- 可重定位的动态装入
- 内存的分配
  - 单一连续分配
  - 分区
  - 动态分区分配
    - 数据结构
    - 分配
    - 回收
    - 分区算法
    - 紧凑
- 对换
  - 对换类型
  - 对换空间管理
- 分页（计算）
  - 页表（计算）
  - 快表
    - 访问内存有效时间EAT
- 分段
- 分段分页区别
- 段页式

第五章 虚拟存储器

- 虚拟存储器
  - 定义
  - 特征
- 请求分页
  - 页表
  - 缺页中断
- 内存分配策略
  - 调页来源
  - 缺页率
- 页面置换算法（计算）
- 页面缓冲算法
- 访问内存时间的影响因素
- 抖动

第六章 IO

- IO系统的层次结构
- IO通道
- IO控制方式
- 设备独立性
- SPOOLing
  - 组成
    - 假脱机管理系统P222，P223
      - 实现
- 磁盘访问时间
  - 磁头调度算法（计算）

第七章 文件管理

- 文件的结构
  - 逻辑结构
  - 物理结构
- 对目录的管理要求

第八章 磁盘管理

- FAT表
- 位示图
- 成组链接法

第一章 概论

操作系统的目标

方便性：使人与机器的沟通更加方便
有效性：1.提高系统资源的利用率 2.提高系统的吞吐量
可扩充性：快速方便地添加功能
开放性：计算机之间的数据互通

操作系统的作用

1.作为用户与计算机硬件系统的接口
2.管理计算机系统资源
3.实现都计算机资源的抽象

操作系统发展的主要动力

1.不断提高计算机资源利用率
2.方便用户
3.器件的不断更新迭代
4.计算机体系结构的不断发展
5.不断提出新的应用要求

不同种类的操作系统

单道批处理系统

目的:
　解决CPU与I/O设备速度不匹配的矛盾
特点:
　脱机输入输出
　利用**监督程序**将作业调入内存
　内存上同时只有一道作业
缺点:
　系统中的资源得不到充分的利用

多道批处理系统

目的:
　进一步提高资源的利用率和系统吞吐量
特点:
　作业在内存中排成**后备队列**
　作业调度程序按照一定的算法从后备队列中选入**数个作业**调入内存
　多个作业共享资源
　CPU中多个程序交替运行
优点:
　资源利用率高
　系统吞吐量大
缺点:

平均周转时间长
无交互能力

分时系统

目的:
解决多道批处理系统无交互能力的问题
实现多个用户共享主机
关键功能:
及时接收，及时处理，时间片
特征:
多路性：多个终端连接同一台主机
独立性：用户感受为独自使用主机
及时性：用户的请求能在很短时间内活的相应（1-3秒）
交互性：用户可通过终端与系统进行广泛的人机对话

实时系统

类型:
控制系统
信息查询系统
多媒体系统
嵌入式系统
目的:
使系统能及时响应外部事件的请求
在规定时间内完成事件处理
控制所有实时任务协调一致地运行
实时任务类型:
周期性与非周期性
软实时与硬实时任务

分时系统与实时系统的比较

1.多路性
信息查询系统与分时系统按照分时原则对多个终端用户服务
实时控制系统周期性地对多路现场信息进行采集，对多个对象或多个执行机构进行控制
2.独立性
信息查询系统与实时控制系统的交互过程都是相互独立互不干扰的
3.及时性
信息查询系统与多媒体系统的实时性要求由人的主观感受决定
实时控制系统的实时性由控制对象要求的截止时间决定，一般为秒级到毫秒级
4.交互性
信息查询系统与多媒体系统对系统的交互性限定于某些特定的功能
分时系统可以提供更多种的如数据处理，资源共享等服务
5.可靠性
实时系统比分时系统对系统的可靠性要求更高

操作系统的基本特性

并发：两个或多个事件在同一时间间隔发生 *并行：~在同一时刻发生*
共享：系统中的资源可供内存中多个并发的进程共同使用
并发和共享互为前提，是多用户（多任务）OS两个最基本的特征
虚拟：将一个物理实体变为若干个逻辑上的对应物
异步：允许进程以人们不可预知的速度推进。

第二章 进程

顺序与并发执行的特征

顺序执行

1.顺序性：严格按照程序规定的顺序执行
2.封闭性：程序运行时独占全机资源，执行结果不受外界影响
3.可再现性：程序执行的环境和初始条件相同，则重复执行的结果相同

并发执行

1.间断性：并发程序会受到其他程序的制约而暂停运行
2.失去封闭性
3.不可再现性

进程

定义

进程是具有独立功能的程序在一个数据集合上运行的过程，是系统进行资源分配和调度的独立单位

组成

程序段，数据段，PCB
创建与撤销进程实际上是创建与撤销PCB

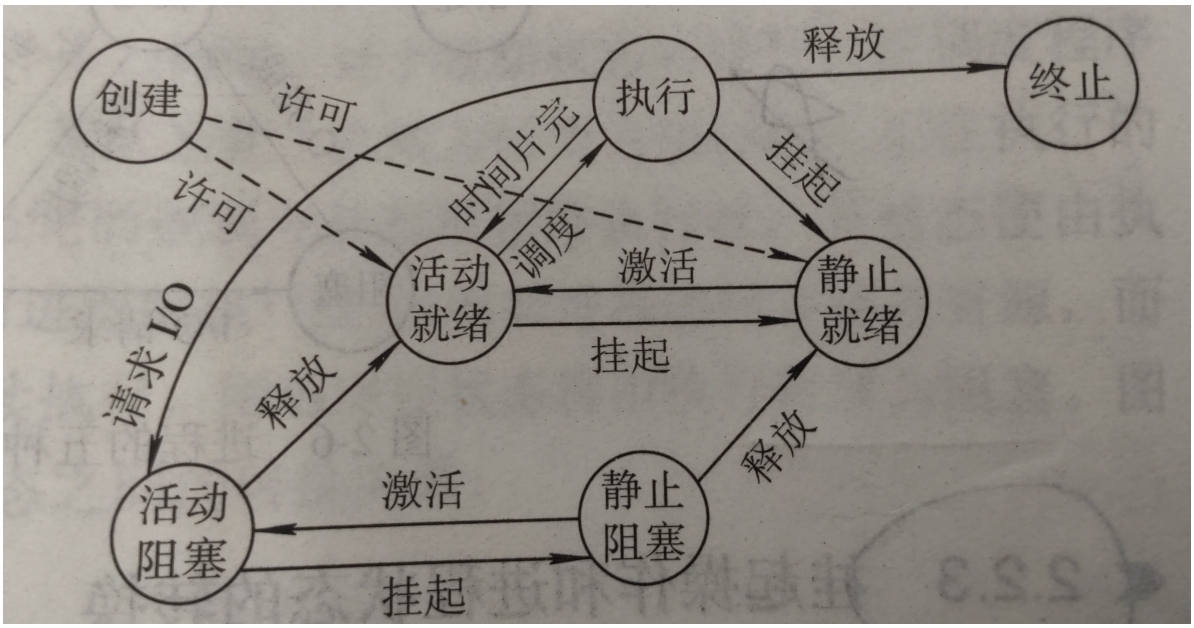
特征

动态性：进程最基本的特征，进程的实质是进程实体的执行过程
并发性：多个进程实体共存于内存中，能在一段时间内同时执行
独立性：进程能独立运行，独立接受资源，进行调度
异步性：进程按异步方式运行

状态转换

重点看执行态，两个就绪态，两个阻塞态五个状态之间的转换
创建态：创建PCB,分配除CPU外资源
就绪态：进程除CPU外所有资源均已分配好
阻塞态：进程由于发生某些事件，在**CPU**里暂停执行
执行态：进程在CPU内运行
终止态：撤销并返回PCB,保存记录
阻塞和就绪可以被挂起，即活动阻塞/静止阻塞与活动就绪/静止就绪，被挂起的进程会被放在外存，不

参与调度，但可以由阻塞态转为就绪态。



PCB功能

作为独立运行基本单位的标志
能实现间断性运行方式（保存现场信息）
提供进程管理所需要的信息
提供进程调度所需要的信息
实现与其他进程的同步与通信

同步

临界区

访问临界资源的代码

临界资源

一段时间内只允许一个进程访问的资源

同步机制原则

空闲让进：临界资源空闲时，允许一个请求进入临界区的进程立即进入临界区
忙则等待：已有进程访问临界资源时，其他试图进入临界区的进程必须等待
有限等待：进程在有限时间内进入自己临界区
让权等待：当进程不能进入临界区是，应立即释放处理机 *对应忙等*

信号量

仅能由同步原语进行操作的整型变量
wait(S):P操作，申请资源
signal(S):V操作，释放资源
　整形信号量：有忙等的缺点
　记录型信号量：S>0时，表示剩余资源数量；S=0时，S表示资源正好被分配完；S<0时，|S|表示阻塞队列中进程数目
　AND型信号量：分配多种资源，要么不分配，要么全分配
　信号量集：分配多种，多个资源

管程（细看）

定义

1.管程是**代表共享资源的数据结构**与**对该共享数据结构实施操作的一组过程**(*即PV操作等*)**所组成的资源管理程序**共同构成的**操作系统资源管理模块**
2.管程定义了一个**数据结构**和能由并发进程在这个数据结构上执行的**一组操作**，这组操作能同步进程，也可以改变管程中的数据
总之，管程就是 用于抽象资源的数据结构 和 用于同步进程的操作 两者的合称
每次仅有一个进程进入管程

组成

管程的名称
局部于管程的共享数据结构说明
对该数据结构进行操作的一组过程
对局部于管程的共享数据设置初始值的语句

与进程区别

1.数据结构
　进程定义的数据结构是私有的，即PCB（每个进程都有一个）
　管程定义的数据结构是全局公共的，如消息队列
2.对数据结构的操作
　进程是由顺序程序执行有关操作
　管程进行初始化操作与同步操作
3.设置目的
　设置进程的目的是实现系统的并发性
　设置管程的目的是解决共享资源的互斥使用问题
4.工作方式
　进程是主动工作方式，主动调用管程
　管程是被动工作方式，被进程调用
5.并发
　进程之间能并发执行
　管程不能与其调用者并发
6.动态性
　进程是动态的
　管程是一个静态的模块

经典进程同步问题（大题）

生产者-消费者问题
　记录型信号量的解决方式：P66
　注意：申请资源信号量与互斥信号量时，应先申请资源，最后申请互斥，防止死锁
　消费者进程在取消息时，也需要申请互斥信号量，防止覆盖（将要取出的结果）
　AND型信号量的解决方式：P67

线程和进程的比较

1.调度
在引入线程前，进程是调度和分派的基本单位
引入线程后，线程是调度和分派的基本单位
同一进程中线程的切换不会引起进程的切换，不同进程间线程的切换会导致进程的切换
2.并发
多个进程之间可以并发进行
一个进程中的多个线程也可以并发执行
3.资源
进程仍然是拥有资源的单位
进程的多个线程共享该进程的资源，线程本身不拥有系统资源，只有能保证独立运行的资源，如TCB，程序计数器，局部变量等
4.独立性
同一进程的不同线程之间的独立性比不同进程之间的独立性低很多
同一进程的不同线程拥有相同的地址空间
5.系统开销
创建/撤销进程的系统开销比线程大
6.多处理机
多线程进程可以在多个处理机上运行

内核态与用户态

内核支持线程

对线程的操作（创建，撤销，调度）在内核空间中完成，内核空间中为每个内核线程设置TCB
优点：
多处理机系统中，内核能同时调度同一进程的多个线程并行执行
进程中的一个线程被阻塞时，内核可以调度同一进程的其他线程，或不同进程中的线程
线程切换速度快，切换开销小（指需要保存的数据等）
内核本身可以采用多线程技术，提高系统执行速度
缺点：
用户的线程切换时，模式切换开销较大（指用户态与核心态之间的切换）

用户级线程

线程的操作与内核无关，TCB设置在用户空间，调度仍以进程为单位执行
优点：
线程切换不需要转换到内核空间，节省模式切换的开销
调度算法可以是进程专用的
用户级线程实现与操作系统无关，可以在不支持线程的平台上实现
缺点：
线程执行系统调用时，除该线程外，同一进程下的所有线程都会被阻塞
不能利用多处理机，进程中同时只有一个线程可以执行

组合态

用户级线程时分多路复用内核支持线程，使得一些内核支持线程能对应多个用户级线程
多对一模型：多个用户线程映射到一个内核控制线程
优点：
线程管理开销小，效率高
缺点：
同用户级线程
一对一模型：用户线程与内核支持线程一一对应

优点：
一个线程阻塞时可以调度另一个线程，并发功能更好
支持多个线程并行地运行在多处理机系统
缺点：
进程管理开销大

多对多模型，将多个用户线程映射到多个或更少的内核线程

第三章 调度

三种调度

高级调度（作业调度，长程调度）：
决定将外存中处于后备队列中的哪几个作业调入内存，为他们创建进程，分配资源，并将其放入就绪队列
对应状态转换中，创建->就绪的过程
对象：作业

中级调度（内存调度）：
将内存中暂时不能运行的进程调至外存等待，将外存中具备运行条件的就绪进程调入内存
对应状态转换中，活动xx->静止xx，静止xx->活动xx的过程，也相当于对换功能
目的为提高内存利用率与系统吞吐量

低级调度（进程调度/短程调度）
决定就绪队列中的哪个进程应获得处理机，并分配处理机
对应状态转换中，就绪->执行的过程

以上三种调度运行频率由上而下从低到高

调度目标

共同目标：
资源利用率
公平性(不发生进程饥饿现象)
平衡性
策略强制执行

对于批处理系统：
平均周转时间短
系统吞吐量高
处理机利用率高

对于分时系统：
响应时间快
均衡性

对于实时系统：
截止时间的保证
可预测性

计算指标

Ti(周转时间：从进入系统到离开系统)，Ts(服务时间，进程真正被执行的时间)

利用率

$$\text{CPU 的利用率} = \frac{\text{CPU 有效工作时间}}{\text{CPU 有效工作时间} + \text{CPU 空闲等待时间}}$$

平均周转时间

$$T = \frac{1}{n}\left[\sum_{i=1}^{n} T_i\right]$$

平均带权周转时间

$$W = \frac{1}{n}\sum_{i=1}^{n} \frac{T_i}{T_s}$$

调度算法（计算）

先来先服务FCFS

优点：

实现简单

缺点：

对短作业不利

短作业SJF

优点：

效率更高

缺点：

必须预知作业运行时间

对长作业不利

人机无法交互

未考虑作业紧迫程度

高响应比HRRN

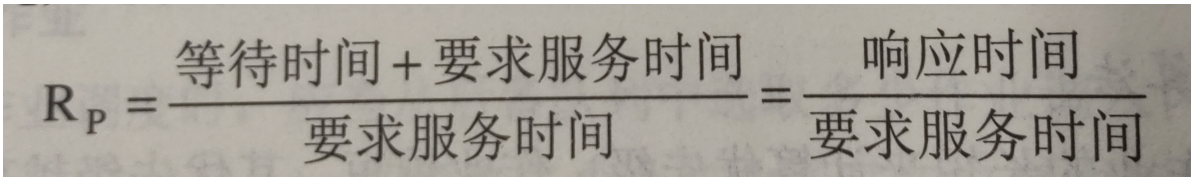
优点：

既考虑作业等待时间，又考虑作业运行时间，兼顾短作业与长作业，从而改善了处理机调度的性能

缺点：

系统开销较大

优先权：



最早截止时间优先EDF（适用于实时系统）：

计算每个进程的截止时间，然后根据紧迫程度进行调度

抢占

允许调度程序根据某种原则，暂停某个正在执行的进程，将已分配该进程的处理机重新分配给另一个进程

原则：

允许高优先级的新到进程抢占低优先级进程的处理机

允许新到的短进程抢占长进程的处理机

当前进程的时间片用完后，需要重新调度

时钟抢占与立即抢占

时钟抢占：

抢占发生在时钟中断时，延迟为几十到几毫秒

立即抢占：

当前进程只要未在临界区便立即进行抢占，延迟为几毫秒至100微秒

优先级倒置

高优先级进程被低优先级进程延迟或阻塞

发生原因

低优先级进程P1与高优先级进程P3共享临界资源，P1进入临界区后被中优先级进程P2抢占，P3由于无法申请临界资源（被P1占用，而P1因P2的抢占而无法执行）而被阻塞

解决方法

1.进入临界区的进程无法被抢占

缺点：若低优先级进程临界区较长，高优先级进程仍会等待较长时间

2.正在使用临界资源的低优先级进程继承申请使用该资源的高优先级进程的优先级

死锁

定义

一组进程中的每一个进程都在等待仅由该组进程中的其他进程才能引发的事件，此时称这组进程是死锁的

条件

互斥条件：进程对所申请的资源的使用是排他性的
请求和保持条件：进程请求自己没有的资源，同时不释放自己已有的资源
不可抢占条件：进程获得的资源在用完前不可由其他进程抢占，只能用完后由自己释放
循环等待条件：出现进程间相互等待资源的循环链

解除

预防：破坏四个必要条件中的一个
避免：在资源分配过程中防止系统进入不安全状态
检测：发生死锁时及时检测，然后采取措施将进程从死锁中解脱出来
解除：撤销进程，回收资源，分配给已处于阻塞态进程
从上到下，防范程度减弱，资源利用率提高，并发程度提高

预防方法

破坏请求与保持
1.一次性申请全部资源（*类似于AND型信号量*）
优点：
简单，易行，安全
缺点：
资源被严重浪费
进程经常会发生饥饿现象
2.进程在获得运行初期所需资源后便开始运行，运行过程中逐步释放自己已用毕的资源，并申请新的所需资源
优点：
提高设备利用率
减少进程发生饥饿的几率
破坏不可抢占：
当保持某些不可被抢占资源的进程，提出新的资源请求不被满足时，必须释放已保持的所有资源
破除循环等待：
对资源进行线性排序，进程必须按照序号递增的顺序请求资源。当已有高序号资源，需要申请低序号资源时，需要先释放高序号资源

银行家算法（计算）

安全序列：一个进程序号序列，系统按照这个序列分配资源推进进程，可以使每个进程都顺利完成
步骤：
1.检查分配的合理性：进程要求分配的资源是否超过进程需要的资源
2.检查分配的可行性：系统内部是否有足够的资源
3.尝试分配资源：修改系统中剩余资源，进程保持的资源和进程需要的资源
4.检查安全性：尝试找到一条安全序列以证明系统处于安全状态

第四章 内存管理

可重定位的动态装入

程序在装入内存后，并不直接将逻辑地址转化为物理地址，转换地址的过程是在程序真正运行时执行
需要重定位寄存器的支持

内存的分配

单一连续分配

整个内存的用户空间由某个程序独占
　只适用于单用户单任务操作系统

分区

将内存用户空间划分为若干个（大小可相同可不同）分区
　会导致存储空间的浪费，适用于控制多个对象的控制系统中

动态分区分配

数据结构

空闲分区表
　记录分区号，分区大小，分区起始地址与分区状态
空闲分区链
　设置分区大小与分区状态，并设置连接各个分区的双向指针

分配

1.找到合适大小的分区
2.根据分区大小，申请大小，剩余部分大小决定是否切割
3.将剩余部分（若有）留在空闲分区表（链）中，将分配区首地址返回给调用者

回收

1.回收区与前一个空闲分区邻接:
　修改前一分区的大小
2.回收区与后一个空闲分区邻接：
　修改后一个分区的首地址与大小
3.回收区同时与前后邻接：
　删除后一个表项，修改前一个表项的大小跟首地址
4.回收区不与别的空闲分区邻接：
　建立一个新表项，根据地址插入到空闲链的适当位置

分区算法

首次适应FF：
　地址递增排序，从链首开始查找
　低址碎片多，能留下大内存空间
循环首次适应NF：
　地址递增排序，从上次分配的下一个分区开始查找
　缺乏大空闲分区，空闲分区分布均匀
最佳适应BF：
　容量递增排序，从链首找第一个满足要求的

碎片多
最坏适应WF：
容量递减排序，从链首找第一个满足要求的
产生碎片可能性最小，对小作业有利
快速适应：
根据容量大小分类，对每一类建立一个链表
分配时根据进程长度找索引表中对应的链表
归还时算法复杂，存在浪费
伙伴系统：略
哈希算法：略

紧凑

通过移动内存中作业的位置，将多个小分区拼接成一个大分区
会导致物理地址的变化，需要动态重定位解决

对换

将内存中暂时不能运行或者暂时不用的程序和数据换出到外存上

解决被阻塞的进程占用大量内存空间，能执行的进程由于内存空间不足驻留外存，导致系统资源浪费，系统吞吐量下降的问题
内存紧张但cpu空闲

对换类型

整体对换
以进程为单位，将整个进程换出到外存上（相当于中级调度）
页面对换
以进程的页面或分段为单位，将进程的某个页面或分段换出到外存

对换空间管理

文件区
主要目的是提高文件存储空间的利用率，需要采取离散分配的管理方式
对换区
主要目的是提高进程换进换出的速度，需要采用连续分配的管理方式

分页（计算）

将程序的地址空间分成若干固定大小的“页”，将内存空间分为若干个与页大小相同的“物理块”
计算内存的块数

页表（计算）

页表寄存器：页表起始地址+页表长度
分页地址结构：页号+位移量
页号占的位数决定作业中最多有多少页
位移量占的位数决定页面的大小
页表结构：页号+块号
用于将页号与块号对应
逻辑地址->分页地址：
带余除法：逻辑地址/页面大小 = 页号 ..... 位移量

分页地址->物理地址：
　根据页号在页表中查找对应块号，块号*块大小=起始地址，起始地址+位移量=物理地址
检索页表前，先判断页号与页表长度，页号大于页表长度会产生越界中断

快表

在分页存储管理中，CPU访问数据时需要访问两次内存，第一次访问页表，第二次根据页表访问数据
快表：
　一个具有并行查询能力的缓冲寄存器
查询流程：
　计算出页号后先在快表中查询，查询成功便直接取物理块号访问内存
　查询失败则访问页表，将该项存入快表中，若快表已满，将快表中较老的一项换出。

访问内存有效时间EAT

t:访问内存时间 a：快表命中率 l:访问快表速度
EAT=a*l+(t+l)(1-a)+t=2t+l-t*a
可以只记前半部分，不记化简后的，前半部分比较贴合实际意义

分段

段表寄存器：段表起始地址+段表长度
分段地址结构：段号+段内地址
　段号的位数决定作业最多有多少个段
　段内地址的位数决定每个段的最大长度
段表结构：段长+基址
　检索段表前，先判断段号是否超过段标长度，然后检查段内地址是否超过该段的段长
　若超过，产生越界中断

分段分页区别

1.单位含义
　页是信息的物理单位，是系统的行为，对用户透明，与信息内容无关
　段是信息的逻辑单位，包含一组相对完整的信息
2.固定大小
　页的大小固定，由系统决定，直接由硬件实现
　段的大小不固定，由用户编写的程序决定，编译程序在对源程序进行编译时根据信息的性质划分
3.地址空间性质
　分页的用户程序地址空间是一维线性的，只需要一个记忆符便可确定
　分段的用户程序地址空间是二维的，标识是既需要给出段名，也需要给出段内地址

段页式

段表寄存器：段表起始地址+段表大小
段页式地址结构：段号+段内页号+页内地址
段表结构：页表起始地址+页表长度
相当于页表寄存器放在了段表中，每个段有对应的页表

第五章 虚拟存储器

虚拟存储器

定义

具有请求调入功能和置换功能，能从逻辑上对内存容量加以扩充的存储器系统

特征

多次性：
　作业中的程序和数据可以分成多次调入内存
　虚拟存储器最重要的特征
对换性：
　作业中的程序和数据可以在作业的运行过程中换进换出
虚拟性：
　能从逻辑上扩充内存容量，使用户所看到的内存容量远大于实际内存容量
　虚拟存储器**表现出来**的最重要的特征
　实现虚拟存储器最重要的目标
虚拟性以多次性和对换性作为基础，多次性和对换性建立在离散分配的基础上

请求分页

页表

相较于传统的页表只有页号与物理块号 新页表的结构如下：
页号 物理块号 状态位 访问字段 修改位 外存地址
状态位：用于记录该页是否已经被调入内存
访问字段：用于记录页面在一段时间内被访问的次数，或本页已有多久未被访问
修改位：用于标识该页在调入内存后是否被修改过
外存地址：记录该页在外存上的地址，通常是物理块号（注意页表中“物理块号”一项指的是内存中的物理块号）

缺页中断

当访问的页面不在内存时，会发生缺页中断
与一般中断的区别：
　在指令执行期间产生和处理中断信号
　一条指令执行期间可能会产生多次缺页中断

内存分配策略

固定分配局部置换：
　每个进程的物理块数固定，进程在缺页时只在分配给本进程的页面中选择一页换出
　无法确定每个进程应分配的物理块数
可变分配全局置换：
　进程的物理块数可在运行的过程中增加或减少，缺页时，挑选系统中保留的（原本不属于该进程的）空闲物理块分配给该进程，或在所有进程的所有物理块中选一块换出，将所缺页换入
　一个进程缺页可能会导致其他进程缺页率增加
可变分配局部置换：
　进程的物理块数在频繁发生缺页中断时增加，在缺页率特别低时减少。缺页时，只允许在该进程的物理块中选择一块换出

调页来源

1.系统对换区空间足够：
　全部从对换区调入，需要先将与该进程有关的文件复制到对换区
2.系统对换区空间不足
　不会被修改的页面从文件区调入，换出时不重新写到磁盘，可能被修改的页面换出时放到对换区，以便之后调入
3.UNIX方式：
　未运行过的页面从文件区调入，曾经运行过又被换出的页面从对换区调入。页面共享情况下，某进程所请求的页面可能被其他进程调入内存，此时无需从兑换区调入

缺页率

注意缺页与页面置换次数的区别
缺页率=缺页次数/页面访问总次数
与 页面大小 分配给进程物理块的数目 页面置换算法 程序固有特性 有关

页面置换算法（计算）

最佳置换算法：
　理想中的，选中未来最长时间不被访问的页面
先进先出算法：
　换出最先进入内存的页面
最近最久未使用：
　换出内存中最久未被使用的页面
CLOCK：
　页面组成循环队列，被访问时将访问位置为1，换出时检查访问位，若为0将其换出，若为1将其置0。由于是循环，最后一定能找到一个页面将其换出
改进CLOCK:
　与CLOCK相比，增加了修改位，检查方法如下：检查未访问未修改->检查未访问已修改，将访问位置0->将所有访问位都置0，重复第一步->重复第二步

页面缓冲算法

设置空闲页面链表与修改页面链表
读入页面时，可以将该页放在空闲页面链首
页面换出时不换出到外存，而放到空闲页面链尾
已修改页面换出时，可以将其放在修改页面链尾，而不立即写回外存

访问内存时间的影响因素

l：查找快表时间 t：实际访问物理地址时间 e:缺页中断处理时间 a:命中率 f:缺页率
被访问页在内存中，页表项在快表中
　EAT=l+t
被访问的在内存中，页表项不在快表中
　EAT=l+t+l+t
被访问页不在内存中：
　EAT=l+a*t+(1-a)*[t+f*(e+l+t)+(1-f)*(l+t)]

抖动

进程的大部分时间用于换进换出，处理机的利用率急剧下降
原因：在系统中运行的进程太多，分配给每一个进程的物理块太少，进程在运行时频繁发生缺页

第六章 IO

IO系统的层次结构

用户层软件：实现用户交互，提供库函数，产生请求，实现SPOOLing
IO系统接口
设备独立性软件:实现设备的统一接口，设备命名，分配，保护，释放，数据缓冲
设备驱动程序：具体实现系统对设备的指令，驱动IO设备工作的驱动程序，由设备厂商提供
中断处理程序：保存中断进程的CPU环境，转到设备的中断处理程序，直接与硬件进行交互
RW/HW接口
设备控制器

IO通道

目的：建立独立的IO操作，将数据的传送和对IO操作的组织，管理与结束操作独立于CPU
分类：
字节多路通道：
含有许多非分配型子通道，按照时间片轮转方式共享主通道，每个子通道每次传输一字节
不适用于高速设备
数组选择通道
只含有一个分配型子通道，一段时间内只允许一个设备独占地数据传送，传输以数据块为单位，数组方式进行
利用率很低
数组多路通道
含有多个*分配型子通道（存疑，与书上不同但书上的不合逻辑）*，综合以上两者优点

IO控制方式

轮询
发出IO指令后，处理机不断查询控制器中状态寄存器的busy标志，=1表示未输入完，=0表示输入完
CPU处于忙等状态，绝大部分时间处于等待IO设备完成的时间，造成CPU的极大浪费
中断
发出IO指令后，CPU与IO设备并行运行。数据进入数据寄存器后，设备控制器向CPU发送中断信号
以字符为单位传输，适用于字符型设备
DMA
需要IO时，向磁盘控制器发送读命令，命令被存到命令寄存器中。同时，需要读入数据的起始地址送入内存地址寄存器，读入字节数送入数据计数器中，磁盘中的源地址送入DMA控制器的IO逻辑上，然后开始数据传输。传输过程为以数据计数器中的值为条件的循环
以数据块为基本单位传输，在传输数据块的开始或结束时，才需要CPU干预。传输数据由设备直接送入内存，或者相反。
IO通道
DMA的发展，可以传输多个数据块。
通道指令格式：操作码，通道程序结束位P，记录结束标志R，计数，内存地址
操作码规定操作，如读，写，控制
通道程序结束位P表示执行完该条指令后通道程序是否结束
记录结束标志R表示本条指令是否为处理某条记录的最后一条指令。

设备独立性

应用程序使用逻辑名，实际执行使用物理名，系统需具有将逻辑名转化为物理名的功能
应用程序所使用的设备不局限于某个具体的物理设备，为每个设备所配置的设备驱动程序是与硬件紧密相关的软件。

SPOOLing

联机情况下同时实现外围操作的技术，外围输入输出操作与CPU对数据的处理同时进行

组成

输入井和输出井：磁盘上开辟的存储区域，模拟脱机输入输出的磁盘
输入缓冲区和输入缓冲区：内存中开辟的两个缓冲区，用于缓和CPU和磁盘之间速度不匹配之间的矛盾，在设备与输入\输出井间起到缓冲作用
输入进程与输出进程：模拟脱机输入\输出时的外围控制机，控制数据在设备-缓冲区-井之间的过程
井管理程序：控制作业与磁盘井之间信息的交换

假脱机管理系统P222，P223

输入设备：磁盘 输入缓冲区：磁盘缓冲区
输出设备：打印机 输出缓冲区：打印缓冲区 输出进程：假脱机打印进程
井管理程序：假脱机管理进程

实现

系统不即时执行程序输出数据的真正打印操作，而是即时将数据输出到缓冲区，此时数据并未被真正地打印，只是让用户感觉系统已经为他打印。真正的打印操作，是在打印机空闲且该打印操作在等待队列已排到队首是进行的。打印操作本事也是利用CPU的一个时间片，没有使用专门的外围机。以上操作对用户透明。

磁盘访问时间

寻道时间Ts:启动磁臂的时间s+磁头移动n条磁道的时间
　Ts = s+m*n
旋转延迟时间Tr:指定扇区移动到磁头下面所经历的时间
　与磁盘转速有关，一般为磁盘旋转一周周期的一半
传输时间Tt:将数据从磁盘读出或向磁盘写入数据所经历的时间
　与磁盘转速和读\写字节数b有关
　Tt=b/r*N r为磁盘每秒转速，N为一条磁道上的字节数

磁头调度算法（计算）

先来先服务FCFS：略
最短寻到时间有限SSTF：找最近的
扫描SCAN：
　只考虑当前磁头移动方向的最近的磁道，外->里->外
　“电梯调度算法”
循环扫描CSCAN：
　磁头只能单向移动的SCAN
　外->里　外->里

第七章 文件管理

文件的结构

逻辑结构

用户观点观察到的文件组织形式，文件是由一系列的逻辑记录组成的，是用户可以直接处理的数据及其结构

物理结构

系统将文件存储在外存上所形成的存储组织形式，是用户不能看见的

对目录的管理要求

1.实现“按名存储”：用户只需要向系统提供所需访问文件的名字，便能快速准确找到指定文件在外存上的存储位置
2.提高对目录的检索速度
3.允许文件共享
4.允许文件重名

第八章 磁盘管理

FAT表

引入“卷”的概念，支持将一个物理磁盘分成四个逻辑磁盘，可以单独进行管理。
FAT后的数字代表表项所占的位数，可以计算出FAT表中最多的表项数。
一个表项可以对一个盘块（512B）进行管理，也可以对一个簇进行管理（1，2，4，8个盘块）
FAT表的大小=FAT表项数*表项的大小（如FAT12占12位，即1.5B）

位示图

分配：
　顺序找到一个值为0的二进制位
　将坐标[i][j]（i行j列）转化为盘块号b
　　设位示图的下标从1开始，n为位示图的字长（即每行的位数）
　　b=n(i-1)+j
　修改位示图，令该二进制位=1
回收：
　将盘块号转换为位示图中的行号与列号
　　带余除法：b-1/n=(i-1).....(j-1)
　修改位示图，令二进制位=0

成组链接法

将空闲盘块的盘块号压栈，每压100个便将这个栈连同栈中的空闲盘块号存入一个新的空闲盘块，并将该盘块压入一个新栈