Hdu 操作系统 2019

author：hjs

操作系统目标：方便性、有效性

批处理操作系统：

单道批处理系统：监督程序；缺点：资源利用率、系统吞吐量低

多道批处理系统：多道程序设计技术、后备队列、作业调度程序；

优点：提高了 CPU 的利用率，提高了内存和 I/O 设备的利用率，增加系统吞吐量；

缺点：作业平均周转时间长，无交互能力

分时操作系统：时间片原则；最重要的特征是交互性；

分时系统的及时性是以用户所要求的响应时间来确定

实时操作系统：及时响应外部事件请求，规定时间内完成处理；

实时控制系统：用于工业、军事、生产；

实时信息处理系统：机票订购、银行财务、情报检索；

最重要特征是及时性（开始截止时间或完成截至时间）

操作系统特征：并发性（提高资源利用率和系统吞吐量）、共享性（互斥共享、同时共享（如磁盘，共享的条件是并发））、虚拟性、异步性

操作系统功能：处理器管理、存储器管理、设备管理、文件管理、提供用户接口（命令接口：shell；程序接口：系统调用（访管指令：trap、INT80、INT21）图形接口）

操作系统内核结构：单体/整体结构：Linux、MS_DOS；层次结构：Unix；微内核结构：Windows NT；混合内核：Windows、macos

处理器特权级：管态（系统态、核心态）：使用系统全部资源、访问整个存储区、使用全部指令（包括特权）；目态（用户态）：只能访问自己的存储区域、不能执行特权指令、必须向操作系统提出资源使用请求，由操作系统统一分配资源

区分目的：保护操作系统

中断目的：（1）提高计算机系统效率（2）维持系统可靠正常工作（3）满足实时处理要求（4）提供故障现场处理手段

中断分类：（1）中断（是异步的，由硬件随机产生）

可屏蔽中断：可用程序忽略中断信号，IO 设备发出的所有中断请求都属于可屏蔽中断

非屏蔽中断：不能由程序屏蔽，紧急事件发出，处理器一定要立即处理，如断电、硬件故障

（2）异常（同步的，程序指令执行时由 CPU 控制单元产生）

处理器探测异常：略

编程异常：又称软中断，如系统调用

中断向量表：为每个中断和异常赋予一个唯一的标识号，称为中断向量，以快速定位一个异常或中断处理程序的入口地址，中断向量组成中断向量表；向量号范围：0-255，0-31 保留为 x86 处理器定义的异常和中断，32-255 用于用户自定义中断。所以系统最多只有 256 个中断或异常向量。

系统调用：用于请求操作系统服务；用户态下使用，是用户在程序一级请求操作系统内核完成某种功能服务的过程调用，每一个系统调用都是完成某种特定内核功能的一个函数，如 read()、printf()；
工作机制：调用程序运行在用户态；被调用过程运行在系统态，需要通过中断和陷入机制，先由用户态转换到系统态，经内核分析检查后，才能转向相应的内核被调用过程执行。从管态恢复到目态通过修改程序状态字来实现
Linux 常用系统调用号：
read 0
write 1
open 2
close 3
exit 60
一组系统调用组成了提供给用户的程序接口

程序并发执行的特征
（1）间断性
（2）不可再现性：会出现与时间有关的错误，这是由于访问了共享变量引起

进程管理功能：（1）进程控制（2）进程互斥与同步（3）进程通信（4）调度
进程：进程是具有一定独立功能的程序关于某个数据集合的一次运算过程，是系统进行资源分配和调度的一个独立单位
进程特征：（1）动态性：最基本特征（2）并发性（3）独立性（4）异步性
进程映像：程序、数据集、栈和 PCB
进程状态：（1）就绪态（2）运行态（3）阻塞态
就绪->运行
运行->就绪：时间片用完、抢占调度（如优先级抢占）
运行->阻塞：等待 IO 操作完成、wait()
阻塞->就绪：IO 操作完成、signal()

PCB 基本信息
（1）进程标识信息：进程内部标识符 pid 等
（2）进程调度信息：进程状态，优先级等
（3）进程现场信息：通用寄存器内容、指令计数器的值、程序状态字 PSW、栈指针
（4）进程控制信息：略

引起进程创建的典型事件
（1）作业调度：eg：批处理系统
（2）用户登录：eg：分时系统、Linux 的 shell 进程
（3）提供特定服务：

（4）应用请求

引起进程撤销的典型事件：
（1）正常结束而撤销
（2）异常终止而撤销
（3）应外界干预而撤销：eg：发生死锁、死循环而撤销进程、父进程请求撤销子进程、父进程被撤销时，子孙进程被系统自动撤销

Linux：动态优先级 prio：[0,139]，实时进程[0,99]，普通进程[100,139]
数值越小，优先级越高
静态优先级 static_prio：[0,139]，进程被创建时从父进程继承而来
常规动态优先级 normal_prio：用于临时提升进程的优先级，工作完成后恢复
普通进程三个优先级的值通常是相同的。

fork：创建一个普通进程，调用一次返回两次：子进程中返回 0，父进程中返回子进程的 pid，创建失败返回-1

vfork：子进程共享父进程的地址空间，父进程会一直阻塞，直到子进程调用 exit()终止运行或调用 exec()加载另一个可执行文件，即子进程优先运行

进程同步机制应该遵循的原则
（1）空闲让进：临界区没有进程访问，则允许任何一个进程进入临界区
（2）忙则等待：有进程正在访问临界资源，其他进程必须等待。即实现互斥
（3）有限等待：即没有进程处于“永远等待”的状态
（4）让权等待：eg: wait(S){ while(S<=0); S--;} //一直死循环占用 cpu，违背让权等待

硬件解决进程互斥问题：
（1）禁止中断
缺点：（1）赋予普通用户权限，增加系统风险（2）只能用于单处理器系统
（2）使用专用机器指令：TSL、Swap
缺点：违背了“让权等待”，并且存在饥饿现象

软件方法解决进程互斥问题
（1）严格交替算法：违背“空闲让进”和“让权等待”原则
（2）Peterson 算法：违背“让权等待”
（3）面包店算法：违背“让权等待”

利用锁机制解决进程互斥问题
整型信号量机制：违背“让权等待”，存在“忙等”现象
记录型信号量机制：实现了“让权等待”
信号量集机制

Linux 自旋锁：常用于多处理器系统中的进程互斥，自旋锁任一时刻最多只能被一个进程持有，所以能够实现多个进程互斥进入临界区。若锁被其它进程所持有，则当前进程空循环等待，直到申请到锁。因此自旋锁只适用于短期间进行轻量型加锁，特别适合可快速完成的临

界区代码的互斥，但不应该被长时间持有。

Linux 用户程序同步机制：
（1）IPC 信号量（System V 信号量）
（2）POSIX 信号量（无名信号量、有名信号量）

进程调度
（1）高级调度（作业调度）
后备队列（外存） -> 就绪队列（内存）
*批处理系统需要用到作业调度，而分时系统和实时系统不需要作业调度
作业调度算法：先来先服务、短作业优先、基于优先级调度、高响应比优先等

（2）低级调度（进程调度）
就绪队列（内存） -> CPU 运行

（3）中级调度（对换调度）
目的：提高内存利用率和系统吞吐量
当内存紧张时：阻塞队列 -> 外存 （挂起状态）
当内存充裕时：外存 -> 就绪队列（内存）

进程调度功能：
（1）排队程序：排序就绪队列
（2）分派程序：就绪队列中选择进程分配 CPU 运行
（3）上下文切换程序：在 CPU 切换过程中，保存被移出进程的所有 CPU 寄存器内容到其 PCB 或堆栈中，恢复新选中进程的 CPU 运行环境信息。在进行 CPU 切换时，会发生两次上下文切换操作：第一次是被移出 CPU 的进程与分配程序间的上下文切换，第二次是分配程序与新选中进程间的上下文切换。

进程调度方式：
（1）非抢占方式：实现简单系统开销小，缺点是不能保证紧迫性任务（实时进程）得到及时处理。
（2）抢占方式：（1）基于优先级原则：实时系统；（2）基于时间片原则：分时系统
缺点是系统开销大

进程调度时机：
（1）运行进程完成所有工作，或出现错误终止运行
（2）运行进程等待 I/O 操作完成或 wait()等而变成阻塞态
（3）分时系统时间片原则
（4）基于优先级抢占式调度
（5）完成系统调用或中断处理后，在返回用户态之前，会产生一次调度

进程调度算法
（1）FCFS 先来先服务调度算法

作业调度：后备队列 -> 就绪队列（内存）

进程调度：就绪队列 -> CPU 运行

周转时间 = 完成时间 – 提交时间　　带权周转时间 = 周转时间/要求执行时间

表格形式：

作业名 提交时间 要求执行时间 开始执行时间 完成时间 周转时间 带权周转时间

特点：FCFS 有利于长作业，不利于短作业，平均周转时间较长，非抢占，不能保证紧迫性任务得到处理。不适用于实时系统。

（2）SJF 短作业优先调度算法

非抢占式：第一个到达的进程先投入运行，之后都从到达就绪队列的进程中选择要求运行时间最短的进程投入运行

最短平均周转时间 = (4a + 3b + 2c + d)/4

抢占式：每当一个进程进入就绪队列时，就需要比较该进程的下一次运行时间是否比当前运行进程的剩余时间短，如果是就抢占当前运行进程的 cpu

特点：获得最短平均周转时间和平均带权周转时间，提高系统吞吐量；在作业调度中使用更多；是一种不公平的调度算法，不利于长作业，让长作业产生饥饿现象；不能保证紧迫型任务得到及时处理

（3）HRRF 高响应比优先调度算法

响应比 = 1 + 等待时间/要求服务时间

特点：首先照顾短作业（进程），长作业会随着等待时间的延长而逐步升高并获得调度机会，改进了短作业优先的不公平情况，避免饥饿现象；更多用于作业调度；不能保证紧迫型任务得到及时处理

（4）优先级调度算法

非抢占式优先级调度算法

可抢占式优先级调度算法：适用于实时系统

（5）RR 时间片轮转调度算法

略

（6）多级队列调度算法

略

（7）多级反馈队列调度算法

Unix 采用。

（1）设置多个就绪队列，各个队列赋予不同的优先级，第一个最高，其余依次降低；同时为每个就绪队列设置不同的时间片，优先级最高的时间片最短，第二个队列是第一个队列的两倍…第 i+1 个队列是第 i 个队列的时间片的 2 倍。

（2）各队列内部按时间片轮转算法进行调度

（3）各队列之间采用抢占式优先级算法调度

仅当第 1 ~ i-1 个队列为空时，才会调度第 i 个队列中的进程运行。

当 CPU 正在运行第 i 个队列中的某个进程时，又有进程进入优先级较高的队列，则系统立即调度高优先级进程运行，而把正在运行的进程重新插入第 i 个队列（原队列）。

Linux 调度器：

Linux2.6：O(1)调度器

关键词：两个长度为 140 的优先级队列数组：active 数组（本轮调度时间片未耗尽）和 expire 数组（时间片已耗尽），每个元素指向一个优先级队列。0-99 实时进程 100-139 普通进程；

为每个队列数组设置一个优先级位图，每个位对应一个优先级队列。有进程相应位为 1，否则为 0。调度时遍历 active 数组位图，找到第一个值为 1 的位。

复杂度 O(1)，调度时间开销与就绪进程数量无关。

动态优先级 = max(100,min(static_prio-bonus + 5，139))

若要提高进程优先级，bonus 应该取值大于 5

Linux2.6.24：CFS 调度器

关键词：一是根据系统中各进程的权重分别为各进程分配下一次 CPU 运行时间长度，进程优先级越高，权重越大，分配到的 CPU 时间越多。二是选择下一个运行进程时，总是选择“运行速度”最慢的进程参与运行。

CFS 不再使用动态优先级，而是使用 nice 值[-20,19]，nice 值越大，优先级越低，权重越小。prio 值与 nice 值转换：prio - 120 = nice

nice 值 0 对应权重 1024

为了维护公平性，CFS 提出虚拟运行时间 vruntime 概念：总把 CPU 分给 vruntime 最小的那个。

权重>1024，vruntime 小于实际运行时间，权重<1024，vruntime 大于实际运行时间。nice 值为 0（1024），vruntime 与实际运行之间相等。

CFS 就绪队列数据结构：红黑树，一种平衡二叉树

进程通信

（1）共享存储器系统通信

存储器中划出共享存储区，诸进程对共享区进行读写。

Linux：shmget() 创建 shmat() 连接映射 shmdt() 断开 shmctl() 撤销

数据量由共享存储区大小决定

（2）消息传递系统通信

可实现不同主机多个进程的通信，进程间数据交换以格式化的消息为单位

1.直接通信方式

send()，receive()

2.间接通信方式

信箱通信

基本思想：发送进程把消息发送到共享的称为信箱的中间实体，接收进程从信箱中取出对方发送给自己的消息。信箱是一种数据结构，用来存放信件，由若干格子组成，每个格子存放一个信件，格子的数量和大小在创建信箱时确定。

私用信箱：由用户进程创建，只有创建者能从该信箱中读取消息，其他进程只能发

公用信箱：由操作系统创建

共享信箱：由某进程创建，指明信箱的共享者，创建者和共享者都可以从信箱中接

收其他进程发送给自己的消息。

（3）管道通信系统

管道——管道文件

文件描述符：fd[0]：从管道读信息；

fd[1]：从管道写信息；

无名管道：临时文件，pipe()建立，使用完后关闭，管道文件不复存在

有名管道：长期存在具有路径名的文件，mkfifo()建立，关闭后存在。FIFO 文件

（4）客户-服务器系统通信

进程死锁

原因：资源有限且分配不当

产生死锁必要条件（Coffman 条件）：只要一个不满足，就不会出现死锁

（1）互斥条件

（2）占有且等待条件

（3）不可剥夺条件

（4）循环等待条件

处理死锁的方法：

（1）预防死锁：破坏产生死锁的四个必要条件的一个或几个

1. 破坏互斥：X
2. 破坏占有且等待条件：

静态分配资源：一开始就要申请到所有资源，只有获得所有资源才能执行。

缺点：可用于批处理系统，但在交互式系统（分时）资源利用率低，系统吞吐量小，会产生饥饿现象。

要求进程在不占有资源时才可申请资源

3. 破坏不可剥夺条件：

不适用于不可剥夺资源，如打印机和磁带

4. 破坏循环等待条件：

按序分配资源：对系统中所有资源类型进行线性排序并编号，如果进程整个生命周期需要申请两个以上资源，则必须按序号递增顺序申请资源；如果进程需要多个同类型资源，则必须同时一起申请。

优点：改善了资源利用率和系统吞吐量

缺点：降低添加新设备的灵活性、造成资源浪费

（2）避免死锁：银行家算法，防止系统进入不安全状态

系统进入不安全状态只是存在死锁的可能性，不一定会发生死锁。

银行家算法：

①T0 时刻的资源分配状态：Max　Allocation　Need　Available

②进程 P1 请求资源：Request1<a，b，c>，进行检查

Request1<a，b，c>　<=　Need1<a，b，c>

Request1<a，b，c> <= Available<a，b，c>

不满足则失败，否则直接修改 P1 分配资源后的系统资源分配状态，并转③

③安全性检查

Work Need Allocation Work+Allocation Finish

初始 Work = Available

随后每次都是 Work = 上一个进程的 Work + Allocation

全为 true 得到安全性序列

算法过程略

（3）检测和解除死锁：死锁定理+资源分配图：略

某类资源的不会发生死锁的最少数量 = (每个进程所需该类资源数 – 1) * 进程数 + 1

线程和进程的比较

（1）把线程作为调度和执行的基本单位，把进程作为资源分配和拥有的基本单位，提高系统并发程度，降低 CPU 切换开销。

（2）进程间可并发执行，一个进程的多个线程间也可并发执行。并发度更高，资源利用率和吞吐量更高。

（3）系统创建进程开销大于创建线程开销。进程切换开销也远大于线程切换开销

第四章 存储器管理

CPU 处理器可直接访问寄存器、Cache 和主存（内存）

存储器管理功能

（1）内存的分配与回收

静态方式：创建进程时分配好内存空间，进程不能申请新的内存，也不能在内存中更换位置

动态方式：以上均可

（2）地址映射

地址映射机构：完成虚地址映射到实际物理地址，即地址映射（地址重定位）

将程序的起始物理地址存储在重定位寄存器中

（3）内存的共享和保护

（4）内存扩充

程序的链接和装入

用户源程序 -> 编译程序来编译并生成若干目标模块 -> 将目标模块与所需要的库模块链接，生成一个可以装入内存执行的程序 -> 装入程序将可执行程序装入内存并创建进程 -> 进程调度

程序的链接方式

（1）静态链接：程序装入内存执行前就链接成完整的可执行程序，无法升级，无法共享

（2）装入时动态链接：程序装入内存执行时边装入边链接，可升级，无法共享

（3）运行时动态链接：程序装入内存后等执行到需要的模块或库才进行链接。可升级，可共享。

程序的装入方式

（1）绝对装入：程序装入到绝对地址开始的内存空间，该地址**编译时就确定**，无法改变。如固件程序烧写

（2）静态重定位装入：程序可加载到内存任一合适位置，**装入时**才一次性**确定**物理地址。之后程序在内存中不能再移动位置。软件方法实现。

（3）动态重定位装入：装入时程序依然使用逻辑地址，当真正执行指令时再将逻辑地址映射为物理地址，即**边执行边确定。**移动位置只需要修改重定位寄存器即可。需要软件+硬件实现

连续存储器管理方式

（1）固定分区

物理内存划分为大小和数量固定的分区，**每个分区只能装入一个进程**。

当有内存分配需求时，遍历分区分配表查找空闲且大小合适的分区分配，然后修改分区表的分配状态；回收则相反。

缺点：分区数量和位置固定，导致并发进程最大数量固定，分区内部空间浪费，内部碎片多。降低了内存利用率。

（2）可变分区

分区大小和数量可变。使用空闲分区链和相应回收算法实现。各分区分配后剩余空间可继续使用

分配算法：

1.首次适应算法

空闲分区按首地址递增进行排序，从头到尾寻找可以第一个可以满足需求的分区就进行划分。

缺点：分区间存在外部碎片

2.最佳适应算法（内存利用率最好）

寻找与请求大小最接近的空闲分区进行分配。从小到大排列

缺点：分区间存在外部碎片。

3.最坏适应算法

寻找与请求大小相差最大的空闲分区进行分配。从大到小排列

缺点：

回收算法：上邻接、下邻接、上下邻接、无邻接

把不连续的空间变成连续的空间？紧凑，但代价巨大。

现代操作系统采用的都是非连续存储管理方式（离散存储管理方式）：

（1）分页存储管理方式（2）分段存储管理方式（3）段页式存储管理方式

以上都需要基于动态重定位

分页存储管理方式：内存利用率最高，只有最后一页有内部碎片。但是不便于程序模块的更新升级和共享保护
逻辑空间：页面　　物理空间：物理块、页框、帧　　都从 0 开始编号
页面大小 = 物理块大小
逻辑地址结构：页号 + 页内地址　以字节 Byte 为单位，即一个地址对应 1 字节空间
一维线性地址空间
页表：[页号]index　物理块号
页表存储在内存中，只存储物理块号，页号不占据存储空间。

物理地址 = 物理块号 * 页面大小 + 页内地址
页表寄存器 PTR：保存页表始址和页表长度

一次越界中断判断，两次访存（访问页表+访问物理地址）才能获取到数据

快表 TLB（本质是高速缓存）
容量只有 32~256 个表项

两级或多级页表
当页表大小超过一个页面大小时，就需要分级
页表偏移位 = 页面大小/页表项(PTE)大小
页表级数 = (逻辑地址-页内地址)/页表偏移位
页内地址 = 页面大小/1B

操作系统仅将顶级页表存储在内存中。
n 级页表访存次数：n+1 次
改善措施：引入快表 TLB 机制

分段存储管理方式：将进程的虚地址空间按逻辑信息划分为若干段，存储有逻辑关系和意义的部分，相对完整的一组逻辑信息组织成一个段，有助于实现程序更新和升级，数据的共享和保护。内存利用率低于分页存储，因为段长度大小可变，而页面大小固定，各分段之间可能存在外部碎片。 dll 文件

逻辑地址结构：段号 + 段内地址　[1,1000]　1 号段段内地址为 1000 的位置
二维非线性地址
注意：段内地址的位数决定了段的最大长度，而不是当前段的长度；
段号 S 的位数决定了一个进程段的最大数量

段表：[段号]index　段长 段基址
段表存放在内存中，段表始址和段表长度保存在进程的段表寄存器中
两次越界中断判断：
（1）段号 >= 段表长度？ 越界中断　　0 开始编址段号

（2）段内地址 >= 该段段长？越界中断

段页式存储管理方式：每段最后一页可能会产生内部碎片
逻辑地址结构：段号 + 段内页号 + 页内地址
一个进程一张段表，每段一张页表。即一个进程一张段表，多张页表。

两次越界中断判断
（1）段号 段表长度
（2）段内页号 页表长度

段表和页表都存储在内存中，所以需要**三次访存**
优化：引入快表提高访存性能

虚拟存储系统
含义：具有请求调入功能和置换功能，能够利用外存储器的空闲空间从逻辑上对内存容量进行扩充的一种存储器系统。

三种实现方式：（1）**请求分页（主流）**（2）请求分段（3）请求段页式
请求分页：在分页存储管理系统基础上增加了请求调页和页面置换功能，换入换出基本单位是页面。需要使用带有**缺页中断机构**的地址映射机构实现逻辑地址到物理地址的转换过程。

页号 物理块号 状态位 访问字段 修改位 外存地址
状态位：不在内存，产生缺页中断
访问字段：记录页面访问情况
修改位：页面装入内存后是否被修改过，
外存地址：页面在外存上的地址

频繁页面换出换入：抖动/颠簸

页面置换策略：
（1）可变分配全局置换
进程分配的物理块数量是可变的，可从内存中的所有进程中寻找可置换的页面
（2）可变分配局部置换
进程分配的物理块数量是可变的，只能从本进程在内存中已存在的页面寻找置换页面
（3）固定分配局部置换
进程分配的物理块数量是不可变的，只能从本进程在内存中已存在的页面寻找置换页面

页面置换算法
（1）最佳置换算法 OPT 无法预知页面走向，只是理论上的算法
思想：选择**将来**永远不再访问或者最长时间内不会访问的页面进行置换

（2）先进先出置换算法 FIFO：会出现 Belady 现象：物理块数增加，缺页率不减反增。

思想：淘汰最早调入内存的页面
基于队列实现

（3）最近最久未使用置换算法 LRU
思想：淘汰最近一段时间内最长时间没有被访问的页面
基于堆栈实现

（4）最近最少使用置换算法 LFU
思想：淘汰最近一段时间里访问次数最少的页面
基于堆栈实现

（5）时钟置换算法 NRU
访问位 u，替换指针始终指向最近被淘汰的页面所在的物理块号，当某页被访问时，其访问位置 1。当需要淘汰一页时，从替换指针的下一个页面检查访问位，如果为 0，就淘汰该页，如果为 1，改为 0，暂不淘汰，往下继续检查，直到找到访问位为 0 的页面为止。并将替换指针指向它。

改进型 Clock 算法：
增加修改位 m
um=00 最近未被访问也未被修改
um=01 最近未被访问，但被修改
um=10 最近被访问，但未被修改
um=11 最近被访问，也被修改
步骤：
（1）从替换指针的**当前位置**开始扫描，对访问位不做任何修改。查找 00 型页面用于置换，如果找到则结束
（2）如果（1）失败，则重新扫描查找 01 型页面，并将所有扫描过的页面的访问位都置为 0。如果找到就结束
（3）如果（2）失败，则将所有页面的访问位 u 改为 0，重复（1），没有找到再重复（2）

（6）页缓冲
淘汰页进入缓冲区，缓冲一段时间，防止错选淘汰页。

伙伴系统：
优点：时间效率高，同时能够尽量保证有足够的连续空间
缺点：空间利用率不高，用空间换时间
内存块包含的页框数量必须 2^k（0 <= k < 11） 即 1~1024
假设分配请求 n 个页框的内存，在大小与 n 最接近的内存块队列 2^m 中查找，若为空则往上递归逐级分解

第 5 章 设备管理

用户程序发出磁盘 I/O 请求后，系统的处理流程是：

用户程序 -> 系统调用处理程序 -> **设备驱动程序 -> 中断处理程序**
其中设备驱动程序用于计算数据所在磁盘的磁头号，扇区号。

设备管理应具有以下几个功能：
（1）设备分配
（2）缓冲管理
（3）设备处理

输入输出系统

块设备是有结构设备，如磁盘、U 盘。数据基本传输单位是固定长度的数据块
特点：传输速率高，可寻址，采用 **DMA 方式**

字符设备是无结构设备，如打印机、终端等。数据基本传输单位是字符
特点：传输速率低，不可寻址，一般采用**中断驱动方式**。

设备控制器：CPU——设备控制器——I/O 设备
功能：

（1）接收和识别命令	控制寄存器
（2）数据交换	数据寄存器
（3）地址识别：8 位，支持 127 个外部设备	地址译码器
（4）数据缓冲	缓冲器
（5）识别和报告设备的状态	状态寄存器
（6）差错控制	

设备控制器组成：
（1）设备控制器与处理器的接口：数据总线、地址总线、控制总线
（2）设备控制器与设备的接口：每个接口都存在数据、控制、状态三种类型信号线
（3）I/O 逻辑：实现对 I/O 设备的控制

I/O 通道：内存——I/O 通道——外设（I/O 设备）
作用：CPU 只需向通道发送一条 I/O 指令，通道在接收到该指令后，从内存中取出对应的通道程序并执行，仅当通道执行完该通道程序后，才向 CPU 发中断信号。**通道没有自己的内存，它与 CPU 共享内存。解放 CPU**

通道类型：
（1）字节多路通道：以字节为单位传输信息，时分复用执行多个通道程序控制多台设备，适用于低速、中速设备
（2）数组选择通道：一次只执行一个通道程序控制一台设备，实现内存和外设的成批数据传送。适用于高速设备
（3）数组多路通道：具备两者特点。

通道程序：
（1）操作码
（2）内存地址
（3）计数：要操作的数据的字节数
（4）通道程序结束位 P
P=1 是通道程序最后一条指令，标志程序结束
（5）记录结束标志 R
P=0 表示当前指令与下一条指令所处理的数据同属一个记录
P=1 表示当前指令是记录的最后一条指令

输入输出控制方式：
1、程序直接控制方式（不采用了，严重降低 CPU 利用率）
2、中断驱动控制方式（用于打印机等字符设备、CPU 和设备可并行工作，是字符设备的 IO 控制方式）
3、DMA 方式（块设备普遍采用 DMA 方式，它要求控制器支持直接存储器存取，整个数据传送过程由 DMA 进行控制，送数据时内存地址+1，字节计数器-1，不断重复，直到字节计数器变为 0，然后 DMA 向 CPU 发送中断）
4、通道控制方式：并行效率最高

缓冲管理
引入缓冲目的：
（1）缓和 CPU 与 I/O 设备速度不匹配的矛盾，提高 CPU 和 I/O 设备的并行性，提高系统吞吐量和设备的利用率。
（2）减少对 CPU 的中断频率，放宽 CPU 对中断响应时间的限制

单缓冲：max(C,T) + M
双缓冲：max((C+M),T)

SPOOLing 系统
假脱机技术，以空间换时间
组成：
输入井：在磁盘上开辟的大存储区，存放预先从 I/O 设备输入的各作业全部信息
输出井：在磁盘上开辟的大存储区，存放各运行作业的输出信息
预输入程序：把作业的全部信息输入到磁盘上的输入井中保存
缓输出程序：把输出井中等待输出的作业信息进行输出
井管理程序：分为输入井读，输出井读
输入井读：当作业要求读信息时，由输入井读程序从输入井中找出作业所需信息并传送给作业。
输出井写：当作业要求输出信息时，由输出井写程序把作业的输出信息存放到输出井中。

共享打印机:
输出井写程序将要打印的数据存放到输出井中
缓输出程序从输出井中取出待打印数据，通过输出缓冲区由打印机打印。

第 6 章 文件系统
Unix 和 Linux 也把设备作为特殊文件处理，文件名区分大小写。MS-DOS 和 Windows 不区分大小写。Linux 本身不使用文件扩展名来识别文件的类型，而是根据文件的头内容来识别其类型。Linux 三种文件类型：普通文件、目录文件、特殊文件（FIFO 文件、字符设备文件、块设备文件、符号链接文件）

文件系统功能：
（1）文件存储空间的管理：文件系统的存储分配通常采用离散分配方式。
（2）文件目录管理：实现“按名存取”，为每个文件建立 FCB
（3）文件地址映射：
（4）文件读写管理：
（5）文件共享和保护

文件的逻辑结构
（1）有结构文件：划分记录，又称记录式文件。基本单位是记录
顺序文件：串结构（按存入时间的先后顺序，只能顺序查找）、顺序结构（按关键字排序，可顺序访问也可随机访问）
顺序文件不利于文件的查找和动态增加或删除。
索引文件：索引表+逻辑文件，可实现随机访问，有利于文件增加和删除，但是索引表增大空间开销。
索引顺序文件
直接文件：关键字的值决定了记录的物理地址
散列文件：关键字转换，转换结果决定记录的物理地址

（2）无结构文件：不划分记录，又称流式文件。基本单位是字节
Unix、Linux、Windows 所有的文件都是流式文件。

同一文件的不同拷贝在**不同物理介质**上存储**可以采用不同的物理结构**
物理块 = 磁盘块 = 簇
FAT 文件系统中，簇的大小一般是 2^n 个扇区

文件分配磁盘空间方式
（1）静态分配：文件建立时一次性分配所需要的全部磁盘空间
（2）动态分配：根据动态增长

文件的物理结构
（1）**连续文件/顺序文件**：连续磁盘块

优点：支持顺序存取和随机存取

缺点：由于存储文件要求有连续的存储空间，**容易产生磁盘碎片**；不支持文件内容经常性的大幅度增加或删减

（2）**链接文件**：链接指针，**可消除磁盘碎片**，提高存储空间利用率

隐式链接（串联文件）：只能对文件进行顺序存取，不适合随机访问，可靠性差

显式链接：实现随机存取，FAT 文件系统采用显式链接，FAT 表在内存中存放，表项内容是链接指针，即文件的下一个盘块号。

FAT 表的表项数量由磁盘块的数量决定，表项长度由最大盘块号（磁盘块数量？）决定。

优点：查找 FAT 表在内存中，提高文件检索速度；实现随机存取

缺点：不支持大文件随机存取，而且 FAT 表占据内存空间。

（3）索引文件：

优点：支持高效随机访问，不产生磁盘外部碎片，支持文件动态增长

缺点：每个文件都需要索引块，增大存储开销。

1. 单级索引文件

一个文件一张索引表，文件索引指针（一级索引）指向索引表，索引表所在的盘块称为一级索引块。每个表项存放一个磁盘块号，指向一个磁盘块。当需要访问文件时，不需要读取整个 FAT 表到内存，只需要将索引块读入内存即可。

2. 多级索引文件

3. 混合索引文件：Unix（system V 文件系统）、Linux

system V 文件系统：索引表有 13 个表项，前 10 个为直接地址，第 11 个为一级索引，第 12 个为二级索引，第 13 个为三级索引。

文件 = 文件体 + 文件控制块（FCB）
文件目录项 = 文件控制块（存放每个文件的起始盘块号）
文件目录/目录文件 = 文件控制块的有序集合

引入索引节点：文件目录项 = 文件名 + i 节点指针
减少了目录文件的大小，减少访存，提高检索速度

文件目录结构形式：
单级目录：要求所有用户用同一个文件名来访问同一个文件
两级目录：主文件目录 + 各个用户文件目录
提高目录检索速度，允许文件重名（不同用户不同文件相同名，同一用户不允许重名）
多级目录：“.”表示当前目录，“..”当前目录的父目录，“/”根目录

目录检索技术：
1. 线性检索法
2. Hash 方法（不适用于现代操作系统）

文件存储空间管理

（1）空闲表法 X
（2）空闲块链表法
（3）位示图
（4）成组链接法 Unix

文件共享
（1）硬链接：基于索引节点实现共享
创建时索引节点的 count = 1
有用户参与共享加 count + 1，断开 count – 1
只有 count = 1 文件主才能删除文件，不能实现跨文件卷的文件共享

（2）软链接：利用符号链接实现共享
文件主可随时删除，能够实现跨文件卷共享

文件保护
对于一个文件的访问，常由用户访问权限和文件属性共同限制

设置文件访问权限
（1）访问控制矩阵
（2）访问控制表：按列进行划分
（3）用户权限表：按行进行划分

磁盘调度

扇区编址方式：
（1）CHS 方式：柱面号 i、磁头号 j、扇区号 k
柱面数 L 磁头数 M：扇区数 N
（2）LBA 方式：LBA = (i * M * N) + (j * N) + k – 1
i = LBA/(M * N)
j = [LBA mod (M * N)] / N
k = [LBA mod (M * N)] mod N +1

磁盘容量 = 柱面数 * 磁头数 * 扇区数 * 每扇区字节数

磁盘访问时间 = 寻道时间 + 旋转延迟时间 + 传输时间
旋转延迟时间 = 1/(2 * 磁盘转速)，即磁盘旋转半周的时间
传输时间 = 读写字节数/(一个磁道字节数 * 磁盘转速)

磁盘调度算法
（1）移臂调度：
FCFS：简单，不会产生饥饿现象，但是寻道时间长
SSTF（最短寻道时间优先）：总是选择与当前磁道最近的磁道访问请求

特点：不公平，远离当前磁道的 IO 请求得不到服务，饥饿现象。不考虑磁头方向，影响磁盘寿命。

SCAN（扫描算法/电梯算法）：选择与当前磁头方向+当前磁道最近的磁道访问请求，直到该方向没有访问请求，改变方向。消除了 SSTF 的不公平性。

CSCAN（循环扫描）：单向移动，与当前最近。完成最里层的请求后快速返回最外面的请求访问磁道，循环扫描。可消除对两端磁道请求的不公平。

N-Step-SCAN：分成若干个长度为 N 的子队列，子队列 FCFS，每个子队列内部 SCAN

FSCAN：分成两个子队列。

（2）旋转调度

Linux Ext 文件系统

Ext2 把一个磁盘分区的所有磁盘块分成若干个组，每个组包含固定数量的磁盘块，称为块组。每个块组都有本块组的索引节点和数据块，尽量让一个文件的索引节点及磁盘块位于一个快组中。索引 i 节点 128B

Linux 的 ext2 采用混合索引表实现文件的物理组织，15 个索引项，12 个直接索引项，1 个一次索引项、1 个二次索引项、1 个三次索引项。

索引项长度（盘块号长度） = 32bit = 4B

一个块 b 字节，则块位图最多只能表示 8*b 个块使用情况

每个块组有一个块位图，每个块位图大小占一个块。每个块组最多只能有 8*b 个块，假如分区有 s 块，则块组总数 = s/8*b