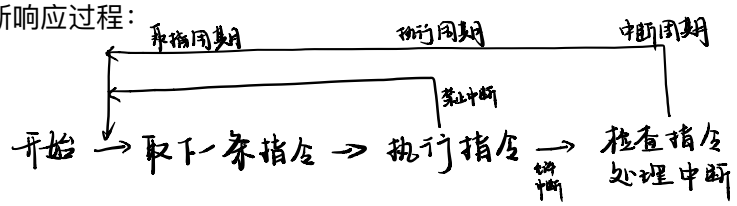
第一章

1.操作系统：资源的管理者；向用户提供各种服务；对硬件机器的扩展
2.操作系统的特征：并发、共享、虚拟、随机

第二章

1.x86支持的四个特权级别：
特权环（由高到低）：R0 R1 R2 R3 。R0相当于内核态，R3相当于用户态
R1:设备驱动程序，I/O处理。R2:受保护的共享的代码（如语言编译环境）
2.用户态到和心态唯一途径：中断/异常/陷入机制
3.中断响应过程：



4.中断向量：中断处理程序入口地址和程序运行时所需的处理机状态字
5.I/O操作出现错误的话，首先需要重新执行失败的I/O操作，重试次数有上线，达到时系统将判定硬件故障。
6.四种类型门描述符：
任务门
中断门：通过中断门后系统会自动禁止中断
陷阱门：通过陷阱门后系统不会自动禁止中断
调用门
7.应用程序可以直接进行系统调用，但一般情况下都是利用C函数库和API接口间接地进行系统调用。在操作系统内核提供了许多内核函数，这些内核函数经封装提供给了C函数库或API接口。系统调用对内核而言，内核函数是系统调用的处理程序，而处理程序经过封装在C函数库或API接口中提供给用户来使用。函数库/API接口中也有不是系统调用的函数。
8.实现用户程序的参数传递给内核：
由陷入指令自带参数、通过通用寄存器传递参数、在内存中开辟专用堆栈区来传递参数

第三章

1.并发：同一时间间隔
2.进程：每个拥有独立的地址空间（最重要的）
3.PCB是系统感知进程存在的唯一标志，进程与PCB一一对应
4.PCB包括：进程描述信息、进程控制信息、所拥有的资源和使用情况、CPU现场信息
5.exec（）：用一段新的程序代码覆盖原来的地址空间
6.fork（）：以一次一页的方式复制父进程地址空间。从父进程处继承共享资源。
对子进程返回标识符0，对父进程返回标识符的PID
7.进程更能准备刻画并发，而程序不能。
程序是静态的，进程是动态的。
进程是有生命周期的，有诞生有消亡，是短暂的；而程序是相对长久的。
一个程序可以对应多个进程。
进程具有创建其他进程的功能。

8.进程映像IMAGE：对进程执行活动全过程的静态描述（在一瞬间进程的快照）
由地址进程空间内容、硬件寄存器内容及与该进程相关的内核数据结构、内核栈组成
用户相关：进程地址空间（代码段、数据段、堆和栈、共享库...）
寄存器相关：程序计数器、指令寄存器、程序状态寄存器、栈指针、通用寄存器的值
内核相关：静态部分：PCB及各种资源数据结构
动态部分：内核栈（不同进程在进入内核后使用不同的内核栈）
9.线程有自己的栈和栈指针
10.用户级线程：在用户空间建立线程库，内核管理的还是进程
11.可重入程序：可被多个进程同时调用的程序，具有下列性质：它是纯代码的，即在执行过程中自身不改变，调用它的进程应该提供数据区。

第四章

1.如果没有就绪进程，系统会安排一个系统空闲进程或者idle进程
2.进程切换：切换全局页目录以加载一个新的地址空间；切换内核栈和硬件上下文
3.上下文切换
直接开销：内核完成切换所用的CPU时间
间接开销：高速缓存、缓冲区缓存、TLB失效
4.最短剩余时间SRTN优先是短作业优先SJF的抢占式版本
5.最高响应比优先HRRN

$$响应比R=\frac{周转时间}{处理时间}=\frac{处理时间+等待时间}{处理时间}$$

6.优先级反转（基于优先级的抢占式）：一个低优先级进程持有一个高优先级进程所需要的资源，使得高优先级进程等待低优先级进程运行。
解决方案：设置优先级上限（进入临界区的进程优先级最高）
优先级继承（低优先级进程继承高优先级进程的优先级）
使用中断禁止（进入临界区的进程不再响应中断）
7.Windows线程调度：
引发线程调度的条件：一个线程的优先级改变了；一个线程改变了它的亲和处理机集合
8.线程的时间配额
时间配额不是一个时间长度值，而是一个称为配额单位的整数。一个线程用完了自己的时间配额时，如果没有其他相同优先级的线程，Windows将重新给该线程分配一个新的时间配额，让它继续运行。
9.当线程被抢占时，它被放回响应优先级的就绪队列的对首。处于实时优先级的线程在被抢占时，时间配额被重置为一个完整的时间配额。处于可变优先级的线程在被抢占时，时间配额不变，重新得到CPU后将运行剩余的时间配额。
10.线程优先级提升后，为了避免不公平，在I/O操作完成唤醒等待线程时会将该线程的时间配额减1

第五章

1.DEKKER算法是第一个解决互斥问题的软件解决方案。会出现忙等待问题。
2.PETERSON算法解决了互斥访问的问题，而且克服了强制轮流法的缺点，可以完全正常的工作。（没有用原语）
3.中断屏蔽方法不适用于多处理器；测试并加锁TSL是把总线锁住，适合多处理器；XCHG指令；自旋

锁适合多处理器。

第六章

1.管程的出现是由于信号量机制的不足：程序编写困难，易出错
2.管程是互斥进入的，管程的互斥性是由编译器负责保证的
3.Hoare管程：先进来的先运行，后面来的那个等待
MESA管程：正在执行的接着执行
4.管程的实现：直接构造（效率高）、间接构造（用某种已经实现的同步机制去构造，如信号量）
5.Hoare管程缺点：两次额外的进程切换
6.MESA：signal（）改为notify（），用while循环取代if。导致对条件变量至少多一次额外的检测（但不再有额外的进程切换），并且对等待进程在notify之后何时运行没有任何限制
改进notify：给每个条件原语关联一个监视计时器，不论是否被通知，一个等待时间超时的进程将被设置为就绪态
引入broadcast
7.MESA管程优于Hoare管程之处在于MESA管程错误比较少。在MESA管程中，由于每个过程在收到信号后都会重新检查管程变量，并且由于使用了while结构，一个进程不正确的broadcast广播或发信号notify，不会导致收到信号的程序出错。
8.引入进程间通信IPC是因为信号量及管程的不足（只能传递少量信息），不适用于多处理器情况。进程间通信IPC可以解决进程间的同步问题、通信问题
9.基本通信方式：消息传递、共享内存、管道、套接字、远程过程调用
10.屏障：一种同步机制，应用于对一组线程进行协调。一组线程协同完成一项任务，需要所有的线程到达一个汇合点后再一起向前推进。

第七章

1.静态重定位：当用户程序加载到内存时，一次性实现逻辑地址到物理地址的转换，一般可以由软件完成。
动态重定位：在进程执行过程中进行地址变换，即逐条指令执行时完成地址转换。需要硬件部件支持。
内存管理单元MMU
2.首次适配、下次适配、最佳适配、最差适配
3.伙伴系统，一次特殊的“分离适配”算法，将内存按2的幂进行划分
4.单一连续区：一段时间只有一个进程在内存。简单，内存利用率低。它总是被加载到同一个内存地址上。
5.内存“扩充”技术：内存紧缩技术（如可变分区）、覆盖技术（程序大小超过物理内存总和，程序执行过程中，程序的不同部分在内存中相互替代，程序员负责生命覆盖结构）、交换技术（操作系统操作，一般系统会指定一块特殊的磁盘区域作为交换空间）、虚拟存储技术

第八章

1.存储保护：确保每个进程都有独立的地址空间，确保进程访问合法的地址范围，确保进程的操作是合法的。地址越界—>陷入操作系统
2.请求调页（主要）、预先调页
3.通常，页表项是硬件设计的。包括：页框号、有效位、访问位、修改位、保护位
4.反转（倒排）页表：从物理空间出发，系统建立一张页表。页表项纪录进程i的某虚拟地址（虚页

号）与页框号的映射关系。将虚拟地址的页号部分映射到一个散列值，散列值指向一个反转页表。反转页表大小与实际内存成固定比例，与进程个数无关。
5.TLB：相联存储器，按内容并行查找
6.缺页异常：硬件检查页表时发现所要访问的页面不在内存。操作系统执行缺页异常处理程序。
7.固定分配局部置换、可变分配局部置换、可变分配全局置换
8.页框锁定：通过设置相应的锁定位，不让操作系统将进程使用的页面换出内存，避免产生由交换过程带来的不确定的延迟。是为了希望运行时间相对稳定。
9.清除策略：从进程的驻留集中收回页框。

两个表：空闲页链表、修改页链表。被修改的页定期写回硬盘（不是一次只写一个，大大减少磁盘访问时间）。被置换的页仍保留在内存中。
10.页面置换算法：最佳页面置换算法OPT、先入先出算法FIFO、第二次机会算法SCR（先检查其访问位，如果0就置换，如果是1给第二次机会，并将访问位置0）、时钟算法CLOCK、最近未使用算法NRU（时钟算法的改进）、最近最少使用算法LRU、最不经常使用算法NFU（LRU的一种软件解决方案）、老化算法AGING
11.最优页面大小：P=根号下（2se），s：平均进程规模，e：页表项长度
12.内存映射文件：进程通过一个系统调用（mmap）将一个文件（或部分）映射到其虚拟地址空间的一部分，访问这个文件就像访问内存中的一个大数组，而不是对文件进行读写。在多数实现中，在映射共享的页面时不会实际读入页面的内容，而是在访问页面时，页面才会被每次一页的读入。磁盘文件则被当作后备存储。在进程退出或显式地解除文件映射时，所有被修改的页面会写回文件。
13.写时复制：新复制的页面对执行写操作的进程是私有的，对其他共享写时复制页面的进程是不可见的。

第九章

1.文件是对磁盘的抽象。
2.文件系统提供与I/O系统的统一接口。
3.文件的分类：普通文件、目录文件、特殊文件（字符设备文件和块设备文件）、管道文件、套接字。
4.文件的逻辑结构（记录式文件、流式文件）由用户决定。
5.UNIX成组链接法
6.

磁盘

柱面号 = 块号 / (磁头数 × 扇区数)
磁头号 = 块号 mod (磁头数 × 扇区数) / 扇区数
扇区号 = 块号 mod (磁头数 × 扇区数) mod 扇区数
块号 = 柱面号 × 磁头数 × 扇区数 + 磁头数 × 扇区数 + 扇区号

7.

位图

块号 = 以字长为单位
字号 = 块号 / 字长
位号 = 块号 mod 字长

8.文件控制块FCB包含的常用属性：文件名、文件号、文件大小、文件地址、创建时间、最后修改时间、最后访问时间、保护、口令、创建者、当前拥有者、文件类型、共享计数、各种标志（只读、隐藏、系统、归档、ASCII/二进制、顺序/随机访问、临时文件、锁）
9.目录是文件控制块的有序集合。
10.文件分配表FAT：把系统所有的块号记录在一张表内

11.索引结构：每个文件一个索引表，在FCB中存放索引表的位置（索引表不一样大小，不适合放在FCB中）
12.同一个文件卷中使用同一份管理数据进行文件分配和磁盘空闲空间管理，不同文件卷中的管理数据是相互独立的。
13.系统打开文件表：整个系统一张，放在内存中，用于保存已打开文件的FCB。FCB（i结点）信息、引用计数、修改标记。
　用户打开文件表：每个进程一个，进程中的PCB中记录了用户打开文件表的位置。不需要记录FCB。文件描述符、打开方式、读写指针、系统打开文件表索引。

第十章

1.FAT16:文件系统的数据记录在“引导扇区”中。表项16字节，目录项32字节。FAT描述簇的分配状态、标注下一簇的簇号等。
2.主引导记录MBR、引导区DBR、引导扇区BIOS参数块、扩展BIOS参数块EBPB
3.FAT状态：未使用、坏簇、系统保留、被文件占用（下一簇簇号）、最后一簇0xFFFF。簇号从0开始编号，簇0和簇1保留。
4.FAT32的根目录区不是固定区域、固定大小，而是数据区的一部分，采用于子目录相同的管理方式。
5. ..：父目录
6.FAT32支持长文件名格式，支持Unicode，不支持高级容错特性，不具有内部安全特性。
7.创建文件：实施是建立文件的FCB。在目录中为新文件建立一个目录项，根据提供的参数及需要填写相关内容。分配必要的存储空间。
　检查参数的合法性、申请空闲目录项、为文件申请磁盘块
8.打开文件：根据文件名在目录中检索，并将该文件的目录项读入内存，建立相应的数据结构，为后续的文件操作做好准备。返回给用户一个文件句柄（文件描述符）。
　先查系统打开目录表。根据打开方式、共享说明和用户身份检查访问合法性。在用户打开文件表中获取一空表项，填写打开方式等，并指向系统打开文件表对应表项。
9.读文件（文件描述符，读指针，要读的长度，内存目的地址）
　根据打开文件时得到的文件描述符，找到相应的文件控制块（目录项），确定读操作的合法性。将文件的逻辑块号转换为物理块号，根据参数中的读指针、长度与文件控制块中的信息，确定块号、块数、块内位移。申请缓冲区。启动磁盘I/O操作。
10.rename给文件重命名：根据文件名在目录中检索，找到对应FCB或目录项，直接更改文件名即可。
11.文件系统备份：
　全量转储：定期将所有文件拷贝到后援存储器。增量转储：只转储修改过的文件，即两次备份之间的修改，减少系统开销。
　物理转储：从磁盘第0块开始，将所有的磁盘块按序输出到磁带。逻辑转储：从一个或几个指定目录开始，递归地转储自给定日期后所有更改的文件和目录。
12.文件系统的写入策略
　通写（就是写穿透法）（FAT）、延迟写（就是写返回）（可恢复性差，如果两次恢复间出现故障）、可恢复写（利用事务日志来实现文件系统的写入，即考虑安全性，又考虑速度性能）、（NTFS）
13.文件的访问控制：（文件的）主动控制：访问控制表，（用户的）能力表（权限表）
14.UNIX 二级存取控制
第一级：对访问者的识别

对用户分类：文件主、文件主的同组用户、其他用户
第二级：对操作权限的识别
对操作分类：读操作r、写操作w、执行操作x、不能执行任何操作-
15.提高文件系统性能的方法：目录项（FCB）分解、当前目录、磁盘碎片整理、块高速缓存、磁盘调度、提前读取、合理分配磁盘空间、信息的优化分布、RAID技术…
16.提前读取具有针对性：数据块可以多读几块，代码段可以少读几块
17.Windows的文件访问方式中，要访问的数据在磁盘、系统缓存和进程地址空间有三份备份，通常用户对数据的修改并不直接反映到磁盘上，而是通过write-back机制由lazy writer定期地更新到磁盘
18.分配磁盘块时，把有可能顺序存取的块放在一起。尽量分配到同一柱面上，从而减少磁盘臂的移动次数和距离。
19.磁盘调度：先来先服务FCFS、最短寻道时间优先、扫描算法SCAN（电梯算法）、单向扫描调度算法C-SCAN、N-step-SCAN策略（新请求加到新队列）、FSCAN策略（使用两个子队列）
20.独立磁盘冗余阵列RAID：通过把多个磁盘组织在一起，作为一个逻辑卷提供磁盘跨越功能；通过把数据分成多个数据块，并行写入/读出多个磁盘，以提高数据传输率（数据分条stripe）；通过镜像或校验操作，提高容错能力（冗余）
21.最简单的RAID组织方式：镜像
最复杂的RAID组织方式：块交错检验
RAID0:条带化（无冗余，无差错控制，多个磁盘可以并行操作，性能最佳）
RAID1:镜像（数据安全性最好）
RAID4:交错块奇偶校验（带奇偶校验、以数据块为单位）

第十一章

1.块设备可寻址，字符设备不可寻址
2.I/O设备一般由机械和电子两部分组成。机械部分是设备本身（物理装置），电子设备又称设备控制器（或适配器）。
3.控制器的作用：操作系统将命令写入控制器的接口寄存器（或接口缓冲区）中，以实现输入/输出，并从接口寄存器读取状态信息或结果。当控制器接受一条命令后，可独立于CPU完成指定操作，CPU可以另外执行其他运算；命令完成时，控制器产生一个中断，CPU响应中断，控制转给操作系统；通过读控制寄存器中的信息，获取操作结构和设备状态。控制器与设备之间的接口常常是一个低级接口。
4.控制器的任务：把串行的位流转换为字节块，并进行必要的错误修正。首先，控制器按位进行组装，然后存入控制器内部的缓冲区形成以字节为单位的块；在对块验证检查并证明无错误时，再将它复制到内存中。
5.内存映像编址（内存映像I/O模式）、I/O独立编址（I/O专用指令）
6.内存映射I/O不需要特殊的保护机制来阻止用户进程进行I/O操作。操作系统必须要避免把包含控制寄存器的那部分地址空间放入任何用户的虚拟地址空间之中。可以引用内存的每一条指令也可以引用控制寄存器。
7.内存映射I/O不能进行高速缓存。
8.I/O软件层次：
用户级I/O软件（系统调用，对I/O数据进行格式化，为假脱机输入/输出做准备）
与设备无关的OS软件（设备的命名、保护、成块处理、缓冲技术和设备分配；
设备驱动程序（设施驱动程序设置设备寄存器、检查设备的执行状态）
中断处理程序

硬件（实现物理I/O操作）
9.与设备无关的OS软件包括：驱动程序的统一接口、缓冲、错误报告、分配与释放设备、提供与设备无关的块大小）
10.设备独立性（设备无关性）：设备分配时的灵活性，易于实现I/O重定向
11.缓冲区：硬缓冲（硬件寄存器实现）、软缓冲（在内存中开辟一个空间，用作缓冲区）
12.UNIX缓冲区由两部分组成：缓冲控制块或缓冲首部+缓冲数据区
13.逻辑设备号和盘块号是缓冲区的唯一标志。
14.独占设备的分配：静态分配、动态分配
15.设备驱动程序与外界的接口：与操作系统的接口、与系统引导的接口、与设备的接口

第十二章

1.可消耗资源：只可使用一次，可创建和销毁的资源
2.活锁：即无进展也没有阻塞，得不到资源，但总能上CPU运行一段时间
3.产生死锁的必要条件：互斥使用（资源独占）、占有且等待（请求和保持、部分分配）、不可抢占（不可剥夺）、循环等待。
4.破坏循环等待条件：资源有序分配算法
5.不安全状态一定导致死锁，但是当前不一定死锁。所以不安全状态不一定是死锁状态。