# 操作系统复习

第三章： 进程

第三章总纲：

* Process Concept
* Process Scheduling
* Operations on Processes
* Interprocess Communication
* Examples of IPC Systems
* Communication in Client-Server Systems

**#进程的概念**

英文定义：a program in execution; process execution must progress in sequential fashion.

中文定义：进程是具有独立功能的程序关于某个数据集合上的一次运行活动，是系统进行资源分配和调度的独立单位，又称任务（Task or Job）。

**文本段（text section）**：程序代码；

**栈段（Stack）**：包含临时数据；

**数据段（data section）：**包括全局变量；

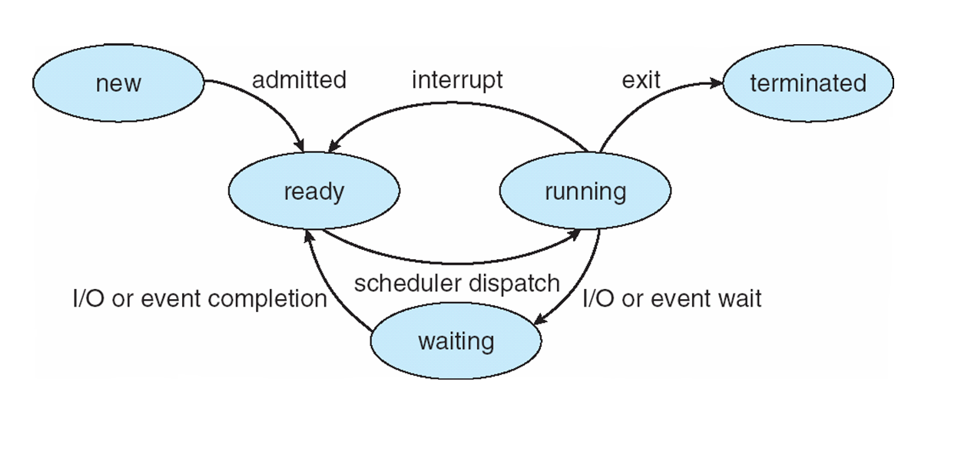
**堆（heap）：**进程运行期间动态分配的内存。

注意：程序本身不是进程，而是一个*被动实体*，而进程是*活动实体*，它有一个程序计数器来表示下一个要执行的命令和相关资源合集。

**#进程状态**

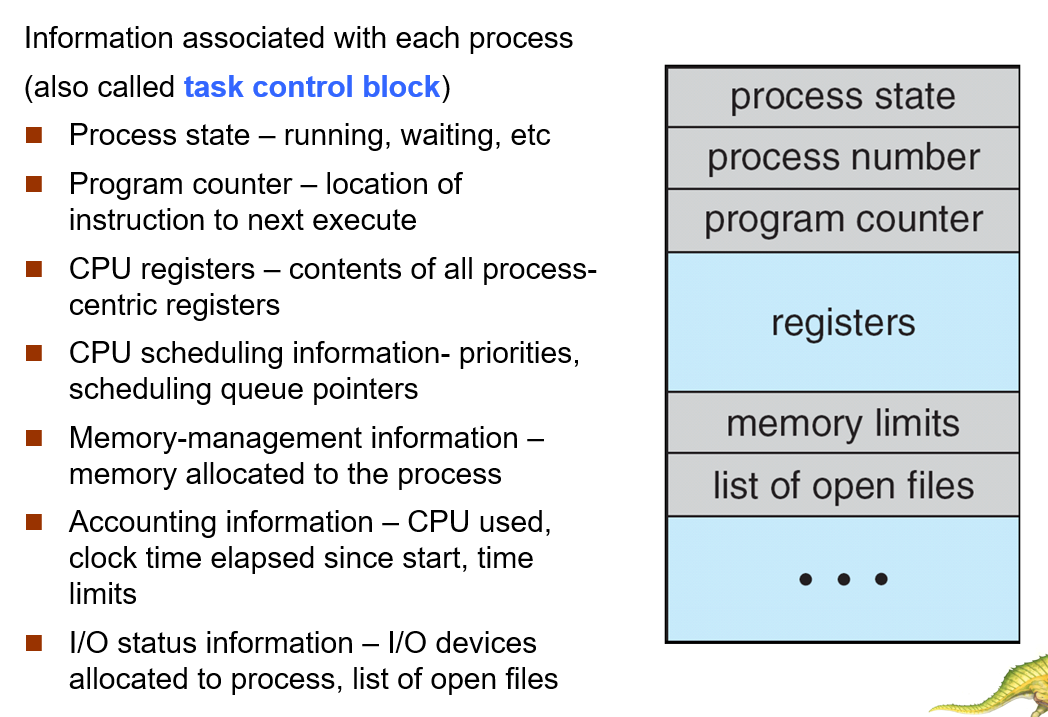
进程总共有五个状态

* + **new**: The process is being created
  + **running**: Instructions are being executed
  + **waiting**: The process is waiting for some event to occur
  + **ready**: The process is waiting to be assigned to a processor
  + **terminated**: The process has finished execution

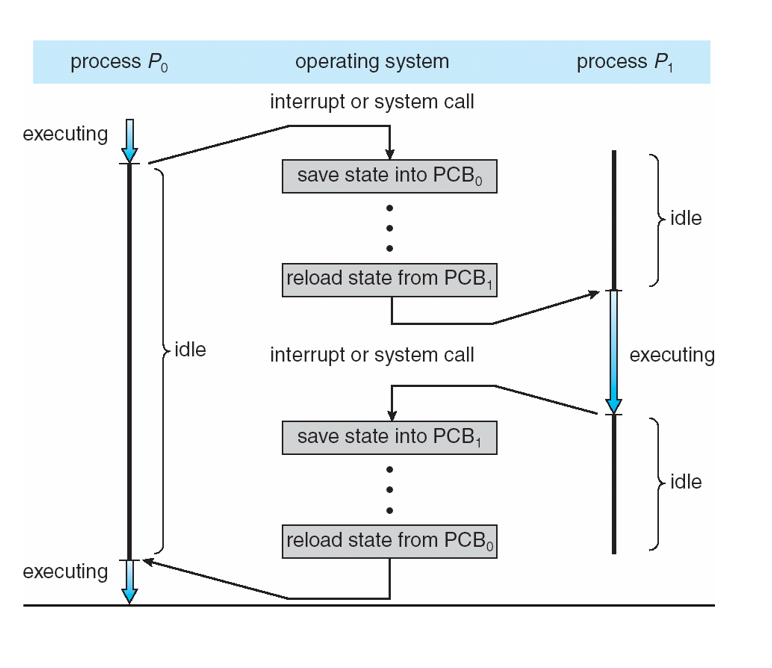


**#进程控制块（Process Control Block）**

每个进程在操作系统内用进程控制块来表示，又称进程描述符，操作系统用于管理控制进程的一个专门的数据结构，记录进程的各种属性，描述进程的动态变化的过程。



Cpu在进程间的切换：



**#进程调度**

目的是最大化CPU利用率。进程调度器选择一个可用的进程到CPU上执行。

**\*调度队列**：

Job queue—包含系统中的所有进程

Ready queue—驻留在内存中就绪的、等待运行的进程

Device queue—等待特定的IO设备的进程

**#调度程序**

进程在其生命周期中会在各种调度队列之间迁移。为了调度，操作系统必须按照某种方式从这些队列中选择进程。

**Short-term scheduler** 或者称为cpu scheduler：从准本执行的进程中选择进程，并为之分配CPU

**Long-term scheduler** (job scheduler)：从缓冲池中选择进程，并装入内存准备执行。

**Mid-term scheduler**：能将进程从CPU或内存中移除，从而降低多道程序设计的程度。--Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**.

这两个调度程序的主要差别是他们执行的频率。短期调度程序更频繁，长期调度程序执行的并不频繁，它控制着多道程序的调度。

**进程也可主要分为：**

**IO-bound process**：在执行IO方面比执行计算花费更多时间。

**CPU-bound process**：在执行计算方面花费更多时间。

**#上下文切换CONTEXT SWITCH**

中断使CPU从当前任务状态改变为运行内核子程序。When CPU switches to another process, the system must **save the state of the old process** and **load the saved state for the new process** via a context switch.

State restore 重新开始运行原进程。

将CPU切换到另一个进程需要保存当前进程的状态并回复另一个进程的状态，这一任务称为上下文切换。当发生上下文切换时，内核会将旧进程的状态保存在其PCB中，然后装入经调度要执行的并已保存的新进程的上下文。上下文切换时间是额外开销，因为切换时系统并不能做什么有用的工作。

**#进程操作**

**1 进程创建**

进程在其执行过程中，能通过创建进程系统调用创建多个新进程。创建进程称为**父进程**，被创建的新进程称为**子进程**。

唯一进程标识符 process identifier（pid）

资源共享情况：父与子同时共享资源；子分父资源的子集；父与子不共享资源。

执行情况：并发执行；父等待，直到某个或全部子进程执行完。

**UNIX：fork/exec**

**WINDOWS：CreateProcess**

**2 进程终止**

当进程执行完最后的语句并使用系统调用exit()请求操作系统删除自身时，进程终止。这时，进程可以返回状态值到父进程（通过系统调用wait（））。所有进程资源会被操作系统释放。

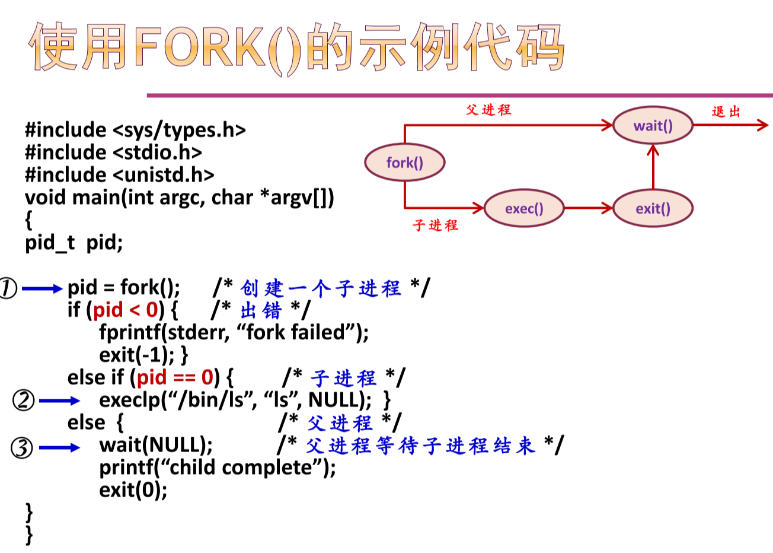
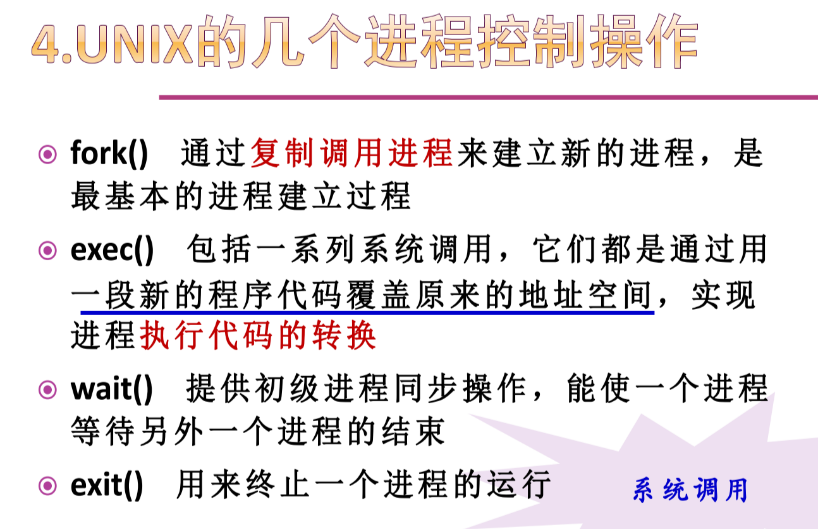
父进程也能通过系统调用abort()来终止子进程。

有些操作系统不允许当父进程终止了，子进程还在运行。

**UNIX：exit**

**WINDOWS：TerminateProcess**

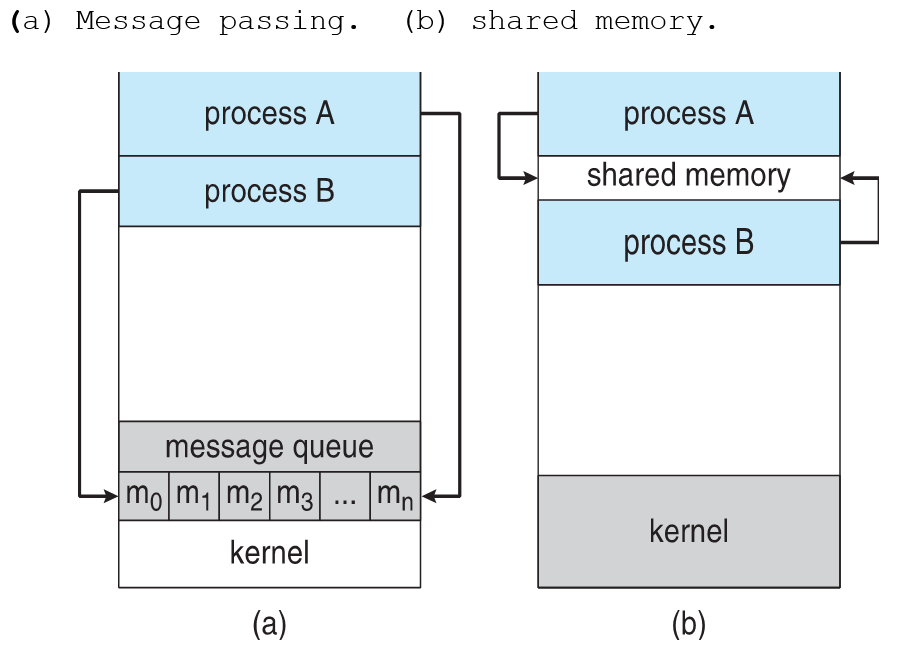
//unix的几个进程控制的操作



**#进程间通讯 inter-process communication**

与其它进程共享数据的进程是协作的（cooperating）。

Two models：Shared memory; Message passing



**生产者-消费者问题**

*Producer* process produces information that is consumed by a *consumer* process. 生产者进程产生信息以供消费者消费。

**两种缓冲：**

无限缓冲：对缓冲大小没有限制。消费者可能不得不等待新的项，但生产者总是可以产生新项。

**有限缓冲：缓冲大小固定。如果缓冲为空，那么消费者必须等待；如果缓冲为满，那么生产者必须等待。**

**#进程间通信—message passing**

**消息传递系统**

这种机制允许进程不必通过共享地址空间来实现通信和同步。

提供两种操作：

发送send

接受receive

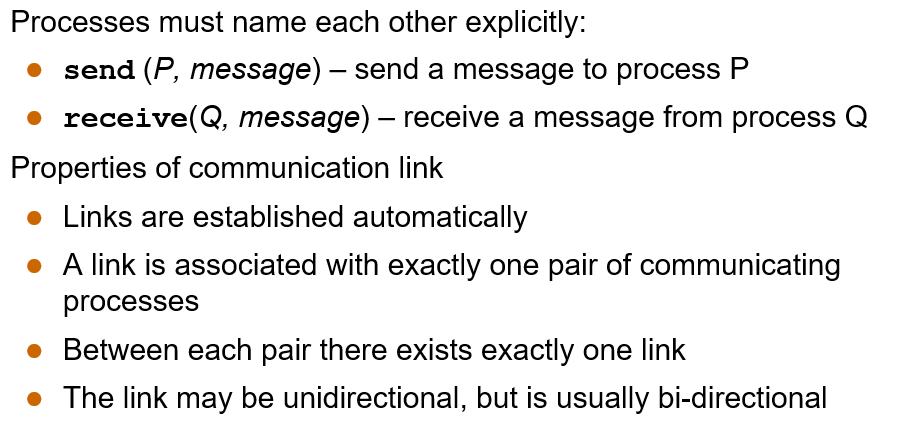
如果进程P和Q要通信，它们必须先建立**通信链路（communication link）**。

实现通信链路：

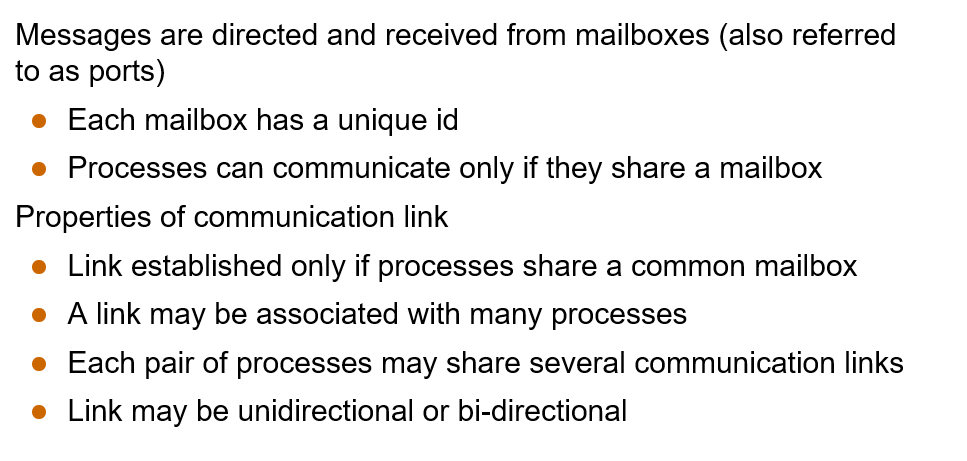
物理：共享内存；硬件总线；网络。

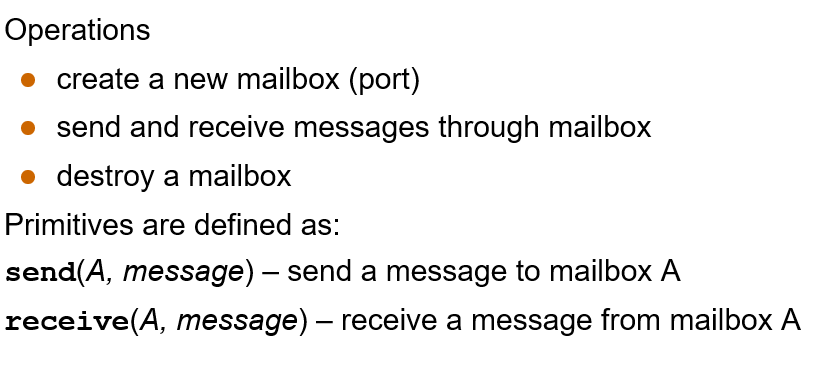
**逻辑：直接或间接通信；同步或异步通信；自动或显式缓冲。**

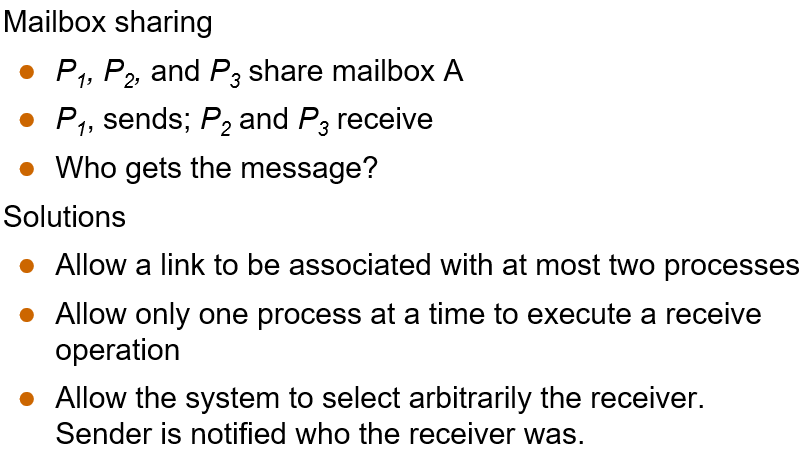
**直接通信：**



**间接通信：**

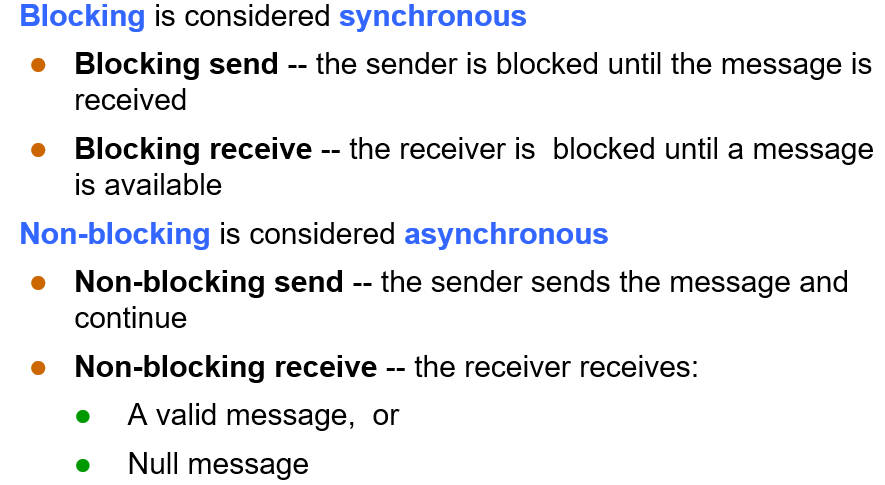






**同步：**

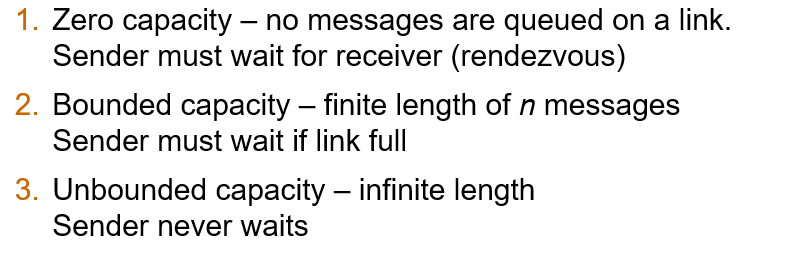
消息的传递可以是阻塞或非阻塞（blocking or non-blocking）--也称同步或异步。



Send和receive可以进行多种组合。当send和receive阻塞时，则在发送者和接受者之间就有一个集合点（rendezvous）。当通信是同步时，生产者消费者问题就不再重要，生产者仅需调用阻塞send并等待，知道消息被送到接收者或邮箱。

**缓冲：**

不管通信是直接的或是间接的，通信进程所交换的消息都驻留在临时队列中。



**#客户机-服务器系统通信**

\*sockets：

套接字可定义为通信的端点。一对通过网络通信的进程需要使用一对Socket。它由IP地址与一个端口号连接组成（**161.25.19.8:1625**）。所有低于1024的端口都被认为是众所周知的，可以用来实现标准服务。

\*远程过程调用（Remote Procedure Calls）

RPC设计成抽象过程调用机制，用于通过网络连接系统。

Stub（存根）：RPC语义允许客户机调用位于远程主机上的过程。通过在客户端提供存根，RPC系统隐藏了允许通信发生的必要细节。

