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。

还要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?

进程----进程表　存储器----存储表　I/O 设备----I/O 设备表　文件----文件表

3　为什么要引入多道程序系统?它有什么特点?

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。

指在主存同时存放若干道程序，使它们在系统中交叉运行，共享系统中的各种资源。当一道程序暂停执行时，CPU 立即转去执行另一道程序。

在单处理机系统中。宏观上，多道程序并行运行；微观上，在任何特定时刻，只有一道程序在处理机上运行，即各程序交叉地在 CPU 上运行。

4　叙述操作系统的基本功能。

(1)处理机管理：进程管理。处理机如何调度的问题：FCFS、优先级、时间片轮转?

(2)存储器管理：主存管理。存储分配、存储保护、主存扩充。

(3)设备管理：涉及对系统中各种输入、输出设备的管理和控制。分配设备，控制设备传输数据。

(4)文件管理：将程序、数据、操作系统软件等组织成文件，存在磁盘或磁带上，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：优点：系统吞吐量大，资源利用率高。适合计算量大、自动化程度高的成熟作业。

缺点：用户与作业无法交互，作业平均周转时间较长。

目标是提高系统资源的利用率。适用于比较成熟的大作业。可在后台执行。不需要用户频繁干预。

分时系统：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。

目标是对用户请求的快速响应。适用于短小作业。终端键入命令。

实时系统：（1）实时性。其响应时间由被控制对象所能承受的延迟来确定。

（2）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。

（3）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。

适用于实时过程控制，实时信息处理。

6　操作系统的特性?

(1)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。

(2)共享性：是指系统中的资源可供内存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

(3)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。

(4)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？

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。

响应时间：用户发出请求或者指令到系统做出反应（响应）的时间。

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。

软件要求固化存储。

通常是一个多任务可抢占式的实时操作系统，只有满足实际需要的有限功能，如任务调度、同步与通信、主存管理、时钟管理等。

9　什么是对称多处理？它有什么好处？

多处理机采用紧耦合方式进行连接，共享主存

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。

优势：①增加了系统的吞吐率。②增加了系统的可靠性。

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 的运行状态分为核心态（管态）和用户态（目态）。通过中断和异常，CPU 能从用户程序的运行转入操作系统内核程序的运行。在核心态下， 允许执行处理机的全部指令集，访问所有的寄存器和存储区；在用户态下，只允许执行处理机的非特权指令，访问指定的寄存器和存储区。用户态到核心态的转换由硬件完成；管态到目态的转换由操作系统程序执行后完成。

11　什么是系统调用，特权指令？特权指令执行时，CPU 处于哪种工作状态？

系统调用就是操作系统内核提供的一些子程序。操作系统内核向用户提供了一组系统调用接口。用户通过系统调用接口，向操作系统提出资源请求或获得系统服务。

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，设置时钟时间，关中断，清主存。只允许操作系统使用，不允许用户使用 CPU 指令集分为特权指令和非特权指令。特权指令执行时，CPU 处于和心态。

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：命令语言或窗口界面是用户使用计算机系统的主要接口。

编程接口：系统调用是用户与操作系统之间的编程接口。

第 2-3 章　进程管理

1　程序顺序执行的特点

程序的顺序（串行）执行：计算机每次只运行一道程序。如单道批处理系统。

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。

优点： 由于顺序程序的封闭性和可再现性，为程序员调试程序带来了很大方便。

缺点： 由于资源的独占性，使得系统资源利用率非常低。

2　何谓进程，进程由哪些部分组成？试述进程的四大特性及进程和程序的区别。

操作系统用“进程” 来描述系统中各并发活动进程（process）又叫做任务（task）进程是程序的一次执行过程进程。

组成：进程是由**程序、数据和进程控制块**三个部分组成。

特性：

（1）动态性。进程是程序的一次执行过程，是临时的，有生命期的。

（2）独立性。进程是系统进行资源分配和调度的一个独立单位。

（3）并发性。多个进程可在处理机上交替执行。

（4）结构性。系统为每个进程建立一个进程控制块。

进程和程序的区别：

（1）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。

（2）进程是暂时的，程序是永久的。

（3）进程包括程序、数据和进程控制块。

（4）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。

进程并发执行的特点

失去了封闭性和可再现性，

以资源共享为基础，并发执行的程序之间产生了制约关系，

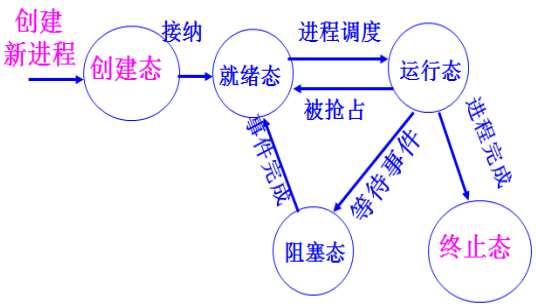
程序与 CPU 执行的活动不在一一对应。（比如编译程序）

3 进程控制块的作用是什么?它主要包括哪几部分内容?

也叫进程描述符，是为了描述进程的运行变化情况，操作系统为每个进程定义了一个数据结构，它是进程存在的唯一标识。它包含了进程的**描述（状态、调度、资源信息）信息和管理控制信息**。具体包括：

1. 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。
2. 进程的状态以及调度、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。
3. 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。
4. 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。
5. 记帐信息：包括使用 CPU 时间量、帐号等。
6. 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。Windows 进程之间不具有父子关系。
7. 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。



进程的三个基本状态：

（1）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。

(2)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。

(3)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。

(4) 创建态：刚刚建立，未进就绪队列。

(5) 终止态：已正常结束或故障中断，但尚未撤消。暂留在系统中，方便其它进程去收集该进程的有关信息。

进程的组织：

(1) 线性表

(2) 链接表：相同状态的 PCB 组成一个队列，也可以处于就绪态的进程可按照某种策略排成多个就绪队列，根据阻塞的原因不同组织成多个阻塞队列。

5 什么是原语?什么是进程控制？

原语是由若干多机器指令构成的完成某种特定功能的一段程序，具有不可分割性，即原语的执行必须是连续的，在执行过程中不允许被中断。

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。进程控制是由操作系统内核实现的。是属于原语一级的操作，不能被中断。

原语类型：

(1) 创建原语 fork（）、CreateProcess()

(2) 撤销原语 exit()\ExitProcess()

(3) 阻塞原语 中断 CPU Unix 阻塞原语 Sleep（）、Pause（）、Wait()、Kill（）Linux block（）

(4) 唤醒原语 Wakeup()

(5) 挂起原语（6）解挂原语 静止状态变为活动状态，被解挂的进程变为活动就绪时，立即转进程调度

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?

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：将处于主存就绪或者主存阻塞等不具备运行条件的进程换出到外存交换区

进程调度：低级调度。进程调度就是为进程分配处理机，也叫处理机调度

功能：

(1) 记录进程执行状况。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。

(2) 选择就绪进程占有 CPU

(3) 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。

进程上下文：

（1） 用户级上下文（进程的程序、数据、用户栈），

（2） 寄存器级上下文（CPU 的现场信息），

（3） 系统级上下文（进程 PCB 和核心站）

栈：记录进程的执行历程，存放有关输入参数、局部变量、过程调用返回地址

进程调度方式：

① 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。

② 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：

（1）**现行进程完成或错误终止**；

（2）**提出 I/O 请求，等待 I/O 完成时**；

（3）在分时系统，按照**时间片轮转**，分给进程的时间片用完时；

（4）**优先级调度**，有更高优先级进程就绪；

（5）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。

处理机调度算法：

作业调度（1-3）：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。

（1）先来先服务(FCFS)：简单，节省机器时间。缺点：容易被大作业垄断，使得平均周转时间延长。

（2）最短作业优先(SJF)：选取运行时间最短的作业运行。对短作业有利，作业的平均周转时间最佳。缺点：若系统不断进入短作业，长作业就没有机会运行，出现饥饿现象。

（3）响应比高者优先(HRN)：Rp =(作业等待时间+作业估计运行时间)/作业估计运行时间 =1+作业等待时间/作业估计运行时间

特点：结合了先来先服务、短作业优先的方法。通常情况下优先运行短作业然后等待时间足够长的长作业也会被运行。

交换调度（4-6）：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。

（4）优先级调度法：将 CPU 分配给就绪队列中优先级最高的进程

（5）轮转法(RR)：用在分时系统，轮流调度所有就绪进程。

（6）多级反馈队列轮转法：系统设置多个就绪队列。

批处理系统：增加系统吞吐量和提高系统资源的利用率；

实时系统：保证对随机发生的外部事件做出实时响应。

分时系统：保证每个分时用户能容忍的响应时间。

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

- 线程(thread)：是进程内的一个可执行实体，是处理机调度的基本单位，**以进程为单位分配资源，以线程为单位调度执行。**

比较：

（1）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。

（2）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。但是，在由一个进程的线程向另一个进程的线程切换时，会引起进程上下文的切换。

（3）并发性

引入线程后，使得系统个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。

（4）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，共享方便，但是和多线程比较，**多进程实现，安全性好。**

系统对线程的支持

1） 用户级线程（用户调用用户态运行的线程库完成，一个线程阻塞，进程也阻塞，多线程对应核心级一个进程）

运行时系统（Run-time system）是一个管理线程的过程集合，包括：thread_create、thread_exit、thread_wait。

2） 核心级线程（线程的管理工作由内核完成，用用程序通过系统调用来创建或者撤销线程，个线程的阻塞，不影响其他线程的执行，用户态的一个线程对应核心态的一个线程或者进程）

3） 两级组合（多个用户级线程，对应等量或者少量的核心级线程）

4） 线程创建：clone（） CreateThread（）

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?

（1）对资源的共享引起的互斥关系

进程之间本来是相互独立的，但由于共享资源而产生了关系。间接制约关系，互斥关系。

（2）协作完成同一个任务引起的同步关系

一组协作进程要在某些同步点上相互等待发信息后才能继续运行。直接制约关系，同步关系。

（3）进程之间的前序关系

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。

并发进程进入临界区四条原则：

(1) 不能同时有两个进程在临界区内执行——互斥使用；

(2) 等待进入临界区的进程，应释放处理机后阻塞等待——让权等待；

(3) 在临界区外运行的进程不可阻止其他进程进入临界区——有空让进；

(4) 不应使要进入临界区的进程无限期等待在临界区之外——有限等待。

10 解决进程之间互斥的办法（进程间的低级通信）：开、关中断，加锁、开锁（又叫测试与设置硬件指令，通常由一条机器指令完成），信号量与 P、V 操作。

(1) 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（时间片到）或其它中断时才会进行进程切换。

(2) 使用测试和设置硬件指令 锁位变量 W ：为每个临界资源设置一个，以指示其当前状态。W=0，表示资源空闲可用；W=1，表示资源已被占用。（忙等）

信号量和 PV 操作：两个或多个进程通过简单的信号进行合作，一个进程被迫在某一位置停止，直到它接收到一个特定的信号。

为了发信号，需要使用一个称作信号量的特殊变量。

显然，P、V 操作的引入，克服了加锁操作的忙等待现象，提高了系统的效率。操作系统正是利用信号量的状态来对进程和资源进行管理的。

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?

P 操作相当于申请一个资源，得不到阻塞；V 操作相当于归还一个资源，如有等待该资源的进程，则唤醒。

S>0 时 S 表示可使用的资源数或表示可使用资源的进程数；

S=0 时 S 表示无资源可供使用或表示不允许进程再进入临界区；

S<0 时 S 表示等待使用资源的进程个数或表示等待进入临界区的进程个数。

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？（作业、课件例子）

（1）利用信号量实现**互斥**。为临界资源设置一个互斥信号量 mutex，初值为 1；每个进程中将临界区代码置于 P/V 原语间。

（2）用 P/V 实现进程**同步**时，同步信号量的初值与相应资源有关，也与 P/V 操作在程序代码中出现位置有关。

13 经典的 IPC 问题。

14 进程高级通信有哪些实现机制？

高级通信：是指进程采用系统提供的多种通信方式来实现通信。如消息缓冲、信箱、管道、共享主存区等。

发送进程和接收进程的消息通信方式：非阻塞发送，阻塞接收、**非阻塞发送，非阻塞接收**、阻塞发送，阻塞接收（双向通信）。

管程：共享变量、条件变量、操作条件变量的同步原语（外部模块）、内部函数（过程）

15 死锁产生的必要条件及解决死锁的方法

一组进程是死锁的，是指这一组中的每个进程都正在等待该组中的其他进程所占用的资源时，可能引起的一种错误现象。

必要条件：

(1)互斥条件。独占性的资源。

(2)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。

(3)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。

(4)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。

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。

解决办法：

① 鸵鸟算法。忽略死锁。

② 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。

③ 死锁的避免。是在资源的动态分配过程中，用某种方法（比如银行家算法）防止系统进入不安全状态，从而避免发生死锁。

④ 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。

1、故意终止一些进程 2、从当前占有者剥夺一部分资源给另一些进程，直至死锁解除 c

16 理解银行家算法的实质。能够利用银行家算法避免死锁。（计算题）

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。

功能：

（1）存储器分配：解决多道程序或多进程共享主存的问题

（2）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。

（3）存储器保护：防止故障程序破坏 OS 和其它信息

（4）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。

（5）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。

符号名字空间：源程序中的各种符号名的集合所限定的空间。如源程序中的数据和子程序通常是用符号名进行访问的。

逻辑地址空间：经编译连接后的目标代码所限定的空间。用地址码替换符号地址。相对地址，**逻辑地址**，虚地址。

存储空间：物理存储器中全部物理存储单元的集合所限定的空间。绝对地址，**物理地址**，实地址。

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。分为静态重定位和动态重定位。

静态：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。

特点：1）无硬件变换机构；2）为每个程序分配一个连续的存储区；3）在程序执行期间不能移动，主存利用率低；4）难以做到程序和数据的共享；5）用于单道批处理系统。

动态：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。多道批处理系统、分时系统

优点：1）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。2）程序不必占有连续的存储空间。 3）便于多用户共享存储器中的同一程序和数据

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。

通常存储器划分为两部分：操作系统占用区和用户进程占用区（或用户区）。存储器管理是指对用户区的管理。

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?

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。

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

根据作业的大小动态地划分分区，使分区的大小正好等于作业大小。各分区的大小是不定的；内存中分区的数目也是不定的。

（1）首次适应(first fit)法：要求空闲区表或空闲区链中的空闲区按地址从小到大排列。分配内存时，从起始地址最小的空闲区开始扫描，直到找到一个能满足其大小要求的空闲区为止。分一块给请求者，余下部分仍留在其中。

（2）最佳适应(best fit)法：存储分配程序要扫描所有空闲区，直到找到能满足进程需求且为最小的空闲区为止。

缺点：因为要查找所有的分区，所以比首次适应算法效率低。可能把主存划分得更小，出现很多无用的碎片。

改进：从小到大对空闲区排序。

（3）最坏适应(worst fit)法：要扫描所有的空闲区，直到找到满足进程要求且为最大的空闲区为止。一分为二，一部分分给进程，另一部分仍留在链表中。目的：使剩下的空闲区可用。

缺点：要扫描所有的空闲区；大空闲区的不断分割，可能满足不了大进程的要求。

改进：从大到小对空闲区排序，以提高查找速度。

地址重定位和存储器保护：

（1） **固定分区采用静态重定位，**进程运行时使用**主存物理地址 设置上、下界寄存器**来实现存储器保护

（2） **可变式分区采用动态重定位，**进程运行时 CPU 给**出逻辑地址，设置基址+限长寄存器**实现存储器保护

分区管理的优缺点：

优点：多进程共享主存、系统设计简单、存储器保护简单

缺点：有碎片，主存利用不充分；主存不扩充。

覆盖和交换技术

覆盖：是指同一主存区可以被不同的程序段重复使用。通常一个进程由若干个功能上相互独立的程序段组成，进程在一次运行时，也只用到其中的几段。让那些不会同时执行的程序段共用同一个主存区。覆盖段和覆盖区一一对应，系统提供覆盖管理程序，用户覆盖结构。

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：系统根据需要把主存中暂时不运行的进程中的部分或全部信息移到外存，而把外存中的进程移到主存，并使其投入运行。

时机：时间片用完或者进程要求扩充其占用的存储区而得不到满足

通常把外存分为文件区和交换区（存放从内存换出的进程）。交换主要是在进程之间进行，而覆盖则主要在同一个进程内进行。

交换技术的关键：设法减少每次交换的信息量，通常将进程的副本保留在外存，每次换出时，只需要换出修改过的信息。

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，使得小容量主存也可以运行多个进程，也可以使得各用户进程，在有限的时间内得到响应。

缺点：交换技术也是利用外存来逻辑的扩充主存，进程的大小也会受到实际主存容量的限制。

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？（存储方式，哪种存储器效率最高）

页表：系统为每个进程建立一张页面映像表，记录逻辑页与主存块的映射关系。

快表和联想存储器

管理内存的数据结构有两种：**存储分块表，位示图**；

7 什么是页式存储器的内零头?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？

在等长固定分区中，进程装入一个分区后，若这个分区还有没用的部分，则这个部分叫做内零头

可变分区时,可能会形成大量较小的，难以再分配的分区这样分区叫外零头

8 段式存储器管理与页式管理的主要区别是什么?

(1)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。

(2)页的大小固定；段的大小不固定。

(3)段式用二维地址空间；页式用一维地址空间。

(4)段允许动态扩充，便于存储保护和信息共享（**各进程的段表项指向共享段的物理地址**）。

(5)段可能产生主存碎片；页消除了碎片。

(6)段式管理便于实现动态链接，页式管理只能进行静态链接。

(7)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**。

其**容量由计算机的地址结构确定**。系统的指令地址部分能覆盖的地址域大于实际主存的容量。

物质基础：有一个大的 CPU 地址结构；采用多级存储结构；地址转换结构 MMU

实现过程：原理+取页置页置换

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。试说明它们的作用。

(1) 有效位（状态位）：用来指示某页是否在主存。为 1 表示该页在主存，完成正常的地址变换；为 0 表示该页不在主存，由硬件发出一个缺页中断，转操作系统进行缺页处理。

(2) 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。

(3) 访问位（引用位）：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?

1）根据当前指令的逻辑地址查页表的状态位。

2）状态位为 0，缺页中断。

3）操作系统处理缺页中断，寻找一个空闲的页框。

4）若有空闲页，则把从磁盘读入信息装入该页框。

5）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。

6）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、抖动现象。（习题）

(1) 最佳置换算法（OPT）：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。（无法实现）

(2) 先进先出置换算法(FIFO)：当淘汰一页时，选择在主存驻留时间最长的那一页。（抖动、Balady 异常）

(3) 最近最少使用的页面置换算法(LRU)：淘汰那些在最近一段时间里最少使用的一页。（复杂）

(4) 时钟页面置换算法：第二次机会算法。淘汰时，检查指针所指页。若引用位为 0，则用新页置换之，指针向前走一个位置。若引用位为 1，清 0，指针前进，直到找到引用位为 0 的页。

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?

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(thrashing)现象：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。

工作集模型法：设法记录每个进程的工作集，并确保在进程运行前将工作集调入主存，以大大减少进程的缺页率。

页式管理中应该考虑的问题：

1. **交换区的管理**
2. **页尺寸**
3. **页的共享：共享页锁在内存**
4. **多级页表的结构**
5. **写时复制技术（copy-on-Write）**

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。

多级页表结构：页表在内存不必连续存放。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

二级页表：**页目录索引（页目录表），页表索引（页表），页内字节索引（物理主存）**

写时复制的页面保护：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。所有未修改的页父子进程共享，子进程独享被复制的页。

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。（重要）

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。

段的动态链接和共享

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。包括文件控制块、文件体两部分。

文件系统是 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。。**由一组文件和一个文件目录结构组成。**

主要功能：

1）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。

2）实现**按名存取**。利用目录结构快速定位文件。

3）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。

4）提供一套使用方便、简单的操作命令。

5）保证文件信息的**安全**性。

6）便于文件的**共享**。

UNIX 系统中的文件分类：

(1)普通文件：通常的文件。

(2)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。**

(3)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（块特别文件），管道文件。

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。

引入文件系统的主要目的是使**用户实现按名存取文件**。

文件目录指记录文件的名字机器存放物理地址的一张映射表，表中包括了许多文件控制块（FCB），文件名，第一个物理块号等

其他管理控制信息。

3 文件的逻辑结构有几种形式？文件的存取方法？

文件的逻辑结构：1）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。

2）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：（1）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。

（2）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

目录结构：一级、二级、多级

4 文件的物理结构有几种组织形式?对于不同的组织形式，文件系统是如何进行管理的?

1）连续文件（顺序文件） 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系统文件。

2）链接文件 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。**索引表又叫做文件映射表可以克服链接文件不能随机存取的缺点**

3）索引文件 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。Unix 和 Linux 的 Ext2 采用多级索引结构。优点：文件可动态修改；随机、顺序存取。缺点：索引表的使用增加了存储空间的开销；降低了文件的存取速度。

4）索引顺序文件 Windows 的 NTFS 文件系统，MFT 主控文件表，采用静态索引结构。

5 DOS 文件卷的结构，DOS 系统的文件物理组织是什么？

MS-DOS 就是使用文件分配表（FAT）来分配和管理磁盘空间的。DOS 系统的文件采用链接结构。（索引顺序文件）

6 为什么要进行记录的组块和分解？

实际一个逻辑记录与物理块的大小是不相等的。当用户文件的逻辑记录远小于物理块大小时，一个逻辑记录存放在一个物理块总，将会造成极大的浪费。为此，可把**多个逻辑记录存放在一个物理块中，也允许一个逻辑记录跨块存放**。这样可以提高存储器的利用率。

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 （是一种最简单的方法）空白文件是一个连续未用的空闲盘块区。系统为所有这些空白文件建立一张表。每个空白文件占用一个表目。适合于文件的静态分配（连续文件的分配）。

空闲块链表

（1）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。

（2）空闲块成组链表

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

是适合文件静态分配和动态分配的最简单方法。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，便于分配和回收空间。

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?

多级目录优点：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。

文件名包含了从根目录开始的路径，因此只要在同一个目录下面不存在相同的文件名，系统就不会有文件重名。

只需要将各用户文件目录中的一个目录项指向同一个文件的存放地址，则各个用户都可以存取该文件，实现了共享。

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

创建(Create)文件 主要功能：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 主要功能：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 根据文件路径名找到目录项，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 释放文件在主存专门区域中的目录项，切断用户与文件的联系。若该目录项被修改过，则复制到磁盘。若文件作过某些修改，应将其写回辅存。

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 常用的文件存取控制方法：保护域和存取控制表 ACL

一个域（用户、进程、过程）是一组（对象、存取权限）偶对。即给出它能操作的对象和对每个对象的存取权限。

文件的存取控制信息，规定各类用户对文件的存取权限。防止核准的用户误用文件和为核准的用户存取文件。

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 理解存储器映射文件。

将文件映射到进程地址空间的一个区域，返回虚拟地址，仅当需要对文件存取时，才传输实际的数据。

12 文件共享与保护：

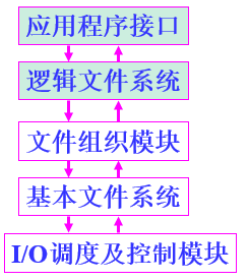
文件共享：允许多个用户共同使用同一文件。硬链接，符号链接文件（类 Unix）MS-DOS 和 Windows 绕道法

文件保护：防止自然灾害导致数据丢失，防止文件被无权使用的用户窃取。

（1）文件复制

（2）增设防护措施（防止病毒程序破坏文件。用户的鉴别----口令。物理鉴定。使用磁卡、指纹和签名等技术。 对文件进行加密。

13 文件系统的组织结构



14 文件的存储介质

FAT 文件卷：磁盘低级格式化（厂家完成）；硬盘分区（FDISK 命令）分区格式化（FORMAT 命令 制作文件系统）

硬盘主引导扇区：对每个硬盘，有一个硬盘主引导扇区 0 面 0 道 1 扇区。包括（1、硬盘主引导程序 2、硬盘分区表 3、引导扇区的有效标志或者分区扇区的结束标志）

FAT文件卷的组成

引导或保留扇区	文件分配表1（FAT1）	文件分配表2（FAT2备份）	根目录区	文件数据区

- 引导或保留扇区：或存引导代码或保留不用。
- FAT：存有整个文件存储空间的使用情况。
- 根目录区：记录文件或子目录在根目录中的占用情况。
- 文件数据区：存放系统文件、子目录文件、各种各样的应用程序、用户文件。

分区规范：一个硬盘有多个主分区，但最多只有一个扩展分区（可以划分成逻辑分区）

MS 的 OS 最多只让划分一个主分区 一个扩展分区 非 DOS 分区

存取设备、存取方法、物理结构之间的关系

存取设备	磁盘			磁带
物理结构	顺序结构	链接结构	索引结构	顺序结构
存取方法	直接/顺序	顺序	直接/顺序	顺序
文件长度	固定	可变/固定	可变/固定	固定

磁盘：寻道时间，旋转延迟时间，读写传输时间

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：网卡，调制解调器。

时钟：

I/O 体系结构：图 6.1 必背诵 P125

I/O 设备一般由机械和电子两部分构成，机械是设备本身，电子部分是设备控制器，设备控制器主要有两个作用：

① 解释从 IO 接口接收到的高级命令，强制设备执行特定的操作

② 转换和解释从设备接受电信号，修改状态寄存器

设备控制器包括：状态寄存器、控制寄存器、数据缓冲寄存器

2 常用的四种数据传输方式。

程序查询（polling）方式：CPU 忙等，串行工作。

中断（interrupt）方式：CPU/设备并行工作，使用最多。

直接存储器访问(DMA)方式：通常，CPU 控制地址总线，与主存交换数据，DMA 控制器取代 CPU，接管地址总线的控制权，直接控制内部数据缓冲区与主存之间的数据交换。整块数据的传输是在 DMA 控制下完成的。仅在开始和结束时才需 CPU 干预。

通道控制方式：是一种专用的 I/O 处理机，一个通道控制多台设备，进一步减轻 CPU 的负担。CPU、通道和 I/O 设备并行工作。

DMA 工作流程：CPU——（DMA 命令块 源、目的、n）——主存——DMA 控制器——挪用 CPU 工作周期传送一个字节的数据（恢复 CPU 对主存的控制权）——主存地址寄存器加 1 字节个数减 1

通道的类型：字节多路通道（分时执行多个通道程序，字节传输慢速设备）、选择通道（一个通道程序，一台设备传输，快）、数组多路通道（快且分时，先为一台执行再为另一台设备执行一条通道指令）

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？

虚拟设备，共享设备，独占设备。

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。常用**可共享的高速设备**来模拟**独占的慢速设备**。能有效提高独占型设备的利用率。 **Spooling 技术（缓输出技术）**是实现虚拟设备的具体技术。它利用可共享磁盘的一部分空间，模拟独占的输入/输出设备。以空间换时间

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.**中断处理程序**　①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。②CPU 响应中断，执行中断处理程序。③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。④中断返回被中断的进程，或转进程调度。

2.**设备驱动程序**　每个设备驱动程序处理一种类型设备。由一些与设备密切相关的代码组成。提供一些与文件类似的 API：open, close, read, write, control 等。是 OS 中唯一知道设备控制器的配置情况，如设置有多少个寄存器以及这些寄存器作用。 通常包含三部分功能：**①设备初始化。②启动设备传输数据的例程。③中断处理例程。**

3.**独立于设备的软件**　（1）实现所有设备都需要的功能，且向用户提供一个统一的接口。（2）设备命名**设备的符号名映射到正确的设备驱动程序。**（3）设备保护。（4）提供与设备无关的块尺寸。（5）缓冲技术（单双多缓冲 缓冲池）。（6）负责设备分配和调度（静态、动态分配、虚拟设备）。（7）出错处理。

4.**用户空间的 I/O 软件（I/O 库函数、 Spooling 技术）**

5　何为设备的独立性？

（设备独立性是指**用户**及用户程序**不受系统**配置的**设备类型和设备台号**的影响。**用户只是使用逻辑设备，具体的映射由操作系统完成。**）

同步异步 IO：

6　什么是 SPOOLING 技术？以输出为例，说明它的实现原理。

是实现虚拟设备的具体技术/。它利用可共享磁盘的一部分空间，模拟独占的输入/输出设备。以空间换时间。

Spooling 实际是一种缓冲技术。进程要打印时，系统并不为它分配打印机，而是把待打印的数据缓冲到一个独立的磁盘文件上，形成待打印文件队列。之后，Spooling 系统一次一个地将打印队列上的文件送打印机打印。这种技术又叫缓输出技术。

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？（计算题）

（寻道时间、旋转延迟时间和读/写传输时间）

9　常用的磁盘调度算法：

先来先服务：最简单，易实现，又公平合理。

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：读/写磁头在由磁盘的一端向另一端移动时，随时处理所到达磁道上的服务请求。（SCAN：扫描（电梯法）、C-SCAN：循环扫描、LOOK：查询、C-LOOK：循环查询）

10 **同步 IO**：进程发出 I/O 请求后阻塞等待，直到数据传输完成后被唤醒，之后才能访问被传输的数据。

异步 I/O：允许进程发出 I/O 请求后继续运行。将来 I/O 完成后的通知（ 调用者）方式：设置进程地址空间内的某个变量；通过触发信号或软件中断；进程执行流之外的某个回调函数。

对于不必进行缓冲读写的快速 IO，使用同步 IO；

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。

进程控制块，又称进程描述符（task_struct）：描述进程的数据结构。

thread_info：当前进程基本信息

mm_struct：指向进程的虚拟内存描述符

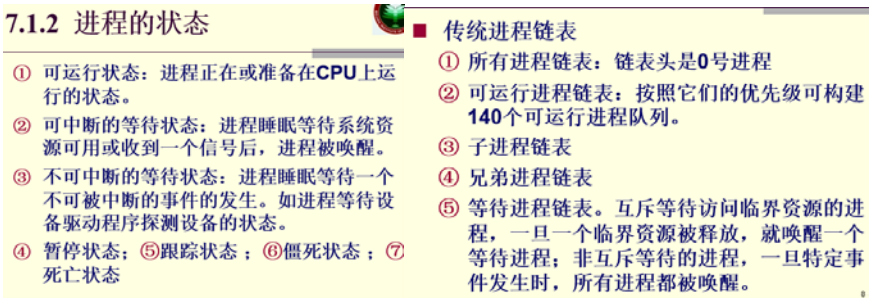
fs_struct：指向文件系统信息

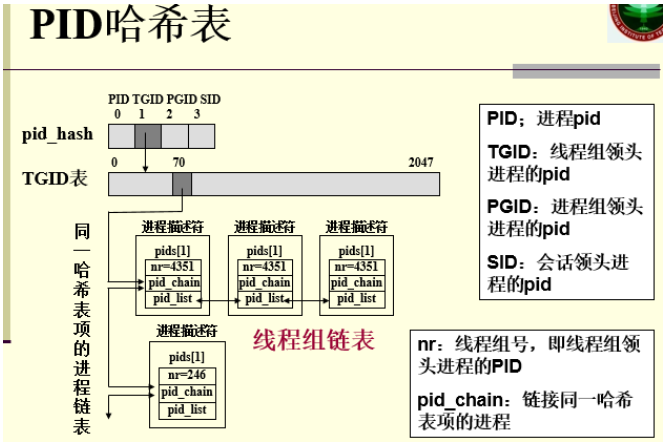
files_struct：指向该进程打开文件信息

signal_struct：所接收的信号

dentry：指向目录结构的指针

pid_t　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。

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。

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。

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。**之后大部分寄存器值存放在进程描述符的 thread_struct 字段（进程硬件上下文 P147）里，一小部分仍在核心栈中。**

进程硬件上下文存放在进程描述符的 thread_struct　thread 中。

第一步，切换页目录表以安装一个新的地址空间；

第二步，切换核心栈和硬件上下文。由 schedule()函数完成进程切换。

4 进程调度方式和算法。进程调度时机。

Linux2.6 系统采用**可抢先式的动态优先级调度方式**。其**内核是完全可重入的**。无论进程处于用户态还是核心态运行，都可能被抢占 CPU，从而使高优先级进程能及时被调度执行，不会被处于内核态运行的低优先级进程延迟。

Linux 系统的调度算法是基于进程过去行为的启发式算法，以确定进程应该被当作交互式进程还是批处理进程。

实时进程调度时机：（1）出现了更高优先级的实时进程。（2）进程执行了阻塞操作而进入睡眠状态。（3）进程停止运行或被杀死。（4）进程调用自愿放弃处理机。（5）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。

- **进程调度可分为：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　　（包括基本时间片和动态优先级）。**
- **实时进程的基本优先数为 1～99，而分时进程和批处理进程的基本优先数为 100～139。**

- 进程调度所涉及的数据结构：**每个处理机都有自己的可运行队列，存放在一维数组 runqueues 中。每个处理机对应数组中的一项。活动进程链表：没有用完自己的时间片。过期进程链表：已经用完了自己的时间片。当活动进程都过期后，过期进程才可运行。**

5 Linux 有很多内核线程，了解 0 号进程和 1 号进程的作用。

0 号进程就是一个内核线程，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。其进程描述符存放在 init_task 变量中。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。

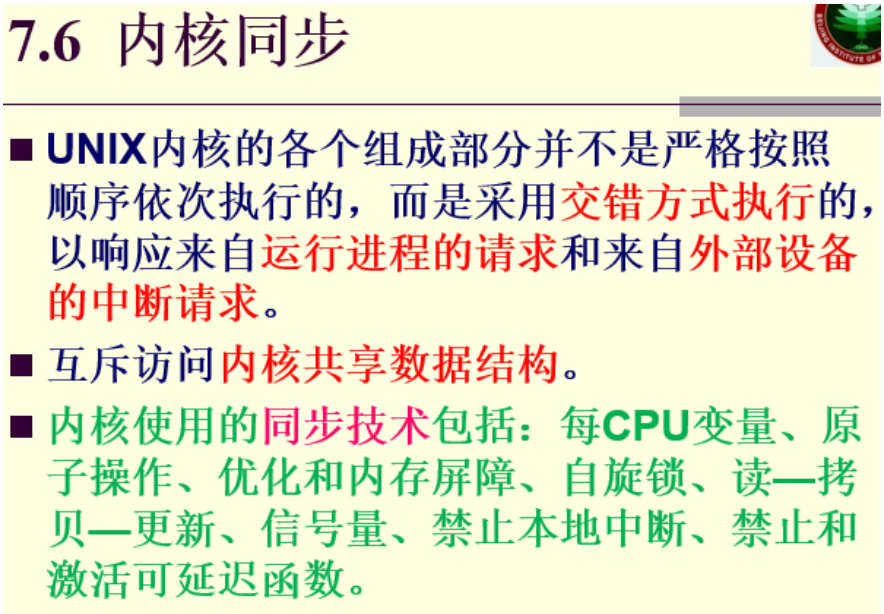
进程撤销：

- **exit()系统调用只终止某一个线程。**
- **exit_group()系统调用能终止整个线程组**

  父进程先结束的子进程会成为孤儿进程，系统会强迫所有的孤儿进程成为 init 进程的子进程。

  Init 进程在用 wait()类系统调用检查并终止子进程时，就会撤消所有僵死的子进程

6 内核同步



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？

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。

Linux 使用红黑树来管理虚存区域。

虚拟内存区域描述符的结构类型 vm_area_struct 中：vm_file：映射文件时指向文件对象

虚拟内存描述符的结构类型 mm_struct 中：pgd_t *pgd：指向页目录表

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。

页框描述符的结构类型 page。

管理区描述符的结构类型 zone。

采用伙伴系统(buddy system)管理连续的空闲内存页框，以解决外碎片问题。伙伴系统算法以页框为单位，适合于对大块内存的分配请求。

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 分配器把小内存区看作对象，slab 分配器对不再引用的对象只是释放但内容保留，以后再请求新对象时，就可直接使用而不需要重新初始化。

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。

32 位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。

虚地址分成 3 个域：页目录索引（前 10 位）、页表索引（中 10 位）和页内偏移（后 12 位）。

Linux 系统的页目录项和页表项的数据结构相同。页表项所包含的字段：

Present 标志：为 1，表示页（或页表）在内存；为 0，则不在内存。

页框物理地址(20 位)：页框大小为 4096，占去 12 位。20+12=32。

Accessed 标志：页框访问标志，为 1 表示访问过。

Dirty 标志：每当对一个页框进行写操作时就设置这个标志

6　请求调页。所缺的页可能存放的地方。

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。

7　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”来描述这些块组本身的结构信息，同时将超级块和所有的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。盘块位图，索引节点位图。

超级块存放整个文件卷的资源管理信息。索引节点存放文件的管理控制信息。

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，可以提高文件目录的检索速度。

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。

3 Linux 文件系统的索引节点中，索引表划分成几级？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。（文件最大长度计算）

最初的 12 个元素是直接索引项，给出文件最初的 12 个逻辑块号对应的物理块号。

索引 12 是一次间接索引块，是一个存放盘块号的一维数组。对应的文件逻辑块号从 12 到（b/4）+11，b 是盘块大小，每个逻辑块号占 4B。

索引 13 是二次间接索引块，对应的文件逻辑块号从 b/4+12 到 (b/4)2+(b/4)+11。

索引 14 是三次间接索引块，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？

5 Linux 文件系统如何管理空闲存储空间？

文件的数据块和其索引节点尽量在同一个块组中。文件和它的目录项尽量在同一个块组中。父目录和子目录尽量在同一个块组中。每个文件的数据块尽量连续存放。

6 VFS 通用文件模型中的四个主要对象？

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：打开的文件对应的…。

目录项对象：dentry (directory entry)

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？

```
Struct tast_struct{
      struct fs_struct *fs; 指向文件系统信息
      struct files_struct *files; 指向进程打开文件信息
…}
struct fs_struct {
      atomic_t count; 共享该结构的进程数
      struct dentry *root, *pwd;  每个进程都有自己的当前工作目录和根目录，通过这两个目录与文件系统进行交互。
      struct vfsmount *rootmnt; 根目录下安装的文件系统对象
      struct vfsmount *pwdmnt; 当前目录下安装的文件系统对象
…}
struct files_struct {
      struct file **fd;       指向文件对象指针数组
      struct file *fd_array[ ]; 文件对象指针数组
…}
struct file {     文件对象
      struct list_head f_list;          文件对象链表
      struct dentry *f_dentry;       指向目录项对象
      atomic_t f_count;          该对象的引用计数
      loff_t  f_pos;                  文件的当前读写位置
```

```
        struct file_operations *f_op;
        struct address_space *f_mapping;   映射
……}
struct dentry {   目录项对象
        atomic_t  d_count;          引用计数
        struct inode *d_inode;    指向文件的 inode
        struct dentry *d_parent; 指向父目录项对象
        struct list_head d_alias; 属于同一 inode 的 dentry 链表
        struct dentry_operations *d_op;   方法
……}
struct inode {    索引节点对象
        struct list_head i_sb_list; 同一超级块的索引节点链表
        unsigned long  i_ino;      磁盘索引节点号
        atomic_t  i_count;           该对象的引用计数
        nlink_t  i_nlink;                硬链接计数
        struct inode-operations  *i_op;
        struct file_operations  *i_fop;
……}
```

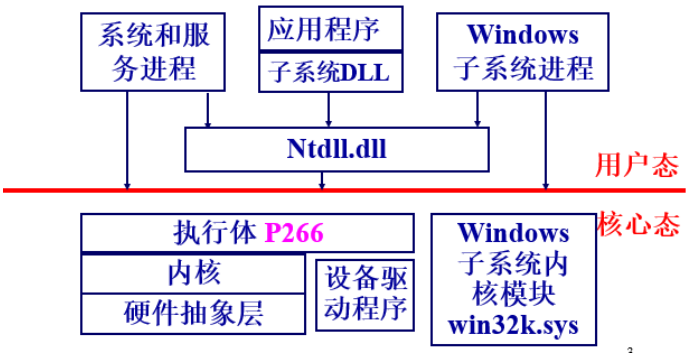
8 一个文件在使用与不用时各占用系统哪些资源？

9 安装表的作用是什么？

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？（核心态系统组件：执行体、内核、设备驱动程序、硬件抽象层、窗口和图形）**



Ntdll.dll：对于内核提供的每一个系统服务，该 DLL 都提供一个以 Nt 作为前缀的存根函数

2. **Windows 系统文件**

1. Ntoskrnl.exe：执行体和内核
2. Hal.dll：硬件抽象层
3. Ntdll.dll：对于内核提供的每一个系统服务，该DLL都提供一个以Nt作为前缀的存根函数。另外，还提供系统级支持函数。
4. Win32k.sys：Windows子系统的内核部分
5. Kernel32.dll，Advapi32.dll，User32.dll，Gdi32.dll：Windows子系统DLL
6. NTFS.sys：ntfs驱动程序
7. 设备驱动程序：可动态加载的模块(.sys)

Windows 系统进程

- idle进程：每个CPU一个线程。
- system进程：包含大多数内核系统线程

1. smss.exe：会话管理器。创建环境变量和启动csrss.exe和winlogon.exe。建立会话Session0
2. csrss.exe：Windows子系统进程
3. winlogon.exe：用户登录进程
4. services.exe：系统服务管理器。系统服务是一些特殊的进程。系统有很多功能组件是以服务的方式实现的，如事件日志、任务调度器和各种网络组件等。
5. svchost.exe：系统提供的通用服务宿主进程
6. lsass.exe：本地安全认证子系统进程
7. Explorer.exe：shell进程

3. **硬件抽象层 HAL 的作用是什么？**

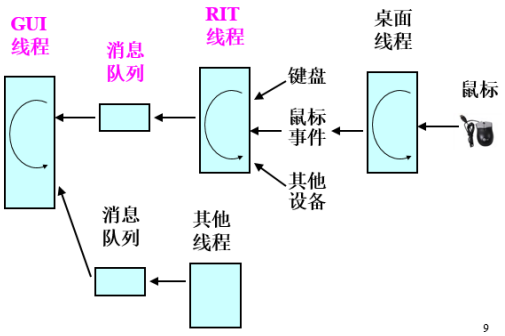
隐藏各种与硬件有关的细节。使内核、设备驱动程序和执行体免受特殊硬件平台差异的影响。系统可移植性好。

4. 环境子系统

Windows、POSIX、OS/2。即在同一个内核基础上配以不同的外围软件 win32k.sys 移到内核里。

5. Windows 子系统内置了 7 个窗口类：按钮、组合框、编辑框、列表框、多文档界面中的子窗口、滚动条、静态文本。

6.



9

7. Windows2000/XP 划分为两层，上层为执行体、下层为内核

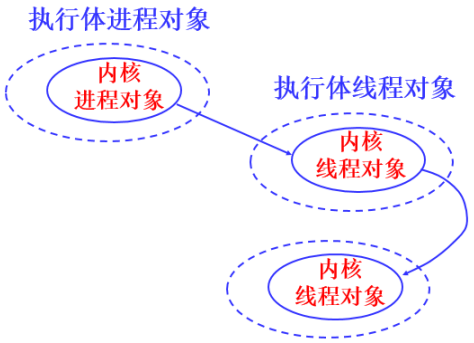
执行体包括：可供用户态调用的系统服务函数 NTDLL.DLL，仅供核心态内部使用的函数、各功能组件函数

内核主要提供以下功能：线程安排和调度、中断和异常调度、多处理机同步、提供执行体使用的一组例程和基本对象

8.

Windows 内核结构

- ntoskrnl.exe：执行体、内核
- 执行体对象绝大多数封装了一个或多个内核对象。执行体组件：P266
- 内核对象是由内核实现的一个初级对象集，对用户态代码不可见，仅供执行体使用。



9.

内核对象

- 控制对象，用于控制内核操作，但不影响线程调度。包括：APC、DPC、中断对象。
- 调度程序对象（dispatcher object）：事件、互斥体、信号量、内核进程P284下、内核线程P286上。实现同步。

（线程终止时处于有信号状态。进程终止时处于有信号状态。）

I/O请求中断处理

1. 当设备中断产生时，中断处理例程ISR被调用，它停留在设备上的时间只够捕获住该设备状态并停止设备中断，之后产生一个DPC。
2. 当IRQL降低到DPC级别以下时，DPC中断就产生，DPC例程被调度，完成此中断处理。
3. 同步I/O，直接返回发起者线程；异步I/O，在发起者线程中插入一个APC对象以便通知I/O完成。
4. 之后需调用IopCompleteRequest在进程地址空间将数据从系统缓冲区复制到用户缓冲区。

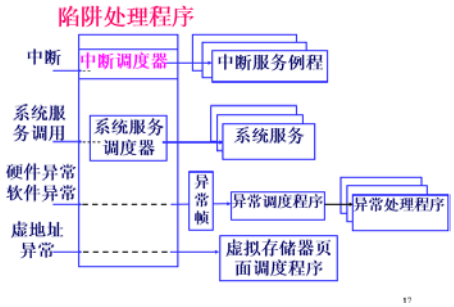
10. Windows 系统组件的基本机制包括：

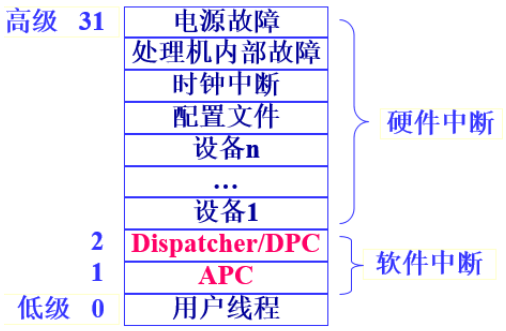
陷阱调度、

执行体对象管理器、

同步（自旋锁、内核调度程序对象）、

本地过程调用（LPC）。





陷阱调度是属于内核的功能。包括中断、DPC 、APC 、异常调度、系统服务调度。中断异步，异常同步，系统调用被视为异常。

执行体对象管理器。

同步。包括自旋锁、内核调度程序对象。

本地过程调用 LPC。服务器进程创建一个 LPC 连接端口对象，然后在该端口上监听客户连接请求。类似 socket 编程。消息传递。

11. 理解：延迟过程调用 DPC，异步过程调用 APC

DPC：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。

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**。

Windows 32 个中断请求优先级 IRQL

12. Windows 中有哪些对象，都有什么作用？

（两种类型对象：执行体对象和内核对象。

执行体对象：进程管理器、主存管理器、I/O 子系统等。

内核对象：由内核实现的一个初级对象的集合，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。）

13. 在多处理机系统中，提供了哪些同步和互斥机制？

内核对象同步：内核引入自旋锁实现多处理机互斥机制。

执行体同步：内核以内核对象的形式给执行体提供其他的同步机构，称为“调度程序对象”，包括进程、线程、事件、信号量、互斥体、可等待的定时器及文件等同步对象。每个同步对象都有“有信号”或“无信号”两种状态。）

14. 线程如何实现等待一个同步对象的操作？

WaitForSingleObject( )\WaitForMultipleObjects( ) 等待一个\多个同步对象变为有信号状态

- **对象管理器提供**

  WaitForSingleObject( ) ，使线程与调度程序对象同步。

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。

2. 创建线程：CreateProcess( ); 创建线程：CreateThread( )

3. 线程的 7 种状态，及其解释。

① 就绪状态(ready)

② 备用状态(standby)。已选好处理机，正等待描述表切换，以便进入运行状态。

③ 运行状态(Running)

④ 等待状态(waiting)

⑤ 传输状态(transition)。核心栈被调到外存的就绪态。

⑥ 终止状态(terminated)

⑦ 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 个就绪线程队列、32 位线程就绪队列位图、32 位处理机空闲位图。

线程调度程序的数据结构：负责记录各线程的状态。

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. 线程优先级的提升时机。

系统会提升线程的优先级，以改善性能。

① I/O 操作完成后的线程。

② 信号量或事件等待结束的线程。

③ 前台进程中的线程完成一个等待操作。

④ 由于窗口活动而唤醒图形用户接口线程。

⑤ 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 管理进程私有空间采用两种数据结构：虚拟地址描述符、区域对象，这两种结构有什么作用？

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。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(section object)被称为文件映射对象，是一个可被多个进程共享的存储区。利用区域对象映射磁盘上的文件（包

括页文件）。然后访问这个文件就象访问主存中的一个大数组，而不需要使用文件的读/写操作。作用：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 **虚拟内存区域：空闲的、保留的、提交的**

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。

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。**

4 **管理内存的数据结构：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，页框的状态转换图 16.9。**

5 **原型页表项，区域对象的页表。虚拟页式中，采用原型页表实现多进程共享页。**

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”的页表来记录这些被共享的页框。

当一个区域对象被创建时，这些原型页表项“按段”同时被创建。

6 **Windows 采用的页替换策略是什么？**

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“时钟页面置换算法”)。

第 17 章　Windows 文件系统

1 **Windows 所支持的文件系统类型有哪些？**

支持 FAT12、FAT16 和 FAT32 文件系统。

2 **虚拟簇号和逻辑簇号的概念。**

3 **NTFS 卷的结构，主控文件表 MFT 的作用。**

分区引导扇区、主控文件表区 MFT、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；

4 **NTFS 文件的物理结构：索引顺序结构。**

5 **管理文件的目录结构采用 B-树。**