

进程

进程常考的知识点

★重点

关于进程的特性

1. 进程失去封闭性（进程执行的结果只取决于进程本身）后，不同速度下的执行结果不同

2. 进程被建立后，随着进程运行的正常或不正常结束而撤销

3. 进程不是一个完整的程序

- 进程是动态的，程序是静态的
- 进程是暂时的，程序是永久的
- 进程至少由代码，数据，pcb组成，程序

4. 一个进程是程序在一个数据集上的一次运行过程；运行于不同的数据集，会形成不同的进程仅需代码和数据

5. 全局变量是针对一个进程而言的，不同的进程拥有不同的代码段和数据段，不能通过全局变量来交换进程之间的数据

6. 临界资源一次只能为一个进程所用

★重点

关于进程与进程的关系

1. 进程与进程之间是完全隔离的，线程与线程之间是没有隔离的，但并不意味着进程之间是毫无关联的

- 如一个进程的状态变化可能会引起另一个进程的状态变化
- 如打印机进程结束后可能会改变另一个等待打印机的进程

★重点

关于各种状态

1. 死锁的进程是处于阻塞态的，不可能所有的进程都处于就绪态，但是可以所有的进程都处于阻塞态(死锁的时候)

2. 就绪队列进程的多少与处理器的效率无关，只有当就绪队列为空时，CPU进入等待态，cpu效率下降

关于进程里的信息

C语言的程序在使用内存时有3个段

	存放什么？	举例
正文段	存二进制代码，常量	全局赋值变量，常量值123
数据堆段	存动态分配的存储区	用malloc动态分配的存储区
数据栈段	存临时使用的变量	未赋值的局部变量，函数调用实参传递值
PCB	进程的描述信息和控制信息等	进程状态（就绪还是阻塞）CPU中的寄存器内容

进程的概念与特征

★重点定义	<ul style="list-style-type: none">进程是<u>程序的一次执行过程</u>进程是一个程序及其数据在处理机上<u>顺序执行时所发生的活动</u>进程是具有独立功能的<u>程序在一个数据集合上运行的过程</u>，是系统进行资源分配和调度的一个独立单位
特征	<ul style="list-style-type: none">动态性并发性独立性异步性
为什么要引入进程？	<ul style="list-style-type: none">为了使多道程序并发执行，提高资源利用率和系统吞吐量为了可以对并发执行的程序加以描述和控制
★重点其他考点	<ul style="list-style-type: none"><u>进程实体/进程映像</u> = PCB + 程序段 + 相关数据段<u>进程映像</u>是静态的，<u>进程</u>是动态的，<u>程序</u>是静态的<u>进程控制块</u> (PCB, Process Control Block) 是进程的唯一标志<u>系统资源</u> = 处理机，存储器和其他设备服务于某个进程的“时间” 【这里的系统资源类似于处理机的时间片】

进程的状态与转换

★重点 五种状态	运行态 Running	该时刻进程占用CPU
	就绪态 Ready	进程获得了除处理机外的一切所需资源，一旦得到处理机，就可以立即运行
	阻塞态 Blocked	该进程正在等待某一事件发生而暂停运行，即使给它CPU控制权，它也无法运行
	创建态 New	进程正在被创建时的状态
	结束态 Exit	进程正在从系统中消失时的状态
状态转换图	<p>The diagram illustrates the transitions between process states. It starts with '创建状态' (New) leading to '就绪状态' (Ready) via '进入就绪队列' (Enter ready queue). From '就绪状态', a process can be '被调度' (Scheduled) to '运行状态' (Running) or '时间片用完' (Time slice used up) back to '就绪状态'. From '运行状态', a process can '结束' (End) to '结束状态' (Exit), '等待事件' (Wait for event) to '阻塞状态' (Blocked), or '事件完成' (Event completed) back to '就绪状态'.</p>	
状态的转换	就绪态→运行态	1. 进程被调度，获得处理机资源（分派处理机时间片）
	运行态→就绪态	1. 时间片用完后，不得不让出处理机 2. 可剥夺的OS中，当有更高优先级的进程就绪时，调度程序将正在执行的进程转换为就绪态
	运行态→阻塞态	1. 进程请求某一资源（如外设）的使用和分配时 2. 等待某一事件的发送时（如I/O操作的完成） • 进程以系统调用的方式请求OS提供服务，这个过程系统从用户态转换为核心态 • 该过程是主动行为
	阻塞态→就绪态	1. I/O操作结束或中断结束时 2. 发送了阻塞队列等待的事件，如发送了V操作，信号量+1，然后阻塞队列被唤醒到就绪队列中 • 该过程是被动行为，需要其他相关进程的协助



进程的控制结构/数据结构

重点

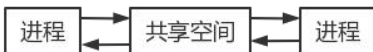
• 进程的控制结构 = PCB + 程序段 + 数据段

1.PCB	<ul style="list-style-type: none">PCB 是进程存在的唯一标识一个进程的存在，必然会有一个 PCB，如果进程消失了，那么 PCB 也会随之消失	
PCB包含什么信息?	进程描述信息	<ul style="list-style-type: none">进程标识符PID: 标识各个进程，每个进程都有一个并且唯一的标识符用户标识符UID: 进程归属的用户，用户标识符主要为共享和保护服务
	进程控制和管理信息	<ul style="list-style-type: none">进程当前状态，如 new、ready、running、waiting 或 blocked 等进程优先级: 进程抢占 CPU 时的优先级
	资源分配清单	<ul style="list-style-type: none">有关内存地址空间或虚拟地址空间的信息所打开文件的列表和所使用的 I/O 设备信息
	CPU 相关信息	<ul style="list-style-type: none">CPU 中各个寄存器的值当进程被切换时，CPU 的状态信息都会被保存在相应的 PCB 中以便进程重新执行时，能从断点处继续执行
如何组织PCB?	链接方式	<ul style="list-style-type: none">把统一状态的PCB链成一个队列，不同状态对应不同的队列也可把处于阻塞态的进程的PCB，根据阻塞原因，排成多个阻塞队列
	索引方式	<ul style="list-style-type: none">将统一状态的进程组织在一个索引表中如就绪索引表，阻塞索引表
2.程序段	<ul style="list-style-type: none">能被进程调度程序调度到CPU执行的程序代码段程序可以被多个进程共享，即多个进程可以运行同一个程序	
3.数据段	<ul style="list-style-type: none">可以是进程对应的程序加工处理的原始数据可以是程序执行时产生的中间或最终结果	



重点

进程的通信

定义	进程通信是指进程之间的信息交换		
低级通信方法	PV操作		
高级通信方法	共享存储	<ul style="list-style-type: none">如共享内存，共享文件系统	
	信息传递	<ul style="list-style-type: none">需要发送消息和接受信息两个原语	
	管道通信	<ul style="list-style-type: none">需要满足的能力 = 互斥 + 同步 + 确定对方的存在缓冲区只允许一边写入，另一边读出，所以管道只能采用半双工通信利用共享文件进行的进程通信机制称为管道，管道即文件	
	示例图		
	共享存储	信息传递	管道通信
			



进程的控制（也就是进程的创建、终止、阻塞、唤醒的过程）

	定义和过程	对应事件
创建	<ul style="list-style-type: none">• 允许一个进程创建另一个进程• 允许子进程继承父进程所拥有的资源• 创建进程的过程如下：<ul style="list-style-type: none">• 申请一个空白的 PCB，并向 PCB 中填写一些控制和管理进程的信息，比如进程的唯一标识等• 为该进程分配运行时所必需的资源，比如内存资源• 将 PCB 插入到就绪队列，等待被调度运行	<ol style="list-style-type: none">1. 终端登陆系统，作业调度2. 系统提供服务3. 用户程序的应用<ul style="list-style-type: none">• 设备分配不需要创建进程
终止	<ul style="list-style-type: none">• 有 3 种终止方式：正常结束、异常结束以及外界干预• 当子进程被终止时，其在父进程处继承的资源应当还给父进程• 当父进程被终止时，该父进程的子进程就变为孤儿进程• 终止进程的过程如下：<ul style="list-style-type: none">• 查找需要终止的进程的 PCB• 如果处于执行状态，则立即终止该进程的执行，然后将 CPU 资源分配给其他进程• 如果其还有子进程，则应将该进程的子进程交给1号进程接管• 将该进程所拥有的全部资源都归还给操作系统• 将其从 PCB 所在队列中删除	<ol style="list-style-type: none">1. 正常结束2. 异常结束3. 外界干预
阻塞	<ul style="list-style-type: none">• 当进程需要等待某一事件完成时，它可以调用阻塞语句把自己阻塞等待• 一旦被阻塞等待，只能由另一个进程唤醒• 阻塞进程的过程如下：<ul style="list-style-type: none">• 找到将要被阻塞进程标识号对应的 PCB• 如果该进程为运行状态，则保护其现场，将其状态转为阻塞状态，停止运行• 将该 PCB 插入到阻塞队列中去	<ol style="list-style-type: none">1. 请求系统资源失败2. 等待某种操作的完成3. 新数据尚未到达或无新任务可做
唤醒	<ul style="list-style-type: none">• 进程由「运行」转变为「阻塞」状态是由于进程必须等待某一事件的完成• 处于阻塞状态的进程是绝对不可能叫醒自己• 如果某进程正在等待 I/O 事件，需由别的进程发消息给它• 只有当该进程所期待的事件出现时，才由发现者进程用唤醒语句叫醒它• 唤醒进程的过程如下：<ul style="list-style-type: none">• 在该事件的阻塞队列中找到相应进程的 PCB• 将其从阻塞队列中移出，并置其状态为就绪状态• 把该 PCB 插入到就绪队列中，等待调度程序调度	<ol style="list-style-type: none">1. 释放该I/O设备的进程2. 提供数据的进程

进程的上下文切换

上下文	上下文是指某一时刻CPU寄存器和PC的内容； 如寄存器值，用户和系统内核栈内容	
进程上下文切换	上下文切换是指一个进程切换到另一个进程的过程	
上下文切换的实质	上下文切换实质是处理机从一个进程的运行转到另一个进程上运行， 这个过程中，进程的运行环境产生了实质性的变化	
上下文切换的消耗	有些处理器提供多个寄存器组，上下文切换只需要简单改变当前寄存器组的指针，不需要用到磁盘和主存	
上下文切换的场景	<div>1. 某个进程时间片耗尽时</div> <div>2. 进程在系统资源不足时，要等到资源满足后才可以运行</div> <div>3. 进程通过sleep将自己主动挂起</div> <div>4. 有优先级更高的基础运行时</div> <div>5. 发送硬件中断时</div>	
上下文切换的流程	<div>1. 挂起一个进程，保存CPU上下文，包括PC和其他寄存器</div> <div>2. 更新PCB信息</div> <div>3. 把进程的PCB移到相应的队列（如就绪队列，阻塞队列）</div> <div>4. 加载另一个进程执行，更新其PCB</div> <div>5. 跳转到新进程PCB中的PC所指向的位置执行</div> <div>6. 恢复处理机上下文</div>	
示例图	<div><div><div>进程 1</div><div>进程 1 上下文 保存</div><div>加载 进程 2 上下文</div><div>进程 2</div></div><div>时钟</div><div>进程上下文切换</div></div>	
调度和切换的区别	先有资源的调度，才有进程的切换	
	调度	• 决定资源分配给哪个进程的行为；是一种决策行为
	切换	• 实际分配的行为；是一种执行行为