有空闲块。
- 若没有，则检查 16 个连续页框块的链表。若找到，则将其分成两等份，一份满足请求，另一份插入 8 个连续页框块的列表。
- 若在 16 个连续页框块的链表中没有找到空闲块，则继续在更大的块链表中寻找，直到找到为止。

4 理解 slab 分配器的原理。slab 分配器的作用?

- slab 分配器用于为只有几十或几百个字节的小内存区分配内存。如，file 对象。
- slab 分配器为不同类型的对象生成不同的高速缓存。

5 进程页表建立的时机？了解页目录表项或页表项所包含的字段。逻辑地址的划分，利用两级页表实现地址转换的过程。

- 处理机普遍采用二级页表模式，为每个进程分配一个页目录表，页表一直推迟到访问页时才建立，以节约内存。
- Linux 系统的页目录项和页表项的数据结构相同。
- 地址转换：
  - 查页目录表，确定对应的页表位置。
  - 查页表，确定对应的物理页框位置。

6 请求调页。所缺的页可能存放的地方。

- 请求调页机制是把页框的分配一直推迟到进程要访问的页不在 RAM 中时引起一个缺页异常，才将所需的页调入内存。请求调页增加了系统中的空闲页框的平均数。
- 页面置换策略是 LFU（Least Frequently Used）
- 所缺页的可能存放处：
  - 该页从未被进程访问过，且没有相应的内存映射。

- ✧ 该页属于非线性内存映射文件。非线性内存映射的是文件数据的随机页。给定文件的所有非线性映射虚拟内存区域描述符都存放在一个双向链表中。
- ✧ 该页已被进程访问过，但其内容被临时保存到磁盘交换区上。
- ✧ 该页在非活动页框链表中。
- ✧ 该页正在由其它进程进行 I/O 传输过程中。

7 了解盘交换区空间的管理方法。

- ◆ 盘交换区用来存放从内存暂时换出的数据页
- ◆ 每个盘交换区都由一组 4KB 的页槽组成。
- ◆ 盘交换区的第一个页槽用来存放该交换区的有关信息，有相应的描述符。
- ◆ 存放在磁盘分区中的交换区只有一个子区，存放在普通文件中的交换区可能有多个子区，原因是磁盘上的文件不要求连续存放。
- ◆ 内核尽力把换出的页存放在相邻的页槽中，减少访问交换区时磁盘的寻道时间。

第 9-10 章 Linux 文件系统

1 Ext2 文件卷的布局？各部分的作用是什么？

- ◆ Ext2 把磁盘块分为组，每组包含存放在相邻磁道的数据块和索引节点。块组的大小相等并顺序安排。
- ◆ Ext2 用“块组描述符”来记录块组的结构信息，同时为了容错，将超级块和所有块组描述符重复存储于每个块组中。
- ◆ Ext2 通过“位图”来管理每个块组中的磁盘块和索引节点，包括数据块位图和索引节点位图。一个块组中的盘块和索引节点的位图必须存放在一个单独的磁盘块中。

2 Linux 系统把一般的文件目录项分成哪两部分？这样做的好处是什么？

- 把通常的文件目录项分成简单目录项和索引节点两部分。
- 简单目录项包含了文件名和索引节点号等，文件的管理控制信息放在文件的索引节点中。
- 优点：
  - 提高文件目录的检索速度和文件系统的使用效率。
  - 系统只保留一个索引节点，就可实现多条路径共享文件，减少信息冗余，提高信息的一致性。

3 Linux 文件系统的索引节点中，索引表划分成几级？计算文件最大长度文件的索引表是如何增长的？要求能够利用索引表实现将文件中的字节地址转换成文件的物理块的操作。

- 索引表共 15 个元素，分成四级。
  - 0~11：直接索引项。给出文件最初的 12 个逻辑块号对应的物理块号。
  - 12：1 次间接索引项。对应的文件逻辑块号从 12 到（b/4）+11。
  - 13：2 次间接索引项。对应的文件逻辑块号从（b/4）+12 到 $(b/4)^2+(b/4)+11$。
  - 14：3 次间接索引项。对应的文件逻辑块号从 $(b/4)^2+(b/4)+12$ 到 $(b/4)^3+(b/4)^2+(b/4)+11$。

4 硬链接和符号链接的区别？

符号链接不与文件的索引节点建立链接，这种链接允许在不同的文件系统之间建立。

5 Linux 文件系统如何管理空闲存储空间？

包括空闲块的管理和空闲索引节点的管理。它们都采用位示图实现分配和回收。

6 VFS 通用文件模型中的四个主要对象？为什么可以支持众多的文件卷

- 超级块对象：Linux 为每个安装好的文件系统都建立一个超级块对象。
- 索引节点对象：代表一个文件，对应于存放在磁盘上的文件控制块。

- 目录项对象：代表一个目录项，是一个文件路径的组成部分，存放目录项与对应文件进行链接的信息。
- 文件对象：记录了进程与打开的文件之间的交互信息。

7 Linux 系统中，进程打开一个磁盘文件要涉及哪些数据结构？它们各有哪些关键字段？他们的作用是什么？参考图 10.2

- struct task_struct
  - struct fs_struct *fs; 指向文件系统信息
    - struct dentry *root, *pwd; 指向根目录和当前工作目录的指针
      - struct inode *d_inode; 文件的索引节点
  - struct files_struct *files; 指向进程的打开文件信息
    - struct file *[] fd_array; 文件对象指针的初始数组
      - struct dentry *f_dentry; 文件对应目录项对象的指针

8 一个文件在使用与不用时各占用系统哪些资源?

不用时：外存

用时：内存、外存、外部设备等

9 了解安装表的作用

内核将安装点与被安装的文件系统信息保存在 vfsmount 结构中，形成一个链式安装表。

第 14 章 Windows 2000/XP 模型

1 Windows 采用什么样的体系结构?

- 融合了分层和客户/服务器系统模型。
- 系统划分为用户态和核心态。

- 用户态：
  - 系统支持进程
  - 系统服务进程
  - 环境子系统进程（csrss.exe）：为用户提供一个图形界面
  - 用户应用程序进程
- 核心态：
  - 操作系统内核（Ntoskrnl.exe）：
    - 执行体
    - 内核
  - 设备驱动程序
    - NTFS 驱动程序（NTFS.exe）
  - 硬件抽象层（Hal.dll）
  - 子系统内核部分（win32k.sys）

2 硬件抽象层 HAL 的作用是什么?

- HAL 直接操纵硬件，是 Windows 操作系统在多种硬件平台上可移植性的组件。
- HAL 是一个可加载的核心态模块 HAL.dll，它为 Windows 运行在硬件平台上提供低级接口。
- HAL 向内核、设备驱动程序和执行体的其他部分隐藏各种与硬件有关的细节，使其免受特殊硬件平台差异的影响。

3 Windows 系统组件的基本机制。

- 陷阱调度
  - 属于内核的功能，是操作系统处理意外事件的硬件机制。
  - 当硬件或软件检测到异常或中断发生时，将暂停正在处理的事情，把控制转交给内核

的陷阱处理程序。陷阱处理程序检测异常和中断的类型，并将控制转交给相应的处理程序。

- ✧ 中断是异步事件；异常是同步事件，系统服务调用被视为异常。

◆ 执行体对象管理器

- ✧ 管理操作系统内的所有对象。
- ✧ 有两种类型的对象：执行体对象、内核对象
- ✧ 对象管理器的工作就是跟踪所有对象，便于对象访问的安全检测。

◆ 同步

- ✧ 自旋锁、内核调度程序对象

◆ 本地过程调用 LPC

4 理解：延迟过程调用 DPC，异步过程调用 APC

◆ 延迟过程调用 DPC：

- ✧ DPC 被用来执行一些相对于当前高优先级的任务来说不那么紧急的任务。
- ✧ 有时内核在进行系统嵌套调用时，检测到应该进行线程调度。为了保证调度的正确性，内核用 DPC 来延迟请求调度的产生。
- ✧ 硬件中断服务例程，可以把一些相对不紧急的事情放到一个 DPC 对象中处理，从而缩短处理机停留在高 IRQL 的时间。
- ✧ DPC 对象主要包含需系统调用的函数地址。
- ✧ DPC 队列是系统范围的，它能记住有哪些工作尚未处理。
- ✧ 当 IRQL 降低到 DPC/Dispatcher 级别以下时，DPC 中断就产生。调度程序依次执行 DPC 队列中的每个例程，直至 DPC 队列为空。

◆ 异步过程调用 APC：

- ✧ 为用户程序和系统代码提供了一种在特定用户线程环境中执行代码的方法。
- ✧ 每个线程都有自己的 APC 队列。 当一个线程被调度时，它的 APC 过程会立刻被执行。
- ✧ 两种 APC：
  - ■ 核心态 APC：可以中断线程，不需要得到目标线程允许。
  - ■ 用户态 APC：需要得到目标线程的允许。
- ✧ APC 队列也由内核管理。
- ✧ 如果需要从内核空间复制一个缓冲区到用户进程地址空间，那么复制过程需要在用户进程上下文进行，这样页表才能包含内核缓冲区和用户缓冲区。

5 Windows 中有哪些对象，都有什么作用?

- ◆ 执行体对象：是执行体的各种组件实现的对象。如进程和线程对象、文件映射对象、I/O 子系统创建的文件对象等。
- ◆ 内核对象：是由内核实现的一批初级对象，对用户态代码不可见，仅供执行体使用。
- ◆ 一个执行体对象可以包含一个或多个内核对象。

6 在多处理机系统中，提供了哪些同步和互斥机制?

- ◆ 内核同步
  - ■ 利用自旋锁
- ◆ 执行体同步
  - ■ 内核以内核对象的形式给执行体提供了用户态可见的附加的同步机构—— “调度程序对象”，包括：进程对象、线程对象、事件对象、信号量对象、互斥体对象、可等待的定时器对象及文件对象等。
  - ■ 每个同步对象都有“有信号”或“无信号”两种状态。

7 线程如何实现等待一个同步对象的操作?

线程通过调用由对象管理器提供的 WaitForSingleObject( )，WaitForMultipleObjects( ) 系统服务来与调度程序对象同步，等待同步对象变为有信号状态。

第 15 章 Windows 进程和线程管理

1 管理进程和线程的数据结构：执行体进程块 EPROCESS、执行体线程块 ETHREAD、内核进程块 KPROCESS、内核线程块 KTHREAD。

- struct EPROCESS 执行体进程块
  - struct KPROCESS Pcb；内核进程快
- struct ETHREAD 执行体线程块
  - struct KTHREAD Tcb； 内核线程块

2 创建进程：CreateProcess( )；创建线程：CreateThread( )

- CreateProcess 创建新进程及其主线程，执行指定的程序。主要流程：
  ① 打开将在进程中执行的映像文件(.exe)，创建一个区域对象，建立映像与主存之间的映射关系。
  ② 创建执行体进程对象，包括申请并初始化执行体进程控制块，创建并初始化进程地址空间、内核进程块和进程环境块等。
  ③ 创建一个主线程。
  ④ 通知 Win32 子系统，对新进程和线程进行一系列初始化。
  ⑤ 完成地址空间的初始化，开始执行程序。
- CreateThread 创建新线程
  ① 在进程的地址空间为线程创建用户态堆栈。

② 初始化线程的 CPU 硬件描述表。

③ 调用 NtCreateThread 创建执行体线程对象，调用 ResumeThread 激活线程。

3 线程的 7 种状态，及其解释。

- 就绪状态(ready)：表明线程可以被调度执行，在就绪队列中排队。
- 备用状态(standby)：作为下一个运行的线程，已选好处理机，正等待描述表切换，以便进入运行状态。
- 运行状态(Running)：对于每个处理机，只有一个线程可以处于运行状态。
- 等待状态(waiting)：等待某个事件或某个对象成为有信号状态。
- 传输状态(transition)：核心栈被调到外存的就绪态。
- 终止状态(terminated)：线程执行完成。
- 初始化状态(Initialized)：正在创建过程中。

4 线程调度：基于优先级的抢先式的多处理机调度系统。

- 线程调度程序的数据结构：
  - 32 个就绪队列。每个优先级对应一个。
  - 32 位掩码的就绪位图。每一位指示一个调度优先级的就绪队列中是否有线程等待运行。
  - 32 位掩码的空闲位图。每一位指示一个处理机是否处于空闲状态。

5 进程优先级

- 空闲优先级：在系统处于空闲状态时执行的程序，如屏幕保护程序
- 普通优先级：能升高或降低
- 高优先级：只在需要时才使用，如 Task Manager
- 实时优先级：用于核心态的系统进程

6 线程优先级的提升时机。

- I/O 操作完成后的线程。
- 信号量或事件等待结束的线程。
- 前台进程中的线程完成一个等待操作。
- 由于窗口活动而唤醒图形用户接口线程。
- 线程处于就绪状态超过一定时间，仍未能进入运行状态(处理器饥饿)。

第 16 章　Windows 存储器管理

1 两种数据结构：虚拟地址描述符 VAD、区域对象，这两种结构各有什么作用?

- 虚拟地址描述 VAD：
  - 当线程要求分配一块连续虚存时，系统并不立即为其构造页表，而是为它建立一个 VAD 结构。
  - VAD 结构：被分配的地址域、该域是共享的还是私有的、该域的存取保护以及是否可继承等信息。
  - 进程页表的构建一直推迟到访问页时才建立。（“懒惰”方式）
- 区域对象：
  - 利用区域对象将一个可执行文件装入主存。
  - 使用区域对象可将一个大于进程地址空间的文件映射到进程地址空间。
  - 缓存管理器利用区域对象访问一个被缓存文件中的数据。

2 虚存内存区域：空闲的、保留的、提交的

- 进程私有的 2G 地址空间的页可能是空闲的，或被保留，或被提交。

- ✧ 空闲：还没有被使用过
- ✧ 被保留：已预留虚存，还没分配物理主存
- ✧ 被提交：已分配物理主存或交换区。

◆ 分配主存时，可以先保留地址空间，后提交物理主存；也允许保留和提交同时实现。

◆ 第一阶段只保留地址空间，特别适合线程正在创建大的动态数据结构的情况。

3 32 位逻辑地址，二级页表。页目录表项和页表项具有相同的数据结构，该数据结构包含哪些数据项？进程页表建立的时机。进程的地址转换过程。

◆ 页目录表的每一项记录一个页表的地址。进程页表不再占用连续的主存空间。

◆ 32 位的虚地址被分解为：页目录索引、页表索引、页内字节索引。

◆ 进程页目录的物理地址被保存在核心进程块中(KPROCESS)。

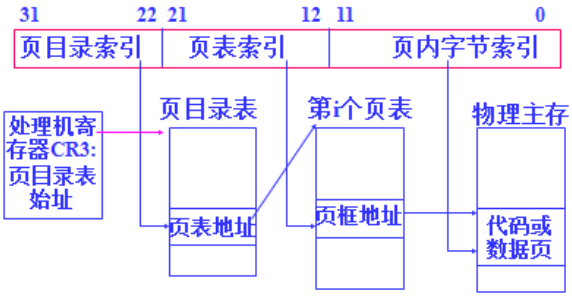
◆ 虚拟地址变换过程：

① 系统把即将运行进程的页目录表始址送处理器的 CR3 寄存器。

② 由页目录索引定位某个页表在页目录表中页目录项的位置，找到某页表所在页框号。

③ 页表索引定位指定页在页表中的位置。如果该页是有效的，找到虚拟页在物理主存的页框号。如果该页是无效的，存储管理器的故障处理程序将失效的页调入主存。

④ 当页表项包含有效页时，页内字节索引定位程序或数据在物理页框内的地址。



4 管理物理内存的数据结构：页框数据库。

◆ 页框的 8 种状态：活动、转换、备用、更改、更改不写入、空闲、零初始化、坏。

5 原型页表项的概念。

- 当一个页框被两个或多个进程共享时，存储器管理器依靠一个称为“原型页表项”（Prototype，PTE）的软件结构来映射这些被共享的页框。引入原型页表项是为了尽可能地减少对各进程的页表项的影响。

6 无效页处理

- 访问一个未知页，其页表项为 0 或页表不存在。
- 所访问的页不在主存，在外存页文件或映像文件。
- 所访问的页在备用链表或更改链表。
- 页访问违约，与访问权限不符。

7 Windows 采用的页替换策略是什么?

- 调页策略：将所缺的页及其前后的一些页装入主存。程序的局部性原理。
- 置页策略：放到物理主存。
- 置换策略：
  - 在多处理器系统中，采用了局部先进先出置换策略。
  - 在单处理器系统中，更接近于最近最久未使用策略（LRU，也称为“最近最少使用算法”）。

第 17 章 Windows 文件系统

1 Windows 所支持的文件系统类型有哪些?

- FAT：支持向下兼容的文件系统
- NTFS：规定使用 64 位的簇编号，但限制用 32 位来表达。
- 卷上簇的大小是在使用 Format 命令格式化卷时确定的。

2 虚拟簇号和逻辑簇号的概念。

- FAT 和 NTFS 将卷划分成若干簇，并从卷头到卷尾进行编号，称为逻辑簇号(LCN)。
- 通过索引表建立文件的虚拟簇号(VCN)与磁盘的逻辑簇号之间的映射。

3 NTFS 卷的结构，主控文件表 MFT 的作用。

- NTFS 卷的结构：分区引导扇区、主控文件表（MFT）区、文件数据区。
- MFT 是 NTFS 卷的管理控制中心，包含了卷上所有的文件、目录及空闲未用盘簇的管理信息。
- 文件数据区依次存放 NTFS 所有系统文件，主控文件表 MFT 的镜像文件，根目录、普通文件和子目录，一些空闲簇等。

4 NTFS 文件的物理结构：索引顺序结构。

5 对于大目录，采用 B+树，使得查找一个特定文件的访盘次数减到最少。大目录文件的索引根属性包含 B+树的第一级并指向包含下一级的索引缓冲区。