# 操作系统定义：管理和控制计算机系统中各种软硬件资源、合理地组织计算机工作流程的系统软件，是用户与计算机之间的接口

# 操作系统特征：并发、共享（两个最基本特征）、虚拟、异步

* 并发：宏观上多个事情在同一时间发生，微观上分时交替执行
* 共享：系统中资源可供内存中多个并发执行的进程共同使用
  + 互斥共享
  + 同时访问
* 虚拟：将一个物理上的实体变为若干个逻辑上的对应物，如虚拟处理器、虚拟内存和虚拟外设
* 异步：进程的执行不是一贯到底，而是走走停停，以不可预知的速度向前推进

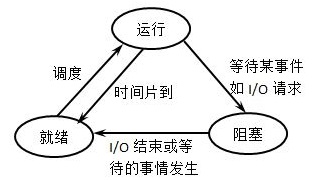
# 操作系统功能

* 管理系统资源
  + 处理机管理（进程管理）：进程控制、进程同步、进程通信和进程调度
  + 存储器管理（内存管理）：内存分配、内存保护、地址映射、内存扩充
  + 设备管理：完成用户的IO请求；为用户进程分配IO设备；提高IO设备利用率；提高IO速度；方便IO的使用
  + 文件管理：磁盘存储空间管理，目录管理，文件读写管理以及文件共享和保护
* 提供用户接口
  + 程序接口：API
  + 命令接口：联机控制方式、脱机控制方式
* 用作扩充机器

# 进程基本概念

* 定义：一个具有独立功能的程序在某个数据集运行的过程
* 和程序的区别
* 进程是动态的，程序是静态的：程序本身可以作为软件资源长期存在；进程是一次执行过程，有一定生命周期
* 进程具有并发特征，程序没有
* 程序和进程没有一一对应关系
  + 多次执行一个程序可以得到多个进程
  + 一个进程可以调用多个程序
* 进程组成：**程序段+数据段+进程控制块PCB（进程标志信息、进程控制信息、进程资源信息、CPU现场信息）**

# 进程常见状态及各种状态的转换条件



# 进程控制

* 进程创建
  + 为新进程分配一个唯一的进程标识号，并申请一个空白的PCB
  + 为进程分配资源，为新进程的程序和数据及用户栈分配必要的内存空间。若资源不足，则进入阻塞态
  + 初始化PCB
  + 若就绪队列能够接纳新进程，则将新进程调入就绪队列，等待被调度运行
* 进程终止
  + 根据被终止进程PID，检索PCB，从中读取出该进程的状态
  + 若处于执行状态，则终止执行，将处理机资源分配给其他进程
  + 若该进程还有子进程，将所有子进程终止
  + 将该进程所拥有的全部资源还给父进程，或归还给操作系统
  + 将该PCB从所在队列中删除
* 进程阻塞：是进程自身的一种**主动**行为，只有处于运行态的进程才可能转换为阻塞态
  + 找到要被阻塞进程的PID对应的PCB
  + 保护现场，将状态转换为阻塞态，停止运行
  + 将该PCB插入相应事件的等待队列
* 进程唤醒
  + 在该事件的等待队列中找到相应进程的PCB
  + 移出等待队列，转换为就绪态
  + 将该PCB插入就绪队列，等待调度
* 进程切换
  + 保存处理机上下文，包括PC和其他寄存器
  + 更新PCB信息
  + 把进程的PCB移入相应的队列，如就绪或某事件的等待队列
  + 选择另一个进程执行，更新其PCB
  + 更新内存管理的数据结构
  + 恢复处理机上下文

# 进程通信方式：共享存储器、消息传递系统、管道通信系统

* + 共享存储器系统：存储器中划分共享存储区
  + 消息传递系统：进程间的数据交换以消息为单位
  + 管道通信系统：管道——连接读写进程以及实现它们之间通信的共享文件；命名管道可以跨网络通信，匿名管道不可以跨网络

# 进程执行模式：用户模式和内核模式

* + 用户模式：进程在用户空间中运行，只能访问用户空间
  + 内核模式：进程在内核空间中运行，可以访问内核空间和用户空间
  + 区别：权限不同
  + 用户模式到内核模式的切换
    - 系统调用：应用程序需要使用OS提供的服务时发出请求
    - 异常：在用户模式下运行时发生了不可知异常，触发处理此异常的内核中断处理程序
    - 外设中断：外设完成用户的操作后向CPU发出中断信号，执行中断处理程序

# 进程同步：同步任务和遵循原则

* + 主要任务：对多个相关进程在执行次序上进行协调，以使并发执行的诸进程之间能有效地共享资源和相互合作，从而使程序的执行具有**可再现性**
  + 同步机制遵循原则
    - 空闲让进
    - 忙则等待：保证对临界区的互斥访问
    - 有限等待：有限代表有限的时间，避免死等
    - 让权等待：当进程不能进入自己的临界区时，应该释放处理机，以免陷入忙等状态
  + 同步机制
    - 原子操作
    - 信号量机制
    - 自旋锁管程（monitor process）
    - 会合
    - 分布式系统

# 进程与线程的区别和联系：定义；区别：地址空间、大小、通信方式、用途、调度消耗；联系：包含、逻辑与物理、资源分配

* + 定义
    - 进程是具有一定独立功能的程序关于某个数据集合上的一次活动，是系统进行资源分配和调度的一个独立单位，**实现了OS的并发**
    - 线程是进程的一个实体，是CPU调度和分派的基本单位，是比进程更小的能独立运行的基本单位，**实现了进程内部的并发**
  + 关系：在没有实现线程的操作系统中，进程既是资源分配的基本单位，又是调度的基本单位，它是系统中并发执行的单元。而在实现了线程的操作系统中，进程是资源分配的基本单位，但是线程是调度的基本单位，是系统中并发执行的单元
  + 区别
    - 一个程序至少有一个进程，一个进程至少有一个线程，线程依赖于进程而存在；
    - 进程在执行过程中拥有独立的内存单元，而多个线程共享进程的内存空间。
    - 属于一个进程的所有线程共享该进程的所有资源，包括打开的文件，创建的Socket等。不同的进程互相独立。
    - 线程又被称为轻量级进程。进程有进程控制块，线程有线程控制块。但线程控制块比进程控制块小得多。线程间切换代价小，进程间切换代价大。
    - 进程是程序的一次执行，线程可以理解为程序中一段程序片段的执行

# 进程调度：三种类型；两种策略；两个度量指标；七个算法

* + 类型
    - 作业调度：把后备作业调入内存运行
    - 进程调度：把就绪队列中的某个进程获得CPU
    - 对换：在内外存对换区进行进程对换
  + 非抢占式调度与抢占式调度
    - 非抢占：分派程序一旦把处理机分配给某进程后便让它一直运行下去，直到进程完成或发生某事件而阻塞时，才把CPU分配给另一个进程
    - 抢占：操作系统将正在运行的进程强行暂停，由调度程序将CPU分配给其他就绪进程的调度方式
  + 调度策略设计
    - 响应时间: 从用户输入到产生反应的时间
    - 周转时间: 从任务开始到任务结束的时间
  + 调度算法
    - FCFS：调度顺序即任务到达就绪队列顺序；公平、简单、非抢占、不适合交互；未考虑任务特性，平均等待时间可以缩短
    - SJF：保证最小平均等待时间，非抢占
    - SRJF：SJF可抢占版本，比SJF更有优势；通过指数平均法预测下一CPU区间大小
    - 优先权调度：调度优先权最高任务；会产生饥饿现象——优先权低的任务一直就绪得不到运行
    - RR调度：设置时间片按时间片来轮转调度；优点: 定时有响应，等待时间较短；缺点: 上下文切换次数较多；时间片太大，响应时间太长；吞吐量变小，周转时间变长；当时间片过长时，退化为FCFS
    - 多级队列调度：按照一定的规则建立多个进程队列，不同的队列有固定的优先级（高优先级有抢占权），不同的队列可以给不同的时间片和采用不同的调度方法；没法区分I/O bound和CPU bound，也存在一定程度的“饥饿”现象
    - 多级反馈队列：在多级队列的基础上，任务可以在队列之间移动，更细致的区分任务，可以根据“享用”CPU时间多少来移动队列，阻止“饥饿”，**最通用的调度算法**

# 死锁：进程被无限期阻塞、相互等待的状态；四个必要条件；四种处理策略

* 必要条件
  + 互斥：资源不能被共享，只能有一个进程使用
  + 持有并等待：一个进程因请求资源而阻塞时，对已获得的资源保持不放
  + 非抢占式调度：采用非抢占策略调度进程
  + 循环等待：系统中若干进程组成环路，该环路中每个进程都在等待相邻进程正占用的资源
* 处理死锁
  + 死锁预防：破坏除互斥外的三个条件之一
  + 死锁避免：动态地检测资源分配状态，以确保循环等待条件不成立，从而确保系统处于安全状态。安全状态是指：如果系统能按某个顺序为每个进程分配资源（不超过其最大值），那么系统状态是安全的
  + 死锁检测和解除
  + 忽视死锁

# 线程同步方式

* 临界区：对多线程的串行化来访问公共资源或一段代码
* 互斥锁：使同时只能有一个进程访问临界资源
* 信号量：使同时有固定多个进程访问临界资源
* 事件：通知线程有一些事件已发生，从而启动后继任务的开始保持多线程同步，也可方便的实现多线程优先级的比较操作

# 线程实现方式：按内核是否对线程可感知

* 用户线程：应用程序管理，创建效率高，并发效率低
* 内核线程：内核管理，创建效率低，并发效率高
* 临界资源
  + 定义：同一时间内只能被一个进程所占用的资源
  + 对于临界资源的访问，必须是互斥进行，也就是当临界资源被占用时，另一个申请临界资源的进程会被阻塞，直到其所申请的临界资源被释放。而进程内访问临界资源的代码被成为**临界区**

# 内存管理方式

* 块式管理：按块分配内存，易于管理但浪费大
* 页式管理：按页分配，空间利用率高
* 段式管理：按段分配，空间利用率更高，但是不方便管理
* 段页式管理：将程序分为若干段，每段分成若干页

# 分段和分页的区别

* 段式管理：符合用户视角的内存分配方案，将程序划分为若干段，如代码段，数据段，堆栈段。这样每个进程都有一个二维地址空间，相互独立，互不干扰。优点：无内碎片；缺点：有外碎片
* 页式管理：用户视角内存与物理内存相分离的内存分配方案，将程序逻辑地址划分为固定大小的页，物理内存划分为同样大小的帧，可将任意一页放入任意一帧，实现离散分配。优点：无外碎片；缺点：有内碎片
* 不同
  + 目的：分页是由于系统管理的需要而非用户的需要，是信息的物理单位；分段是为了更好地满足用户的需要，是信息的逻辑单位
  + 大小：页的大小固定且由系统决定，段的长度不固定且由所完成的功能决定
  + 地址空间：段向用户提供二维地址空间，页向用户提供一维地址空间
  + 信息共享：段是信息的逻辑单位，便于存储保护和信息共享；页的保护和共享收到限制
  + 内存碎片：段式产生外碎片，页式产生内碎片

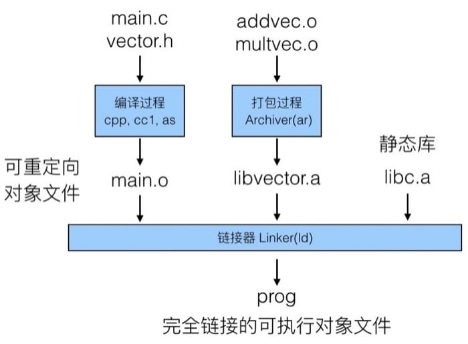
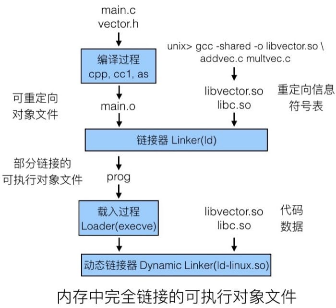
# 程序的执行流程：4个

* 预处理
* 编译
* 汇编
* 链接

# 内存池、进程池、线程池

* **池化技术：提前保存大量的资源，以备不时之需以及重复使用**
* 线程池：先启动若干数量的线程，并让这些线程都处于睡眠状态，当需要一个开辟一个线程去做具体的工作时，就会唤醒线程池中的某一个睡眠线程，让它去做具体工作，当工作完成后，线程又处于睡眠状态，而不是将线程销毁
* 进程池与线程池同理
* 内存池：内存池是指程序预先从操作系统申请一块足够大内存，此后，当程序中需要申请内存的时候，不是直接向操作系统申请，而是直接从内存池中获取；同理，当程序释放内存的时候，并不真正将内存返回给操作系统，而是返回内存池。当程序退出(或者特定时间)时，内存池才将之前申请的内存真正释放

# 动态与静态链接库区别

* 静态链接库：一个外部函数与变量的集合体，要使用其中一部分代码则需要全部包含并编译，最终产生完全链接的可执行文件，速度快，但会产生大量重复
* 
* 动态库：可被所有程序共享，在运行时被装载，速度慢
* 

# 虚拟内存

* 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得一种存储器系统。其逻辑容量由内存之和和外存之和决定
* 特征
  + 多次性，是指无需在作业运行时一次性地全部装入内存，而是允许被分成多次调入内存运行
  + 对换性，是指无需在作业运行时一直常驻内存，而是允许在作业的运行过程中，进行换进和换出
  + 虚拟性，是指从逻辑上扩充内存的容量，使用户所看到的内存容量，远大于实际的内存容量
* 实现方式
  + 请求分页存储管理
  + 请求分段存储管理
* 页面置换算法
  + 背景：操作系统将内存按照页面进行管理，在需要的时候才把进程相应的部分调入内存。当产生缺页中断时，需要选择一个页面写入。如果要换出的页面在内存中被修改过，变成了“脏”页面，那就需要先写会到磁盘
  + 目的：选出一个要换出的页面，使得页面置换的效率最高
  + 种类
    - 最优页面置换算法：调出最不可能再次被用到的页面，只存在理论中
    - FIFO算法：调出最先进入队列的算法
    - LRU算法：调出最近最少未使用的页面，赋予每个页面一个访问字段，用来记录一个页面自上次被访问以来所经历的时间 t，当须淘汰一个页面时，选择现有页面中其 t 值最大的
* 中断与系统调用
  + 中断定义：在计算机执行程序的过程中，由于出现了某些特殊事情，使得CPU暂停对程序的执行，转而去执行处理这一事件的程序。等这些特殊事情处理完之后再回去执行之前的程序
  + 中断种类
    - 内部异常中断：由计算机硬件异常或故障引起的中断
    - 软中断：由程序中执行了引起中断的指令而造成的中断，**如系统调用**
    - 外部中断：由外部设备请求引起的中断
  + 中断优先级：中断需要被处理的紧急程度。每个中断都有一个对应的优先级，当处理器在处理某一中断的时候，只有比这个中断优先级高的中断可以被处理器接受并且被处理。优先级比这个当前正在被处理的中断优先级要低的中断将会被忽略。
* 逻辑地址、物理地址、虚拟内存
  + 逻辑地址：计算机用户看到的地址，是相对的连续的
  + 物理地址：CPU看到的地址，是数据在内存中的真实地址
  + 虚拟内存：操作系统可以通过将部分不太常用的数据移出内存，“存放到价格相对较低的磁盘缓存，以实现内存扩展。虚拟内存空间相对磁盘而言要小很多，因此，即使搜索虚拟内存空间也比直接搜索磁盘要快。唯一慢于磁盘的可能是，内存、虚拟内存中都没有所需要的数据，最终还需要从硬盘中直接读取。计算机通过TLB（联想寄存器）来实现虚拟地址到物理地址的快速转换
    - Resident Set：当一个进程在运行的时候，操作系统不会一次性加载进程的所有数据到内存，只会加载一部分正在用，以及预期要用的数据。其他数据可能存储在虚拟内存，交换区和硬盘文件系统上。被加载到内存的部分就是resident set
    - Thrashing（颠簸）：CPU大部分时间在处理缺页中断的现象
* 碎片
  + 内碎片：已分配空间大于请求的所需空间，出现在页式虚拟内存系统
  + 外部碎片：还没有分配的但是由于太小而无法分配给新进程的空间，出现在段式虚拟内存系统
* 分页和分段有什么区别
  + 段是信息的逻辑单位，它是根据用户的需要划分的，因此段对用户是可见的 ；页是信息的物理单位，是为了管理主存的方便而划分的，对用户是透明的。
  + 段的大小不固定，有它所完成的功能决定；页大大小固定，由系统决定
  + 段向用户提供二维地址空间；页向用户提供的是一维地址空间
  + 段是信息的逻辑单位，便于存储保护和信息的共享，页的保护和共享受到限制
* 同步和互斥的区别
  + 同步：不同进程之间的程序片段，运行必须严格按照某种次序来完成特定的任务。同步往往实现了互斥，尤其是对写资源的情况。互斥是同步的必要条件
  + 互斥：不同进程之间的程序片段，当某个进程运行其中一个程序片段时，其他进程中的所有程序片段无法运行
* 线程安全：多线程程序运行的结果是可预期的，而且与单线程运行的结果一样，则说明线程安全
* 同步与异步
  + 同步
    - 定义：是指一个进程在执行某个请求的时候，若该请求需要一段时间才能返回信息，那么，这个进程将会一直等待下去，直到收到返回信息才继续执行下去
    - 特点
      * 阻塞模式
      * 按顺序执行，需要等待，协调运行
  + 异步
    - 定义：进程不需要一直等下去，而是继续执行下面的操作，不管其他进程的状态。当有消息返回时系统会通知进程进行处理，这样可以提高执行的效率
    - 特点
      * 阻塞模式，无需等待
      * 彼此独立，等待过程中可以继续执行。线程是异步的一个实现方式
  + 优缺点
    - 同步可以避免死锁、读脏数据的发生，但效率低
    - 异步效率高，安全性低
* 系统调用和库函数
  + 系统调用：程序向系统内核请求服务的方式，发生在内核空间，移植性差，空间切换开销大
  + 库函数：把常用的函数编写在一个文件里，编写应用程序时调用，发生在用户空间，移植性好，开销小
* 守护进程、僵尸进程、孤儿进程
  + 守护进程：运行在后台的一种特殊进程，独立于控制终端并周期性地执行某些任务
  + 僵尸进程：一个进程 fork 子进程，子进程退出，而父进程没有wait/waitpid子进程，那么子进程的进程描述符仍保存在系统中，这样的进程称为僵尸进程
  + 孤儿进程：一个父进程退出，而它的一个或多个子进程还在运行，这些子进程称为孤儿进程
* IO多路复用
  + 定义：内核一旦发现进程指定的一个或者多个IO条件准备读取，它就通知该进程
  + 适用场合
    - 当客户处理多个描述字时（一般是交互式输入和网络套接口），必须使用I/O复用
    - 如果一个TCP服务器既要处理监听套接口，又要处理已连接套接口，一般也要用到I/O复用
    - 如果一个服务器即要处理TCP，又要处理UDP，一般要使用I/O复用
    - 如果一个服务器要处理多个服务或多个协议，一般要使用I/O复用
  + 优点：系统不用创建进程/线程也不必维护他们，系统开销小
* 线程共享资源和独占资源问题：一个进程中的所有线程共享该进程的地址空间，但它们有各自独立的（/私有的）栈(stack)
* 缓冲区溢出
  + 定义：计算机向缓冲区填充数据时超出了缓冲区本身的容量，溢出的数据覆盖在合法数据上。
  + 危害：利用堆栈溢出，在函数返回时改变返回程序的地址，让其跳转到任意地址
    - 程序崩溃，导致拒绝服务
    - 跳转并且执行一段恶意代码
  + 主要原因：程序中没有仔细检查用户输入