### CPU调度

#### 调度的分类

1. 高级调度：将外存作业调入内存，分配内存和外设资源，创建PCB，然后把数据调入内存，包含整个从外存到内存再被cpu调度的过程

2. 中级调度（交换调度）：把进程从外存调入内存

3. 低级调度（进程调度或线程调度）：从就绪队列中选择一个调入CPU

#### 调度的时机

\*\*非抢占式调度\*\*：进程中止、进程从运行到等待（等待IO）

\*\*抢占式调度\*\*：进程从运行到就绪（时间片用光或者有更高优先级的进程）、进程从等待到就绪

#### 调度的准则

按照以下几条来定制调度的准则

- \*\*CPU利用率高\*\*

- \*\*吞吐量大\*\*：单位时间内完成的任务数量大

- \*\*响应时间短\*\*：从用户输入到产生反应的时间

- \*\*周转时间\*\*：从任务开始到结束的时间=等待时间+运行时间<=>等待时间短（在就绪队列的时间短）

#### 调度算法（平均等待时间出计算）

- \*\*FCFS\*\*：先来先服务，\*\*不利于短作业\*\*

- \*\*SJF\*\*：短作业优先，优先调度下次CPU时间最短的，\*\*下一CPU区间大小的预测较难\*\*

- \*\*SRTF\*\*:抢占式调度，最短剩余时间优先

- \*\*优先级调度\*\*，\*\*低优先级的任务可能永远得不到时间运行\*\*

- \*\*RR\*\*，分时算法，每个进程分配固定的时间片，按照FCFS的顺序执行，\*\*任何进程的等待时间不会超过（n-1）q,但上下文切换次数较多，开销较大\*\*，\*\*时间片一般为10-100ms\*\*

- \*\*多级队列\*\*，多个就绪队列，不同的队列采用不同的调度方法

- \*\*多级反馈队列\*\*，任务可以在多级队列之间移动，实现aging

- \*\*高响应比优先\*\*：响应比=周转时间/运行时间

#### 一些概念

- \*\*多处理器调度\*\*：非对称多处理器、对称多处理器（大概就是cs和p2p吧）

- \*\*处理器亲和性\*\*：线程尽量长时间运行在指定的CPU上

- \*\*负载平衡\*\*：将负载平均分配在所有cpu上

# 操作系统期末复习

## 第一章

1. \*\*操作系统的定义\*\*：计算机系统中的一个系统软件，一些程序模块的集合；计算机与用户之间的接口。

2. \*\*操作系统的功能\*\*：文件管理、设备管理、进程管理、存储管理

3. \*\*目态和管态\*\*：管态(内核态)：可以使用所有指令；目态（用户态）：禁止使用特权指令

目态=>管态：系统调用

管态=>目态：修改psw

管态指令:存取IO、更改PSW、存取特殊寄存器（时钟、中断）

4. 操作系统的发展:

- 晶体管和批处理系统：把作业合成一批，放入计算机（不支持交互）

- 多道程序系统：在内存中放多道程序，宏观上并行，微观上串行。\*\*优点\*\*：吞吐量大、资源利用率高

- 分时操作系统：为了满足多用户的需要而设计，共享主机，可以交互

5. \*\*例题\*\*：\*\*见P35.1.6\*\*，这里注意：陷阱是用户态发起系统调用的指令，不属于内核态的指令

## 第二章

### 操作系统的结构

1. 简单结构：如Dos系统，在尽可能小的空间中提供大多数功能

2. 分层结构：每一层只能使用较低层的函数和服务，缺点是效率低

3. 微内核结构：公共部分提供最基本的服务，内核常驻内存，其他部分在用户空间作为服务进程，内核模式下运行的较少。

4. 模块结构：内核模块可加载，模块间通过接口通信，windows保留了微内核的特征，linux新功能可以直接添加到内核。

## 第三章

### 进程的定义

1. 执行中的程序

2. 一个正在执行程序的实例，包括程序计数器、寄存器和变量的当前值

3. 并发执行的程序在执行过程中分配和管理资源的基本单位

### 进程的特征

1. \*\*动态性\*\*：由创建而产生，由调度而执行、由得不到资源而阻塞，由撤销而消亡

2. \*\*并发性\*\*：多个进程可存在内存中，在一段时间内同时运行。

3. \*\*独立性\*\*：独立获得资源的基本单位

4. \*\*异步性\*\*：进程的执行顺序和执行时间的不确定性

### 进程的状态

\*\*就绪态\*\*：已得到除CPU以外的其他资源,被调度进入运行态

\*\*运行态\*\*：占用CPU，时间片用完进入就绪态

\*\*等待态\*\*：I/O或事件的等待

### 进程控制块PCB

\*\*作用\*\*：进程存在的唯一标志

\*\*内容\*\*：描述信息：pid、uid、家族关系 控制信息：当前状态、运行统计时间 资源管理信息：占用内存大小指向文件系统的指针等 CPU保护现场信息：程序计数器、CPU寄存器

\*\*进程创建过程\*\*：

- 查PCB链表，有PCB取新的PCB，填入参数

- PCB入对应的就绪队列

- PCB入对应的进程家族

\*\*进程中止的过程\*\*

级联中止

### fork函数

\*\*返回值\*\*：两个返回值，大于0（子进程的标识号）在父进程中，等于0（可以理解为它的子进程，还没有所以是0）在子进程中。

\*\*子进程的内容\*\*：刚创建时数据复制父进程的数据，子进程已耗费的进程时间设置为0，父子进程可以修改各自数据而互不影响

\*\*wait函数\*\*：父进程等待子进程，子进程运行完之后向父进程的wait函数发一个信号，然后等待子进程先结束，\*如果父进程先于子进程运行完，子进程的父进程就变成init进程，称之为孤儿进程\*，如果子进程在父进程调用wait之前中止，那父进程就变成\*僵尸进程\*。

\*\*exec函数\*\*：加载并执行exe文件，以新程序取代子进程的地址空间，\*代码段、堆栈段完全重写\*

### 进程通信

#### 低级通信：控制信息的交换

##### 锁、信号量、软中断

#### 高级通信：大批量数据的交换

##### 消息机制、管道、共享内存

1. 共享内存：生产者消费者问题

2. 消息通信

3. 管道通信：互斥：一个进程对pipe进行读写时，另一个进程必须等待。同步：管道空时，读进程等待，管道满时，写进程等待。

## 第四章

比进程更小的单位=>[[线程]]

## 第五章

进程的问题=>[[进程同步]]

## 第六章

CPU的调度，如何切换进程在CPU中运行？=>[[CPU调度]]

## 第七章

[[内存管理]]

## 第八章

[[虚拟内存]]，几个算法记一下就行了

## 第九章

### 文件的基本概念

为了实现共享、和存储大量信息

### 文件系统的特点

- 文件可以按名存取

- 文件可以存储大量信息

- 文件对用户有接口

### 文件的分类

#### 逻辑结构

- 流式，无结构文件

- 记录式，有结构文件

- 树

#### 物理结构

- 顺序结构

- \*\*逻辑上连续的文件依次存到连续的物理块，目录项包括始址、总块数、最后一块字节数\*\*，类似于数组

- 链接文件

- \*\*文件信息存储在不连续的物理块中，目录项包含链表的首指针\*\*，类似于链表，\*\*不支持随机存取\*\*

- 索引文件

- \*\*将文件信息存放在若干不连续的物理块中，然后每个文件建立一张索引表，目录项是索引表的物理地址\*\*

- 当索引表大小超过一个物理块时，可以采用\*\*多级索引\*\*

### 文件存储设备

#### 顺序存储设备

\*\*磁带\*\*：存取速度高，是连续文件但是\*\*不支持随机存取\*\*

#### 直接存取设备

\*\*磁盘\*\*：允许直接存取磁盘上的任意物理块，这里要记住磁盘的概念，包括\*\*磁头号、磁道号、扇区号，柱面等\*\*

### 文件目录管理

#### 文件的组成

\*\*文件\*\*=\*\*文件说明FCB（创建的时间、组成目录文件、存在标志灯）+文件体\*\*

#### 目录的组成

- 直接法（win/dos）：\*\*目录项=文件名+FCB\*\*

- 间接法（unix/linux）：\*\*目录项=文件名+FCB地址\*\*

#### 目录结构

- \*\*单级目录\*\*：整个系统一个目录表，\*\*不允许重名\*\*

- \*\*两级目录\*\*：一个主目录表，每个用户有一个用户目录表，\*\*允许重名和文件共享\*\*，\*\*不适合用户太多\*\*

- \*\*树形结构\*\*：\*\*查找速度快，不能实现文件共享\*\*

#### 空闲块存储方法

- \*\*空闲文件目录\*\*

- \*\*空闲块链\*\*

- 位示图

### 第十章

#### linux文件系统和dos文件系统的区别

dos文件系统是多个树的结构、linux是一个目录树的结构

#### 文件详细信息

横线是普通文件，-d是目录文件

三个安全级别分别是：所有者、组、其他用户

三种权限分别是：r、w、x（可执行）

#### linux文件系统

linux向进程提供了一个接口，可以创建各种类型的文件

##### 文件系统的内容

- \*\*引导块\*\*：boot manager

- \*\*超级块\*\*：整个文件系统的信息

- \*\*inode节点表\*\*：/为2号节点，存储文件的属性信息

- \*\*数据区\*\*：存储目录文件，即文件名和inode节点的对应关系、以及普通的数据文件

- \*\*系统打开文件链表\*\*：存放一个已打开文件的管理控制信息，f\_node指针指向文件对应的inode

##### 硬链接与软连接

- \*\*硬链接\*\*：加一个指向inode的指针，通过新的文件名也可以访问到它，inode中有一个count，count为0时真正删除

- \*\*软连接\*\*:相当于创建一个新的inode，这个inode指向的文件其实是一个路径，原文件被删除后无法使用，\*\*相当于快捷方式\*\*

##### 进程文件管理

- \*\*fs\_struct\*\*:存储文件系统根目录和当前目录，pwd=>相对路径inode，root=>系统根目录inode

- \*\*files\_struct\*\*:包含进程的打开文件表，打开文件时，建立一个file结构体，加入系统打开文件表，然后在进程打开文件表中建立一个fd指针指向它，重复打开时，重复此操作

### 第十章

[[设备管理]]

### 进程同步和实现进程同步的各种方法

#### 临界区问题

\*\*临界资源\*\*：一次仅允许一个进程访问的资源

\*\*临界区\*\*：访问临界资源的代码

\*\*临界区原则\*\*：互斥、有空让进、有限等待

#### Peterson方案

\*对于每一个进程而言，应该遵循以下原则：\*

\*自己想进\*

\*让对方先进\*

\*是否能进（是否满足互斥变量）\*

{

}

\*修改自己互斥的变量，以便让对方有限等待\*

#### 关中断

#### 硬件原子指令

一次执行，无法被打断，所以实现了同步

#### 优先级反转情况

\*什么是优先级翻转？\*=>优先级调度+忙等待

#### 信号量

##### 信号量的含义

\*\*S=0\*\*=>无资源可用

\*\*S<0\*\*=>等待队列中的进程个数

\*\*S>0\*\*=>有S个资源可用

除了初始化以外，只能通过PV操作访问，PV操作也是原语操作，包括\*\*创建、撤销、阻塞、挂起、激活\*\*等

\*\*信号量的实现\*\*：

`

Wait(S){

S->value--;

if(S->value<0){

添加这个进程到阻塞队列；

BLOCK()阻塞该进程；

}

}

`

`

Signal(S){

S->value++

if(S->value<=0){

撤销一个进程从阻塞队列；

唤醒一个进程；

}

}

`

\*\*信号量的种类\*\*：

1. 公有信号量：解决进程互斥问题，进程共享公有资源，初始化为1，每个进程均可以实现PV操作

2. 私用信号量：解决进程同步问题，进程共享对方的私有资源，初始化为0或者n，只允许拥有的进程施加P操作

#### 进程同步的几个问题

\*\*生产者消费者问题\*\*

- mutex:公用信号量，保证互斥

- avail:生产者的私有信号量，空缓冲区的个数，初始化为m

- full：消费者的私有信号量，满缓冲区的个数，初始化为0

\*\*同步和互斥P操作在一起，同步在前，互斥在后，必须保证进入临界区最后一个操作是互斥操作\*\*

\*\*哲学家就餐问题\*\*

- 保证两个相邻哲学家不会拿起同一根筷子

- 保证两个相邻哲学讲不会同时拿起同一根筷子，并且没有人饿死（有限等待）\*\*采用奇数偶数的方法、或者只让四个哲学家入场的方法（初始化一个为4的同步信号量）\*\*

\*\*读者写者问题\*\*

- 定义：多个进程可以同时读，但是只能有一个进程写

- 进程之间的互相制约关系：\*\*写互斥信号量\*\*：mutexW，用于两个作用：读者写者之间、写者之间互斥写。\*\*当前在读的读者个数\*\*：readCount，\*\*读者之间的信号量，用于对readcount的加减变成原子操作\*\*：mutexCount

#### 管程

互斥：管程内的数据只能通过其中的操作访问，并且一次只允许一个进程调用

同步：通过条件变量做PV操作来实现同步

操作系统本身不支持，是编译器的工作

#### 死锁

##### 定义：一组并发进程彼此互相等待对方所拥有的资源

##### 死锁发生的四个必要条件

1. \*\*互斥条件\*\*：资源为临界资源

2. \*\*不可抢占条件\*\*：进程占用的资源不会被强制拿走

3. \*\*部分分配\*\*：进程等待新资源时仍不放弃自己已有的资源

4. \*\*环路等待条件\*\*：存在一个互相等待的进程链

##### 死锁的处理方法

1. \*\*死锁预防\*\*：破坏死锁的必要条件，如资源变得可抢占，采用一次分配而不是部分分配等

2. \*\*动态避免\*\*：

- 发生的一瞬间避免=>\*\*银行家算法\*\*：\*在每次分配资源之后，找到一个分配序列，使得每个进程的最大需求得到满足，则此次分配是安全的\*；

- \*\*资源分配图\*\*：选择一个节点，消去它所有的分配边和请求边，别的进程也这样重复执行，如果所有进程都变为固定节点，则可以化简，否则是不可以化简的。

4. \*\*检测并解除\*\*：通过死锁的定理和资源分配图来检测死锁，通过\*\*撤销进程、剥夺资源、进程回滚、无为而治\*\*等来解除死锁

##### 例题

\*z个资源，k个进程，每个进程需要m个资源，够不够分\*

\*满足k\*(m-1)+1<=z，则够分\*

## 内存管理

### 存储器管理方式

#### 连续内存分配

- \*\*单一连续区分配\*\*

- \*\*固定分区\*\*：包含请求表和内存分配表，有\*\*内部碎片\*\*，易于实现

- \*\*可变分区\*\*：运行过程中根据进程大小动态建立，有\*\*请求表、已分配分区表、空闲区表三个数据结构\*\*，

1. 可变分区有计算题，包括三种分区算法\*\*最先适应：尽可能找低地址、最佳适应：满足空间的最小空闲区、最坏适应：找最大的可用区分配\*\*

2. 可变分区的回收（用完之后）：合并下分区需要改首址，不邻接需要建立新表项

3. 可变分区的碎片问题：外碎片问题：解决方式为内存紧缩或者离散分配方式

#### 离散分配方式（考计算）

##### 分段：

- \*\*定义\*\*：程序员把程序按照含义分成若干段，\*\*每段是连续的\*\*,创建进程时，建立进程段表，段表包括各个段的起始位置和长度

- \*\*段地址映射的方法\*\*：根据段号找段表，映射到段的基物理地址，再加上偏移地址

##### 分页：

- \*\*定义\*\*：将内存划分为若干个长度相等的页，由\*\*硬件自动完成\*\*

- \*\*地址映射方法\*\*：通过页号访问页表找到页框号，页框号对应的一个页面的初始地址+偏移地址=物理地址，根据进程所需空间的大小分配页框的个数

- \*\*地址计算方法\*\*：给出一个虚地址，除以页大小取整=>页号，剩下的是偏移量，通过页号找页框号，页框号\*页大小+偏移量=物理地址

- \*\*数据结构\*\*：

- 页表：每个进程一个

- 请求表：整个系统一个，包括进程的页表初始地址、页表长度（\*\*可以用于找到每个进程的页表\*\*）

- 位示图/空闲页面链表：每个系统一个，代表整个内存空间的内容，可以得知内存空间的空闲程度

- \*\*快表TLB\*\*:存放着当前页的部分页表项，遵循局部性原理，支持并行查找

- \*\*有效访问时间的计算\*\*：命中率x(TLB+MA)+非命中率x(TLB+2MA）

- \*\*多级页表\*\*：大地址空间用

- \*\*哈希页表\*\*：每个表目包含一个链表，处理冲突，每个元素包括逻辑页号、物理页号和指向下一元素的指针，每次查找时，通过比对逻辑页号对应的物理页号

- \*\*倒置页表\*\*：整个系统一个倒置页表，表的内容有进程的pid和页号

#### 内存管理技术的优缺点

- \*\*静态页式管理\*\*：优点：没有外碎片，缺点：必须一次性全部装入内存，有内碎片

###### 段页式管理

\*\*定义\*\*：将用户程序划分为段，然后查找段表，段表内给的是页表的长度和起始地址，然后查找页表获取各个页面，\*\*段表是一个程序一个，页表是一个段一个，获取物理地址需要三次访存\*\*

### 地址绑定的方法

\*什么是重定位？\*

\*将逻辑地址映射到物理地址的操作\*

- \*\*动态重定位\*\*：利用硬件实现，需要BR和VR

- \*\*静态重定位\*\*：编译时生成重定位代码，修改代码里的地址为物理地址

### 内存保护方法

- \*\*硬件法\*\*:采用寄存器来保护，\*\*上界寄存器<=物理地址<=下界寄存器\*\*，\*\*0<=逻辑地址<限长寄存器\*\*

- \*\*软件法\*\*：采用保护键实现，对应的地址必须送对应的保护键和读写指令才能成功操作

### 外存调入内存的方法

- \*\*覆盖技术\*\*：\*\*没啥说的，会算就行\*\*

- \*\*交换技术\*\*：从外存调入时，交换整个进程的地址空间

## 设备管理

### 设备的分类

这里只提一下：

\*\*独占设备\*\*：打印机、磁带

\*\*共享设备\*\*：磁盘、硬盘

### 数据传输方式

- 查询

- DMA

- 中断

- 通道：不需要CPU实现I/O,且无中断

### 缓冲技术

#### 引入的原因：

1. 解决I/O设备和CPU之间不匹配的问题

2. 减少CPU的中断频率

3. 提高CPU和I/O并行性（通道和DMA）

#### 缓冲的实现方法：

- 专用硬件缓冲器

- 在内存中开辟一段缓冲区

#### 设备管理的数据结构：

SDT（系统）=>DCT（设备）=>COCT（控制器）=>CHCT(通道)

#### 磁盘寻道过程

先找到柱面，再通过磁头找到扇区，再在磁道进行移动

##### 花费的时间：查找时间（磁头移动到指定柱面）+等待时间（扇区转到磁头下）+传输时间（磁头读写完成信息传送的时间）

##### 磁盘调度算法(要会算划过的磁道数和平均寻道长度)

- \*\*FCFS\*\*

- \*\*SSTF\*\*:距离磁头当前位置最近的请求作为下一次调度的对象

- \*\*SCAN\*\*：沿着移动臂的移动方向选择距离磁头当前位置最近的请求作为下一次调度的对象

- \*\*CSCAN\*\*:从0号往里移动，直到最后一个柱面，\*\*马上返回到0号柱面\*\*

## 线程

### 线程的定义

\*\*CPU调度的基本单位\*\*、\*\*可在CPU中运行的基本单位\*\*

特点:共享代码段、堆内存，栈为不同的线程独有

### 并发和并行

\*\*并发\*\*：两个或多个事件在同一时间间隔发生

\*\*并行\*\*：两个或多个事件在同一时刻发生，多核CPU中实现

### 线程的种类

\*\*用户线程\*\*：TCB在进程的用户空间，用于解决上下文切换的开销

- 优点：无需用户态和内核态的切换，速度快

- 缺点：内核以进程为单位调度，无法发挥多核优势，\*一个线程阻塞，整个进程阻塞\*

\*\*内核线程\*\*：TCB在进程的内核空间，可以发挥多处理器的并发优势

- 优点：内核以线程为单位调度，实现了同一进程内多个线程并行运行

- 缺点：线程的调度和同步均通过系统调用实现，开销大。

\*\*多线程模型(用户线程和内核线程的映射关系)\*\*:

- 多对多：复杂

- 一对一：内核线程数目有限

- 多对一：不再使用

### 线程和进程的比较

1. \*\*线程没有独立的资源\*\*

2. 线程是CPU调度单位、进程是资源分配单位

3. 线程中的内容：PC、register、stack

4. 进程的切换开销\*\*远大于\*\*线程

5. 进程绝对隔离