导学阶段：

一、操作系统概述

1.1 操作系统的定义与目标

定义：操作系统是控制管理计算机系统的硬软件，分配调度资源的系统软件。

目标：方便性，有效性（提高系统资源的利用率、提高系统的吞吐量），可扩充性，开放性。

Shell（命令行接口）、GUI（图形用户接口）、Kernel（操作系统的内部）

1.2 操作系统的基本功能

统一管理计算机资源：处理器资源，IO设备资源，存储器资源，文件资源;

实现了对计算机资源的抽象：IO设备管理软件提供读写接口，文件管理软件提供操作文件接;

提供了用户与计算机之间的接口：GUI（图形用户界面），命令形式，系统调用形式。

1.3 操作系统的特征

最基本的特征，互为存在条件：并发，共享；

（1）并行：指两个或多个事件可以在同一个时刻发生，多核CPU可以实现并行，一个cpu同一时刻只有一个程序在运行；

（2）并发：指两个或多个事件可以在同一个时间间隔发生，用户看起来是每个程序都在运行，实际上是每个程序都交替执行。

（3）共享性：操作系统的中资源可供多个并发的程序共同使用，这种形式称之为资源共享。

互斥共享：当资源被程序占用时，其它想使用的程序只能等待。

同时访问：某种资源并发的被多个程序访问。

虚拟和异步特性前提是具有并发性。

（4）虚拟性：表现为把一个物理实体转变为若干个逻辑实体。

时分复用技术：资源在时间上进行复用，不同程序并发使用，多道程序分时使用计算机的硬件资源，提高资源的利用率。

空分复用技术：用来实现虚拟磁盘（物理磁盘虚拟为逻辑磁盘，电脑上的C盘、D盘等）、虚拟内存（在逻辑上扩大程序的存储容量）等，提高资源的利用率，提高编程效率。

（5）异步性：在多道程序环境下，允许多个进程并发执行，但由于资源等因素的限制，使进程的执行以“停停走走”的方式运行，而且每个进程执行的情况（运行、暂停、速度、完成）也是未知的。

1.4 操作系统的中断处理

中断机制的作用：为了在多道批处理系统中让用户进行交互；

中断产生：

发生中断时，CPU立马切换到管态，开展管理工作；（管态又叫特权态，系统态或核心态，是操作系统管理的程序执行时，机器所处的状态。）

发生中断后，当前运行的进程回暂停运行，由操作系统内核对中断进行处理；

对于不同的中断信号，会进行不同的处理。

中断的分类：

内中断（也叫“异常”、“例外”、“陷入”）------- 信号来源：CPU内部，与当前执行指令有关；

外中断（中断）----------信号来源：CPU外部，与当前执行指令无关。

外中断的处理过程：

每执行完一个指令后，CPU都需要检查当前是否有外部中断 信号；

如果检查到外部中断信号，则需要保护被中断进程的CPU环境（如程序状态字PSW，程序计数器PC、各种通用寄存器）把他们存储在PCB（进程控制块中）；

根据中断信号类型转入相应的中断处理程序；

恢复原进程的CPU环境并退出中断，返回原进程继续执行。

二、进程管理

2.1 进程管理之进程实体

为什么需要进程：

进程是系统进行资源分配和调度的基本单位；

进程作为程序独立运行的载体保障程序正常执行；

进程的存在使得操作系统资源的利用率大幅提升。+

进程控制块（PCB）：用于描述和控制进程运行的通用数据结构,记录进程当前状态和控制进程运行的全部信息，是进程存在的唯一标识。

进程（Process）与线程（Thread）：

线程：操作系统进行\*\*运行调度的最小单位\*\*。

进程：系统进行\*\*资源分配和调度的基本单位\*\*。

区别与联系：

一个进程可以有一个或多个线程；

线程包含在进程之中，是进程中实际运行工作的单位；

进程的线程共享进程资源；

一个进程可以并发多个线程，每个线程执行不同的任务。

2.2 进程管理之五状态模型

就绪状态：其它资源（进程控制块、内存、栈空间、堆空间等）都准备好、只差CPU的状态。

  执行状态：进程获得CPU，其程序正在执行。

  阻塞状态：进程因某种原因放弃CPU的状态，阻塞进程以队列的形式放置。

  创建状态：创建进程时拥有PCB但其它资源尚未就绪。

  终止状态：进程结束由系统清理或者归还PCB的状态。

2.3 进程管理之进程同步

生产者-消费者问题：有一群生产者进程在生产产品，并将这些产品提供给消费者进程进行消费，生产者进程和消费者进程可以并发执行，在两者之间设置了一个具有n个缓冲区的缓冲池，生产者进程需要将所生产的产品放到缓冲区中（+1操作），消费者进程可以从缓冲区取走产品消费（-1操作）。

****进程间同步的四原则****：

****空闲让进****：资源无占用，允许使用；

****忙则等待****：资源被占用，请求进程等待；

****有限等待****：保证有限等待时间能够使用资源；

****让权等待****：等待时，进程需要让出CPU。

## 三、作业管理

### 3.1 作业管理之[进程调度](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E8%BF%9B%E7%A8%8B%E8%B0%83%E5%BA%A6&zhida_source=entity" \t "_blank)

定义：指计算机通过决策决定哪个就绪进程可以**获得CPU使用权**。

**什么时候需要进程调度**？

1. 主动放弃：进程正常终止；运行过程中发生异常而终止；主动阻塞（如等待I/O）；
2. 被动放弃：分给进程的时间片用完；有更高优先级的进程进入就绪队列；有更紧急的事情需要处理（如I/O中断）；

**进程调度方式**：

**非抢占式调度**：只能由当前运行的进程**主动放弃CPU**；

* 处理器一旦分配给某个进程，就让该进程一直使用下去；
* 调度程序不以任何原因抢占正在被使用的处理器；
* 调度程序不以任何原因抢占正在被使用的处理器；

**抢占式调度**：可由**操作系统剥夺当前进程的CPU使用权**。

允许调度程序以一定的策略暂停当前运行的进程；

保存好旧进程的上下文信息，分配处理器给新进程；

**进程调度的三大机制**：

**就绪队列的排队机制**：为了提高进程调度的效率，将就绪进程按照一定的方式排成队列，以便调度程序可以最快找到就绪进程。

**选择运行进程的委派机制**：调度程序以一定的策略，选择就绪进程，将CPU资源分配给它。

**新老进程的上下文切换机制**：保存当前进程的上下文信息，装入被委派执行进程的运行上下文。

进程[调度算法](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E8%B0%83%E5%BA%A6%E7%AE%97%E6%B3%95&zhida_source=entity" \t "_blank)：

1. 先来先服务算法：按照在就绪队列中的先后顺序执行。
2. [短进程优先](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E7%9F%AD%E8%BF%9B%E7%A8%8B%E4%BC%98%E5%85%88&zhida_source=entity" \t "_blank)调度算法：优先选择就绪队列中估计运行时间最短的进程，不利于长作业进程的执行。
3. 高优先权优先调度算法：进程附带优先权，优先选择权重高的进程，可以使得紧迫的任务优先处理。
4. [时间片轮转调度](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E6%97%B6%E9%97%B4%E7%89%87%E8%BD%AE%E8%BD%AC%E8%B0%83%E5%BA%A6&zhida_source=entity" \t "_blank)算法：按照FIFO的原则排列就绪进程，每次从队列头部取出待执行进程，分配一个时间片执行，是相对公平的调度算法，但是不能保证就是响应用户。

### 3.2 作业管理之死锁

### 3.2.1 [进程死锁](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E8%BF%9B%E7%A8%8B%E6%AD%BB%E9%94%81&zhida_source=entity" \t "_blank)、饥饿、死循环的区别：

死锁：两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去。永远在互相等待的进程称为死锁进程。

饥饿：由于长期得不到资源导致进程无法推进；

死循环：代码逻辑BUG。

死锁的产生：竞争资源（共享资源数量不满足各进程需求）、进程调度顺序不当，当调度顺序为A->B->C->D时会产生死锁，但改为A->D->B->C则不会产生。

死锁的四个必要条件：

1. 互斥条件：必须互斥使用资源才会产生死锁；
2. 请求保持条件：进程至少保持一个资源，又提出新的资源请求，新资源被占用，请求被阻塞，被阻塞的进程不释放自己保持的资源；
3. 不可剥夺条件：进程获得的资源在未完成使用前不能被剥夺（包括OS），只能由进程自身释放；
4. 环路等待条件：发生死锁时，必然存在进程-资源环形链,环路等待不一定造成死锁，但是死锁一定有循环等待。

死锁的处理策略：

一.预防死锁的方法：破坏四个必要条件的中一个或多个。

1. 破坏互斥条件：将临界资源改造成共享资源（Spooling池化技术）；（可行性不高，很多时候无法破坏互斥条件）
2. 破坏请求保持条件：系统规定进程运行之前，一次性申请所有需要的资源；（资源利用率低，可能导致别的线程饥饿）
3. 破坏不可剥夺条件：当一个进程请求新的资源得不到满足时，必须释放占有的资源；（实现复杂，剥夺资源可能导致部分工作失效，反复申请和释放造成额外的系统开销）
4. 破坏环路等待条件：可用资源线性排序，申请必须按照需要递增申请；（进程实际使用资源顺序和编号顺序不同，会导致资源浪费）

二.**[银行家算法](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E9%93%B6%E8%A1%8C%E5%AE%B6%E7%AE%97%E6%B3%95&zhida_source=entity" \t "_blank)**：检查当前资源剩余是否可以满足某个进程的最大需求；如果可以，就把该进程加入安全序列，等待进程允许完成，回收所有资源；重复1，2，直到当前没有线程等待资源；

三.**死锁的检测和解除**：死锁[检测算法](https://zhida.zhihu.com/search?content_id=224401617&content_type=Article&match_order=1&q=%E6%A3%80%E6%B5%8B%E7%AE%97%E6%B3%95&zhida_source=entity" \t "_blank)，资源剥夺法，撤销进程法（终止进程法），进程回退法；

四、存储管理

存储管理为了确保计算机有足够的内存处理数据；确保程序可以从可用内存中获取一部分内存使用；确保程序可以归还使用后的内存以供其他程序使用。

4.1 存储管理之内存分配与回收

内存分配的过程：单一连续分配（已经过时）、固定分区分配、动态分区分配（根据实际需要，动态的分配内存）。

动态分区分配算法：

首次适应算法：分配内存时，从开始顺序查找适合内存区，若无合适内存区，则分配失败，每次从头部开始，使得头部地址空间不断被划分；

最佳适应算法：要求空闲区链表按照容量大小排序，遍历以找到最佳适合的空闲区（会留下越来越多的内部碎片）。

快速适应算法：要求有多个空闲区链表，每个空闲区链表存储一种容量的空闲区。

内存回收的过程：

回收区在空闲区下方：不需要新建空闲链表节点；只需要把空闲区1的容量增大即可；

回收区在空闲区上方：将回收区与空闲区合并；新的空闲区使用回收区的地址；

回收区在空闲区中间方：将空闲区1、空闲区2和回收区合并；新的空闲区使用空闲区1的地址；

仅仅剩余回收区：为回收区创建新的空闲节点；插入到相应的空闲区链表中去；

4.2 存储管理之段页式存储管理

页式存储管理：将进程逻辑空间等分成若干大小的页面，相应的把物理内存空间分成与页面大小的物理块，以页面为单位把进程空间装进物理内存中分散的物理块。

页面大小应该适中，过大难以分配，过小内存碎片过多；页面大小通常是512B~8K；

现代计算机系统中，可以支持非常大的逻辑地址空间(232~264)，具有32位逻辑地址空间的分页系统，规定页面大小为4KB，则在每个进程页表中的页表项可达1M(2个20)个，如果每个页表项占用1Byte，故每个进程仅仅页表就要占用1MB的内存空间。

段式存储管理：将进程逻辑空间分成若干段（不等分），段的长度由连续逻辑的长度决定。

页式和者段式存储管理相比：

段式存储和页式存储都离散地管理了进程的逻辑空间；

页是物理单位，段是逻辑单位；

分页是为了合理利用空间，分段是满足用户要求页大小由硬件固定，段长度可动态变化；

页表信息是一维的，段表信息是二维的；

段页式存储管理：现将逻辑空间按照段式管理分成若干段，再将内存空间按照页式管理分成若干页，分页可以有效提高内存利用率，分段可以更好的满足用户需求。

4.3 存储管理之虚拟内存

虚拟内存概述：是操作系统内存管理的关键技术，使得多道程序运行和大程序运行成为现实，把程序使用内存划分，将部分暂时不使用的内存放置在辅存，实际是对物理内存的扩充。

  局部性原理：指CPU访问存储器时，无论是存取指令还是存取数据，所访问的存储单元都趋于聚集在一个较小的连续区域中。

  虚拟内存的置换算法：先进先出（FIFO）、最不经常使用（LFU）、最近最少使用（LRU）

虚拟内存的特征：

多次性：无需再作业运行时一次性全部装入内存，而是允许被分成多次调入内存；

对换性：无需在作业运行时一直常驻内存，而是允许在作业运行过程中，将作业换入、换出；

虚拟性：从逻辑上扩充了内存的容量，使用户看到的内存用来，远大于实际的容量；

4.4 Linux的存储管理

Buddy内存管理算法：经典的内存管理算法，为解决内存外碎片的问题，算法基于计算机处理二进制的优势具有极高的效率。

  Linux交换空间：交换空间（Swap）是磁盘的一个分区，Linux内存满时，会把一些内存交换至Swap空间，Swap空间是初始化系统时配置的。

  Swap空间与虚拟内存的对比：



五、文件管理

5.1 操作系统的文件管理

文件的逻辑结构：

逻辑结构的文件类型：有结构文件（文本文件，文档，媒体文件）、无结构文件（二进制文件、链接库）。

顺序文件：按顺序放在存储介质中的文件，在逻辑文件当中存储效率最高，但不适合存储可变长文件。

索引文件：为解决可变长文件存储而发明，需要配合索引表存储。

辅存的存储空间分配：

辅存的分配方式：连续分配（读取文件容易，速度快）、链接分配（隐式链接和显式链接）、索引分配

辅存的存储空间管理：空闲表、空闲链表、位示图。

目录树：使得任何文件或目录都有唯一的路径。

六、设备管理

I/O设备的基本概念：将数据输入输出计算机的外部设备；

广义的IO设备：

按照使用特性分类：存储设备（内存、磁盘、U盘）和交互IO设备（键盘、显示器、鼠标）；

按照信息交换分类：块设备（磁盘、SD卡）和字符设备（打印机、shell终端）；

按照设备共享属性分类：独占设备，共享设备，虚拟设备；

按照传输速率分类：低速设备，高速设备；

IO设备的缓冲区：减少CPU处理IO请求的频率，提高CPU与IO设备之间的并行性。

SPOOLing技术：虚拟设备技术，把同步调用低速设备改为异步调用，在输入、输出之间增加了排队转储环节(输入井、输出井)，SPoOLing负责输入（出）井与低速设备之间的调度，逻辑上，进程直接与高速设备交互，减少了进程的等待时间。

七、实现支持异步任务的线程池

线程池：线程池是存放多个线程的容器，CPU调度线程执行后不会销毁线程，将线程放回线程池重新利用。

使用线程池的原因：

线程是稀缺资源 ，不应该频繁创建和销毁；

架构解耦，业务创建和业务处理解耦，更加优雅；

线程池是使用线程的最佳实践。

实现线程安全的队列Queue

队列：用于存放多个元素，是存放各种元素的“池”。

实现的基本功能：获取当前队列元素数量，往队列放入元素，往队列取出元素。

注意：队列可能有多个线程同时操作，因此需要保证线程安全，如下两种情况：

实现基本任务对象Task

实现的基本功能：任务参数，任务唯一标记（UUID），任务具体的执行逻辑

实现任务处理线程ProcessThread：任务处理线程需要不断地从任务队列里取任务执行，任务处理线程需要有一个标记，标记线程什么时候应该停止。

实现的基本功能：基本属性（任务队列、标记），线程执行的逻辑（run），线程停止（stop）。

实现任务处理线程池Pool：存放多个任务处理线程，负责多个线程的启停，管理向线程池的提交任务，下发给线程去执行。

实现的基本过程：基本属性，提交任务（put，batch\_put），线程启停（start，join），线程池大小（size）。

实现异步任务处理AsyncTask：给任务添加一个标记，任务完成后，则标记为完成；任务完成时可直接获取任务运行结果；任务未完成时，获取任务结果，会阻塞获取线程。

主要实现的两个函数：设置运行结果（set\_result），获取运行结果（get\_result)