CSDN:111辄。祝学业顺利！

1. **进程和线程的概念及区别？**

**进程**：程序的一次执行，是系统进行资源调度分配的一个独立单位

**线程**：进程的一个实体，是CPU调度分配的一个基本单位

一个程序至少有一个进程，一个进程至少有一个线程。资源分配给进程，同一个进程下所有线程共享该进程的资源

1. **线程哪些资源共享？哪些资源不共享？**

**共享**：堆、全局变量、静态变量、文件等共用资源

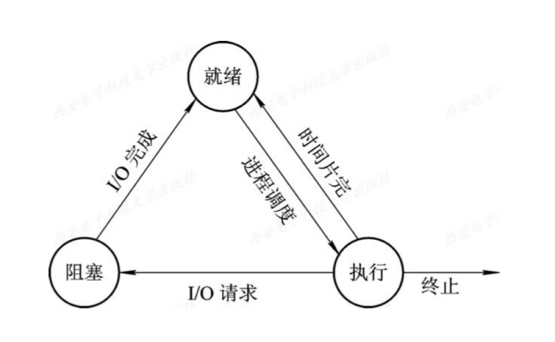
**独享**：栈、寄存器

1. **进程的状态及转换**

进程的三种基本状态：就绪、执行、阻塞

转换过程：

* 处于就绪状态的线程，在调度程序位之分配处理机后，进入执行状态
* 执行的线程，如果因分配给它的时间片已完而被剥夺处理机暂停执行，其状态变为就绪
* 执行的线程，如果因某事件（例如，进程访问某临界资源，而该资源正被其它进程访问），其变为阻塞态



1. **为什么没有就绪->阻塞/阻塞->执行？**
2. 没有就绪->阻塞：就绪状态没有占用处理机。不经过执行，其状态便不会改变
3. 没有阻塞->就绪：阻塞进程完成I/O后，会先进入就绪队列，才会被调度进程选中，变为执行态
4. **进程的调度策略**

**①先来先服务FCFS**

每次调度从就绪队列中选一个最先进入该队列的进程，为之分配处理机，并投入运行

**②短作业优先SJF**

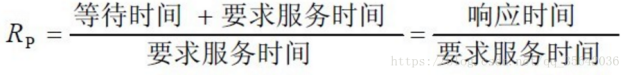
每次调度从就绪队列中选出一个估计运行时间最短的进程，为之分配处理机，并投入运行

**③高优先权优先调度（抢占式、非抢占式）**

将处理机分配给就绪队列中优先级最高的进程

* 非抢占式优先级：一旦把处理及分配给优先级最高的进程，便一直执行到完成，或者其因发生某事件放弃处理机时，系统重新将其分配给另一优先级最高进程。
* 抢占式优先级：执行期间，如果出现优先级更高的进程，进程调度立刻停止当前（原最高优先级进程）的执行，重新分配处理机给优先级更高的进程

**④高响应比优先（短作业+优先）**

更改优先级的算法，（需要事先知道执行时间）

**⑤时间片轮转**

就绪进程按先来先服务排成队列，每次为队首进程分配一个CPU时间片执行。当时间片用完后，该进程被送往就绪队列末尾，再为队首进程分配时间片，周而复始

**⑥多级反馈队列（比较好，不需知道时间）**

* 设置多个就绪队列，为每个队列赋予不同优先级

第一个队列优先级>第二个>第三个。每个队列时间片大小不同，长一倍

* 当新进程进入内存后，首先将它放入第一队列末尾，按FCFS原则排队等调度

在时间片内完成，撤离系统；未完成，放入下一队列末尾

* 仅当第一队列空闲时，调度程序才调度第二队列中的进程运行

1. **进程间的通信方式？**

**①管道**

它是半双工的（即数据只能在一个方向上流动），且只能用于亲缘进程（一般指父子进程）间通讯

**②命名管道FIFO**

可以在无关进程间通讯

**③消息队列**

是消息的链表，它面向记录。每条消息具有特定的格式及特定的优先级。

消息队列独立于发送和接收进程。进程终止时，消息队列及其内容不会被删除。

消息队列可以实现消息的随即查询。消息不一定要以先进先出的次序读取，也可以按消息的类型读取。

**④共享内存**

共享内存就是映射一段能被其它进程访问的内存。这段内存由一个进程创建，但多个进程都可访问。

[参考链接1](https://www.php.cn/faq/469981.html)

[参考链接2](https://blog.csdn.net/zhaohong_bo/article/details/89552188)

1. **什么是死锁？**

**死锁**是指多个进程因竞争资源而造成的一种僵局（互相等待）。即这组死锁进程中的每一个进程，都在等待另一个进程死锁所占有的资源。若无外力作用，这些进程都将无法向前推进

1. **产生死锁的原因？**
2. 竞争不可抢占型资源

例如：系统中只有一台打印机，可供进程P1使用，假定P1已占用了打印机，若P2继续要求打印机打印将阻塞

②竞争可消耗型资源

可消耗型资源包括硬件中断、信号、消息、缓冲区内的消息等，通常消息通信顺序进行不当，则会产生死锁

③进程推进顺序不当

若P1保持了资源R1,P2保持了资源R2，系统处于不安全状态，因为这两个进程再向前推进，便可能发生死锁。当P1运行到P1：Request时，将因R2已被P2占用而阻塞；当P2运行到P2：Request时，也将因R1已被P1占用而阻塞，于是发生进程死锁）

1. **死锁产生的四个必要条件**

**①互斥**：进程要求对所分配的资源排它性控制，即在一段时间内某资源仅为一进程所占用。

**②请求和保持**：当进程因请求资源而阻塞时，对已获得的资源保持不放。

**③不抢占**：进程已获得的资源在未使用完之前，不能抢占，只能在使用完时由自己释放。

**④环路等待**：在发生死锁时，必然存在一个进程--资源的环形链。

1. **如何预防死锁**

**①破坏请求和保持条件**（两种方法）：

* 资源一次性分配（破坏请求）：所有进程开始运行之前，必须一次性申请在整个运行过程中需要的所有资源
* （破坏请求和保持）程序运行过程中逐步释放已用完的资源，然后再请求新的资源

**②破坏不可抢占条件**：

当一个已经保持了某些不可抢占资源的进程，提出新的资源请求而得不到满足时候，要释放已保持的所有资源

**③破坏环路等待条件**：

系统给每类资源赋予一个编号，每一个进程按编号递增的顺序请求资源，释放则相反

1. **中断、硬件中断（外中断）、软件中断（内中断/陷入）、系统调用**

**中断**：CPU对中断信号的一种响应。CPU暂停正在执行的程序，保存现场，执行中断处理程 序，而后回到断点，继续原来的程序

根据中断源，分为硬件中断、软件中断

**硬件中断（外中断）**：由I/O设备（外部设备）发出

**软件中断（内中断/陷入）**：由CPU内部事件引起的中断，例如进程在运算中发生了上/下溢， 程序出错（非法指令、地址越界），电源故障等

**系统调用是软中断的一种**

中断向量表：

每种设备有响应中断处理程序，该程序的入口地址放在中断向量表中，由中断号对应。

根据中断号，查中断向量表，得程序入口地址，转入中断处理程序执行

1. **操作系统的作用及基本特征**

**作用**：操作系统处于用户和计算机硬件之间；管理多种硬件资源和软件资源，如处理机、存 储器、I/O设备、文件等；隐藏硬件操作细节（实现对硬件资源的抽象）

**基本特征**：①**并发**：两个或多个事件在同一时间间隔内发生

②**共享**：互斥共享（临界资源：在一段时间内只允许一个进程访问的资源。如系 统中大部分物理设备，栈、变量和表格）；同时访问（宏观上“同时”，其实是 交替访问的，如磁盘设备）。并发和共享是多用户（多任务）OS的2最基本特征

③**虚拟**：通过时分复用/空分复用将1个物理实体变为若干**逻辑上**的对应物（用户 错觉以为自己有一个多余的）。**时分复用**：利用设备为一用户服务的空闲时间，服 务其它用户。分为虚拟处理机器（为进程）、虚拟设备技术（为I/O）。**空分复用**： 利用存储器的空闲空间分区域存放和运行多道程序。引入虚拟存储可逻辑上扩大存 储器容量（本质上是时分），使一道程序在远小于它的内存空间中运行（用户程序 各个部分分时进入内存运行）。

③**异步**：进程的执行并非“一气呵成”，而是“走走停停”（请求某种资源，如 打印机，需要等待）

1. **虚拟存储器**

虚拟存储器从逻辑上扩充内存容量（其容量=内存+外存）（成本近于外存，速度近于内存）。它使应用程序认为它有连续可用的内存，而实际上，它只有少部分进入内存，部分存储在外存中，需要时进行数据交换。

1. **共享内存**

共享内存是最快的进程间通信机制。操作系统在几个进程的地址空间上映射一段内存，然后这几个进程可在不需操作系统函数的情况下在那段内存上读/写

1. **分页/分段**

**①分页存储管理**

将用户程序的地址空间（逻辑上）分为若干个固定大小的区域，称“页”

将内存空间分为若干个物理块。页和块的大小相同

则可将用户程序的任一页放入任一物理块中，实现离散分配，解决内存碎片问题

**②分段存储管理**

将用户程序的地址空间分为若干个大小不同的段，每段可定义一组相对完整的信息

存储器分配时，以段为单位。这些段在内存中可以不邻接，页实现了离散分配

1. **页面置换算法（虚存时用）**

①**最佳置换算法**：选择的淘汰页是未来最长时间内不再被访问的页面

②**先进先出算法FIFO**：淘汰在内存驻留最久的页面

③**最近最久未使用算法**

④**最少使用算法**

⑤**CLOCK算法**：为每页设置一位访问位，将内存中的所有页面链接成一个循环队列。当页面 被访问，访问位置1，否则为0。选择淘汰也时，若遇0，换出；否则将1置 为0，接着。一直没找到，再回队首重来。

⑥**改进的CLOCK算法**：除了页面访问情况，又添加一个置换代价。淘汰页优先选择未访问、 未修改的；再未访问、已修改的顺次

1. **文件组织方式**

**逻辑组织**：

**①有结构文件（记录文件）**

文件由若干记录构成。每个记录是一组相关的数据集合，用于描述一个对象某方面的属性。

分为定长记录文件（记录长度相同）、不定长记录文件

**②无结构文件（字符流文件）**

文件内部不划分记录，由一组相关信息组成有序字符流，即流式文件，其长度直接按字节计算。如大量源程序、可执行程序、库函数等都是无结构。

**物理结构**：连续文件（放在连续物理块）、串联文件（分散至离散物理块，指针指向下一物 理块；只能顺序访问，不能随机存取）、索引文件（为每个文件建立索引表，放 物理块块好）

1. **银行家算法**

避免死锁。系统在对进程分配资源前，先计算安全性（找到一个安全序列），若安全则正是分配。

在银行家算法中，设置了四个数据结构：Available、Allocation、Max和Need。在判断安全序列时，看Available是否能满足某个进程的Need，如果满足则在该进程释放资源后，即Available+Allocation得到新的空闲资源量，继续看Available是否能满足下个进程的Need，看是否能找到一个进程的安全序列

1. **经典进程同步问题**

**①生产者-消费者问题**

合作/竞争：生产者（产生数据-往缓冲区放数）；消费者（处理数据-从缓冲区取出数据）

循环链表

n个缓冲区（放n个数），信号量empty、full表示空缓冲区、满缓冲区数量；互斥信号量mutex实现互斥使用（临界资源，不能同时放/取）

void producer()

wait(empty);//先资源wait，后互斥信号量wait，否则可能死锁

wait(mutex);

signal(mutex);

signal(full);

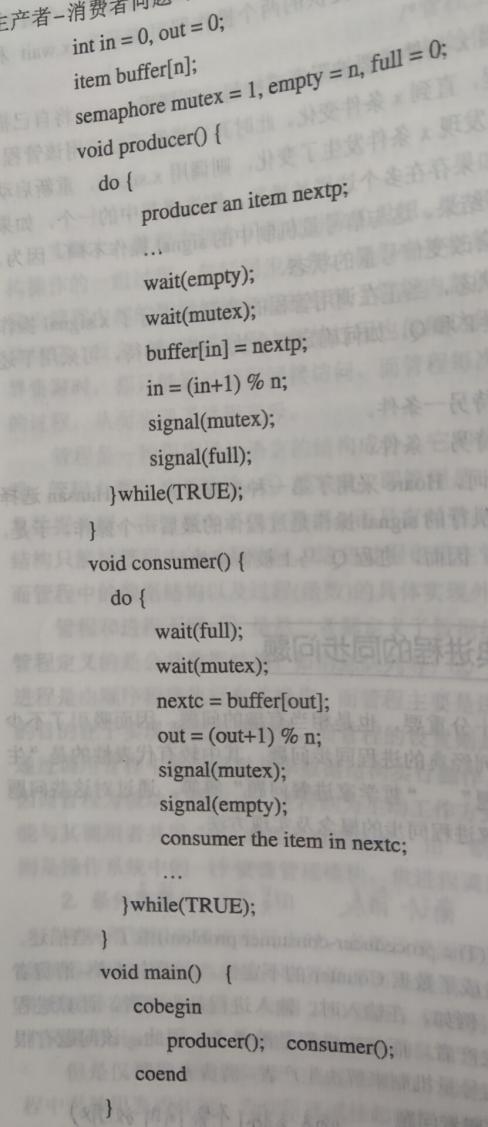
void consumer()

wait(full);

wait(mutex);

wait(mutex);

wait(full);



使用AND信号量解决死锁：

Swait(empty,mutex);

Signal(mutex,full);

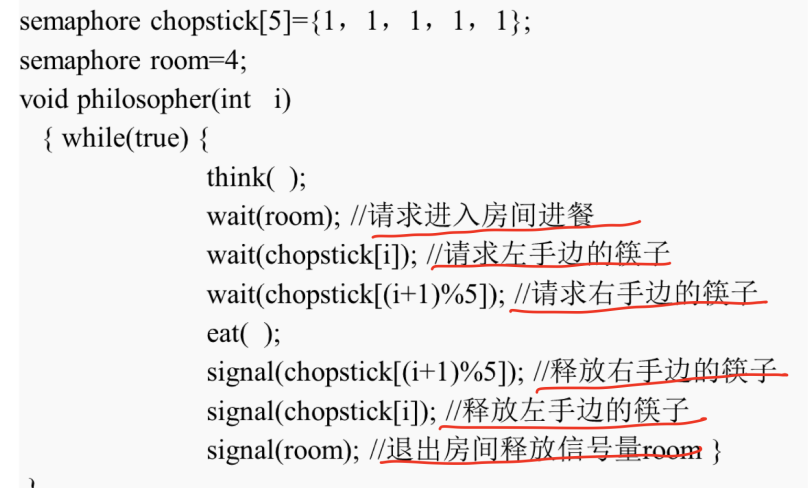
使用管程（排他性，不用mutex）：

建立Producer-Consumer管程，包括put(item)生产者和get(item)消费者过程

**②哲学家问题**

五个哲学家吃面，5个面碗只各放了1根筷子。哲学家交替地思考和进餐。当哲学家吃饭时试图取其左右最靠近他的筷子就餐，要2只

每次最多4个哲学家同时取左边筷子；然后同时竞争右边筷子。一定有一位哲学家拿到2只筷子吃饭，释放后其他哲学家再取



用AND信号量：

Swait(chopstick[(i+1)%5],chopstick[i]);

Signal(chopstick[(i+1)%5],chopstick[i]);

**③读者-写者问题**

允许多个进程Read一个文件，但是Write进程必须是各个进程互斥地

要求读的进程是reader进程，要求写的进程是writer进程

对读操作计数rmutex（不是对读互斥，而是对读操作数量计数时计数，readcount+1）

互斥信号量wmutex，readcount表示正在读的进程数目

只要用一个Reader在读（readcount不为0），便不允许writer写。只有readcount为0，才可wait(wmutex)。若wait(wmutex)成功，reader可读，readcount+1。读完后释放，readcount--。当readcount减1后为0时，才可signal(wmutex)，即writer可写。

因为readcount是临界资源，所以给它也设了个rmutex，在其前后使用

Wait(rmutex);

If(readcount==0) wait(wmutex);

Readcount++;

Signal(rmutex);

Wait(rmutex);

Readcount--;

If(readount==0) signal(wmutex);

Signal(rmutex);

1. **磁盘调度算法**

FCFS先来先服务（by各进程请求访问磁盘先后顺序）、STF短任务优先（by各进程的请求磁盘读写数据少）