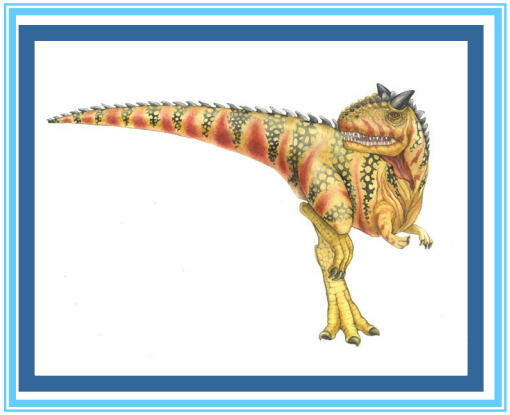
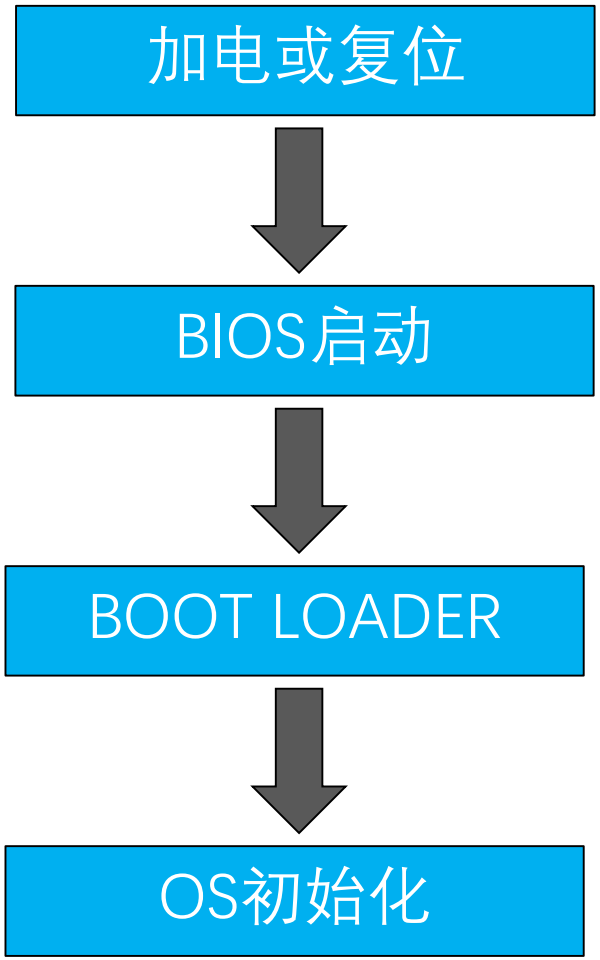
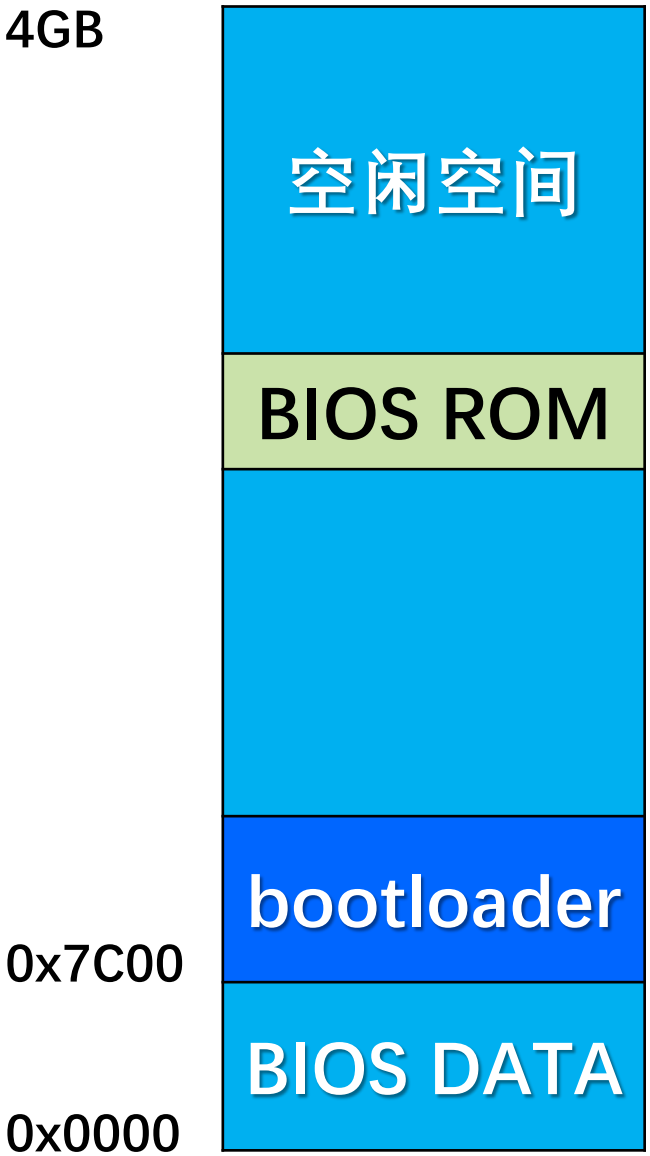
Revision & Exercise (2)

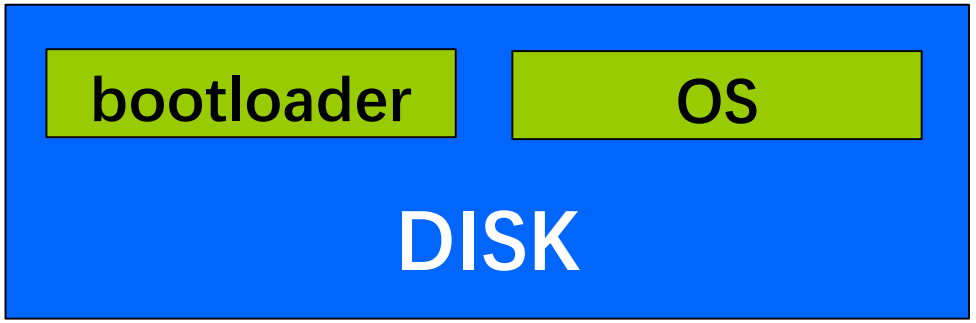


Operating System Boot



Operating System Boot



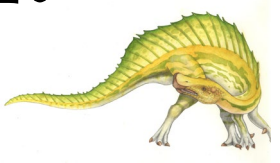


CS:IP = 0xf000:fff0

CS：段寄存器
IP：指令寄存器

Operating System Boot

- CS:IP组合：指向BIOS入口，作为处理器运行的第一条指
- CS（Code Segment）寄存器：存放当前运行段的起始地址；
- IP（Instruction pointer）寄存器：存放指令在代码段内的偏移量；
- CS:IP组合：确定下一条执行指令的物理地址。



BIOS启动

- BIOS：提供CPU需要的启动指令
- 启动程序的运行过程：
  - 上电自检->监测并连接系统硬件->从磁盘引导扇区读入Boot Loader到0x7C00

    （引导扇区：硬盘的0面0道1扇区中）
- BIOS还提供一组中断，以便对硬件设备的访问 。在OS未装入前，负责响应中断。

Operating System Initialization

- 当Boot Loader将控制权交给OS的初始化代码以后，OS开始其初始化工作
- OS负责：
  - 完成存储管理、设备管理、文件管理、进程管理的初始化
- 当OS的初始化工作完成以后，进入用户态，等待用户的操作

Operating System Services

- We can view OS from several points:
  - The services that the OS provides
  - The interface provided to users and programmers
  - Its components and interconnections

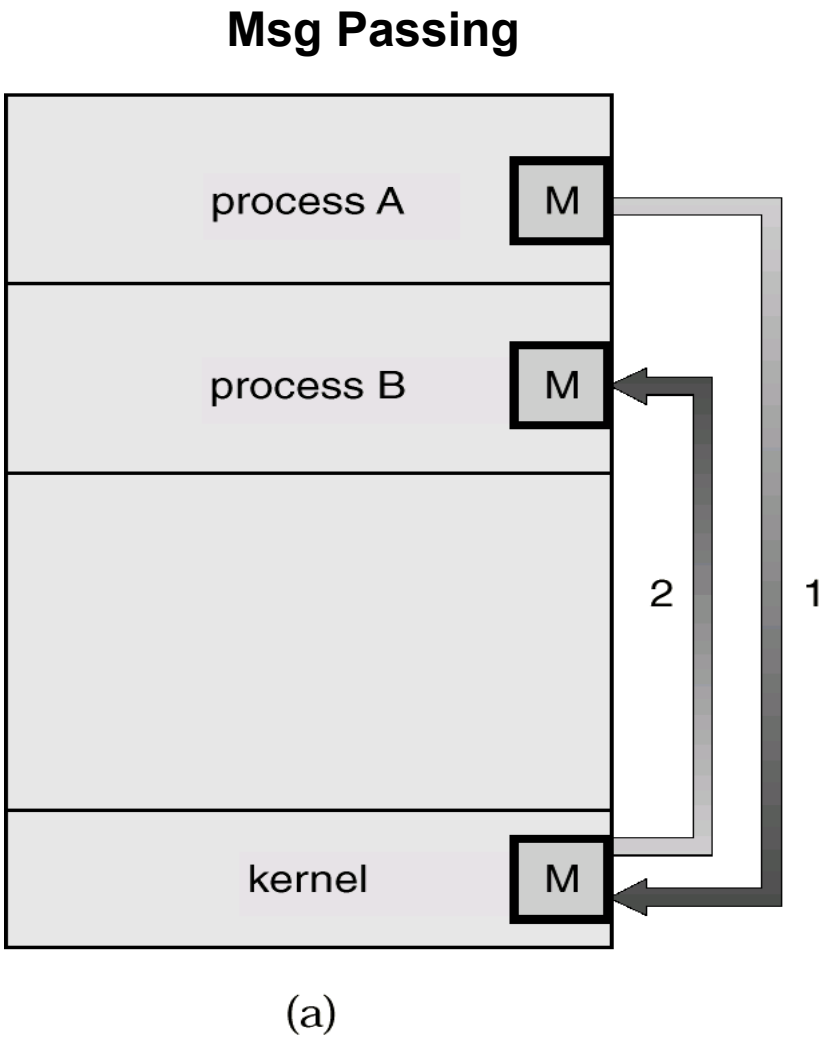
Operating System Interaction

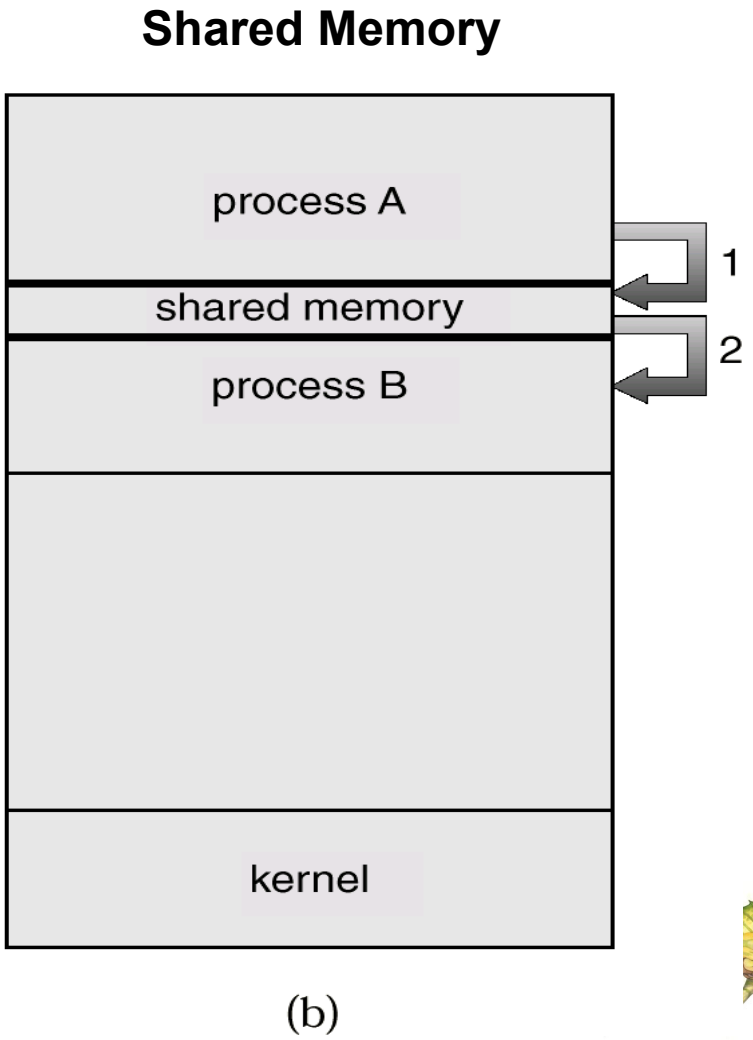
- **系统调用**（源于应用程序）
  - 应用程序主动向操作系统发出服务请求
- **异常**（源于不良应用程序）
  - 非法指令或其他异常状态（如：内存出错）
- **中断**（源于外设）
  - 来自不同的硬件设备的计时器和网路中断

Operating System Interaction

- 响应
  - 中断：持续，对用户应用程序透明
  - 异常：杀死或重新执行（不良）程序指令
  - 系统调用：等待和持续

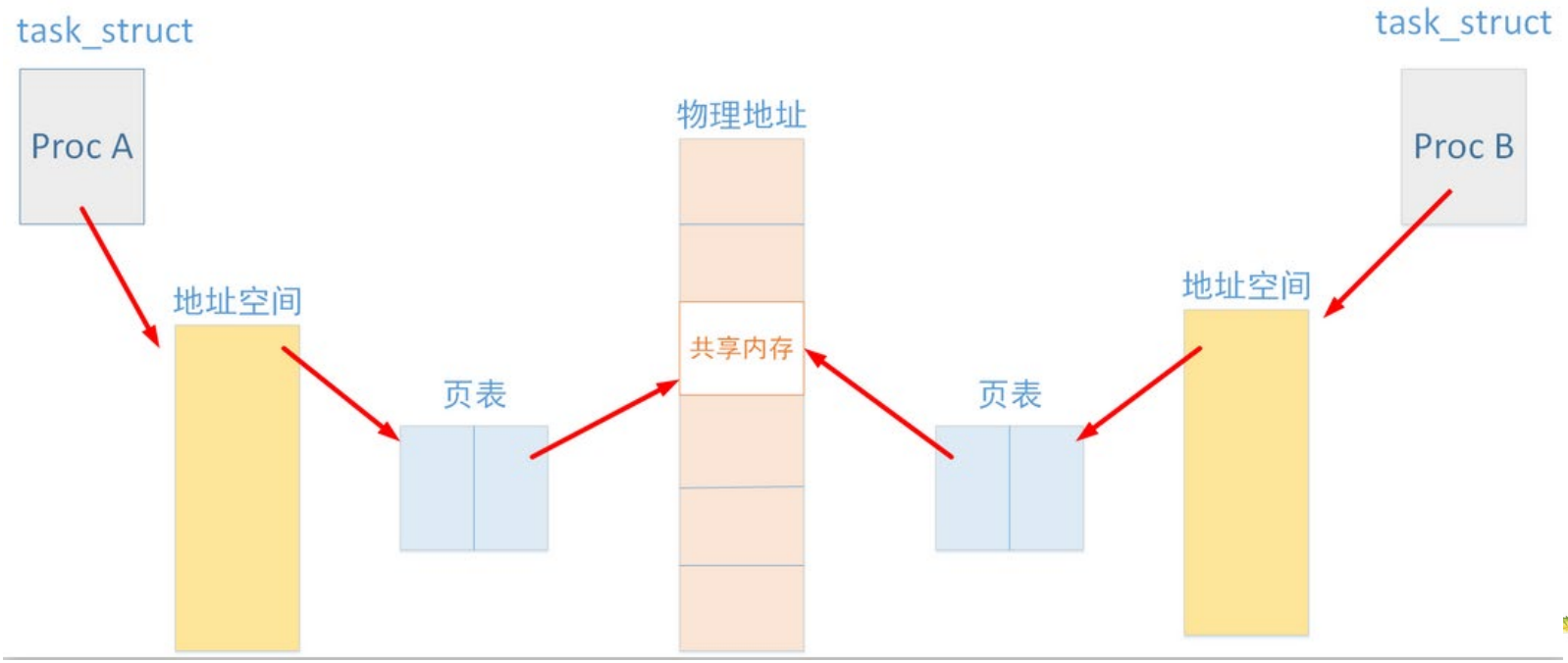
Communication Models





Communication Models

- 共享内存在不同的操作系统的实现机制不尽相同
- 在Linux中，每个进程都有属于自己的进程控制块（**PCB**）和**地址空间**，并且都有一个与之对应的页表，负责将进程的虚拟地址与物理地址进行映射，通过内存管理单元（MMU）进行管理。两个不同的虚拟地址通过页表映射到物理空间的同一区域，它们所指向的这块区域即共享内存。



Multiple Choice Questions

- 1. What is an operating system?
  - a) interface between the hardware and application programs
  - b) collection of programs that manages hardware resources
  - c) system service provider to the application programs
  - d) all of the mentioned

Answer: d
Explanation: An Operating System acts as an intermediary between user/user applications/application programs and hardware. It is a program that manages hardware resources. It provides services to application programs.

Multiple Choice Questions

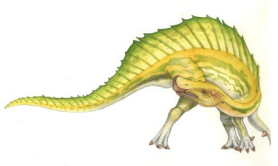
- 2. What is the main function of the command interpreter?
  - a) to provide the interface between the API and application program
  - b) to handle the files in the operating system
  - c) to get and execute the next user-specified command
  - d) none of the mentioned

Answer: c
Explanation: The main function of a command interpreter is to get and execute the next user-specified command. Command Interpreter checks for valid command and then runs that command else it will throw an error.

Multiple Choice Questions

- 3. To access the services of the operating system, the interface is provided by the _____
  - a) Library
  - b) System calls
  - c) Assembly instructions
  - d) API

Answer: b
Explanation: To access services of the Operating System an interface is provided by the System Calls. Generally, these are functions written in C and C++. Open, Close, Read, Write are some of most prominently used system calls.



Multiple Choice Questions

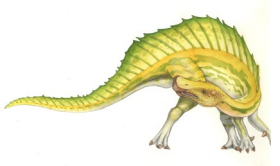
- 4. Which one of the following is not true?
  - a) kernel remains in the memory during the entire computer session
  - b) kernel is made of various modules which can not be loaded in running operating system
  - c) kernel is the first part of the operating system to load into memory during booting
  - d) kernel is the program that constitutes the central core of the operating system

Answer: b
Explanation: Kernel is the first program that is loaded in memory when OS is loading as well as it remains in memory till OS is running. Kernel is the core part of the OS which is responsible for managing resources, allowing multiple processes to use the resources and provide services to various processes. Kernel modules can be loaded and unloaded in run-time i.e. in running OS.

Multiple Choice Questions

- 5. Which one of the following errors will be handle by the operating system?
  - a) lack of paper in printer
  - b) connection failure in the network
  - c) power failure
  - d) all of the mentioned

Answer: d
Explanation: All the mentioned errors are handled by OS. The OS is continuously monitoring all of its resources. Also, the OS is constantly detecting and correcting errors.



Multiple Choice Questions

- 6. Where is the operating system placed in the memory?
  - a) either low or high memory (depending on the location of interrupt vector)
  - b) in the low memory
  - c) in the high memory
  - d) none of the mentioned

Answer: a

Multiple Choice Questions

- 7. In operating system, each process has its own _____
  - a) open files
  - b) pending alarms, signals, and signal handlers
  - c) address space and global variables
  - d) all of the mentioned

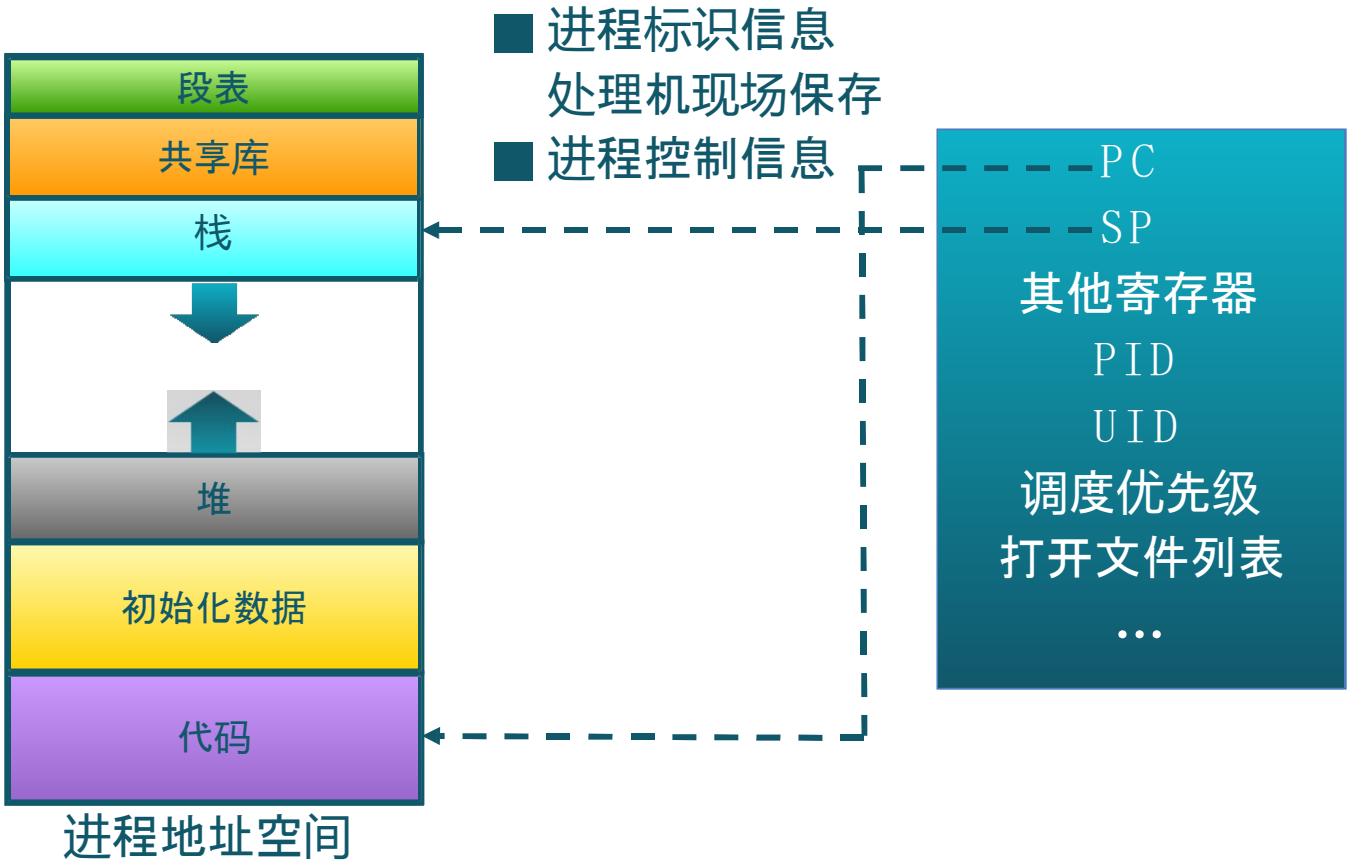
Answer: d
Explanation: In Operating Systems, each process has its own address space which contains code, data, stack, and heap segments or sections. Each process also has a list of files that is opened by the process as well as all pending alarms, signals, and various signal handlers.

Multiple Choice Questions

- 8. When a process is in a “Blocked” state waiting for some I/O service. When the service is completed, it goes to the _____
  - a) Terminated state
  - b) Suspended state
  - c) Running state
  - d) Ready state

Answer: d
Explanation: Suppose that a process is in “Blocked” state waiting for some I/O service. When the service is completed, it goes to the ready state. Process never goes directly to the running state from the waiting state. Only processes which are in ready state go to the running state whenever CPU allocated by operating system.

Process Control Block (PCB)



Process Control Block (PCB)

- 进程存在的唯一标识
  - 进程的创建：生成PCB
  - 进程的终止：回收PCB
  - 进程的组织管理：对PCB的组织管理来实现

PCB包含3类信息

- （一）进程标识信息
  - 本进程的标识，本进程的产生者标识，用户标识
- （二）处理机状态信息保存区
  - 用户可见寄存器：用户使用的数据、地址寄存器
  - 控制和状态寄存器：PC、程序状态字（PSW）
  - 栈指针：过程调用/系统调用/中断处理和返回时需要用到

PCB包含3类信息

- （三）进程控制信息
  - 调度和状态信息
  - 进程间通信信息
  - 存储管理信息
  - 进程所用资源
  - 有关数据结构连接信息

PCB的组织方式

- **链表**

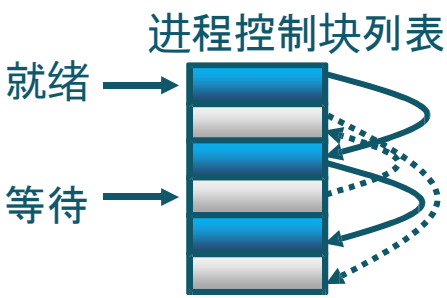
  同一状态的进程其PCB成一链表，多个状态对应多个不同的链表

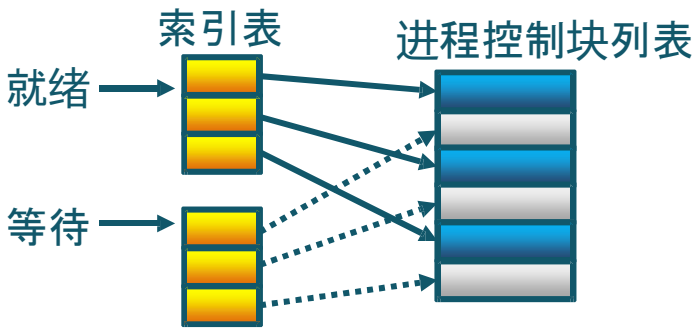
  - 各状态的进程形成不同的链表：就绪链表、阻塞链表

- **索引表**

  同一状态的进程归入一个索引表（由索引指向PCB），多个状态对应多个不同的索引表

  - 各状态的进行形成不同的索引表：就绪索引表、阻塞索引表

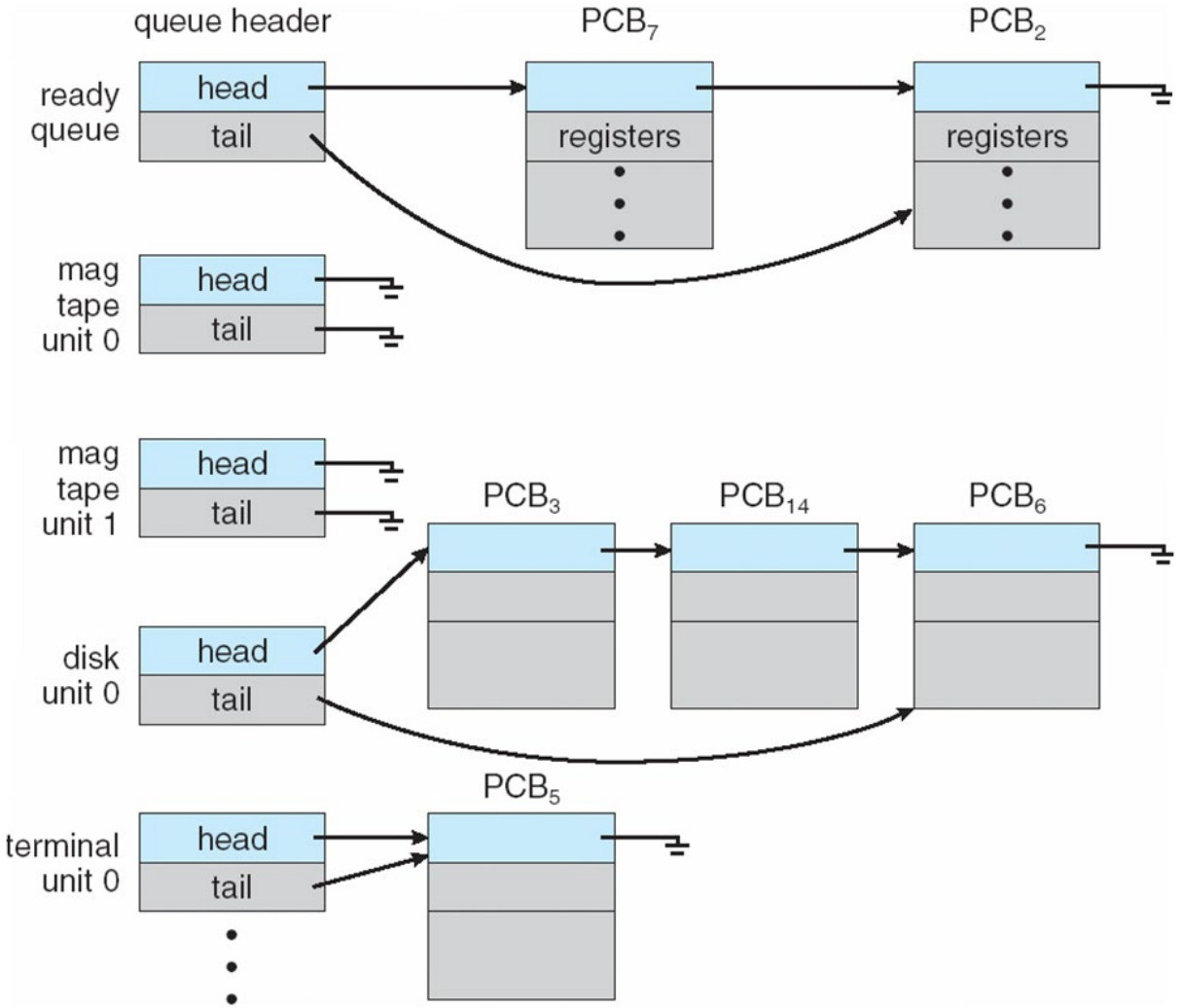




Process Scheduling Queues

- **Job queue** – set of all processes in the system
- **Ready queue** – set of all processes residing in main memory, ready and waiting to execute
- **Device queues** – set of processes waiting for an I/O device
- Processes migrate among the various queues

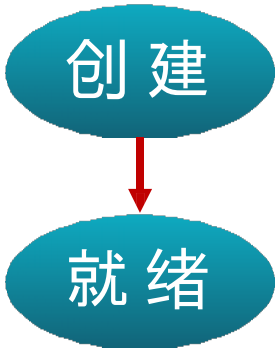
Ready Queue And Various I/O Device Queues



进程创建

引起进程创建的情况

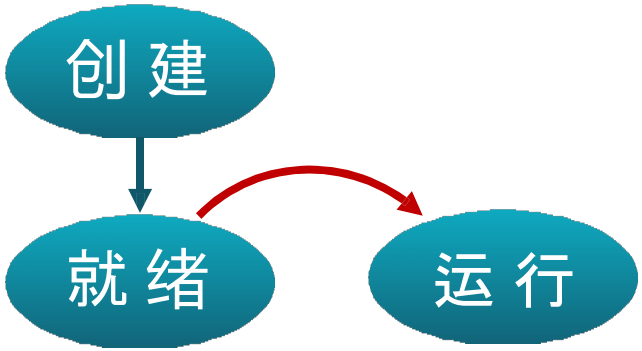
- 系统初始化时
- 用户请求创建一个新进程正在运行的进程执行了创 建进程的系统调用



进程执行

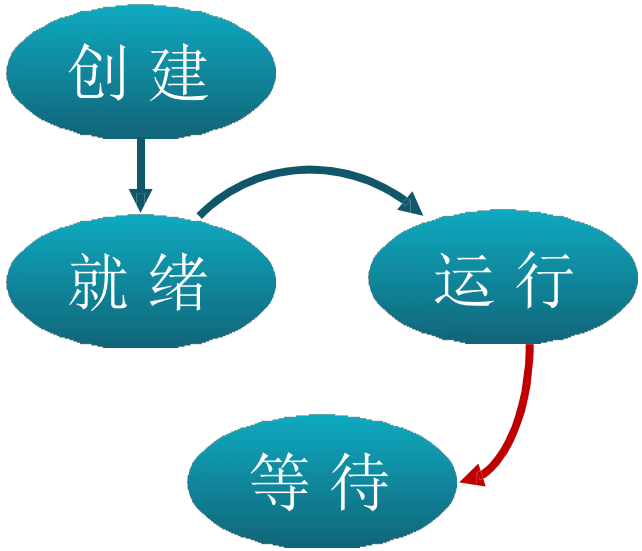
内核选择一个就绪的进程，让它占用处理机并执行

- 如何选择？



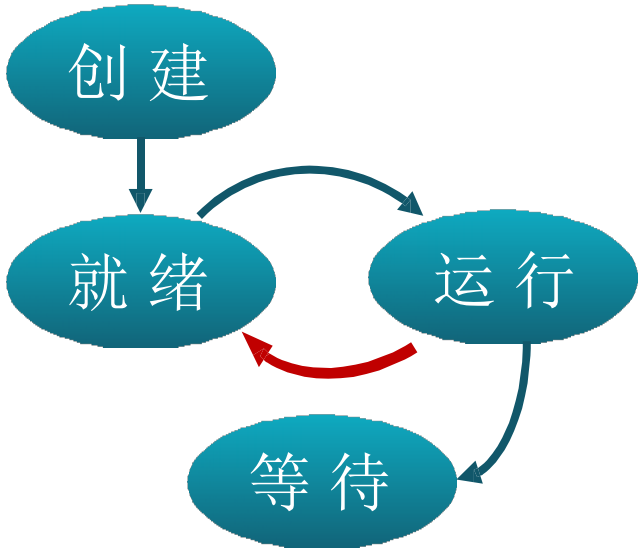
进程等待

- 进程进入等待(阻塞)的情况：
  - 请求并等待系统服务，无法马上完成
  - 启动某种操作，无法马上完成
  - 需要的数据没有到达
- 只有进程自身才能知道何时需要等待某种事件的发生



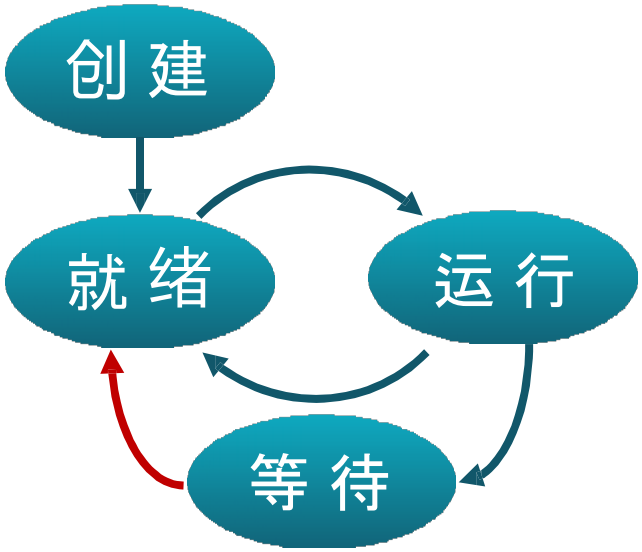
进程抢占

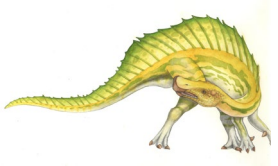
- 进程会被抢占的情况
  - 高优先级进程就绪
  - 进程执行当前时间用完



进程唤醒

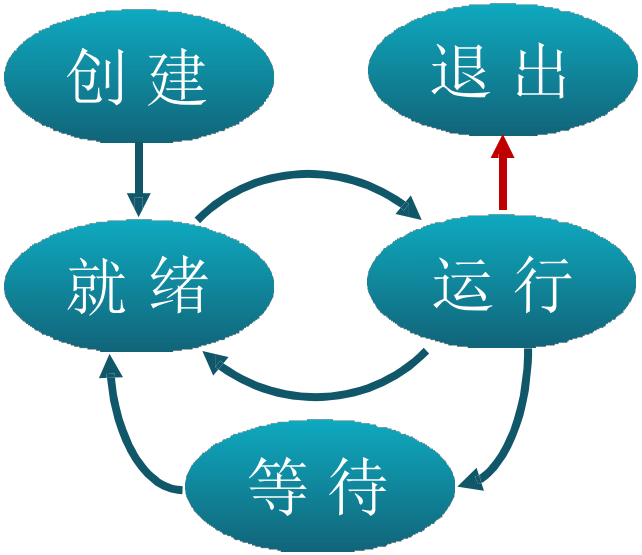
- 唤醒进程的情况：
  - 被阻塞进程需要的资源可被满足
  - 被阻塞进程等待的事件到达
- 进程只能被别的进程或操作系统唤醒



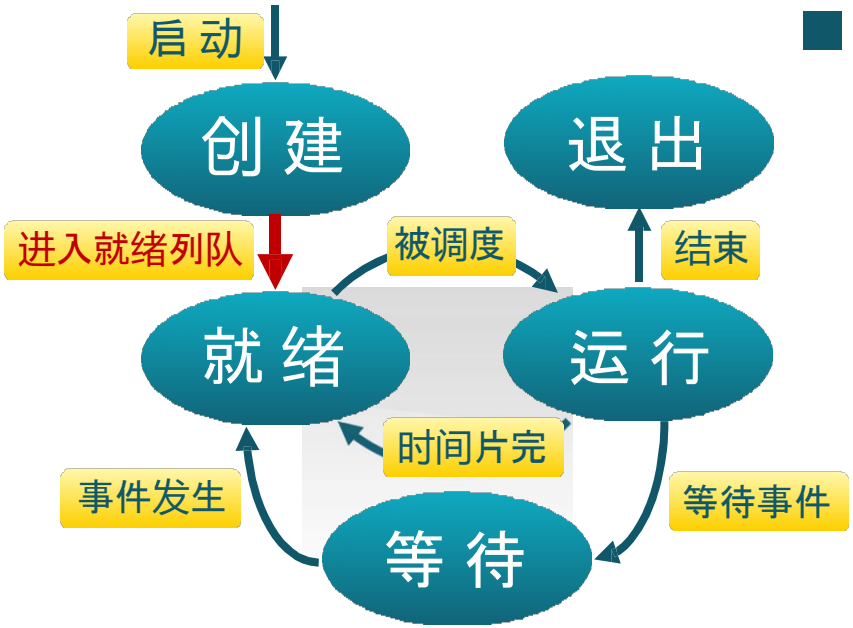


进程结束

- 进程结束的情况：
  - 正常退出（自愿的）
  - 错误退出（自愿的）
  - 致命错误（强制性的）
  - 被其他进程所杀（强制性的）

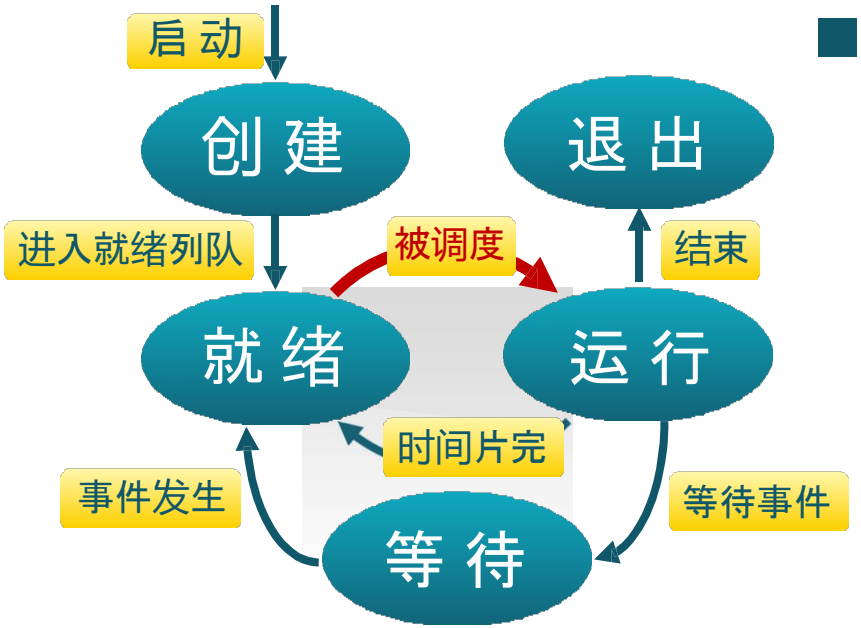


三状态进程模型



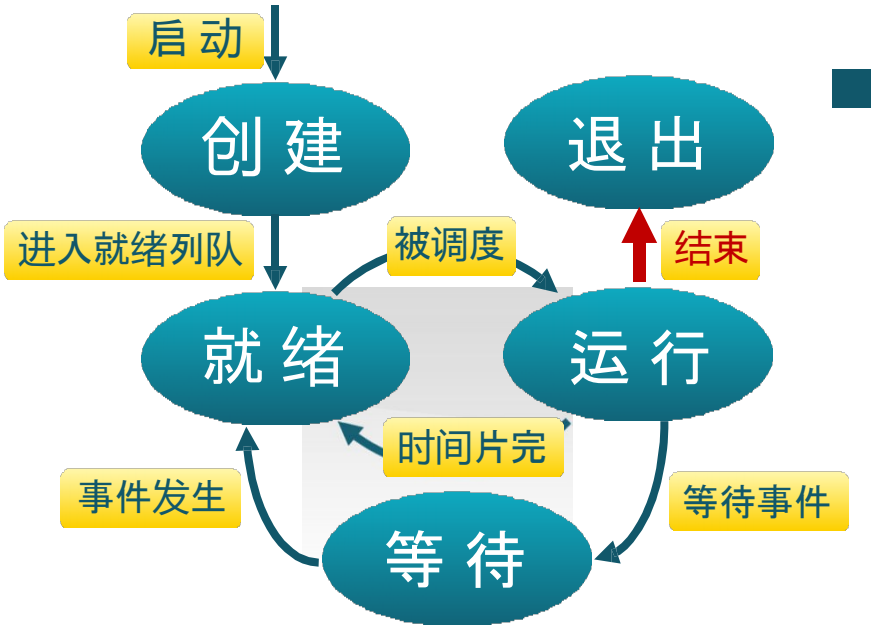
- 创建→就绪
  当进程被创建完成并初始化后，一切就绪准备运行时，变为就绪状态

三状态进程模型



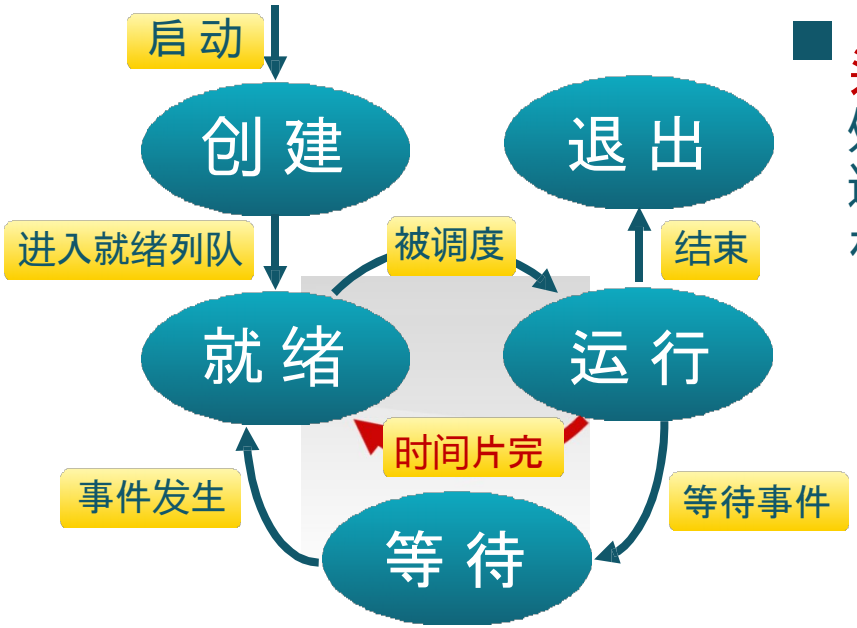
- 就绪→运行
  处于就绪状态的进程被进程调度程序选中后，就分配到处理机上来运行

三状态进程模型



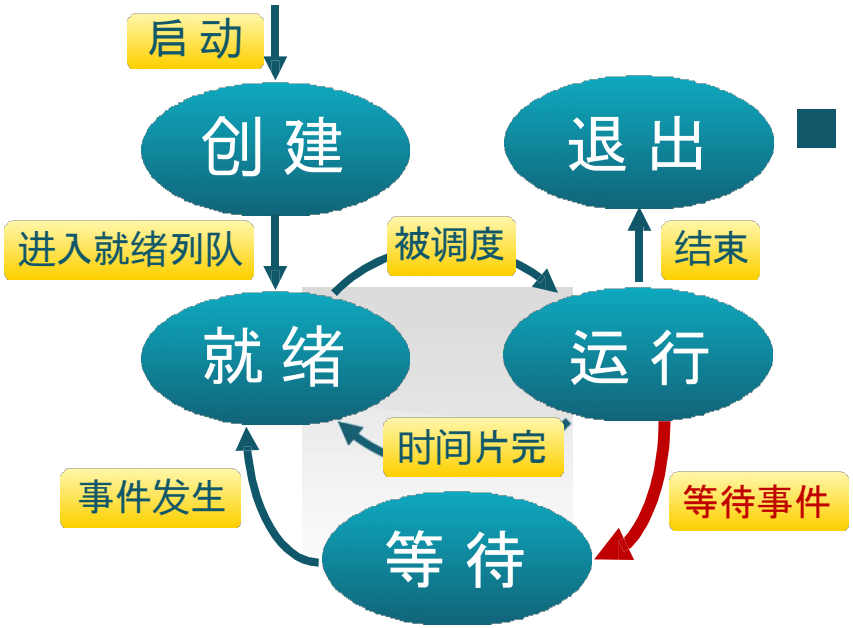
- 运行→结束
  当进程表示它已经完成或者因出错，当前运行进程会由操作系统作结束处理

三状态进程模型



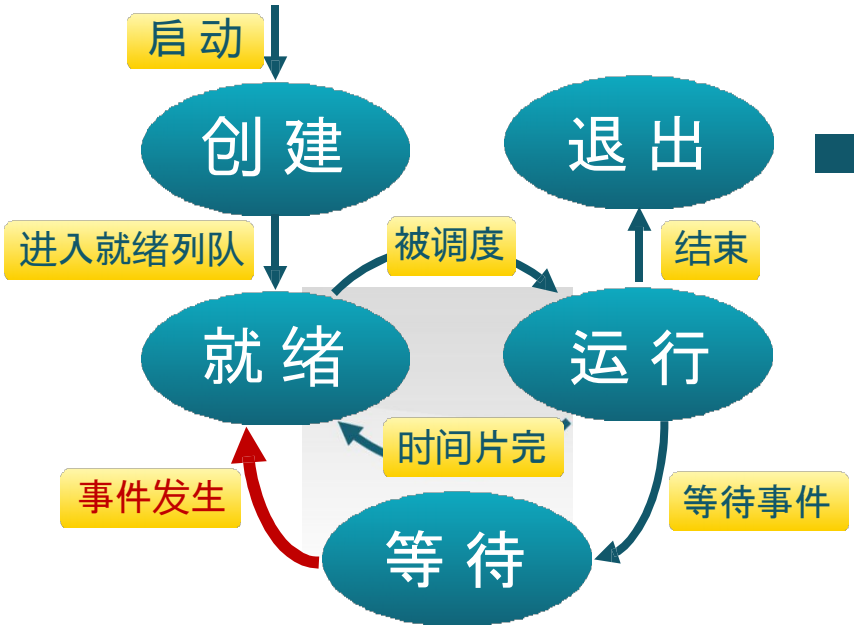
- 运行→就绪
  处于运行状态的进程在其运行过程中，由于分配给它的处理机时间片用完而让出处理机

三状态进程模型



- 运行→等待
  当进程请求某资源且必须等待时

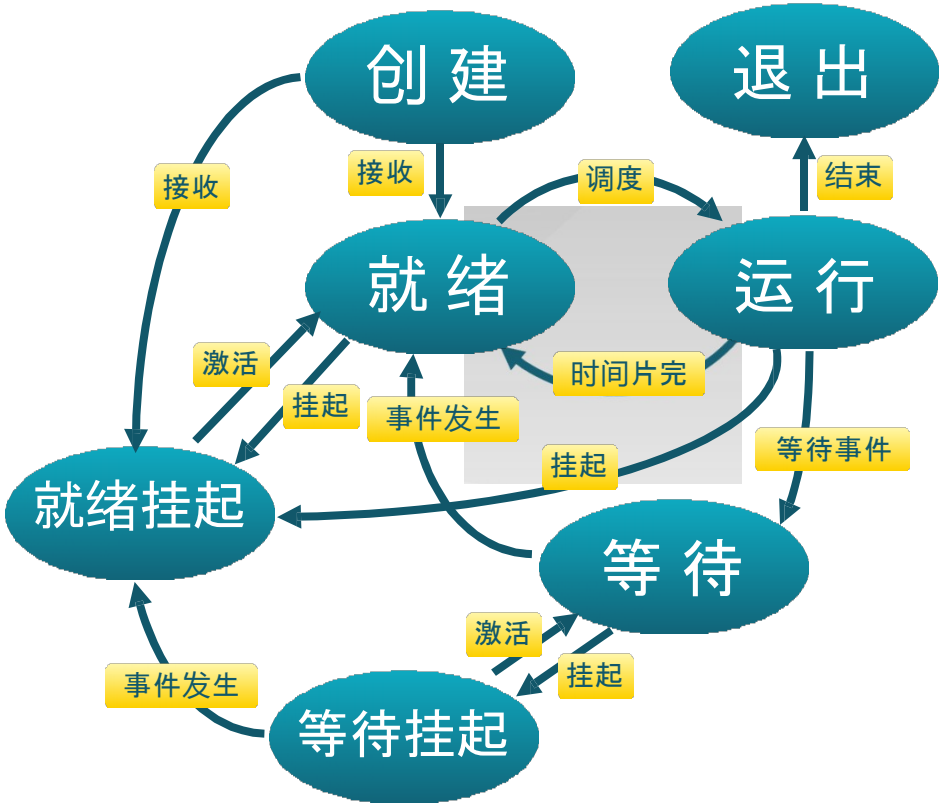
三状态进程模型



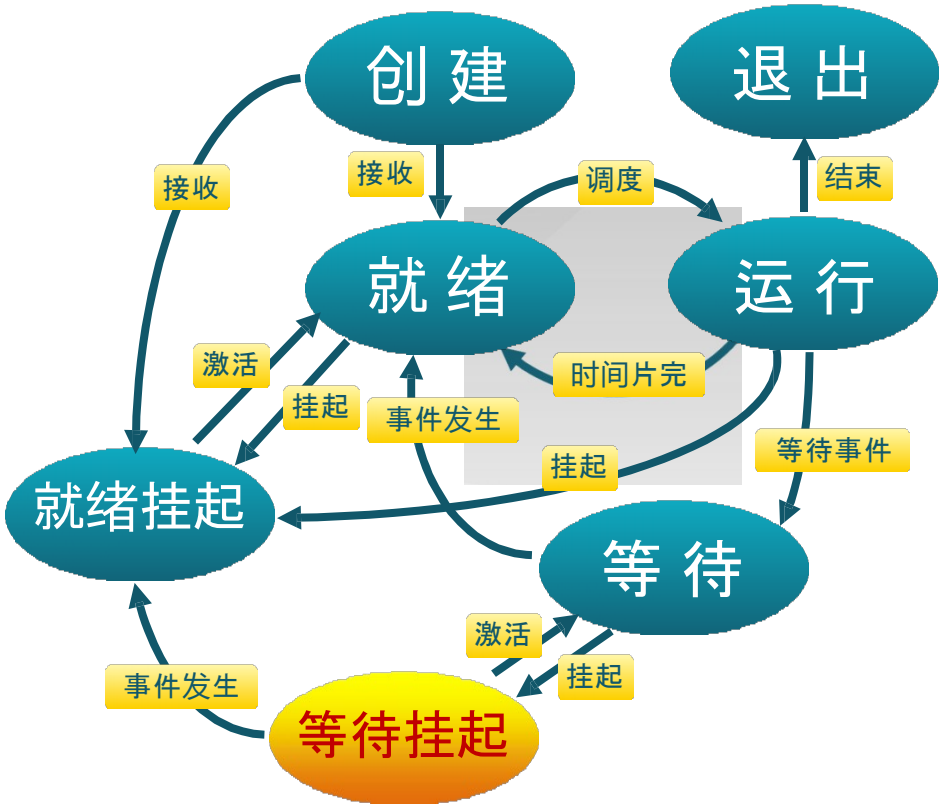
- 等待→就绪
  当进程要等待某事件到来时，它从阻塞状态变到就绪状态

进程挂起

处在挂起状态的进程映像在磁盘上，目的是减少进程占用内存

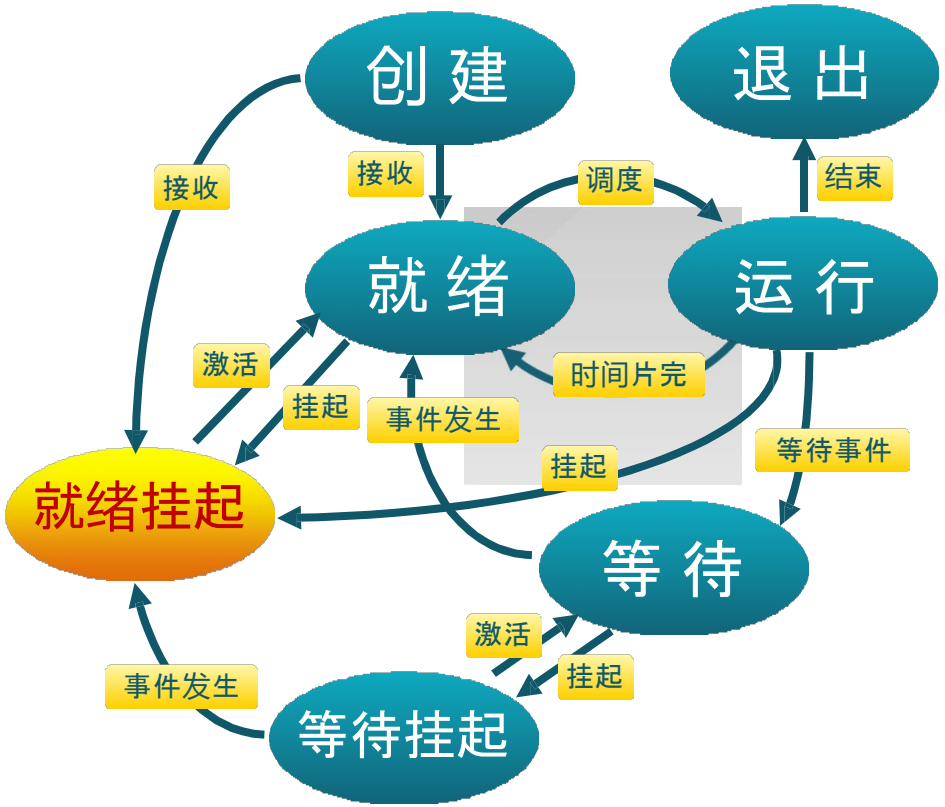


进程挂起



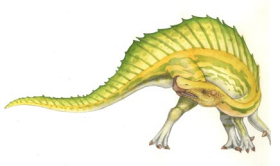
- 等待挂起状态 (Blocked-suspend)
  进程在外存并等待某事件的出现

进程挂起

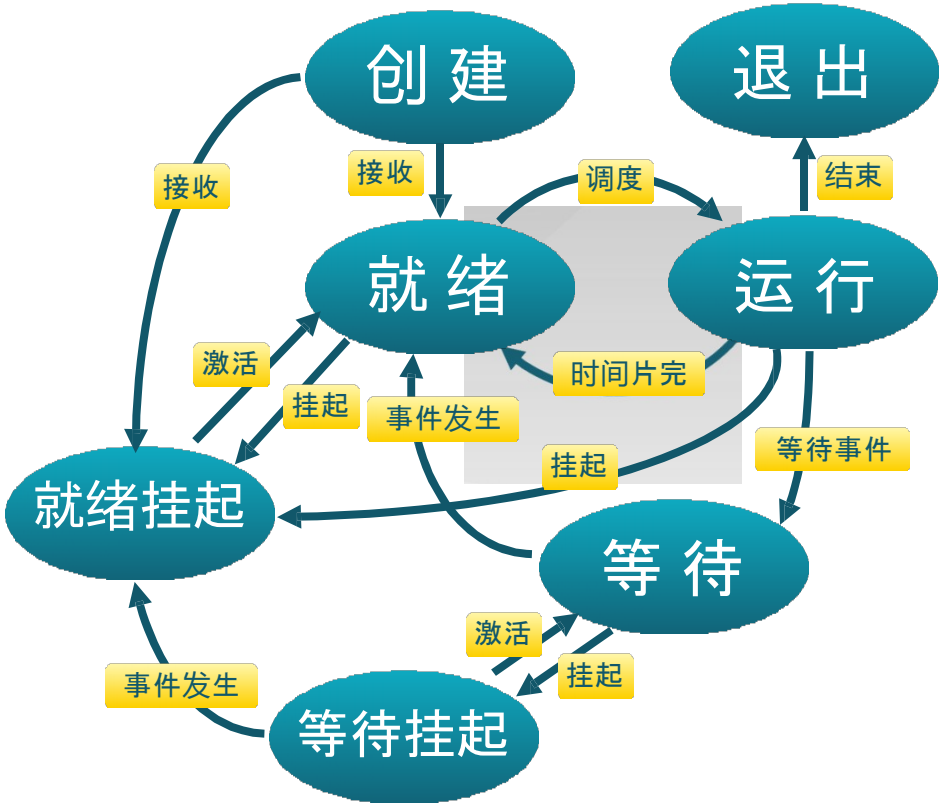


- 等待挂起状态 (Blocked-suspend)
- 就绪挂起状态 (Ready-suspend)

进程在外存，但只要进入内存，即可运行

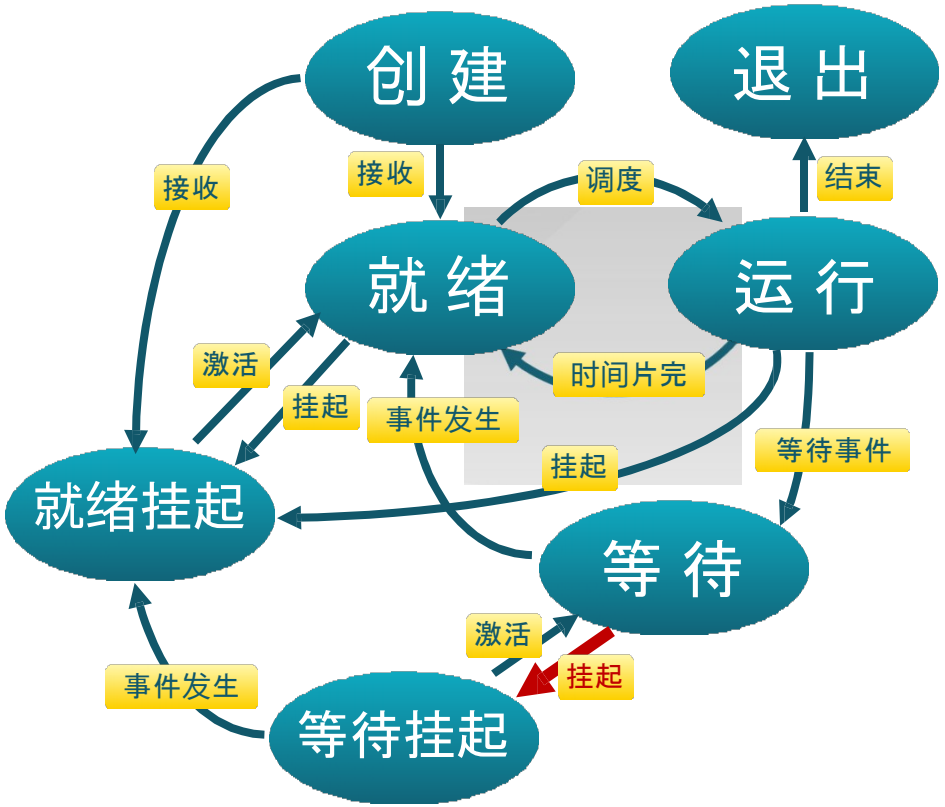


与挂起相关的状态转换



- 挂起(Suspend)：把一个进程从内存转到外存

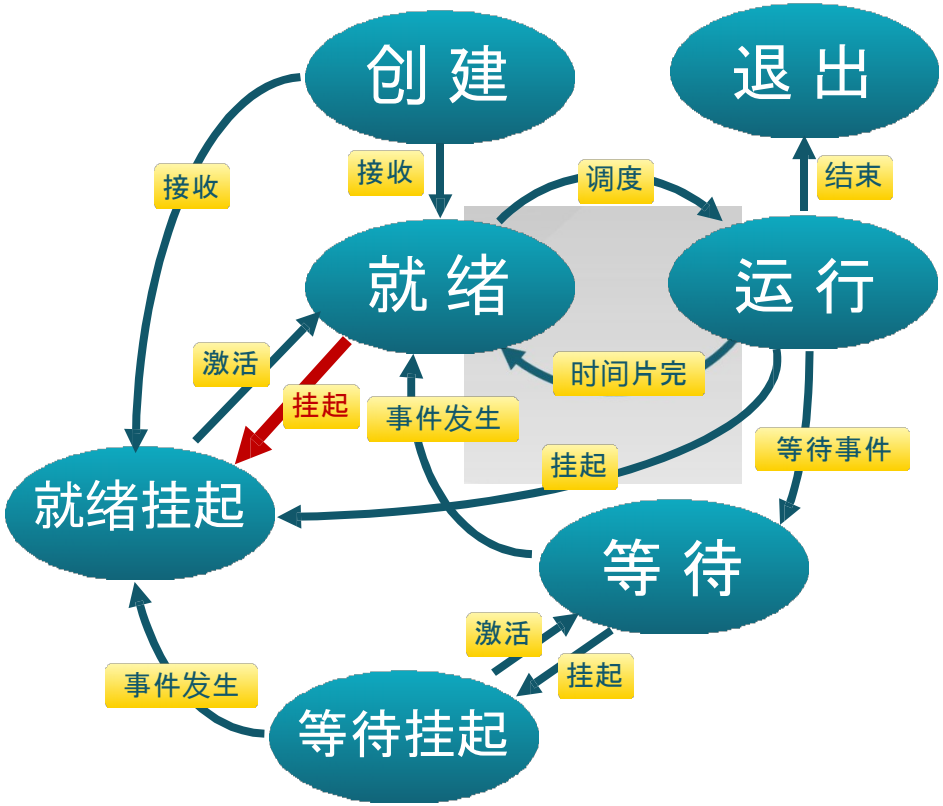
与挂起相关的状态转换



- 挂起(Suspend)：把一个进程从内存转到外存
  - 等待到等待挂起

    没有进程处于就绪状态或就绪进程要求更多内存资源

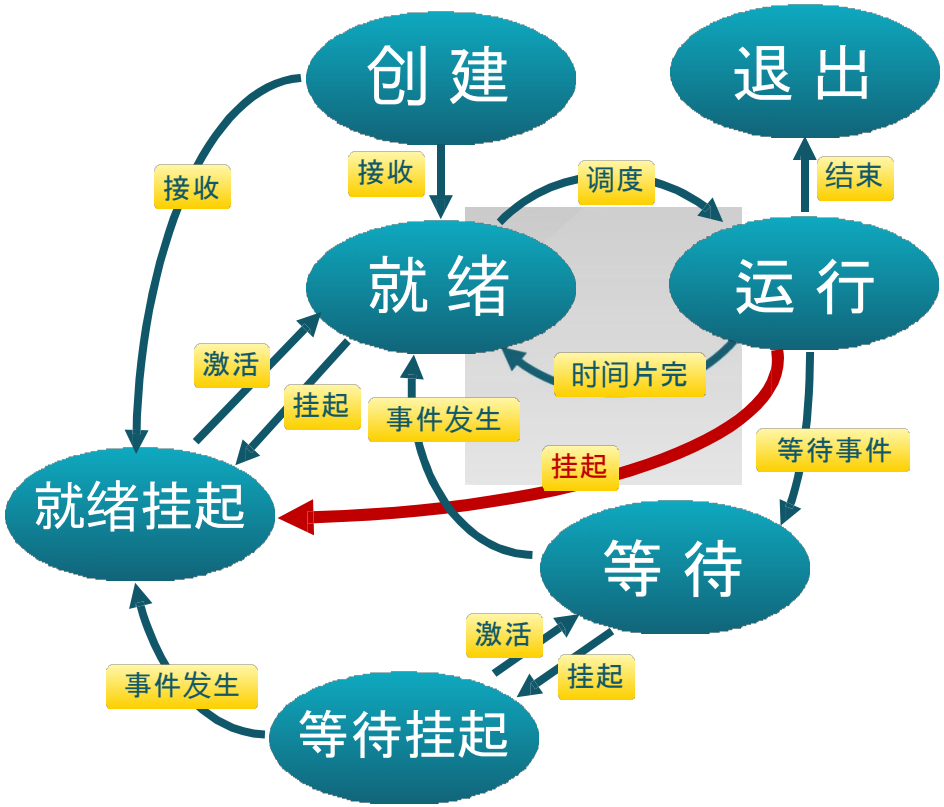
与挂起相关的状态转换



- 挂起(Suspend)：把一个进程从内存转到外存
  - 等待到等待挂起
  - 就绪到就绪挂起

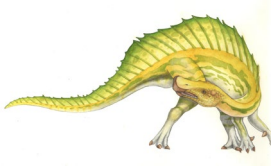
    当有高优先级等待(系统认为会很快就绪的)进程和低优先级就绪进程

与挂起相关的状态转换

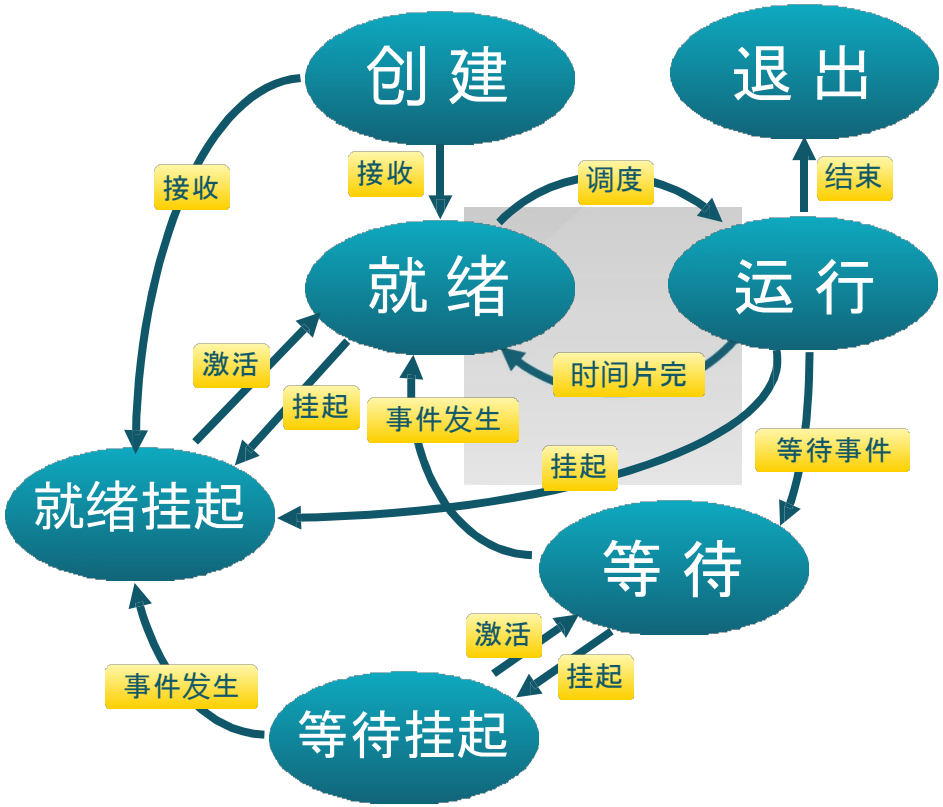


- 挂起(Suspend)：把一个进程从内存转到外存
  - 等待到等待挂起
  - 就绪到就绪挂起
  - 运行到就绪挂起

  对抢先式分时系统，当有高优先级等待挂起进程因事件出现而进入就绪挂起

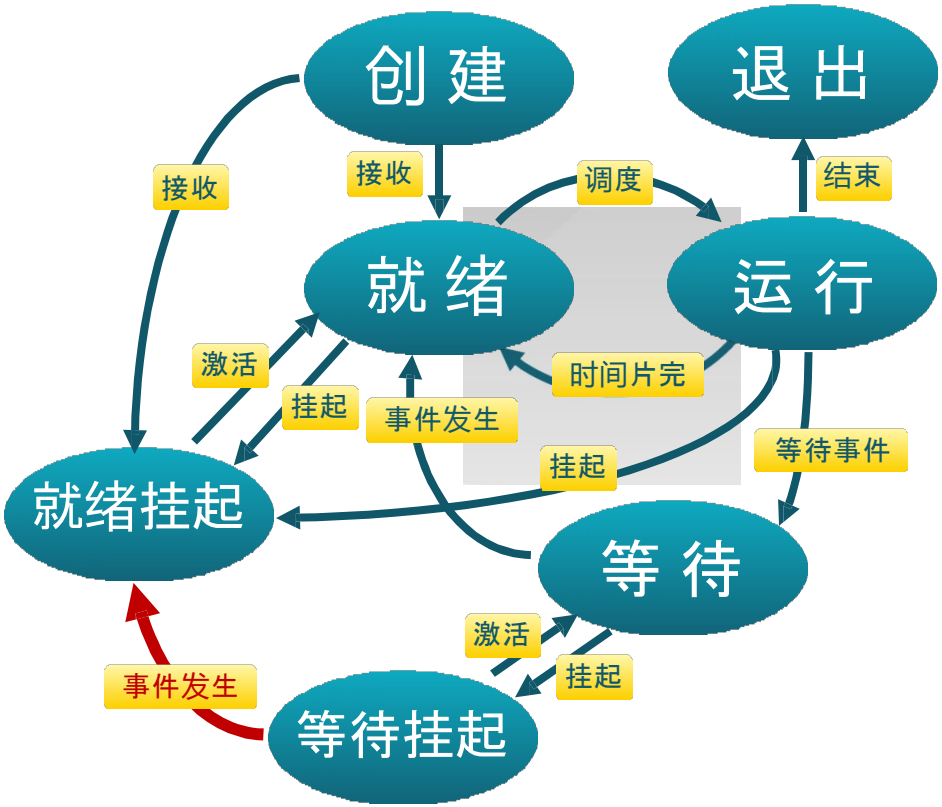


与挂起相关的状态转换



- 在外存时的状态转换

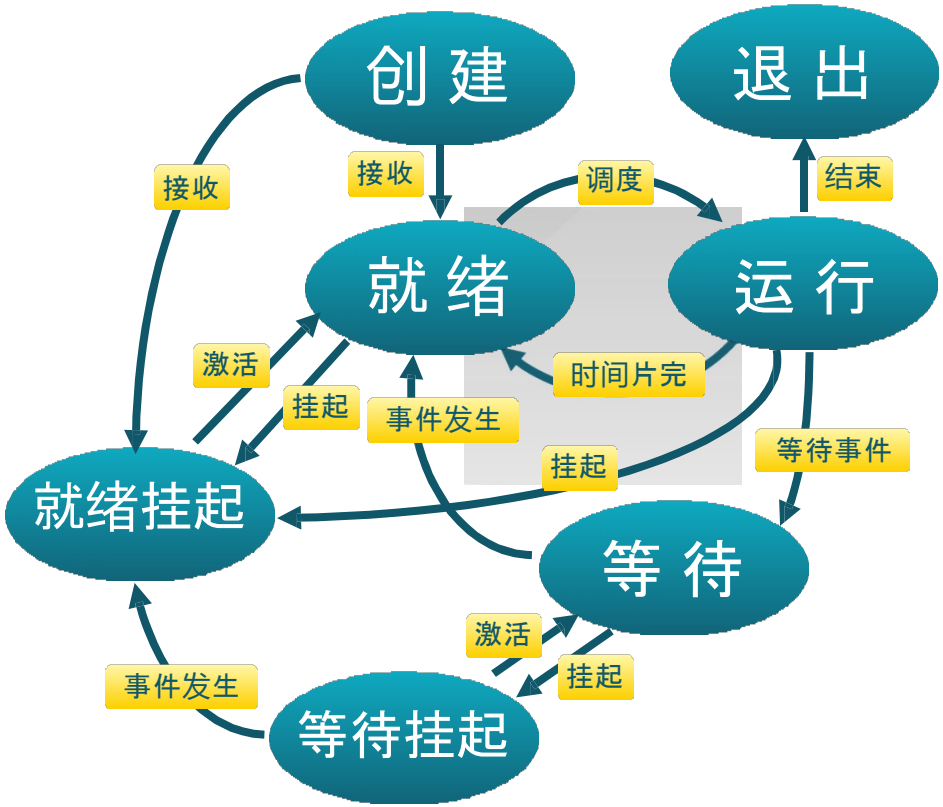
与挂起相关的状态转换



- 在外存时的状态转换
  - 等待挂起到就绪挂起

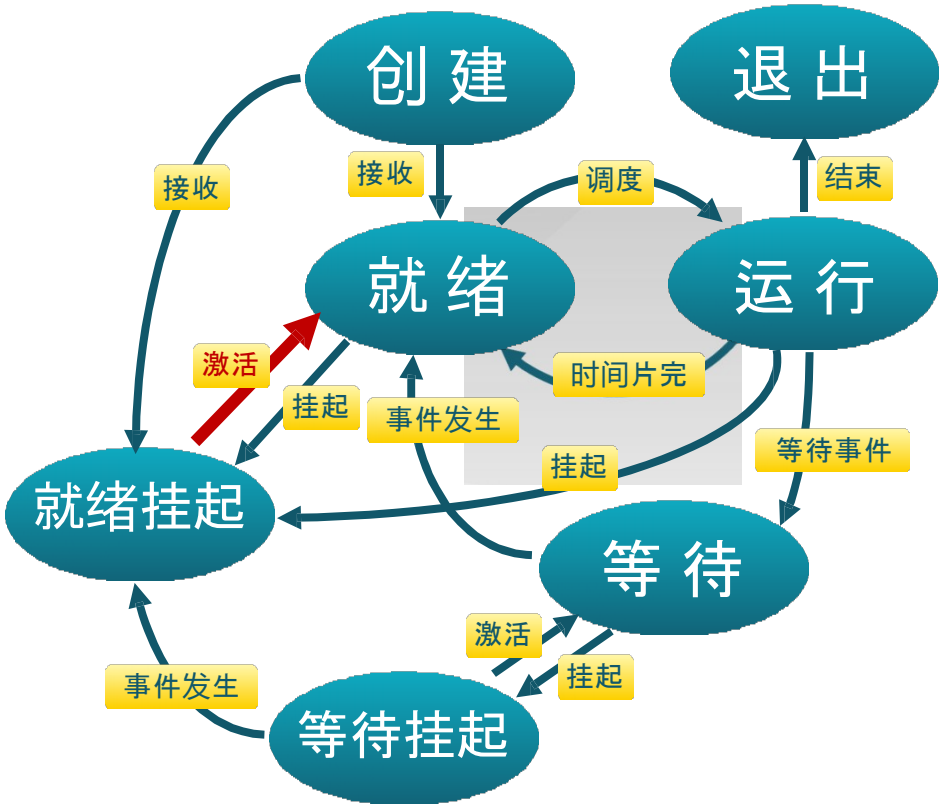
    当有等待挂起进程因相关事件出现

与挂起相关的状态转换



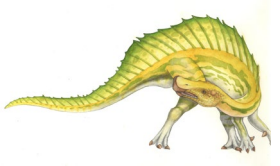
- 激活(Activate):把一个进程从外存转到内存

与挂起相关的状态转换

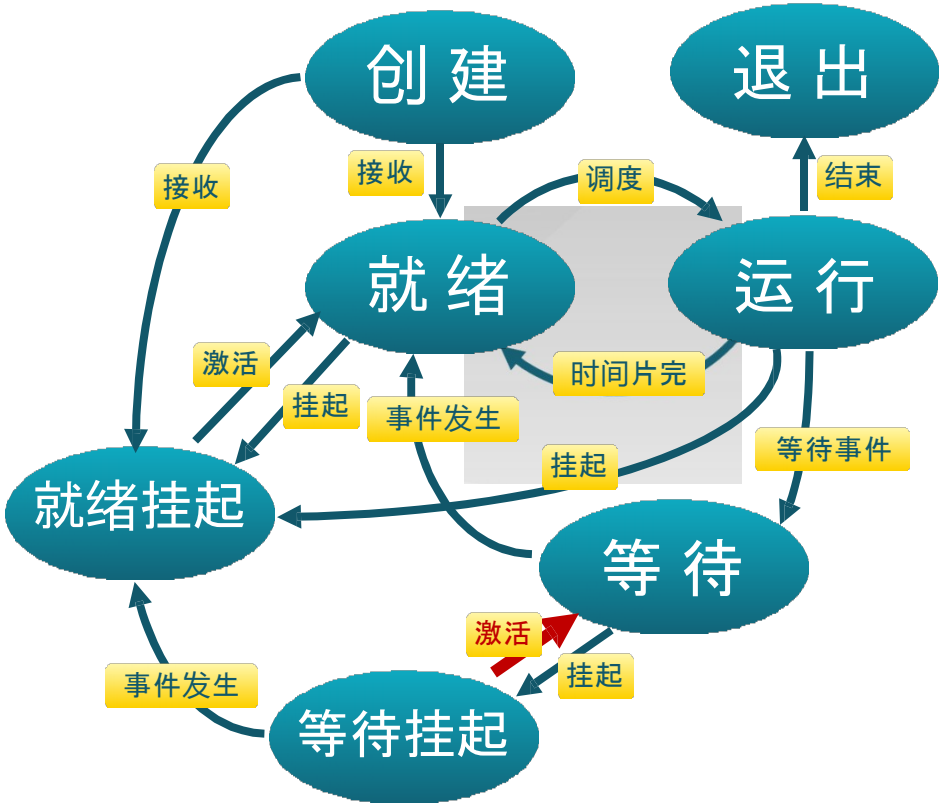


- 激活(Activate):把一个进 程从外存转到内存
  - 就绪挂起到就绪

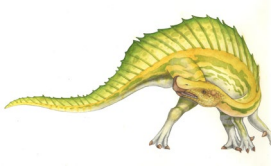
    没有就绪进程或挂起就绪进程优先级高于就绪进程



与挂起相关的状态转换



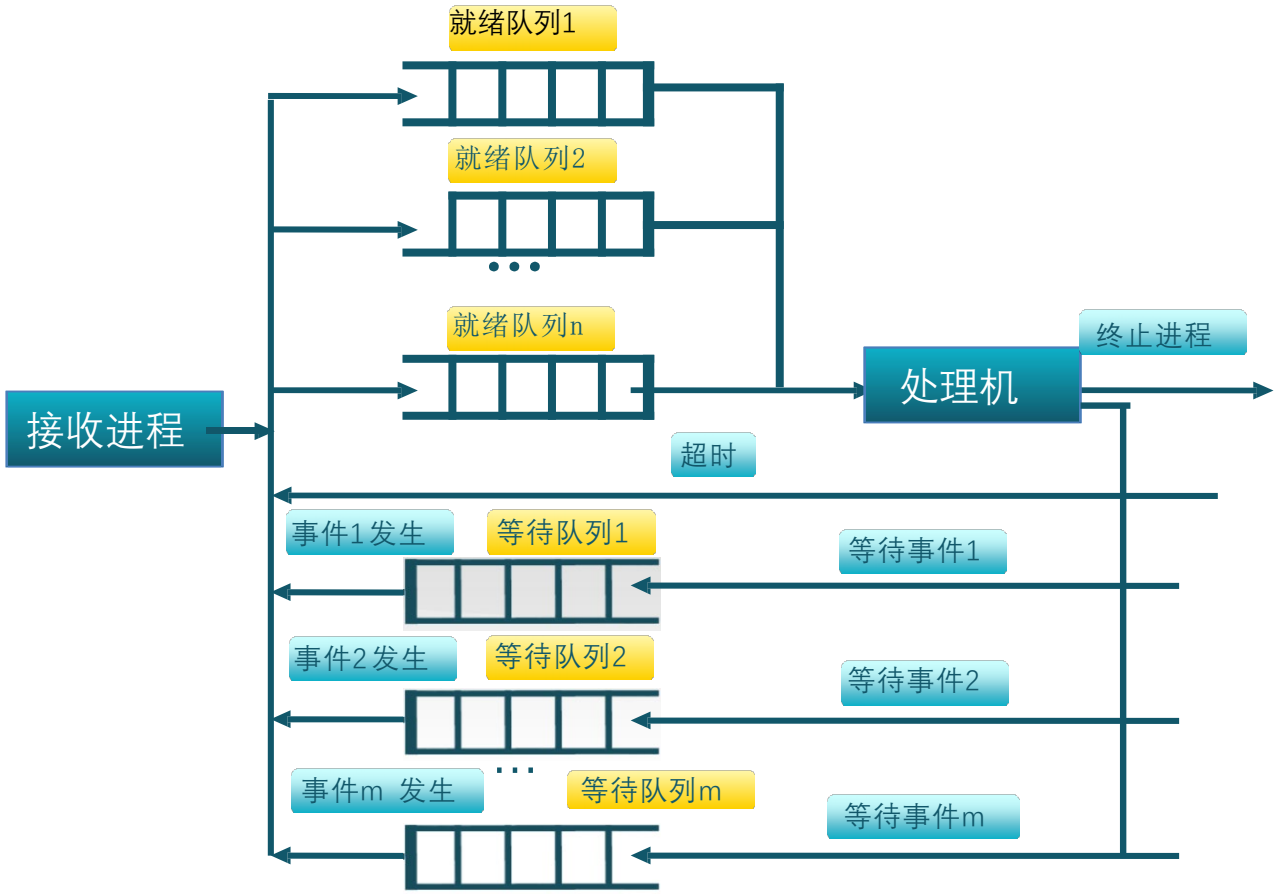
- 激活(Activate):把一个进程从外存转到内存
  - 就绪挂起到就绪
  - 等待挂起到等待
    当一个进程释放足够内存，并有高优先级等待挂起进程



状态队列

- 操作系统来维护一组队列，表示系统中所有进程的当前状态
- 不同队列表示不同状态
  - 就绪队列、各种等待队列
- 根据进程状态不同，进程PCB加入相应队列
  - 进程状态变化时，它所在的PCB会从一个队列换到另一个

状态队列


就绪队列1
就绪队列2
就绪队列n
接收进程
处理机
终止进程
超时
事件1发生
等待队列1
等待事件1
事件2发生
等待队列2
等待事件2
事件m 发生
等待队列m
等待事件m

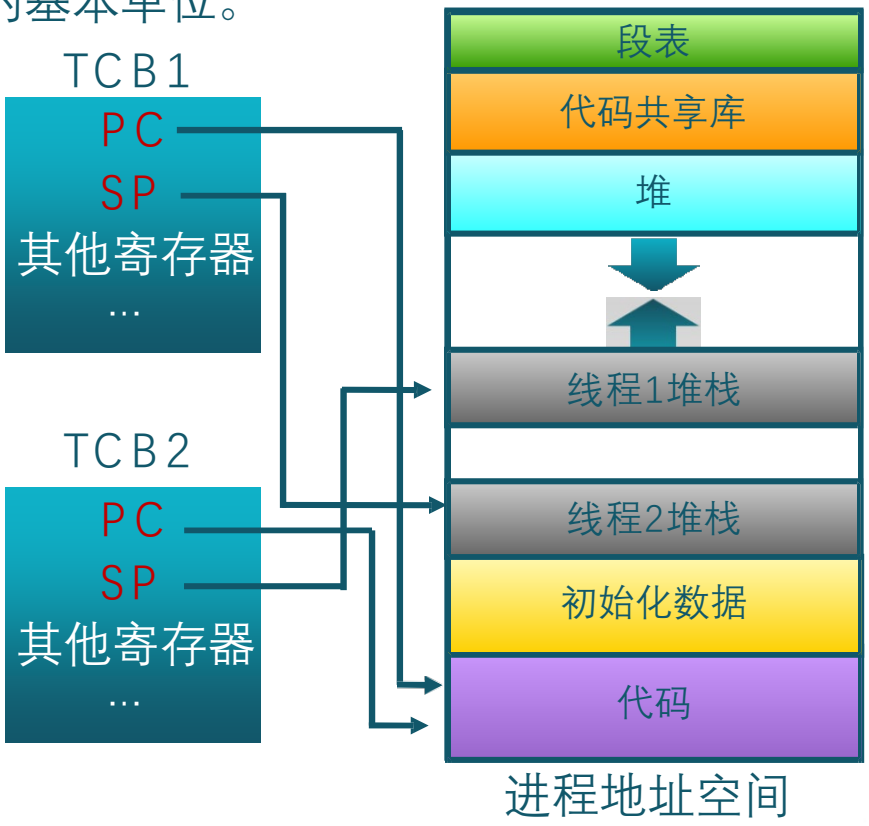
Processes Overheads

- A full process includes numerous things:
  - an address space (defining all the code and data pages)
  - OS resources and accounting information
  - a “thread of control”,
    - defines where the process is currently executing
    - That is the PC and registers
- Creating a new process is costly
  - all of the structures (e.g., page tables) that must be allocated
- Communicating between processes is costly
  - most communication goes through the OS

线程的概念

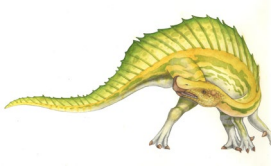
线程是进程的一部分，描述指令流执行状态。它是进程中的指令执行流的最小单元，是CPU调度的基本单位。

- 进程的资源分配角色：进程由一组相关资源构成，包括地址空间（代码段、数据段）、打开的文件等各种资源
- 线程的处理机调度角色：线程描述在进程资源环境中的指令流执行状态



线程= 进程-共享资源

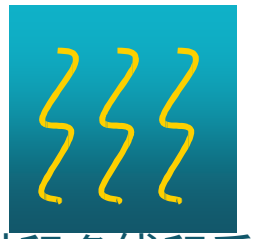
- 线程的优点：
  - 一个进程中可以同时存在多个线程
  - 各个线程之间可以并发地执行
  - 各个线程之间可以共享地址空间和文件等资源
- 线程的缺点：
  - 一个线程崩溃，会导致其所属进程的所有线程崩溃



不同操作系统对线程的支持

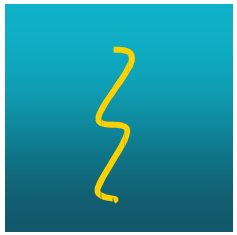
单进程系统

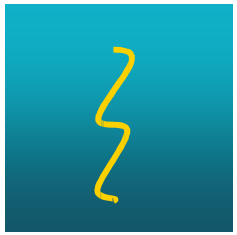
实例：MS-DOS



单进程多线程系统

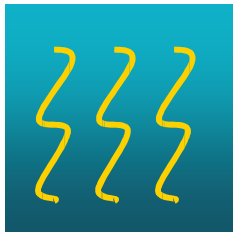
实例：pSOS

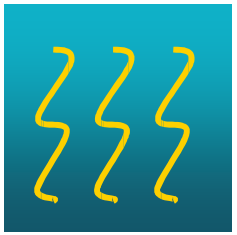




多进程系统

实例：传统UNIX





多线程系统

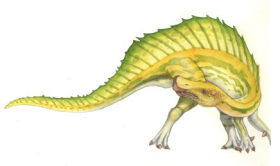
实例：现代UNIX

线程与进程的比较

- 进程是资源分配单位，线程是CPU调度单位
  进程拥有一个完整的资源平台，而线程只独享指
- 令流执行的必要资源，如寄存器和栈
  线程具有就绪、等待和运行三种基本状态和
- 状态间的转换关系
  线程能减少并发执行的时间和空间开销
  - 线程的创建时间比进程短
  - 线程的终止时间比进程短
  - 同一进程内的线程切换时间比进程短
  - 由于同一进程的各线程间共享内存和文件资源，
    可不通过内核进行直接通信

并发进程的正确性

- 独立进程
  - 不和其他进程共享资源或状态
  - 确定性： 输入状态决定结果
  - 可重现： 能够重现起始条件
  - 调度顺序不重要
- 并发进程
  - 在多个进程间有资源共享
  - 不确定性
  - 不可重现
- 并发进程的正确性
  - 执行过程是不确定性和不可重现的
  - 程序错误可能是间歇性发生的



Questions & Answers – Processes

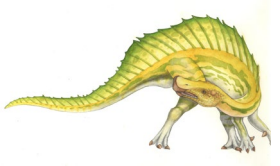
- 1. A process can be terminated due to _____
  a) normal exit
  b) fatal error
  c) killed by another process
  d) all of the mentioned

Answer: d
Explanation: A process can be terminated normally by completing its task or because of fatal error or killed by another process or forcefully killed by a user. When the process completes its task without any error then it exits normally. The process may exit abnormally because of the occurrence of fatal error while it is running. The process can be killed or terminated forcefully by another process.

Questions & Answers – Processes

- 2. What is the ready state of a process?
  a) when process is scheduled to run after some execution
  b) when process is unable to run until some task has been completed
  c) when process is using the CPU
  d) none of the mentioned

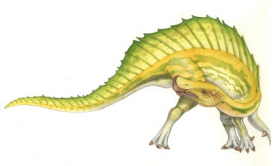
Answer: a
Explanation: Ready state of the process means process has all necessary resources which are required for execution of that process when CPU is allocated. Process is ready for execution but waiting for the CPU to be allocated.



Questions & Answers – Processes

- 3. A process stack does not contain _____
  a) Function parameters
  b) Local variables
  c) Return addresses
  d) PID of child process

Answer: d
Explanation: Process stack contains Function parameters, Local variables and Return address. It does not contain the PID of child process.



Questions & Answers – Processes

- 堆(heap)

用于动态分配内存，位于BSS和栈中间的地址区域，由程序员申请分配和释放。堆是从低地址位向高地址位增长，采用链式存储结构。频繁的malloc/free造成内存空间的不连续，会产生碎片。**如何解决内存碎片？**

当申请堆空间时，库函数按照搜索可用的足够大的空间，因此堆的效率比栈要低的多。注：与数据结构中的堆不是一个概念，分配方式类似于链表。

- 栈(stack)

由编译器自动释放，存放函数的参数值、局部变量等。每当一个函数被调用时，该函数的返回类型和一些调用的信息被存放到栈中，这个被调用的函数再为它的自动变量和临时变量在栈上分配空间。每调用一个函数一个新的栈就会被使用。栈区是从高地址位向低地址位增长的，是一块连续的内存区域，最大容量是由系统预先定义好的，申请的栈空间超过这个界限时会提示溢出。

Questions & Answers – Processes

- 4. Which system call can be used by a parent process to determine the termination of child process?
  a) wait
  b) exit
  c) fork
  d) get

Answer: a
Explanation: wait( ) system call is used by the parent process to determine termination of child process. The parent process uses wait( ) system call and gets the exit status of the child process as well as the pid of the child process which is terminated.

Questions & Answers – Processes

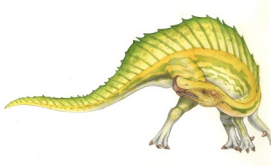
- 5. The state of a process is defined by _____
  a) the final activity of the process
  b) the activity just executed by the process
  c) the activity to next be executed by the process
  d) the current activity of the process

Answer: d
Explanation: The state of a process is defined by the current activity of the process. A process state changes when the process executes. The process states are as New, Ready, Running, Wait, Terminated.

Questions & Answers – Processes

- 6. What is a Process Control Block?
  a) Process type variable
  b) Data Structure
  c) A secondary storage section
  d) A Block in memory

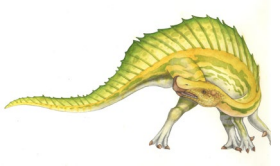
Answer: b
Explanation: A Process Control Block (PCB) is a data structure. It contains information related to a process such as Process State, Program Counter, CPU Register, etc. Process Control Block is also known as Task Control Block.



Questions & Answers – Processes

- 7. The entry of all the PCBs of the current processes is in _____
  a) Process Register
  b) Program Counter
  c) Process Table
  d) Process Unit

Answer: c
Explanation: The entry of all the PCBs of the current processes is in Process Table. The Process Table has the status of each and every process that is created in OS along with their PIDs.



Questions & Answers – Processes

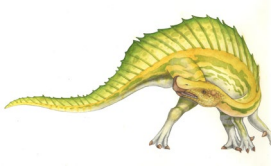
- 8. What is a long-term scheduler?
  a) It selects processes which have to be brought into the ready queue
  b) It selects processes which have to be executed next and allocates CPU
  c) It selects processes which heave to remove from memory by swapping
  d) None of the mentioned

Answer: a
Explanation: A long-term scheduler selects processes which have to be brought into the ready queue. When processes enter the system, they are put in the job queue. Long-term scheduler selects processes from the job queue and puts them in the ready queue. It is also known as Job Scheduler.

Questions & Answers – Processes

- 9. In a multiprogramming environment _____
  a) the processor executes more than one process at a time
  b) the programs are developed by more than one person
  c) more than one process resides in the memory
  d) a single user can execute many programs at the same time

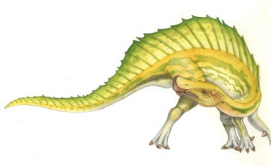
Answer: c
Explanation: In a multiprogramming environment more than one process resides in the memory. Whenever a CPU is available, one process amongst all present in memory gets the CPU for execution. Multiprogramming increases CPU utilization.



Questions & Answers – Processes

- 10. Message passing system allows processes to _____
  a) communicate with each other without sharing the same address space
  b) communicate with one another by resorting to shared data
  c) share data
  d) name the recipient or sender of the message

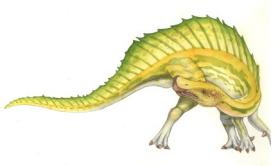
Answer: a
Explanation: Message Passing system allows processes to communicate with each other without sharing the same address space.



Questions & Answers – Processes

- 11. Which module gives control of the CPU to the process selected by the short-term scheduler?
  a) dispatcher
  b) interrupt
  c) scheduler
  d) none of the mentioned

Answer: a



Questions & Answers – Processes

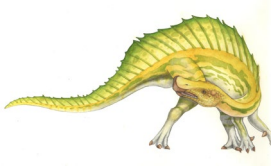
- 12. Which one of the following can not be scheduled by the kernel?
  a) kernel level thread
  b) user level thread
  c) process
  d) none of the mentioned

Answer: b
Explanation: User level threads are managed by thread library and the kernel is unaware of them.

Questions & Answers – Processes

- 13. Which of the following scheduling algorithms gives minimum average waiting time?
  a) FCFS
  b) SJF
  c) Round – robin
  d) Priority

Answer: b



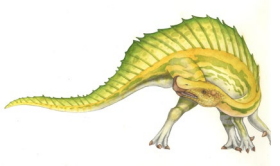
Semaphore

- **Counting semaphore** – integer value can range over an unrestricted domain
- **Binary semaphore** – integer value can range only between 0 and 1
  - Same as a **mutex lock**
- Can implement a counting semaphore ***S*** as a binary semaphore
- With semaphores we can solve various synchronization problems

Questions and Answers

- 1. Concurrent access to shared data may result in _____
  a) data consistency
  b) data insecurity
  c) data inconsistency
  d) none of the mentioned

Answer: c.



Questions and Answers

- 2. Mutual exclusion implies that _____
  a) if a process is executing in its critical section, then no other process must be executing in their critical sections
  b) if a process is executing in its critical section, then other processes must be executing in their critical sections
  c) if a process is executing in its critical section, then all the resources of the system must be blocked until it finishes execution
  d) none of the mentioned

  Answer: a.

Questions and Answers

- 3. Bounded waiting implies that there exists a bound on the number of times a process is allowed to enter its critical section _____
  a) after a process has made a request to enter its critical section and before the request is granted
  b) when another process is in its critical section
  c) before a process has made a request to enter its critical section
  d) none of the mentioned

Answer: a.

Questions and Answers

- 5. TestAndSet instruction is executed _____
  a) after a particular process
  b) periodically
  c) atomically
  d) none of the mentioned

Answer: c.

Questions and Answers

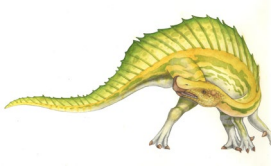
- 6. What are Spinlocks?
  a) CPU cycles wasting locks over critical sections of programs
  b) Locks that avoid time wastage in context switches
  c) Locks that work better on multiprocessor systems
  d) All of the mentioned

Answer: d.

Questions and Answers

- 7. The wait operation of the semaphore basically works on the basic _____ system call.
  a) stop()
  b) block()
  c) hold()
  d) wait()

Answer: b.



Questions and Answers

- 8. The signal operation of the semaphore basically works on the basic _____ system call.
  a) continue()
  b) wakeup()
  c) getup()
  d) start()

Answer: b.

Questions and Answers

- 9. If the semaphore value is negative _____
  a) its magnitude is the number of processes waiting on that semaphore
  b) it is invalid
  c) no operation can be further performed on it until the signal operation is performed on it
  d) none of the mentioned

Answer: a.

Questions and Answers

- 10. The following program consists of 3 concurrent processes and 3 binary semaphores. The semaphores are initialized as S0 = 1, S1 = 0, S2 = 0.

```
Process P0
while(true)
{
   wait(S0);
   print '0';
   release(S1);
   release(S2);
}

Process P1
wait(S1);
release(S0);

Process P2
wait(S2);
release(S0);
```

How many times will P0 print '0'?

a) At least twice
b) Exactly twice
c) Exactly thrice
d) Exactly once

Answer: a.

Questions and Answers

The minimum number of times 0 printed:

- S0 =1 then P0 enter into the critical section
- **print '0'**
- then release S1 and S2 means S1 =1 and s2 =1
- now either P1 or P2 can enter into the critical section
- if P1 enter into the critical section
- release S0
- then P2 enter into the critical section
- release S0
- P1 enter into the critical section
- **print '0'**

The minimum number of time **0 printed** is **twice** when executing in this order **(p0 -> p1 -> p2 -> p0)**

The Maximum number of times 0 printed:

- S0 =1 then P0 enter into the critical section
- **print '0'**
- Then release S1 and S2 means S1 =1 and s2 =1
- Now either P1 or P2 can enter into the critical section
- If P1 enter into the critical section
- Release S0 means S0 =1
- S0 =1 then P0 enter into the critical section
- **print '0'**
- Then P2 enter into the critical section
- Release S0 means S0 =1
- S0 =1 then P0 enter into the critical section
- **print '0'**

Maximum no. of time **0 printed** is **thrice** when execute in this order **(p0 -> p1 -> p0 -> p2 -> p0)**

Questions and Answers

- 11. Each process Pi, i = 0,1,2,3,……,9 is coded as follows.

```
repeat
  P(mutex)
  {Critical Section}
  V(mutex)
forever
```

The code for P10 is identical except that it uses V(mutex) instead of P(mutex). What is the largest number of processes that can be inside the critical section at any moment (the mutex being initialized to 1) ?

a) 1 b) 2 c) 3 d) None of the mentioned

Answer: c

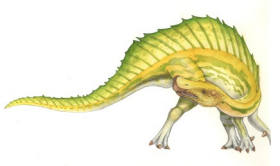
Explanation: Let the mutex be initialized to 1. Any one of the 9 processes Pi, i = 1 , 2 , 3 , ... 9 can get into the critical section after executing P (mutex) which decrements the mutex value to 0. At this time P10 can enter into the critical section as it uses V (mutex) instead of P(mutex) to get into the critical section. As a result of this, mutex will be incremented by 1. Now any one of the 9 processes Pi, i — 1, 2 , 3 , ... 9 (excepting the one that is already inside the critical section) can get into the critical section after decrementing the mutex to 0. None of the remaining processes can get into the critical section.

If the mutex is initialized to 0, only 2 processes can get into the critical section. So the largest number of processes is 3.

Questions and Answers

- 12. A monitor is a type of _____
  a) semaphore
  b) low level synchronization construct
  c) high level synchronization construct
  d) none of the mentioned

Answer: c.



Questions and Answers

- 13. Which is the process of invoking the wait operation?
  a) suspended until another process invokes the signal operation
  b) waiting for another process to complete before it can itself call the signal operation
  c) stopped until the next process in the queue finishes execution
  d) none of the mentioned

Answer: a.

Questions and Answers

- 14. If no process is suspended, the signal operation _____
  a) puts the system into a deadlock state
  b) suspends some default process execution
  c) nothing happens
  d) the output is unpredictable

Answer: c.

Questions and Answers

- 15. A state is known as _____, if the system can allocate resources to each process in some order and still avoid a deadlock
  - a) Safe
  - b) Protected
  - c) Unsafe
  - d) Locked

Answer: a.

Questions and Answers

- 16. What is the drawback of banker's algorithm?
  a) in advance processes rarely know how much resource they will need
  b) the number of processes changes as time progresses
  c) resource once available can disappear
  d) all of the mentioned

Answer: d.

Questions and Answers

- 有3个进程P1、P2、P3共享某个资波原F，P1对F只读不写，P2对F只写不读，P3对F既读也写，当一个进程在写F时则其它进程不能对F进行读写，但多个进程何以人同时读F。试用P、V操作正确实现这三个进程的同步与互斥，要求

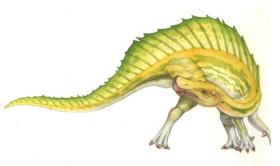
  (1)正常运行时不产生死锁

  (2)使用F的并发性要高。

Questions and Answers

```
int readcount=0;
semaphore rmutex=1, mutex=1;

P1() {
    while(1){
       P(rmutex);
       if(readcount==0) P(mutex);
       readcount++;
       V(rmutex);
       读F;
       P(rmutex);
       readcount--;
       if(readcount==0)  V(mutex);
       V(rmutex);
     }
}
```



Questions and Answers

```
P2(){
    while(1){
        P(mutex);
        写F;
        V(mutex);
    }
}
```

Questions and Answers

```
P3( ) {
        while(1){
            P(rmutex);
            if(readcount==0)  P(mutex);
            readcount++;
            V(rmutex);
            读F;
            P(rmutex);
            readcount--;
            if(readcount==0)  V(mutex);
            V(rmutex);
            P(mutex);
            写F;
            V(mutex);
         }
}
```