操作系统 A 部分　　数学系 05 级使用

第一章概论

1．简述现代计算机系统的组成及其层次结构

答：组成分硬件和软件；层次结构 应用程序-〉系统程序-〉操作系统-〉硬件。

2．操作系统的资源分类

答：硬件 处理器、寄存器、存储器、各种 I/O 设施和设备（输入型，输出型，存储型）。

信息资源 程序、数据

3．什么是操作系统,配置操作系统的主要目标

答：操作系统是管理系统资源、控制程序执行、改善人机界面、提供各种服务、合理组织计算机工作流程、为用户有效使用计算机提供良好运行环境的系统软件。

目标：方便用户使用、扩大机器功能、管理系统资源、提高系统效率、构筑开发环境。

7．什么是系统调用

答：系统调用是为了扩充机器功能、增强系统能力、方便用户使用而在系统中建立的过程；

功能分类 进程和作业管理、文件操作、设备管理、内存管理、信息维护、通信。

8．什么是系统程序

答：又称标准程序或实用程序，是 os 的高层功能，借助于系统调用实现。

分类 文件管理、状态信息、程序设计语言支持、程序的装入和执行支持、通信、其他软件工具。

9．系统调用实现原理

答：实现系统调用功能的机制成为陷入或异常处理机制

a、 编写系统调用处理程序-〉

b、 设计一张系统调用入口地址表…….

c、 陷入处理机制（参数传递）

10．系统调用与过程调用的主要区别

答：a、调用形式不同　　前者按功能号调用；

后者适用一般调用指令；

b、被调用代码位置不同　　前者属于动态调用，它的处理代码在操作系统中；

后者属于静态调用，调用程序和被调代码在同一程序内，使目标代码的一部分。过程改动后，必须重新编译连接；

c、提供方式不同，　　前者由操作系统提供；

后者由编译系统提供，不同编译系统提供的过程可以不同；

d、调用的实现不同　　前者程序通过中断机构实现，要从用户态-〉核心态，在管理状态运行；

后者程序使用一般机器指令调用过程，在用户态运行；

===➔程序执行系统调用安全性好。

11．系统调用与库函数关系

答：在一些程序设计语言中，提供一些与各系统调用对应的库函数。因此，在高级语言中，应用程序可通过对应的库函数来使用系统调用；库函数的目的是隐藏访管指令的细节，使系统调用更像过程调用。一般说来，库函数属于用户程序，而非系统程序。

12、假脱机-〉ch5 spooling 系统

15、什么是多道程序

答：多道程序是允许多个作业同时进入一个计算机系统的内存储器并启动进行交替计算的方法；

特点 提高了 CPU 的利用率

提高了内存和 I/O 设备的利用率

改进了系统的吞吐率

充分发挥了系统的并行性

16、实现多道程序的三个基本问题

答：存储保护与程序浮动

处理器的管理与分配

系统资源的管理和调度

29、什么是虚拟性，怎样实现虚拟性

答：虚拟性是把物理上的一个实体变为逻辑上的多个对应物，或把物理上的多个实体变为逻辑上的一个对应物。

方法 物理计算机资源通过多重化和共享技术变为多个虚拟机

用一类物理设备来模拟另一类物理设备　分时地使用一类物理设备

31、什么是操作系统的内核

答：内核是提供支持系统运行的基本功能和基本操作的一组程序模块，分为微内核和单内核

功能　中断处理&&短程调度（分配处理器）&&原语管理

内核的执行有以下属性　内核是由中断驱动的

内核的执行是连续的

内核在屏蔽中断状态下执行

内核可以使用特权指令

内核+ 裸机 = 虚拟机

虚拟机特性　没有中断

虚拟机为每个进程提供了一台虚拟处理器

它为进程提供了功能较强的指令系统

补充：类程 管理私有资源，对类程的调用表示对私有资源的操作，它仅能被进程及起源于同一进程的其它类程或管程嵌套调用链所调用。

32、微内核操作系统的优缺点

答：操作系统的绝大多数功能由用户态进程来实现，内核主要起信息验证、交换的作用。是现代操作系统的特征之一。

优点　一致性接口　可扩充性　可移植性　可靠性

支持分布式系统　支持面向对象的操作系统

缺点　所有进程只能通过微内核相互通信，在通信频繁的系统中无法提供高效率。

34、层次式操作系统优缺点

答：把操作系统划分为内核和若干模块（或进程），这些模块（或进程）按功能的调用次序排列成若干层次，各层之间只能是单向依赖或单向调用关系。

优点　把整体问题局部化

有利于系统的维护和扩充（增加、修改或替换一个层次不影响其它层次）

缺点　分层单向依赖，必须建立模块间的通信机制，通信开销较大。

35、虚拟机结构操作系统优缺点

答：所谓虚拟是指把一个物理上的实体变为若干个逻辑上的对应物。前者是实际存在的，而后者是虚的。在构造 os 时，把 os 分成若干层，每层完成特定的功能，从而形成一个虚拟机。下层的虚拟机为上层的虚拟机提供服务。这样逐次扩充以完成操作系统的功能。

37、从执行方式看，操作系统的各种实现模型

答：　非进程内核模型

OS 功能（函数）在用户进程内执行的模型

OS 功能（函数）作为独立进程执行的模型

补充：OS 功能在用户进程内执行的模型

当发生一次中断或系统调用后，处理器状态将被置成内核模式，控制权从用户进程手中被剥夺并传递给操作系统例行程序。此时，发生了模式切换，模式上下文（现场）信息被保存。但是进程上下文切换并没有发生，操作系统仍在该用户进程中执行，提供单独的内核堆栈用于管理进程在核心态下执行时的调用和返回。操作系统的例行程序和数据放在共享地址空间，且被所有用户进程共享。

当 OS 例程完成了工作之后，如果应该让当前程序继续运行的话，做一次模式切换恢复执行原先被中断的用户进程。

****

第二章 处理器管理

1、PSW 的主要作用

答：Program Status Word 程序状态字，用于区别不同的处理器工作状态（处于何种状态，能否执行特权指令）

主要作用是方便地实现程序状态的保护和恢复。

3、为什么要设置多个 CPU 状态

答：处理器上的不同执行程序对资源和机器指令有不同的使用权限。

一般设有管态、目态，管态可以执行全部指令访问所有资源，且可以改变处理器状态；目态下只能执行非特权指令。

5、为什么要分特权指令和非特权指令

答：指令系统，是机器指令的集合，分为 数据处理类，转移类，数据传送类，I/O 类，移位与字符串类。

用户程序执行一些有关资源管理的指令很容易导致系统混乱，造成系统或用户信息的破坏。因此，用户程序只能使用指令系统的一个子集。

6、硬件如何发现中断，如何处理中断

答：a、中断是程序执行过程中，当发生某个事件时，中止 CPU 上现行程序的运行，引出处理该事件的服务程序执行的过程。

b、中断装置：发现中断源并产生中断的硬件（中断逻辑线路，中断寄存器）

发现多个中断源时，根据规定的优先级，先后提出中断请求；

保护现场（即运行程序的执行上下文）

启动处理中断的中断处理程序，处理器状态从目态切换到管态；

中断寄存器记录中断事件，中断字的每一位对应一个中断事件；

c、中断处理程序

寻找中断处理程序的向量地址表

保护第二现场（未被硬件保护的一些必需的处理状态）

识别各个中断源，分析产生中断的原因

处理发生的中断事件

恢复正常操作

7、中断性质分类中断

答：a、强迫性中断事件　机器故障中断事件

程序性中断事件，如溢出、越位等（异常）

外部中断事件，如时钟的定时中断

输入输出中断事件，如传输结束

b、自愿性中断事件　正在运行的程序期待的事件

原因：执行了一条访管指令

8、中断事件来源分类中断

答：a、外中断　处理器和主存之外的中断

b、内中断　又称异常，处理器和主存内部的中断

异常不能被屏蔽，一旦出现应立即响应并加以处理

9、处理程序性中断时，什么情况可转用户中断续元处理

答：a、纯属程序错误而又难以克服的事件（非法使用特权指令），报告操作员并请求干预

b、其它，转交用户程序自行处理　on<条件><中断续元入口>

11、为什么要中断分级

答：分级　按中断请求的轻重缓急的程度预定的顺序成为中断的优先级

为了确定中断装置响应中断的顺序

15、中断在操作系统中的重要性及其主要作用

答：重要性 用户程序请求操作系统服务，实现并行工作，处理突发事件，满足实时要求

作用　处理突发事件。

16、时钟中断在操作系统中的作用

答：时钟是操作系统进行调度工作的重要工具（绝对时钟、间隔时钟），如让分时进程作时间片轮转，让实时进程定时发出或接受控制信号；系统定时唤醒或阻塞一个进程，对用户进程进行记帐。

17、中断屏蔽的作用

答：中断屏蔽可以禁止主机对某类中断的响应。

不可被屏蔽的中断有：计算机的断电中断、自愿性访管中断。

补充：禁止中断嵌套是指当一个中断发生时，应当处理完当前中断后再处理另外一个中断，而不应当在中断处理过程中再转去处理更高级别的中断。可以在中断处理程序中进行中断屏蔽，以保证中断处理的完整性。

18、操作系统如何处理多重中断

答：多重中断事件：同时出现中断或者同时发现中断，产生中断的嵌套。

a、同一中断类型的不同中断源，使用同一个中断处理程序按照预定的次序分别处理

b、不同类型的中断

禁止发生中断（屏蔽其它中断） 缺点：没有考虑相对优先级和时间限制

定义中断优先级

响应并进行中断处理，适用情况：运行中断处理例行程序时，出现程序性中断源

补充：某个异常事件在处理过程中又发生了新的异常事件（如处理溢出过程中又产生了溢出），可以再转改种中断程序吗

答：不能

因为在处理异常时系统处于和心态，这时又出现了改种异常事件，此时属于同级中断，所以不能再转入该种中断处理程

序。处理方法：在该种异常事件处理结束后，判断中断寄存器中是否有该种异常，如果有则立即报告错误。

21、解释 windows 的中断、异常和陷阱

答：中断和异常把处理器转向正常控制流之外的代码

中断是异步事件，可能随时发生，与处理器正在执行的内容无关，主要由 I/O 设备、处理器时钟或定时器产生可以启用或禁用。

异常是同步事件，它是某一个特定指令执行的结果。内核将系统服务视为异常

陷阱是指处理意外事件的一种硬件机制，相当于前面的中断响应和处理机构。

25、linux 底半处理

答：引入底半处理原因：发生中断时，系统把中断送到相应的设备驱动程序去处理（和心态），此时系统需要关闭中断，资源非常低。

原理 将中断处理分为两部分，底半处理和上半处理。将耗时较多的工作放在开中断的状态下处理，即底半处理，以提高系统对中断的处理效率。

26、什么是进程，为什么要引入进程

答：进程 理论角度 对正在运行的程序活动规律的抽象

实现角度 一种数据结构

目的 刻画系统的动态性，发挥系统的并行性，提高资源利用率，描述程序动态执行过程。

解决共享性，正确描述程序的执行状态，引入可再入程序和可再用程序的概念

补充：可再入程序（纯代码，自身不改变），可再用程序（自身修改）

进程是一个可并发执行的具有独立功能的程序关于某个数据集合的一次执行过程。

属性 结构性 程序块，数据块，进程控制块

共享性 多个进程可共享同一程序

动态性 与程序不同，程序作为一种系统资源永久存在。

独立性 系统中进行资源分配和保护的基本单位

并发性

28、进程基本状态及其切换

答：运行态 进程占有处理器正在运行

就绪态 进程具备运行条件，等待系统分配处理器以便运行

等待态 不具备运行条件，等待某事件的完成

运行----等待 等待使用资源或者某事件发生，如等待外设传输

等待----就绪 资源得到满足或某事件已经发生

运行----就绪 运行时间片到，或出现有更高优先权的进程

就绪----运行 CPU 空闲时调度选中一个就绪进程执行

29、新建态和终止态的主要作用

答：新建态对应于进程刚刚被创建的状态，此时进程并没有被提交执行，而是在等待操作系统完成创建进程的必要操作。

终止态 一个进程 到达自然结束点

or 出现了无法克服的错误

or 被操作系统终结

or 被其它有终止权的进程终止

进入终止态的进程不再执行，依然保留在操作系统中等待善后。

30、引起创建一个进程的主要事件

答：4 个事件 提交一个批处理作业

在终端上一个交互式作业登陆

操作系统创建一个服务进程

存在的进程创建新的进程

32、什么是进程的挂起状态，其主要特征

答：系统资源不能够满足进程运行的要求，就必须挂起某些进程，对换到磁盘镜像区中，释放它占有的某些资源，暂时不参与低级调度。

主要特征 a、该进程不能立即被执行

b、挂起进程可能会等待一个事件，但所等待的事件是独立于挂起条件的，事件结束并不能导致进程具备执行条件。

c、进程进入挂起状态时由于操作系统、父进程或进程本身阻止它的运行。

d、结束进程挂起状态的命令只能通过操作系统或父进程发出。

33、什么情况下会产生挂起等待态和挂起就绪态

答：等待态-〉挂起等待态　当前不存在就绪进程

挂起等待态-〉挂起就绪态　引起进程等待的事件发生

35、进程控制块

答：Process Control Block,PCB

操作系统用于记录和刻画进程状态及有关信息的数据结构，也是 OS 掌握进程的唯一资料结构，是 OS 控制和管理进程的主要依据。

从进程管理角度　进程标识　　通信角度　消息队列首指针

进程状态　　访问小子队列互斥信号量

进程优先级　　消息计数

队列指针

中断处理角度　上下文信息&&中断源及类型

40、模式切换与进程切换

答：当从用户态转到核心态时，内核要保留足够的信息以便后来能返回到用户态并让进程从它的断点继续执行。用户态和核心态的切换时 CPU 模式的改变，而不是进程上下文切换。

当中断发生时，暂时中断正在执行的用户进程，把进程从用户态切换到内核状态，去执行 OS 例行程序以获得服务，这就是一次模式切换。

步骤：　a、保存被中断进程处理器现场信息

b、根据中断号设置程序计数器

c、把用户状态切换到核心状态，以便执行中断处理程序

45、UNIX SVR4 进程管理的特点

答：采用基于用户进程的运行模型，OS 功能在用户进程的环境中执行，需要在用户模式和内核模式间切换。

unix 允许两类进程：用户进程（在用户模式下执行用户程序）系统进程（在内核模式下运行，完成系统的一些重要功能）

系统调用、中断和异常将引起模式切换。

48、操作系统引入线程的原因

答：在传统的 OS 中，进程是系统资源分配的基本单位，也是 CPU 调度的基本单位。

但是　a、进程时空的开销大，频繁的进程调度将耗费大量 CPU 时间，要为每个进程分配存储空间限制了操作系统中进程个数。

b、进程通信的代价大，每次通信均要涉及通信进程之间或通信进程与操作系统之间的信息传递。

c、进程的并发性粒度较粗，并发度不高，过多的进程切换和通信延迟使得细粒度的并发得不偿失。

d、不适合并行计算和分布式并行计算的要求，对于多 CPU 和分布式的计算机来说，进程之间大量频繁的通信和切换会大大降低并行度。

e、不适合 C/S 计算的要求。对于 C/S 结构来说，需要频繁 I/O 操作并同时大量计算的服务器进程很难体现效率

因此引入线程，减少了程序并发执行时所付出的时空开销，使得并发粒度更细，并发性更好。

50、叙述多线程环境中，进程和线程的定义

答：进程是操作系统中进行保护和分配资源的基本单位，

它具有　一个虚拟地址空间，用来容纳进程的镜像

对处理器、其它（通信的）进程、文件和 I/O 资源等的有控制有保护的访问。

线程是 OS 中能够独立执行的实体（控制流），是处理器调度和分派的基本单位。它是进程的组成部分，每个进程内允许包含多个并发执行的实体（控制流），这就是多线程。

同一进程中的所有线程共享进程获得的主存空间和资源，但不拥有资源

线程具有　线程执行状态

当线程不运行时，有一个受保护的线程上下文，用来储存现场信息

一个执行堆栈

一个容纳局部变量的主存储区

52、比较进程和线程

答：

调度	进程--切换	vs	线程--若同一进程内有就绪线程，调度运行
并发性	并发性差		并发性好
拥有资源	yes		no
系统开销	多		少

具体来说，进程具有独立的虚地址空间，以进程为单位进行任务调度，系统必须交换地址空间，切换时间长；而在同一进程中的多线程共享同一地址空间，因而可快速切换线程。

对多个进程的管理，系统开销大，如响应客户请求建立一个新的服务进程的服务器应用中，创建的开销比较显著。而线程的创建、终止，系统的开销小得多。

线程对所有全局数据均可自由访问，而进程通信则相当复杂，必须借助通信机制、消息缓冲、管道机制等设施

并发程度提高

多线程合用进程地址空间，而不同进程独占地址空间。

53、叙述 ULT 和 KLT 的区别

答：KLT 线程管理的所有工作由操作系统内核来做，任何应用都可被程序设计成多个线程，当提交给操作系统执行时，内核为它创建一个进程和一个线程。内核为整个进程和每个线程维护现场信息。

在内核空间建立和维护 PCB 和 TCB，内核的调度是在线程的基础上进行的。

优点　多 CPU，内核可同时调度同一进程中的多个线程执行

某一线程阻塞，则调度其它线程运行

内核线程仅有很小的数据结构和堆栈，KLT 切换快

缺点　应用程序线程在用户态运行，而线程调度和管理在内核实现。在同一进程中，控制权从一线程传到另一线程需要“用户态—内核态—用户态“模式切换，系统开销大。

ULT 线程管理的所有工作由应用程序来做，在用户空间实现。

用户级多线程由用户空间运行的线程库来实现，任何应用程序均需通过线程库进行程序设计，再与线程库连接后运行来实现多线程

线程库是一个 ULT 管理的例行程序包，线程库是线程的运行支撑环境

此时，内核按照进程为单位进行调度。

优点　线程切换不需要内核特权模式，节省内核的宝贵资源和模式切换的开销

线程库的调度算法与操作系统的低级算法无关

缺点　线程执行系统调用时，阻塞同进程的所有线程（需要用护套 jacketing 技术解决）

多线程应用不能利用多重处理的优点

56、挂起状态与线程

答：由于线程不适资源的拥有单位，挂起状态对线程是没有意义的。由挂起操作引起的状态是进程级状态，此进程的所有线程共享了进程的地址空间，作相同操作。

59、叙述 solaris 中的进程与线程概念

答：在 solaris 中，

进程 process　通常的 UNIX 进程，它包含用户的地址空间、堆栈和 PCB

用户级线程 ULT 通过线程库在用户地址空间中实现，对操作系统来讲是不可见的，ULT 是应用程序并行机制的接口

轻量进程 LWP light weight process

每个 LWP 可看作 ULT 和 KLT 之间的映射，每个 LWP 支持多个 ULT，并映射到一个 KLT 上。LWP 与 KLT 对应，由内核独立调度，可以在多个处理器上并行执行

内核级线程 KLT 是能被调度和指派到处理器上运行的基本实体

补充：作业控制块 JCB job control block

批作业进入系统时，由 spooling 系统建立的，它是作业存在于系统的标志。它的主要内容是从作业说明书（用户利用 JCL 编写的一个控制作业执行的程序）中获得

包括　作业情况　资源需求 资源使用情况 作业控制 作业类型

72、处理器调度分类

答：a、高级调度（long-term scheduling）

按照系统预定的调度测量决定把后备队列作业中的部分满足其资源要求的作业调入主存，为它们创建进程，分配所需资源。为作业做好运行前的工作并启动它们运行，作业完成后做好善后工作。

对于分时系统来说，高级调度决定　是否接受一个终端用户的连接

一个交互式作业能否被计算机系统接纳并构成进程

一个新建态的进程是否能够立即加入就绪进程队列

b、中级调度（mediun level scheduling）又称平衡负载调度，中程调度

它决定主存中所能容纳的进程数，这些进程将允许参与竞争处理器和有关资源，而有些暂时不能运行的进程则被调出主存，这时这个进程处于挂起状态。

当进程具备了运行条件，且主存中又有空闲区域时，再由中级调度决定把一部分这样的进程重新调回主存工作。

作用　短期平滑和调整系统负荷。

c、低级调度（low level scheduling）又称进程调度或线程调度

它的主要功能是按照某种原则决定就绪队列中的哪个进程或内核级线程能获得处理器，并将 CPU 出让给它进行工作。

低级调度中执行 CPU 分配的程序称为 dispatcher 分派程序，它是操作系统最为核心的部分，执行十分频繁。

方式　第一类 剥夺方式 preemptive scheduling

当一个进程在处理器上执行时，系统可以根据规定的原则剥夺分配给它的 CPU，而把 CPU 分配给其它进程使用。

常用的剥夺原则　高优先级进程或线程可以剥夺低优先级进程或线程

当运行时间片用完后被剥夺处理器

第二类　非剥夺方式 nonpreemptive scheduling

一旦某个进程或线程开始执行后便不再出让处理器,除非该进程或线程运行结束或发生了某个事件不能继续执行。

73、叙述衡量一个处理器调度算法好坏的主要标准

答：　资源利用率

响应时间　分时系统和实时系统衡量调度性能的一个重要指标

周转时间　批处理系统衡量调度性能好坏的一个重要指标

吞吐率

公平性

补充：进程调度指标　CPU 利用率

等待时间　进程在就绪状态中的等待时间

响应时间

I/O 设备利用率　以 I/O 为主的进程优先运行，提高 CPU 与 I/O 间的并行度

时空代价

74、叙述作业调度和低级调度的关系

答：首先看作业和进程间的主要关系：作业是任务实体，进程是完成任务的执行实体

作业调度属于高级调度层次，它选中了一个作业且把它装入主存时就为该作业创建了一个用户进程，这些进程将在低级调度的控制下占有 CPU 运行。

作业调度与低级调度的配合能实现多道程序作业的同时执行。

76、解释

答：作业周转时间　批处理用户从作业提交给系统开始，到作业完成为止的时间间隔

作业带权周转时间　带权周转时间=周转时间/需要运行的时间

响应时间　交互式进程从提交一个请求（命令）到接收到响应之间的时间间隔

吞吐率　单位时间内处理的作业数

补充：JCL job control language，作业控制语言

系统提供给用户描述其作业控制意图的工具。

81、响应比最高者优先算法

答：highest response ratio first 介乎 FCFS 和 SJF 之间的一种折中的策略

响应比=作业响应时间/作业估计计算时间

83、优先权调度是否会导致进程饥饿状态

答：不会导致饥饿。一个进程在队列中等待 CPU 的时间越长，那么在它再次获得调度时的优先数就越高。

89、叙述典型的实时调度算法

答：a、单比率调度算法

事先为每个进程分配一个与事件发生频率成正比的优先数，运行时调度程序总是调度优先数最高的就绪进程，并采取抢占式分配策略。

b、限期调度算法

就绪队列按照截至期限排序

c、最少裕度法　laxity

裕度= 截止时间—（就绪时间+ 计算时间）

第三章 并发进程（编程题另外再写）

2、叙述并发程序设计的特点

答：进程的并发性（concurrency）是指一组进程的执行在时间上是重叠的。

并发的实质是一个处理器在几个进程间的多路复用，是对优先的物理资源强制行使多用户共享，消除计算机部件之间的互等现象，以提高系统的资源利用率。

特点　并发性 & 共享性 & 制约性 & 交互性

优点　a、若为单 CPU 系统，可有效利用资源，让 CPU 和 I/O 设备、I/O 设备和 I/O 设备之间同时工作，充分发挥机器部件的并行能力。

b、若为多 CPU 系统，可让进程在不同 CPU 上物理地并行工作，从而加快计算速度

c、简化了程序设计任务

缺点　程序的运行环境不再是封闭的，程序结果可能是不确定的，计算过程具有不可再现性。

4、解释并发性与并行性

答：并发性是指进程的并发性，两个或多个事件在同一时间间隔内发生，执行在时间上是重叠的。

并行性是指硬件的并行性，parallel 两个或多个事件在同一时刻进行。

9、说明进程的互斥和同步访问的异同

答：　进程互斥指若干个进程要使用同一共享资源时，任何时刻最多允许一个进程去使用，其它要使用的进程必须等待，直到占有资源的进程释放该资源；

进程同步指两个以上进程基于某个条件来协调它们的活动。一个进程的执行依赖于另一个协作进程的消息或信号。当一个进程没有得到来自于另一个进程的消息或信号时则需等待，直到消息或信号到达才被唤醒。

进程互斥是一种特殊的进程同步关系。

补充：快表

在 MMU（内存管理单元）中设置一个高速缓冲存储器（TLB，translation look-aside buffer）。在 TLB 中的页表成为快表。

补充：为了让用户互斥地进入临界区，可以把整个临界区实现成为不可中断的过程，即让用户具有屏蔽所有中断的能力。

但是这样做有缺点：

用户进程进入临界区时屏蔽所有中断，应当也包括系统程序。但系统发出的中断也被屏蔽，则会引起错误。因为系统外中断往往与当前运行的程序无关，却可能是一些重要的硬件中断，如电源故障等。故不可盲目屏蔽所有中断；又或者当时发生故障中断的中断源恰好是该临界资源，则更应及时处理。

17、管程及其属性

答：基本思路　把分散在各进程中的临界区集中起来进行管理，并把系统中的共享资源用数据结构抽象地表示出来。

代表共享资源的数据结构及在其上操作的一组过程就构成了管程。

属性　共享性　管程中的移出过程可被所有要求调用管程的过程的进程所共享

安全性　管程的局部变量只能由该管程的进程存取，不允许进程或其它管程来直接存取。一个管程的过程也不应该存取任何非局部于它的变量。

互斥性

19、比较管程和进程

答：

管程	进程
公用数据结构	私有数据结构
把共享变量上的同步操作集中	临界区分散在每个进程中
为管理共享资源而建立的	为占有系统资源和实现系统并发性而引入的
管程是被欲使用共享资源的进程所调用的	进程之间能并行工作，并发性是其固有特性
管程和调用它的进程不能并行工作	
管程是语言或操作系统的成分，不必创建或撤销	进程有生命周期，由创建而产生，至撤销便灭亡

21、为什么要有消息传递机制

答：系统中的交互式进程通过信号量及有关操作可以实现进程的互斥与同步。在用信号量解决生产者问题时，不是单靠信号量而是要另外引入有界缓冲来存放产品，既不方便，局限性也大。

有时进程间可能需要交换更多的信息，这种大量的信息传递可使用 message passing。由于 OS 提供的这类机制隐蔽了许多实现细节，通过消息传递机制就能够简化程序编制的复杂性，方便易用。

23、简述消息缓冲通信机制的实现思想

答：基本思想 由操作系统统一管理一组用于通信的消息缓冲存储区，每一个消息缓冲存储区可存放一个消息（信件）。

当一个进程要发送消息时，先在自己的消息发送区生成待发送的消息（包括接受进程名，消息长度，正文），然后向系统申请一个消息缓冲区把消息从发送区复制到消息缓冲区中，在复制过程中系统将接受进程名换成发送进程名，

以便接受者识别 。随后该消息缓冲区被挂到接受消息的进程的消息队列上，供接受者在需要的时候从消息队列摘下并复制大消息接受区中使用，同时释放消息缓冲区。

24、通过管道机制实现进程间通信

答：pipeline 是连接读写进程的一个特殊文件，允许进程按先进先出方式传输数据，也能使进程同步执行操作。

管道和消息队列的区别：管道中的消息是无界的，它存在于外存；消息队列是位于内存的。

一个进程正在使用某个管道写入或读出数据时，另一个进程就必须等待；

发送者和接受者双方必须能够知道对方是否存在；

发送信息和接受信息之间一定要实现正确的同步关系；

进程在关闭管道的读出或写入端时，应唤醒等待写或读此管道的进程。

27、进程的低级通信工具和高级通信工具

答：IPC inter-process communication

高低级区分：　　低　进程间控制信息的交换

　　高　进程间大批数据的交换

低级：　信号（singal）通信机制

　　信号量及其原语操作（PV、读写锁、管程）控制的共享存储区（shared memory）通信机制

交换的信息量少且效率低下，仅适用于集中式操作系统

高级：　管道（pipeline）提供的共享文件通信机制

　　通道（I/O 处理机）是实现 I/O 操作的硬件装置，通道对管道的实现提供硬件支持。

　　信箱和发信/收信原语的消息传递通信机制

适用于 集中式操作系统& 分布式操作系统

29、死锁产生的条件

答：mutual exclusion　　互斥条件

hold and wait　　占有和等待条件

no preemption　　不剥夺条件

circular wait　　循环等待条件

30、死锁防止策略

答：a、静态分配策略

破坏占有和等待条件，但是严重降低了资源利用率。

b、层次分配策略

阻止循环等待条件的出现

资源被分成多个层次，一个进程得到某一层的一个资源，只能在申请较高一层的资源：当一个进程要释放某层的一个资源时，必须先释放所占用的较高层的资源；当另一个进程获得了某一层的一个资源后，它想再申请该层中的另一个资源，必须先释放在该层中已占资源。

c、按序分配策略

把系统的所有资源安排一个顺序，按顺序给每个资源一个编号，规定每个进程申请两个以上资源时，总是先申请编号小的再申请编号大的资源。这样，在进程集合中总存在某个进程，它占有了已申请资源最大的资源。因而，它无权申请其他资源。当它运行结束后，就可以释放占用的全部资源。

按序分配通过破坏死锁的循环等待条件而防止死锁。

31、银行家算法及其基本思想

答：约束条件　每个客户必须预先说明自己所要求的最大资金量

　　每个客户每次提出部分资金量申请和获得分配

　　如果银行家满足客户对资金的最大需求量，那么客户在资金运作后，应在有限时间内全部归还银行。

****

第四章 存储管理

1、简述存储管理的基本功能

答：负责管理主存储器　主存空间的分配和去配

　　地址转换和存储保护

　　主存空间的共享

　　主存空间的扩充

补充：解决大作业和小内存矛盾的途径

答：a、覆盖技术

由用户把一个程序划分为若干个功能相对独立的程序段，并根据程序的逻辑结构让不会同时执行的程序段共享同一块内存区。程序运行时依需要把程序段调入覆盖区。

b、虚拟技术　部分装入、部分对换

3、逻辑地址空间与物理地址空间

答：把用户目标程序使用的地址单元成为逻辑地址（相对地址），一个作业的目标程序的逻辑地址集合称为该作业的逻辑地址空间。

把主存中的实际存储单元成为物理地址（绝对地址），物理地址的总体构成了用户程序实际运行的物理地址空间。

物理地址空间是由存储器地址总线扫描出来的空间。大小取决于主存容量。

4、地址转换（重定位）

答：把程序和数据的逻辑地址转换为物理地址。

用两种方式 a、静态重定位　在作业装入时由作业装入程序实现地址转换

b、动态重定位　在程序执行过程中，CPU 访问程序和数据之前实现地址转换，必须借助于硬件的地址转换机构

5、分区存储管理中的分配策略

答：　基本思想：给进入主存的用户作业划分一块连续存储区域

固定分区（fixed partition）存储管理/定长分区/静态分区模式

静态地把可分配的主存分割成若干个连续区域，每个区域位置固定，每个分区在任何时刻只装入一道程序执行

优点：　解决单道程序运行在并发环境下不能与 CPU 速度很好匹配的问题

解决了单道程序运行主存空间利用率低的问题

缺点：　预先规定了分区大小，不方便

主存利用率不高

作业运行中要求动态扩充主存困难

共享程序和数据难以实现

限制了多道运行的程序数

可变分区（variable partition）存储管理/变长分区模式

按照作业的大小划分分区，划分时间、大小、位置都是动态的。

补充：常用的可变分区算法

答：a、最先适应 first fit　从未分配区表头顺序查找

b、下次适应 next fit　从上次扫描结束处查找

c、最优适应 best fit　扫描整个未分配区表

d、最坏适应 worst fit 挑一个最大的空闲区

e、快速适应 quick fit 为经常用到的长度的空闲区设立单独的空闲区链表

补充：程序的局部性

答：principle of locality

a、程序中只有少量分支和过程调用，大都是顺序执行的；

b、往往包含若干个循环；

c、很少会出现连续不断的过程调用序列；

虚拟存储器是基于程序局部性原理上的一种假想的，而不是物理存在的存储器。

11、请求分段虚拟存储管理的实现原理

答：在作业执行中访问某段时，由硬件的地址转换机构查段表。若该段在主存，则按分段式存储管理的办法进行地址转换得到绝对地址；若该段不再内存，则硬件发出一个缺页中断。操作系统处理这个中断时，查找主存分配表，找出一个足够大的连续区域容纳该分段。如果找不到足够大的连续区域，则检查空闲区的总合：移动，将该段装入内存 || 调出数个分段到辅存，将该分段装入主存。

12、分页虚拟存储管理器中有哪些页面淘汰算法（中科大，1998）

答：理想算法 Belady 算法/最佳替代算法 optimal

a、随机页面替换算法

b、FIFO　低开销的页面替换算法，适用于具有线性顺序特性的程序

c、LRU least recently used 最近最少用页面替换算法，算法的操作复杂，代价高。

模拟：　NRU not recently used，最近没有使用页面替换

老化算法

LFU least frequently used，最不常用页面替换

d、第二次机会页面替换算法 second chance

e、时钟页面替换算法 clock policy

采用循环队列机制构造页面队列　一个页面首次装入主存时，其引用位置 0；

在主存中的任何一个页面被访问时，其引用位置 1；

遇 1 清 0 条过，遇 0 淘汰，指针推进一步；

f、改进的时钟页面替换算法

考虑了淘汰已修改的页面的情形

补充：对于任意给定的驻留集尺寸，在什么样的引用串情况下，FIFO 与 LRU 替换算法一样

答：若发生页面故障时被替换的页均一样　FIFO 替换最早进入主存的页

LRU 替换上次访问距当前最远的页

===〉当出现页故障时，最先进入主存的页仅被访问一次

故“访问串中所有页号均不同，若相同则必须排列在一起”。

13、比较分页式存储管理和分段式存储管理（清华，1999）

答：

	分页	分段
目的：	提高主存空间利用率	满足用户（程序员）编成和使用上的要求
	缺点 得到的是一维地址结构的可装配模块，但页面与源程序无逻辑关系难以实现对源程序已模块为单位进行分配、共享和保护	
方式：	信息的物理单位，与源程序逻辑结构无关，用户不可见 页长由系统决定，页面只能以页大小的整数倍地址开始。 源程序经连接装配后变成一维地址	分段式信息的逻辑单位，由源程序的逻辑结构所决定，用户可见； 段长可根据用户需要来规定，段起始地址可以从任何主存地址开始， 源程序（段号、段内位移）经连接装配后仍保持二维地址结构。

14、给出几种存储保护方法

答：分页管理提供 地址越界保护，即由地址变换机构中的页表长度值和所要访问的逻辑地址相比较完成

通过页表中的访问控制信息对内存信息提供保护。

分段管理提供 存取控制保护法

地址越界保护法

利用段表寄存器中的段表长度与逻辑地址中的段号比较，若段号越界则产生越界中断；再利用段表项中的段长与逻辑地址中的段内位移进行比较，若段内位移大于段长，产生越界中断。

不过在允许段内动态增长的系统中，允许越界，不过应设置相应的增补位）

15、存储管理中的碎片

答：内零头（内部碎片）：若存储单元长度为 n，该块存储的作业长度为 m，则（n-m）为内零头；

外零头（外部碎片）：若存储单元长度为 n，在该系统所采用的调度算法下，较长时间内无法选出一道长度不超过该块的作业

固定分区		内& 外
可变分区	！	内& 外
页式虚拟分区		内&！外
段式虚拟分区	！	内& 外

补充：动态链接

当程序运行到需要调用某一模块时再去链接。对于未使用的模块，就可以不必链接。

采用段式内存分配方法可以实现这种技术。

21、页式存储器共享

答：分页存储管理在实现共享时，必须区分数据共享和程序共享

实现数据共享时，可允许不同的作业对共享的数据页用不同的页号，只要让各自页表中的有关表目指向共享的数据信息块。

实现程序共享时，由于页式存储结构要求逻辑地址空间是连续的，所以程序运行前它们的页号是确定的。

可再入代码（纯代码） 允许多个进程同时访问的代码，不允许任何进程对其进行修改。

23、叙述段页式存储器的优缺点

答：段式存储 是基于用户程序结构的存储管理技术，有利于模块化程序设计，便于段的扩充、动态链接、共享和保护。但往

往会生成段间碎片浪费存储空间。

页式存储 是基于系统存储结构的存储管理技术，存储利用率高，便于系统管理。但不易实现存储共享、保护和动态扩充。

补充：IPT inverted page table 反置页表

IPT 维护了一个页表的反置页表，它为内存的每一个物理块建立一个页表项并按照块号排序，该表的每个表项包含正在访问该页框的进程标识号、特征位和 hash 链指针等，用来完成内存页框到访问进程的页号，即物理地址到逻辑地址的转换。

****

第五章 设备管理

1、叙述设备管理器的基本功能

答：基本功能 外围设备中断处理
缓冲区管理
外围设备的登记和使用情况跟踪、分配、去配
外围设备驱动调度

提高系统效率 虚拟设备及其实现

2、简述各种 I/O 控制方式及其优缺点

答：按照 I/O 控制器功能的强弱，以及与 CPU 之间联系方式的不同分类

a、询问方式/程序直接控制方式 查询指令----查询设备是否就绪
传送指令----当设备就绪时，执行数据交换
转移指令----当设备未就绪时，执行转移指令转向查询指令继续查询

缺点： CPU 在反复查询过程中，浪费了宝贵的 CPU 时间
CPU 参与数据的传递工作，不能执行原程序
CPU 和 I/O 设备串行工作

b、中断方式
中断机构引入后，外围设备有了反映其状态的能力

缺点 输入输出操作直接由 CPU 控制，每传送一个字符或一个字，都要发生一次中断，消耗大量 CPU 时间

c、DMA 方式
direct memory access 直接主存存取
主存和 I/O 设备之间有一条数据通路，在主存和 I/O 设备之间成块地传送数据过程中，不需要 CPU 干预，由 DMA 直接执行完成。

特点：外围设备在硬件支持下直接与内存交换成批数据而不需要 CPU 干预，地址总线、数据总线及相关控制信号线均与 CPU 共用。当 DMA 时，使用窃取总线控制权的方法，DMA 控制器接管总线，控制外设与内存间成批交换数据。当 DMA 传送的数据完成后发出一个中断，由 CPU 相应中断并回收控制权。

d、通道方式/输入输出处理器（又称 I/O 处理机）
DMA 中，每发出一次 I/O 指令，只能读写一个数据块。
采用通道技术主要解决了输入输出操作的独立性和各部件（设备与 CPU）工作的并行性。

分类 字节多路通道 ：连接大量慢速外围设备
选择通道 ：连接磁带和磁盘快速设备，同一时刻只能为一台设备服务，一个输入输出请求完成后才选择其它设备
数组多路通道

补充：DMA 方式与中断方式主要区别（国防科大，2001）

答：a、中断方式在每个数据完成后中断 CPU
DMA 在一批数据传完后中断 CPU

b、中断方式的数据传送是在中断处理时，由 CPU 完成
DMA 方式则是在 DMA 控制器控制下完成

补充：DMA 和通道的区别

答：a、DMA 要求 CPU 执行设备驱动程序启动设备，并做好传送数据的有关准备工作

b、通道完全是一个相对独立的 I/O 控制系统，仅当 CPU 发出 I/O 启动命令后，它便接收控制，完成全部 I/O 操作。

7、叙述 I/O 系统层次及其功能

答：各层次 用户进程 ：进行 I/O 调用、格式化 I/O、spooling
设备无关软件 ：命名、保护、阻塞、缓冲、分配
设备驱动程序 ：建立设备寄存器、检查状态

中断处理程序 ：当 I/O 结束时，唤醒驱动程序
硬件 ：执行 I/O 操作

9、通道命令与通道程序

答：channel command word（CCW， 通道命令）是通道从主存取出并控制 I/O 设备执行 I/O 操作的命令字，用 CCW 编写的程序称为通道程序（由多条通道命令组成，每次启动可完成复杂的 I/O 控制）。

12、缓冲技术及其基本思想

答：引入缓冲的目的 改善中央处理器与外围设备之间速度不配的问题
协调逻辑记录大小与物理记录大小不一致
提高 CPU 和 I/O 设备的并行性
减少 I/O 对 CPU 的中断次数和放宽对 CPU 中断相应时间的要求

基本思想 进程执行写操作输出数据时，向系统申请一个缓冲区，若为顺序写请求，则不断把数据填到缓冲区，直到装满。此后，进程继续计算，系统将缓冲区内容写到 I/O 设备上。

进程执行输入数据操作时，向系统申请一个缓冲区，系统将一个物理记录的内容读到缓冲区，根据进程的要求，把当前需要的逻辑记录从缓冲区中选出并传送给进程。

在输入数据时，仅当缓冲区空而进程又要从中读取数据时，它才被迫等待

在输出数据时，只有在系统还来不及腾空缓冲区而进程又要写数据时，它才需要等待。

14、驱动调度

答：按照一定次序执行要求访问的诸请求

作用：减少为若干个 I/O 请求服务所需的总时间，提高系统效率。

常用移臂调度算法：

a、电梯调度算法 elevator algorithm
选择沿臂的移动方向最近的柱面

b、最短时间查找优先 shortest seek time first algorithm
执行查找时间最短的磁盘请求

c、扫描 scan algorithm
沿一个方向移动，到最后一个柱面，再向相反方向移动过来

d、分步扫描 N-steps scan algorithm
将 I/O 请求分组，每组不超过 N 个，每次选一组进行扫描

e、循环扫描 circular scan algorithm
一次扫描完成后，从 0 号柱面重复进行。

补充：磁盘不不仅用于存放文件，还可作为主存的延伸，即提供虚拟管理。在虚存的设计思想中，主存作为实际的物理空间，仅存放目前较为活跃的程序部分，其它不活跃的部分暂存于辅存，等待调度程序在主、辅存间进行交换调度。

补充：叙述 RAID

答： redundant array of independent disks
一种 大容量外存系统，用一组较小容量、独立、可并行工作的磁盘组成存储阵，借助冗余存储技术，实现数据的多组织分布存储，从而能够并行开展单个或多个 I/O 请求，提高系统性能和效率。

补充：提高磁盘 I/O 速度的方法

答： 为磁盘设置高速缓存（内容由操作系统控制）
提前读
延迟写
虚拟盘（内容完全由用户控制）

23、叙述 spooling 系统

答：spooling 系统 = spooling 技术 + 速度匹配技术

spooling 技术是用一类物理设备模拟另一类物理设备的技术，它使独占使用的设备变成可共享的设备。

补充：分四层讨论 I/O 软件的功能

答： 设备中断处理程序 分析中断类型并作出相应处理，检查和修改进程状态
设备驱动程序（包括所有与设备相关的代码） 设备控制，把用户提交的逻辑 I/O 转化为物理 I/O 的启动和执行
与设备无关的 I/O 软件 执行适用于所有设备的常用 I/O 功能，并向用户层提供一个一致的接口。
用户层 I/O 软件 包括 spooling 程序、在用户空间运行的 I/O 库例程、调用、格式化、假脱机

31、设备独立性

答：设备独立性是指用户程序独立于所使用的具体物理设备，即用户只使用逻辑设备名。

设备独立（无关）性体现

a、从程序设计的角度看待设备，各种设备所体现的接口应该都是一致的，程序中可使用相同的命令读出不同设备上的数据，也可以用相同的命令将输出数据送到各种不同的设备上。不同设备之间的差异有操作系统处理，对程序加以屏蔽。

b、在操作系统管理设备和相关操作时，对所有的设备都采用统一的方式进行。一般采用层次式、模块化的思想来实现设备管理子系统。

补充：联想存储器（各命名有所不同）associate memory（lookaside 缓冲器）

虚拟地址直接映射到物理地址

与普通存储器不同，它不是通过地址引用。

它得到一个搜索值，同时它搜索所有元素来查找一个相符的索引值。分页系统使用小型的高速联想存储器来改进性能。

****

第六章 文件管理

1、叙述概念

答：卷是物理介质的存储单位

块是存储介质上连续信息所组成的一个区域，也叫做物理记录

记录是信息的单位

文件是由文件名字标识的一组信息的集合

2、记录的成组和分解操作及其优缺点

答：若干个逻辑记录合并成一组，写入一个块叫做记录成组。每块中逻辑记录的个数成为块因子。

当存储介质上的一个物理记录读进缓冲区后，把逻辑记录从块中分离出来的操作叫做记录的分解。

由于顺序文件是顺序存取得，可采用成组和分解的操作加速文件的输入输出。

优点： 节省存储空间

减少输入输出操作次数，提高系统效率

缺点： 需要软件进行成组和分解的额外操作

需要能容纳最大块长的输入输出缓冲区

补充：文件系统提供给用户程序的一组系统调用，通过这些系统调用用户能获得文件系统的各种服务。

10、windows 文件系统特点之考点

答：文件加密，EFS encrypting file system

补充：使用文件系统时，通常要显式进行文件的 open、close 操作，叙述目的、能否取消（北大，1992）

答：a、显式的 open 操作完成文件的打开功能，将基本文件目录中的内容读入用户的文件表中，并在系统活动文件表中记录文件打开次数。

显式的 close 操作完成文件的关闭操作。撤销用户的活动文件表中的相应的表项，改变系统活动文件表中的文件打开次数信息。如果需要，还要将被改动的文件目录信息写回基本文件目录中。

b、可以取消二者。此后系统在进行文件操作之前需判断文件是否打开。若未打开，则应自动完成文件的打开功能，以建立用户与文件间的关系。同时，在系统结束时还应自动关闭所有被打开的文件，更新系统的基本文件目录。

c、取消后市的文件的读写操作变得复杂。因为在每次读写前都要判断文件是否已打开，此外，系统在结束时也要做一些额外的工作，以完成 close 应该完成的操作。

19、文件共享的分类和实现思想

答：文件共享是指不同用户（进程）共同使用同一个文件

a、文件的静态共享：提高文件资源的利用率，节省文件的物理存储空间

文件链接－－一个文件同属于多个目录，但仅有一处物理存储。

只允许链接到文件

b、文件的动态共享

多进程动态共享使用　　系统中不同的用户进程或同一用户的不同进程并发地访问同一文件。这种关系只有当用户进程存在时才可能出现。

c、符号链接

通过路径名间接访问

第七章　操作系统的安全与保护

1、叙述计算机系统的可靠性和安全性之间的联系与区别

答：可靠性是指硬件系统正常持续运行的程度，目标为反故障
安全性是指不因人为疏漏或者蓄谋作案而导致信息资源被泄漏、篡改和破坏，目标为反泄密
可靠性是基础，安全性更为复杂。

3、叙述操作系统安全性的主要内容

答：安全策略
安全模型
安全机制　认证机制　authentication
授权机制　authorization
加密机制　encryption
审计机制　audit

4、计算机网络系统四项安全要求

答：机密性 confidentiality
完整性 integrity
可用性 availability
真实性 authenticity

21、策略与机制

答：策略规定要达到的特定目标，系统地安全策略制定了对本组织人员和非本组织人员资源的共享方式
机制是完成任务和特定目标的方法，系统提供用于强制执行安全策却得特定步骤和工具
优点：留有灵活性，策略发生变化时，整个系统变化小

30、简述操作系统地安全保护技术

答：状态隔离 对计算机系统设置不同工作状态
运行在管态下的程序比在目态下的程序有更多的访问权
限制用户使用容易造成系统混乱的那些及其指令，达到保护系统程序或其它用户程序的目的
空间隔离 为不同作业分配不同的地址空间，避免相互干扰
每个用户进程的内存空间可以通过虚拟存储技术来实现内存保护。
隔离技术能保证系统程序和用户程序的安全性

37、试说明 DES 加密解密过程

答：decrypt (key1 , encrypt (key , plain text)) = plain text

补充：什么是 RPC

答：remote procedure call 远程过程调用
允许不同计算机上的进程使用简单的过程调用和返回结果的方式进行交互

补充：如何实现进程迁移，如何处理已打开的文件

答：将系统中已迁移进程撤销，在目标系统中建立一个相同的新进程
所迁移的是进程映像，包括进程控制快、程序、数据和栈。此外，被迁移的进程于其它进程之间的关联应作相应修改。
对于已经打开的文件：　法 a、将已打开的文件随进程一起迁移；
法 b、仅当迁移后的进程又提出对该文件的访问要求时，再进行迁移。

****

书后简答题到此结束

没有做的题目是我认为不需要掌握（对于初试）的内容。

做题时间：2008 年 4 月 2 日 8:00-12:20，13:40-18:00，19:00-23:10
2008 年 4 月 3 日 9:00-13:00，15:00-19:00，20:20-22:30

所有内容来自高亮考研 OS 笔记