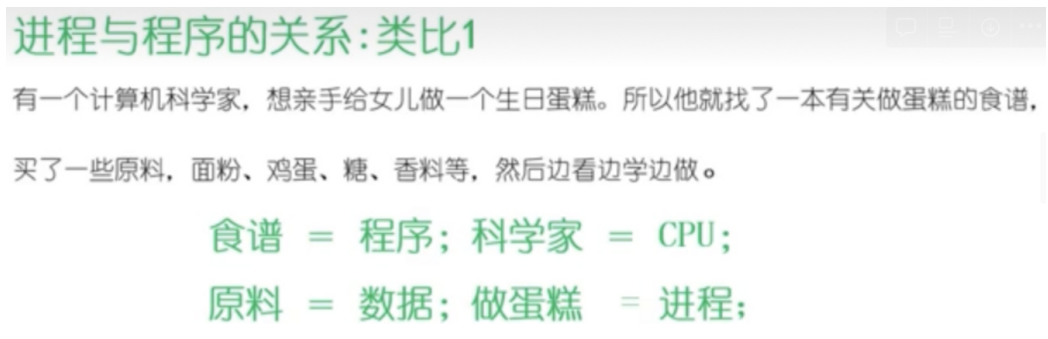
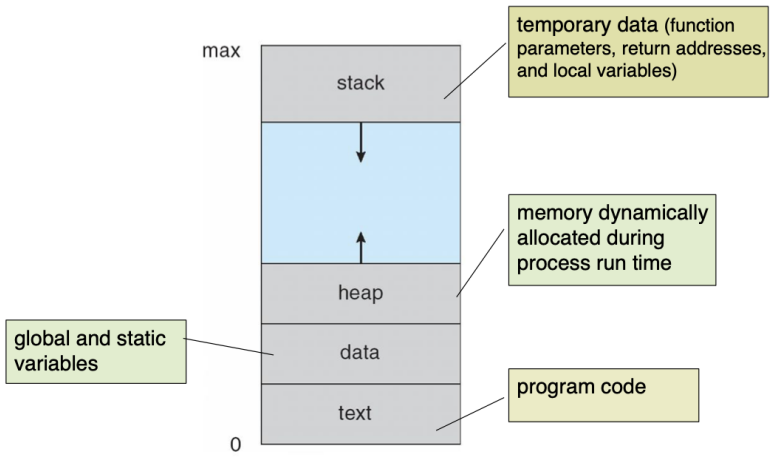
2 Process

- os
  - 概念
    - An interface between users and hardware - an environment"architecture”
    - Allows convenient usage; hides the tedious stuff
    - Allows efficient usage; parallel activity, avoids wasted cycles
    - Provides information protection
    - Gives each user a slice of the resources
    - Acts as a control program.
    - 用户和硬件之间的接口——环境“体系结构”
    - 使用方便;隐藏那些乏味的东西
    - 允许有效使用;并行活动，避免浪费周期
    - 提供信息保护
    - 为每个用户提供资源的一部分
    - 作为一个控制程序
  - Characteristics
    - Time Sharing - multiprogramming environment that's also interactive.
      - 分时-多程序环境，也是交互式的。
    - Multiprocessing - Tightly coupled systems that communicate via shared memory. Used for speed improvement by putting together a number of off-the-shelf processors.
      - 多处理——通过共享内存进行通信的紧密耦合系统。通过将许多现成的处理器组合在一起来提高速度。
    - Distributed Systems - Loosely coupled systems that communicate via message passing. Advantages include resource sharing, speed up, reliability, communication.
      - 分布式系统——通过消息传递进行通信的松散耦合系统。优点包括资源共享、速度、可靠性、通信。
    - Real Time Systems - Rapid response time is main characteristic. Used in control of applications where rapid response to a stimulus is essential.
      - 实时系统-快速响应时间是主要特点。用于必须对刺激作出快速反应的控制应用。
  - O.S. carry out the most commonly required operations:
    - Process Management,
    - Memory management,
    - File System Management,

        - I/OSystemManagement,
        - Protection and Security.
            - 流程管理,
            - 记录管理,
            - 文件系统管理,
            - i/ OSystemManagement,
            - 保护和安全。
- Process Concept
    - Process Concept
        - Process = a program in execution 正在执行的程序
            - process execution must progress in sequential fashion按照顺序执行
        - An operating system executes a variety of programs:
            - 操作系统执行各种程序
        - A process is considered an 'active' entity
            - 流程被认为是一个“活动”实体
        - A program is considered to be a 'passive' entity (stored on disk (executable file)).
            - 程序被认为是一个“被动的”实体(存储在磁盘上(可执行文件))
        - Program becomes process when executable file loaded into memory
            - 当可执行文件载入内存时，程序就变成了进程
        - Process in Memory
            - temporary data (function parameters, return addresses, and local variables)
                - 临时数据(函数参数、返回地址和局部变量)
            - memory dynamically allocated during process run time
                - 进程运行时动态分配的内存
            - global and staticvariables
                - 全局变量和静态变量
            - program code
                - 程序代码
            -





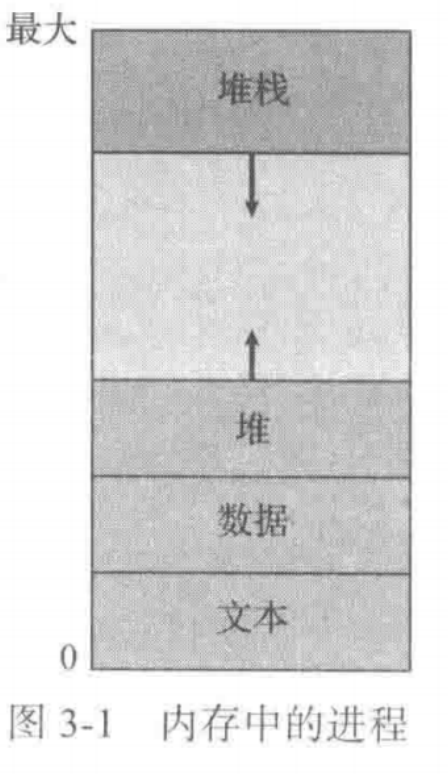
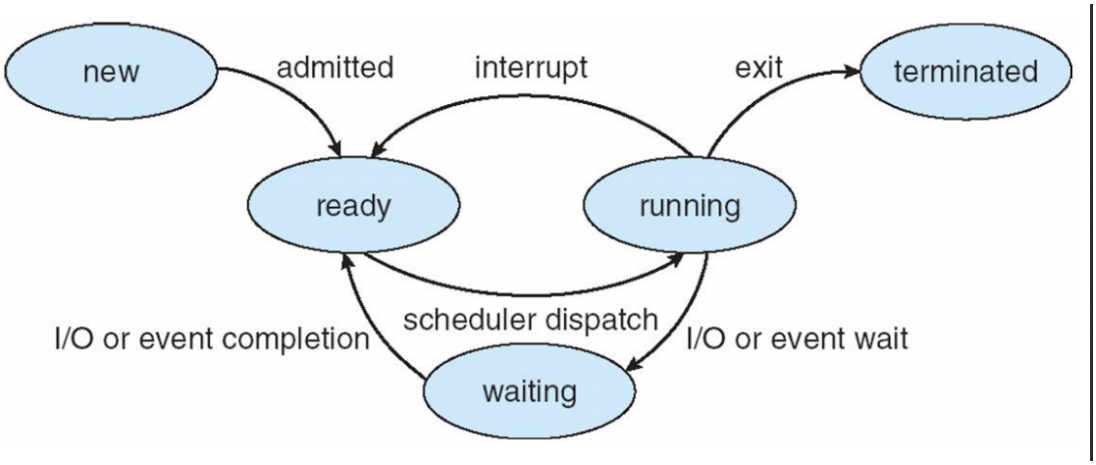
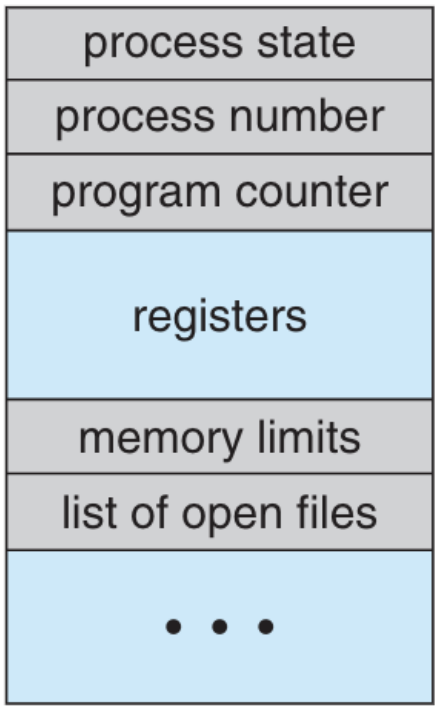


图 3-1　内存中的进程

- Process State
  - As a process executes, it changes state.
  - 当流程执行时，它会改变状态。
  - 流程的状态部分是由该流程的当前活动定义的
  - The state of a process is defined in part by the current activity of that process.
  - new: The process is being created
    - 创建过程中
  - running: Instructions are being executed
    - 正在执行指令
  - waiting: The process is waiting for some event to occur
    - 进程正在等待某个事件发生

    - 消费者waiting着供货商提供菜 0 1
  - ready:The process is waiting to be assigned to a processor
    - 该进程正在等待分配给处理器
  - terminated: The process has finished execution
    - 进程已经执行完毕
  - 一次只有一个进程在处理器运行,但是可以有很多在就绪或者等待
  - 

- Process Control Block (PCB)
  - 进程控制块:操作系统管理控制进程运行所用的信息集合
  - 操作系统使用pcb描述进程的基本情况和运行变化的过程
  - Process Control block is a data structure used for storing the information about a process.
    - 进程控制块是一种用于存储进程信息的数据结构。
    - pcb的组织方式
      - 链表
        - 同一状态的进程其pcb成一链表,多个状态对应多个不同的链表
      - 索引表
        - 同一个状态的进程归入一个index表 多个状态对应多个不同的index表
  - Each & every process is identified by its own PCB.
    - 每个过程都由自己的PCB识别
  - PCB of each process resides in the main memory.
    - 每个进程的PCB驻留在主存中
  - PCB of all the processes are present in a linked list.
    - 所有进程的PCB都在一个链表中
  - PCB is important in multiprogramming environment as it captures the information pertaining to the number of processes running simultaneously.
    - PCB在多路编程环境中很重要，因为它可以捕获有关同时运行的进程数量的信息

- PID = Process identifier Unique number
  - 

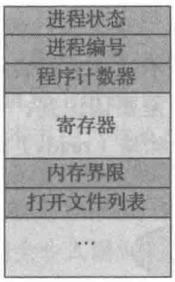
  - 

图 3-3　进程控制块（PCB）

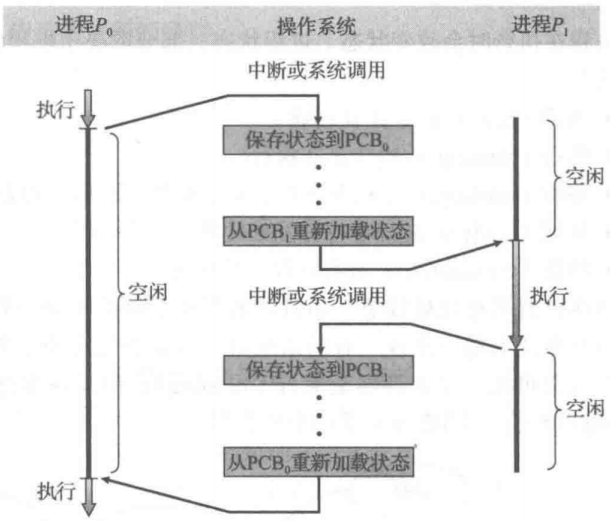
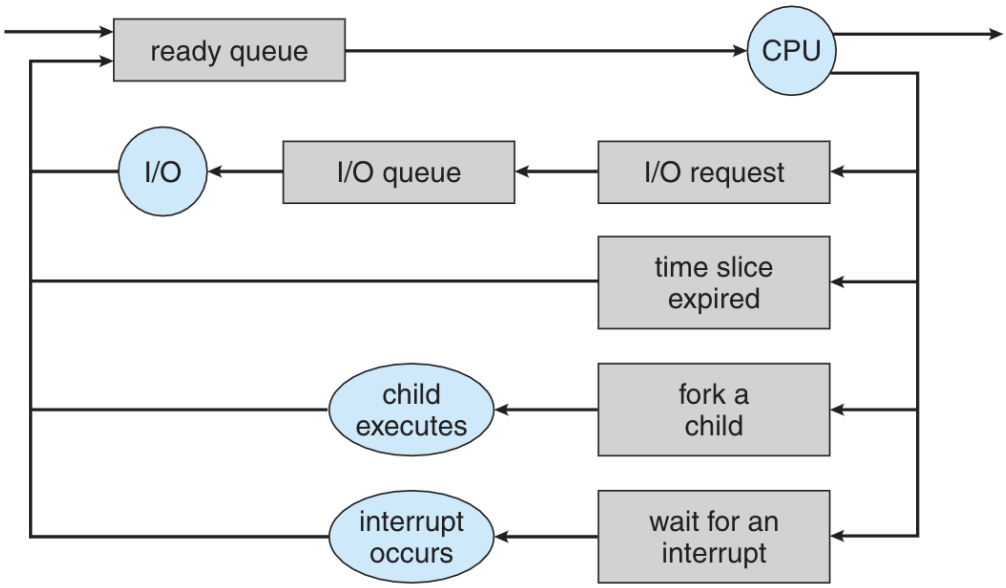


图 3-4　进程间的 CPU 切换
  -



A common representation of process scheduling is a **queueing diagram**, such as that in Figure 3.6. Each rectangular box represents a queue. Two types of queues are present: the ready queue and a set of device queues. The circles represent the resources that serve the queues, and the arrows indicate the flow of processes in the system.

A new process is initially put in the ready queue. It waits there until it is selected for execution, or **dispatched**. Once the process is allocated the CPU and is executing, one of several events could occur:

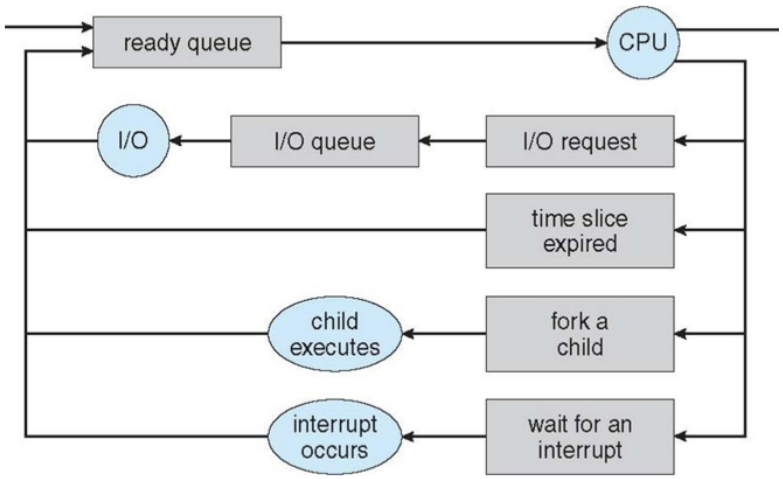
- The process could issue an I/O request and then be placed in an I/O queue.
- The process could create a new child process and wait for the child's termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.

In the first two cases, the process eventually switches from the waiting state to the ready state and is then put back in the ready queue. A process continues this cycle until it terminates, at which time it is removed from all queues and has its PCB and resources deallocated.

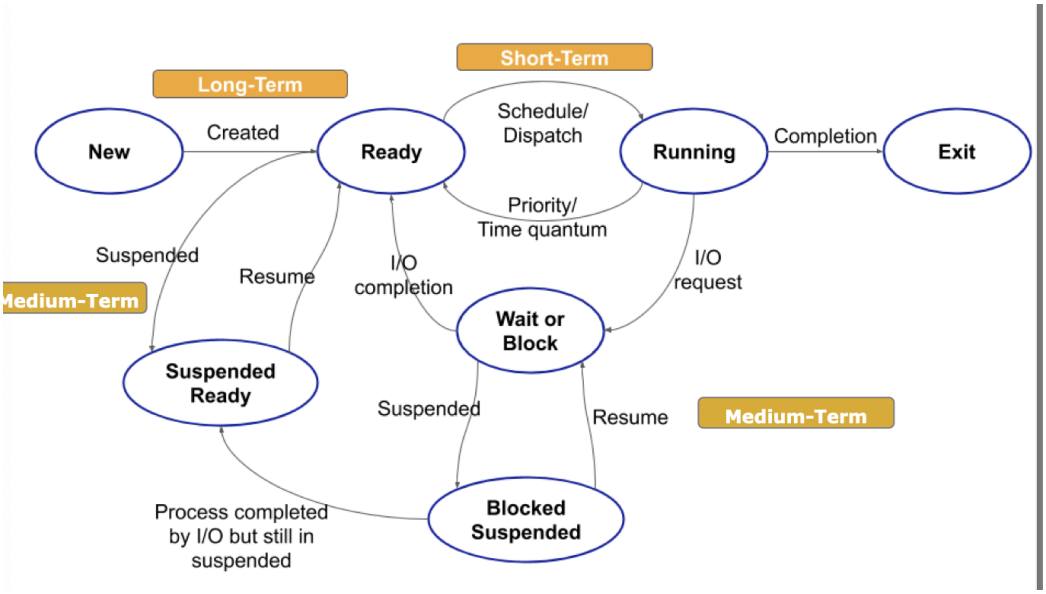
- 队列图中进程调度的常见表示，如图3.6所示。每个矩形框表示一个队列。有两种类型的队列：就绪队列和一组设备队列。圆圈表示为队列提供服务的资源，箭头表示系统中的进程流。新的进程在队列中执行。它一直在那里，直到它被选择执行或跳过。一旦进程被分配了CPU并正在执行，可能会发生以下几个事件之一：•进程可能会发出I/O请求，然后被放入I/O队列。•该进程可以创建一个新的子进程，并等待子进程的终止。•由于中断，进程可能会被强制从CPU中删除，并被放回就绪队列。在前两种情况下，进程最终从等待状态切换到就绪状态，然后被放回准备队列。进程继续这个循环，直到它终止，此时它将从所有队列中删除，并释放其PCBand资源

- Process Scheduling
  - Process execution consists of alternating sequence of CPUexecution and I/O wait.
    - 进程执行由CPU执行和I/O等待的交替顺序组成。

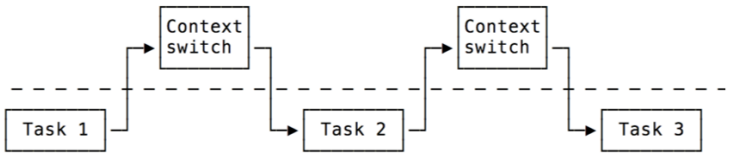
- The scheduling problem:
  - • The system has k processes ready to run•
  - The system has n ≥ 1 CPUs that can run them
- Process scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
  - 进程调度器从内存中准备执行的进程中进行选择，并将CPU分配给其中一个进程
- Scheduling queues of processes:
  - Job queue – set of all processes in the system
    - 作业队列-系统中所有进程的集合
  - Ready queue – set of all processes residing in main memory, ready and waiting to execute
    - 准备队列——所有驻留在主存中的进程的集合，准备就绪并等待执行
  - Device queues – set of processes waiting for an I/O device
    - 设备队列—等待I/O设备的进程集
    -



- Long-Term Scheduler is also called Job Scheduler and is responsible for controlling the Degree of Multiprogramming i.e.the total number of processes that are present in the ready state.长期调度器也称为作业调度器，负责控制多路编程的程度，即处于就绪状态的进程总数。
- 差别在于执行频率
- Short-Term Scheduler is also known as CPU scheduler and is responsible for selecting one process from the ready state for scheduling it on the running state.也称为CPU Scheduler，负责从就绪状态中选择一个进程，将其调度到运行状态
- Medium-term scheduler is responsible for swapping of a process from the Main Memory to Secondary Memory an dvice-versa (mid-term effect on the performance of the system).中期调度器负责将进程从主存交换到从存(硬盘)，反之亦然(对系统性能的中期影响)

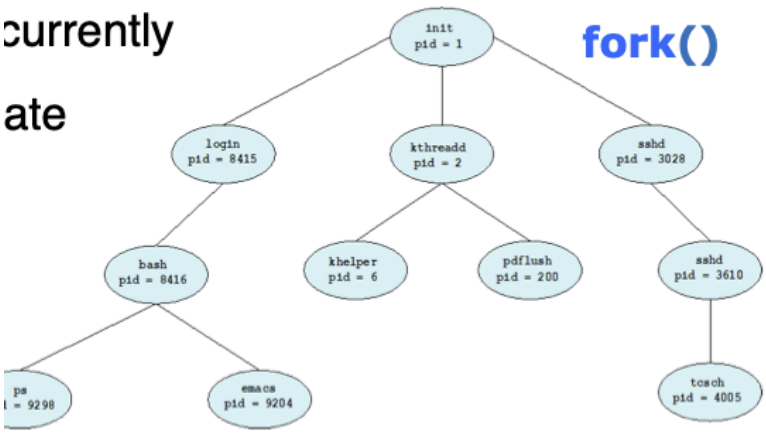


- Context Switch上下文切换
  - When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a CONTEXT SWITCH
  - Context of a process represented in the PCB
  - 当CPU切换到其他进程时，系统必须保存旧进程的状态，并通过CONTEXT SWITCH将保存的状态加载给新进程
  - 把旧的进程状态保存在pcb中,
  -

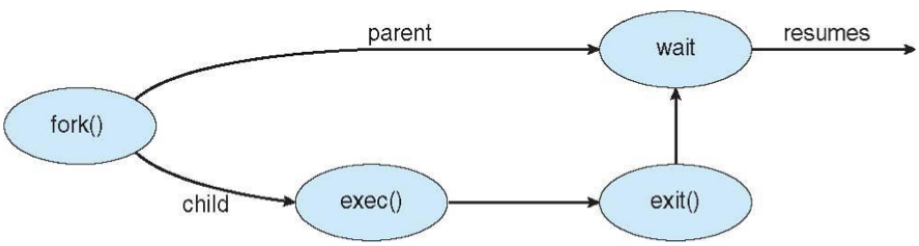


- Operations on Processes
  - System must provide mechanisms for:
    - process creation, process termination,
  - process creation进程创建
    - 通过系统调用fork创建新进程
    - Parent process create children processes, which, in turn cr ..0
      - 父进程创建子进程，子进程又创建其他进程，形成进程树
    - Resource sharing options
      - 资源共享选项
      - Parent and children share all resources
        - 父母和孩子共享所有资源
      - Children share subset of parent's resources
        - 子节点共享父节点资源的子集

- Parent and child share no resources
  - 父母和孩子不共享资源
- 进程创建新进程的两种可能
  - Parent and children execute concurrently
    - 父级和子级同时执行
  - Parent waits until children terminate
    - 父节点等待子节点终止
    -

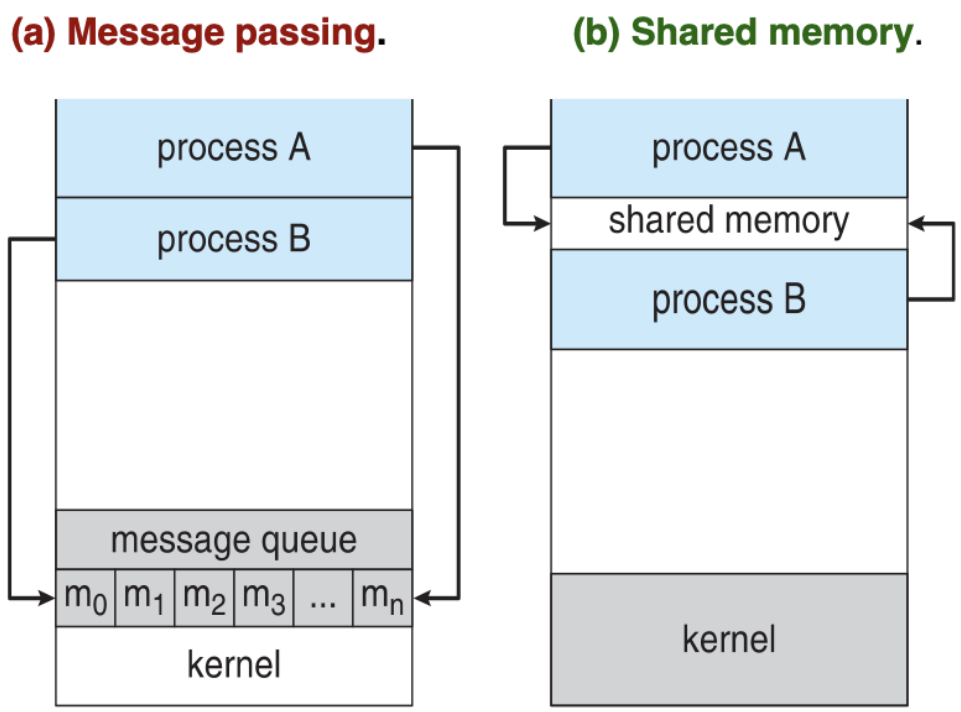


- process termination进程终止
  - Process executes last statement and then asks the operating system to delete it using the exit() system call.
    - 进程执行最后一条语句，然后使用exit()系统调用请求操作系统删除它。
    - ❖ Returns status data from child to parent
      - 返回从子节点到父节点的状态数据
    - ❖ Process' resources are deallocated by operating system
      - 进程的资源被操作系统释放
  - ❖Parent may wait terminate the execution of children processes.
    - 父进程可以等待终止子进程的执行。
    - ❖ Child has exceeded allocated resources
      - 已经超过分配的资源
      -



- Inter-process Communication

- 消息传递允许进程不必通过共享地址空间实现通信和同步
- INDEPENDENT PROCESSES - neither affect otherprocesses or be affected by other processes.
  - 独立进程——既不影响其他进程，也不受其他进程影响。
- COOPERATING PROCESSES - can affect or be affected byother processes.
  - 协作过程——可以影响或被其他过程影响。
- There are several reasons why cooperating processes are allowed:
  - 允许合作过程有以下几个原因:
  - Information Sharing - processes which need access to the same file for example.
  - Computation speedup - a problem can be solved faster if the problem can be broken down into sub-tasks to be solved simultaneously
  - Modularity - break a system down into cooperating modules. (e.g. databases with a client-server architecture.)
  - Convenience - even a single user may be multi-tasking, such as editing, compiling, printing, and running the same code in different windows.
  - 信息共享——例如需要访问相同文件的进程。
  - 计算加速——如果一个问题可以被分解成同时解决的子任务，那么这个问题可以更快地解决
  - 模块化——将系统分解为相互协作的模块。(例如，采用客户-服务器架构的数据库。)
  - 便捷性——即使是一个用户也可能同时进行多项任务，比如编辑、编译、打印和在不同的窗口中运行相同的代码。
- Communications Models
-



- Message-Passing Systems
  - communication takes place by way of messages exchanged among the cooperating processes.
  - A message-passing facility provides at least twooperations:
    - □send(message)
    - □receive(message)
  - The message size is either fixed or variable
  - 通信通过在合作进程之间交换消息的方式进行。
  - 消息传递工具至少提供两个操作:
    - □发送(消息)
    - □接收(消息)
  - 消息大小是固定的或可变的
  - If processes P and Q want to communicate: a communication link must exist between them.
  - Are several methods for logically implementing a link and the send()/receive() operations:
    - ▪ Direct or indirect communication
    - ▪ Synchronous or asynchronous communication
  - 如果进程P和进程Q想要通信:它们之间必须存在通信链路。
  - 有几个逻辑上实现链接和send()/receive()操作的方法:
    - ▪直接或间接沟通 a-b b-a
    - ▪同步或异步通信 a-b
- Shared-Memory Systems
  - a region of memory is shared by cooperating processes.processes can exchange information by reading and writing
  - all the data to the shared region.
  - Two types of buffers can be used:
    - unbounded-buffer places no practical limit on the size of the buffer
    - bounded-buffer assumes that there is a fixed buffer size
  - 一个内存区域由合作进程共享。进程可以通过读写来交换信息
  - 所有数据到共享区域。
  - 可以使用两种类型的缓冲区:
    - unbound -buffer对缓冲区的大小没有实际限制
    - bound -buffer假设有一个固定的缓冲区大小
- Direct or Indirect communication
  - exchanged messages by communicating processes reside in a temporary queue.

- 通过通信进程交换的消息驻留在临时队列中。

- BUFFERING
  - Zero capacity. The queue has a maximum length of zero; thus, the link cannot have any messages waiting in it.
    - 零能力。队列的最大长度为0;因此，链接中不能有任何等待的消息。
  - Bounded capacity. The queue has finite length n; thus, at most nmessages can reside in it.
    - 有限的能力。队列长度为n;因此，最多nmessages可以驻留在其中。
  - Unbounded capacity. The queue's length is potentially infinite.
    - 无限容量。队列的长度可能是无限的。
- Direct Communication
  - ❑ Processes must name each other explicitly:
    - □send (P, message) – send a message to process P
    - □receive(Q, message) – receive a message from process Q
  - ❑ Direct Communication is implemented when the processes use specific process identifier for the communication, but it is hard to identify the sender ahead of time.
  - ❑进程之间必须显式命名:
  - □send (P, message) -向进程P发送消息
  - □receive(Q, message) -从进程Q接收消息
  - ❑当进程使用特定的进程标识符进行通信时，实现了直接通信，但很难提前识别发送方。
- Indirect Communication
  - create a new mailbox (port)send and receive messages through mailbox
  - □send(A, message) – send a message to mailbox A
  - □receive(A, message) – receive a message from mailbox A
  - destroy a mailbox
  - 创建一个新的邮箱(端口)通过邮箱发送和接收消息
  - □send(A, message) -发送消息到邮箱A
  - □receive(A, message) -从邮箱A接收消息
  - 销毁邮箱temp
- Synchronous and Asynchronous Message Passing
  - Message passing may be either blocking or non-blocking
  - ▪ Blocking is considered synchronous
    - ▪ Blocking send -- the sender is blocked until the message is received
    - ▪ Blocking receive -- the receiver is blocked until a message is available
  - ▪ Non-blocking is considered asynchronous

        - ▪ Non-blocking send -- the sender sends the message and continue
        - ▪ Non-blocking receive -- the receiver receives:
            - ▪ A valid message, or
            - ▪ Nullmessage
- 消息传递可以是阻塞的，也可以是非阻塞的
- ▪阻塞被视为同步的
    - ▪阻塞发送—发送方被阻塞，直到收到消息
    - ▪阻塞接收——接收端被阻塞，直到消息可用
- ▪非阻塞被认为是异步的
    - ▪非阻塞发送—发送方发送消息并继续
    - ▪无阻塞接收—接收端接收:

以上内容整理于 幕布文档