操作系统期中考前整理

二、运行环境和运行机制

1. 操作系统为什么需要硬件提供基本运行机制？

①处理器具有特权级别，能在不同的特权级别运行不同的指令集合；
②硬件机制可将操作系统和用户程序隔离。

2. 为什么操作系统要将指令集合划分为不同的特权级别？

有些指令错用会导致系统崩溃。为了防止不恰当的使用，操作系统将部分指令封装在内核中，确保其安全的情况下才允许执行。

3. 常见的特权指令和非特权指令

特权指令：启动 I/O，内存清零，修改程序状态字，设置时钟，允许/禁止中断，停机；
非特权指令：控制转移，算术运算，访管指令，取数指令。

4. CPU 状态之间的转换

用户态到内核态：通过中断/异常/陷入机制；
内核态到用户态：设置程序状态字。

5. 为什么要引入中断和异常？

引入中断是为了支持 CPU 和设备之间的并行操作；引入异常表示 CPU 执行指令本身遇到了问题。

6. 中断/异常机制的工作原理

处理中断/异常时，操作系统从用户态切换到内核态，硬件保存用户态的上下文环境，捕获中断源发出的中断/异常请求，将中断触发器内容按规定编码送入 PSW 相应位，硬件执行流程根据中断号，查中断向量表转移控制权给中断处理程序。软件识别中断/异常类型，保存相关寄存器信息，分析中断/异常具体原因，执行对应处理功能。执行完后，硬件恢复现场并返回被时间打断的程序。

7. 系统调用的作用

系统调用时操作系统提供给编程人员的唯一缺口，使 CPU 从用户态陷入内核态，从而允许用户在编程时调用操作系统的功能。

8. 系统调用机制的设计（为了实现系统调用，操作系统需要做什么）

①需要有中断/异常机制，支持系统调用服务的实现；
②用户需要选择一条特殊的指令（陷入指令，又称访管指令），引发异常，完成用户态到内核态的切换；
③内核需要为每个系统调用分配一个编号，成为是系统调用号；
④内核需要有系统调用表，存放系统调用服务例程的入口地址。

9. 系统调用的执行过程

①通过中断/异常机制，硬件保护现场，通过查中断向量表把控制权转给系统调用总入口程序；
②通过系统调用总入口程序，保存现场，将参数保存在内核堆栈里，通过查系统调用表将控制权转给相应系统调用处理例程或内核函数；
③内核执行系统调用例程；
④执行完毕后，内核恢复现场并返回用户程序。

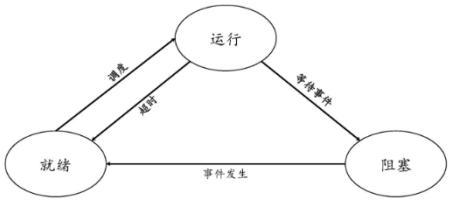
10. 系统调用和函数调用的区别、与 API 的关系

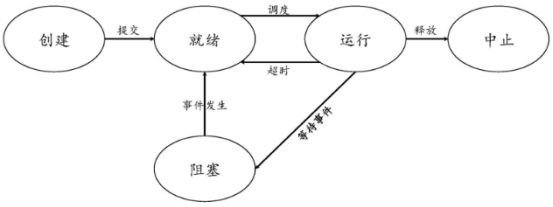
与函数调用的相同点在于都改变了指令的执行顺序，都必须返回原代码继续执行。区别：
①系统调用存在用户态和内核态的切换，调用地址不固定；
②系统调用中，内核使用不同的堆栈，存在堆栈的切换；
③系统调用不允许递归调用，函数调用允许递归调用；

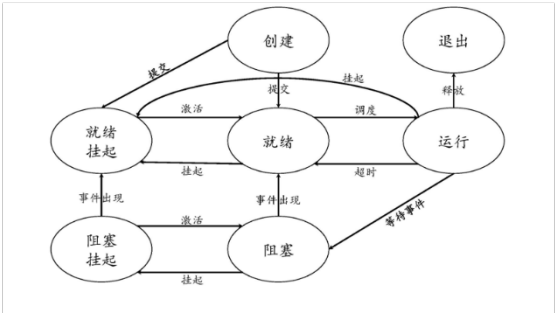
④系统调用通过 call 或 jmp 指令进入调用，函数调用通过 int 或 trap 指令进入调用；
与 API 的关系：API 是封装了的内核函数，API 执行过程可能经历了多次系统调用，多个 API 也可能都执行了同一个系统调用。

三、进程线程模型

1. 三、五、七状态进程模型







2. PCB 的作用

PCB（进程控制块）是操作系统中表示进程的数据结构，记录进程的各种属性，描述进程的动态变化过程，与进程一一对应，是系统感知进程存在的唯一方式。

3. PCB 的主要内容

包括进程描述信息（PID，进程名，用户标识符），进程控制信息（状态，优先级，代码执行入口地址…），所拥有的资源和使用情况（虚拟地址空间，打开文件列表），CPU 现场信息（寄存器值，页表指针）。

4. PCB 如何描述进程地址空间？

操作系统给每个进程都分配了一个地址空间。PCB 通过虚拟地址描述地址空间。

5. 进程是如何创建的？

①分配 PID，PCB；
②为进程分配地址空间；
③初始化 PCB；
④设置相应队列指针（将新进程添加进就绪队列链表中）；
⑤创建/扩充其他数据结构。

6. 进程是如何撤销的？

①结束子进程/线程；
②收回进程占有的资源（关闭打开文件，断开网络连接，回收分配的内存）；
③撤销 PCB。

7. Unix 的 fork()实现

①为子进程分配进程描述符、proc 结构和唯一的 PID；
②以一次一页的方式复制父进程地址空间（Linux 采用写时复制）；
③继承父进程的共享资源，包括打开文件和工作目录；

④将子进程状态设为就绪；
⑤为父进程返回子进程的 PID，子进程返回 0。

8. 为什么要引入线程？

①应用的需要：单进程无并行性；
②开销的考虑：创建线程花费时间少，线程切换花费时间少，线程之间相互切换无需调用内核；
③性能的考虑：多核，多处理器。

9. 进程和线程的区别

①进程是操作系统资源分配的基本单位，线程是处理器任务调度和执行的基本单位；
②进程都有自己独立的地址空间和资源，上下文切换开销大；线程共享地址空间和资源，上下文切换开销小；
③线程是进程的一部分，是轻量级的进程；
④一个线程崩溃后所有线程都会崩溃，而一个进程崩溃不会影响其他进程，多进程比多线程更加健壮；
⑤每个独立进程都有程序运行的入口，但线程不能独立执行，必须依附于进程中。

10. 线程的属性

①有状态和状态切换；
②不运行时，需要保存程序计数器等上下文；
③有自己的栈和栈指针；
④共享地址空间和资源；
⑤可以创建或撤销另一个进程。

11. 为什么线程要有自己的栈？

每个线程都独立执行，函数参数和局部变量都必须保存在栈中，所以每个线程要有独立的栈。

12. 线程的实现方式

线程的实现方式有用户级线程（在用户空间中实现）、核心级线程（在内核中实现），也有二者结合的方法（在用户空间创建，在核心态调度）。

13. 用户级线程和核心级线程的区别

①用户级线程在用户空间中建立，线程由用户管理，内核不知道线程的存在（只知道进程的存在）；核心级线程由内核管理，内核维护进程和线程的上下文；
②用户级线程切换不需要内核特权；核心级线程切换需要内核支持；
③典型例子：POSIX Pthreads 支持用户级线程，Windows 支持核心级线程。

四、进程线程调度

1. CPU 调度的时机

①进程执行完毕并退出；
②进程由于某种错误或异常而终止；
③新进程的创建；
④运行进程要等待 I/O 操作或其他资源时进入阻塞态；
⑤被唤醒的阻塞进程重新回到就绪态；
⑥进程用完所分配的时间片回到就绪队列。

2. 上下文切换的具体步骤（A 下 B 上）

①保存进程 A 的上下文环境；
②用新状态和其他信息更新进程 A 的 PCB；
③将进程 A 移动至合适的队列（就绪，等待…）；

④将进程 B 的状态设置为就绪态；
⑤从进程 B 的 PCB 中恢复上下文

3. 调度算法的衡量指标及主要考虑因素

衡量指标：公平性，吞吐量，周转时间，响应时间，CPU 利用率，等待时间。
主要考虑因素：PCB 中需要记录的信息，进程优先级及就绪队列的组织，抢占/非抢占，I/O 密集型/CPU 密集型，时间片。

4. 批处理系统中采用的调度算法

先来先服务（FCFS），最短作业优先（SJF），最短剩余时间优先（SRT），最高响应比优先（HRRN），其中响应比定义为

$$R=\frac{\text{作业周转时间}}{\text{作业运行时间}}=1+\frac{\text{作业等待时间}}{\text{作业运行时间}}.$$

5. 交互式系统中采用的调度算法

①轮转调度（RR）

时间片轮转调度：为每个进程分配一个时间片，在时间片结束时切换进程。优点：公平，响应时间快；缺点：由于进程切换，时间片轮转调度需要的开销大。

虚拟轮转调度：区分 CPU 密集型和 I/O 密集型。

②优先级调度算法（HPF）

选择优先级最高的进程优先执行，通常系统进程高于用户进程，前台进程高于后台进程，偏好于 I/O 型进程。

HPF 会出现优先级反转的现象，低优先级进程持有高优先级进程所拥有的资源，从而使高优先级进程等待低优先级进程。此时会导致系统错误，高优先级进程停滞不前导致系统性能降低。解决方案为设置优先级上限，优先级继承或中断禁止。

③多级队列调度算法（MQ）与多级反馈队列调度算法（MFQ）

设置多个优先级队列，第一级队列优先级最高。为不同队列分配长度不同的时间片（第一级最短），上一级队列为空时进入下一级队列调度，各级队列按照时间片轮转调度。若允许抢占，被抢占进程回到原一级队列队尾。

6. 各种调度算法的比较

调度算法	FCFS	RR	SJF	SRTN	HRRN	Feedback
选择函数	$\max\{w\}$	常数	$\min\{s\}$	$\min\{s-e\}$	$\max\left\{\frac{w}{s}\right\}$	队列
抢占	×	√	×	√	×	√
吞吐量	不强调	时间片小时低	高	高	高	不强调
响应时间	可能很高	短进程低	短进程低	低	低	不强调
开销	最小	最小	可能高	可能高	可能高	可能高
对进程影响	对短、I/O 进程不利	公平	对长进程不利	对长进程不利	平衡	可能对 I/O 进程有利
饥饿	×	×	√	√	×	√

五、进程同步机制

1. 进程的特征

①并发，共享，不确定性；
②程序执行结果的不可再现性；

③并发环境下进程间断执行；
④资源共享；
⑤独立性、制约性；
⑥程序和计算不再一一对应。

2. 进程互斥、临界资源、临界区

进程互斥：由于各要求使用共享进程，而这些资源需要排他使用，各进程之间这种竞争使用资源的关系称为是进程互斥。
临界资源：系统中某些一次只允许一个资源使用的资源称为是临界资源（互斥资源，共享变量）。
临界区（互斥区）：各个进程中对临界资源实施操作的程序片段。

3. 临界区的使用原则、前提

原则；
①有空让进：当没有进程在临界区时，任何有权使用共享资源的进程都有权进入临界区；
②无空等待：不允许两个以上进程同时进入临界区；
③有限等待：任何进入临界区的要求应该能在有限时间内满足。
前提：任何进程无权停止其他进程的运行；进程之间相对运行素的无硬性规定。

4. 什么是进程同步

进程同步是指系统中多个进程中发生的事件存在某种时序关系，需要相互合作共同完成一项任务。

5. 自旋锁（忙等锁）的优势和劣势

优点：尽量减少线程的阻塞，避免两次上下文切换的开销；
缺点：由于进程长时间占有 CPU 却不进行任何工作，导致获取锁的时间过长，自旋消耗远大于上下文切换的消耗，导致 CPU 资源的浪费。

6. Hoare 管程和 Mesa 管程的区别：

①Mesa 管程出错较少；
②Hoare 管程用 if 语句判断，Mesa 管程用 while 语句判断；
③当出现进程 P 唤醒进程 Q 的情形时，Hoare 管程会让进程 Q 执行，Mesa 管程的执行顺序是任意的。

7. 锁和条件变量

锁是一个互斥量，有两种状态，可以加锁/解锁；条件变量是条件等待。区别在于条件变量的等待用的是 if，锁的等待用的是 while。

8. 进程的基本通信机制

消息传递，共享内存，管道，套接字，远程过程调用。