第三章进程管理

3.1进程与程序

3.2进程的定义与控制

3.3进程调度

3.4进程间的相互作用

3.5进程通信

3.6线程

3.7UNIX和Windows的进程和线程模型

3.1 进程与程序

**程序的顺序执行和并发执行**

**顺序执行的特征**

顺序性：CPU严格按照程序结构所指定的次序执行。

封闭性：独占全部资源，资源的状态只能由该程序本身改变，不受其它程序和外界因素影响。

可再现性：如果程序执行环境和初始条件相同，则其执行的结果相同。

多道程序设计：把一个以上的程序放入内存中，并且同时处于运行状态，这些程序共享CPU和其它资源。特点如下：

多道：内存中有多道程序，它们在任一时刻必须处于就绪、运行、阻塞三种状态。

宏观上并行：从宏观上看，它们在同时执行。

微观上串行：从微观上看，它们在交替、穿插执行

**并发执行的特征**：

失去封闭性：共享资源，程序之间互相制约。

间断性：程序之间的制约关系致使程序执行时间不连贯。

不可再现性：失去封闭性，也就失去了可再现性，程序执行的结果随速度、环境的不同而不同。

综上所述，由于程序的并发执行**破坏了程序的封闭性和可再现性**，使得程序和程序的执行不再一一对应，因此，程序这个静态的概念已经不能切实反映程序执行的各种特征。于是，引入“进程”，能够反映程序执行的独立性、并发性和动态性等特征

3.2 进程定义与控制

**进程定义**

进程是程序的一次执行

进程是可以和别的计算并发执行的计算

进程是定义在一个数据结构上并能在其上进行操作的一个程序

进程是程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位

**进程与程序的区别**

进程是动态的，程序是静态的：程序是有序代码的集合；进程是程序的一次执行。

进程是暂时的，程序的永久的：进程是一个状态变化的过程，程序可长久保存。

进程与程序的组成不同：进程的组成包括程序、数据和进程控制块（即进程状态信息）。

进程与程序的对应关系：通过多次执行，一个程序可对应多个进程；通过调用关系，一个进程可包括多个程序。

**进程定义**：是程序的一次执行，该程序可与其它程序并发执行；它是一个动态实体，在传统的操作系统设计中，进程既是基本的分配单位，也是基本的执行单位。

**进程组成**：有程序段、数据段和进程控制块(PCB)组成。

程序和数据是进程存在的物理基础，是进程的实体

进程控制块是进程的灵魂，是进程存在的唯一标志

操作系统为进程创建进程控制块和分配地址空间的过程就是进程创建的过程

进程控制块(PCB)：

**标识信息**：唯一的标识一个进程，主要有进程标识、用户标识和父进程标识。

**状态信息**：与CPU有关的各种现场信息，包括寄存器状态、堆栈指针。以便该进程重新占用CPU后能够继续执行。

**控制信息**：操作系统对进程进行调度管理时用到的信息。主要有进程状态、调度信息、队列指针、资源占有使用信息等。

**进程控制块的组织方式**

PCB在内存中是以表的形式存在的－PCB表。还可以将相同性质的进程组织在一张表中，形成多个索引表。

有些操作系统将PCB分为常驻内存和非常驻内存两部分，如UNIX。

**进程基本状态**

运行态（Running）：进程已经获得所需资源，并占有CPU

就绪态（Ready）：已经获得所需资源，只等待CPU

阻塞态（Blocked）：也称为等待态、挂起态或睡眠态等，进程等待某个事件，如等待I/O完成，等待某个资源

此外，还可以有新建态、终止态。

**进程状态转换**

① 就绪 → 运行

* + 调度：调度程序选择一个新的进程运行

② 运行 → 就绪

* + 超时：运行进程用完时间片被中断, 在抢占调度方式中, 因为一高优先级进程进入就绪状态

③ 运行 → 阻塞

* + I/O请求：当进程发生I/O请求或等待某事件时

④ 阻塞 → 就绪

* + I/O完成：当I/O完成或所等待的事件发生时

挂起 （suspend）用于调节负载

收回内存空间，其进程映像交换到磁盘上

**进程控制**－通过原语（Primitive）操作实现

原语是指由机器指令构成的可完成特定功能的程序段。它是一个机器指令的集合，在执行时不能被中断。多采用屏蔽中断方法实现。进程控制原语有：

进程创建原语（create primitive）

进程撤消原语（destroy primitive）

进程阻塞原语（block primitive）

进程唤醒原语（wakeup primitive）

进程挂起原语（suspend primitive）

进程激活原语（active primitive）

**进程关系的树型结构**

主要优点

资源分配严格：子进程仅能分配到父进程所拥有的资源，用完后归还。

进程控制灵活：可根据需要给进程以不同的控制权限。

进程结构清楚，关系明确。

**进程特征**

动态性：“执行”、“计算”、“运行过程”都强调动态性，进程的最基本特征。

并发性：进程的重要特征，同时也是操作系统的重要特征。

异步性：进程按各自独立的不可预知的速度向前推进，即进程按异步方式进行，这导致了进程执行的不可再现性，因此，操作系统必须采用某些措施来限制各进程推进序列以保证各程序间正常协调运行。

独立性：进程是一个独立运行的基本单位，即是一个独立获得资源和独立调度的单位。

制约性：一个进程的执行可能依赖其他进程的执行结果。

结构性：每个进程有固定结构，包括程序、数据和PCB三部分。

3.3进程调度

进程调度属于低级调度，就是从就绪队列中，按照一定的算法选择某个进程占用CPU。

**进程调度时机**

现运行进程或者因任务完成而正常结束，或者因错误而异常结束

现运行进程因某种原因，比如I/O请求，从运行态进入阻塞状态

现运行进程执行某种原语操作，如P操作、阻塞原语等，进入阻塞状态

一个具有更高优先数的进程要求CPU，即已进入就绪队列

分配给该进程运行的时间片已用完

**进程调度算法**

先来先服务进程调度算法（FCFS）

基于优先数的进程调度算法：根据优先数大小确定优先级高低，分为

静态优先数法：进程创建时就规定好优先数

动态优先数法：优先数在执行过程中根据情况改变，例如UNIX

时间片轮转进程调度算法

固定时间片

可变时间片

多级队列轮转调度算法

**进程调度方式**

当一个进程正在CPU上运行时，若有一个更为紧迫或更为重要的进程需要进行处理，或者说，如果有更高优先数的进程进入就绪队列时，如何分配CPU。通常有两种方式：

不可剥夺方式（不可抢占方式， non-preemptive）

可剥夺方式（可抢占方式，preemptive）

3.4 进程间的相互作用

**同步**：一个进程的某个操作与协作进程的某个操作之间在时序上有一定的关系。如果协作进程的某个操作没有完成，那么该进程就要等待这个操作完成才能继续下去，这种需要相互合作、协同工作的进程之间的相互关系称为进程的同步。

临界资源（独占资源）：指在一段时间内只允许一个进程访问的资源。

**互斥**：当两个或两个以上进程竞争同一临界资源时，进程间的制约关系称为进程的互斥。

临界资源：critical resource

系统中某些资源一次只允许一个进程使用，称这样的资源为临界资源或互斥资源或共享变量

临界资源可以是一些硬件设备（如打印机、磁带机或绘图仪等），也可以是进程共享变量、数据、队列、或使用权限等“有形”或“无形”的资源。

临界区（互斥区）：critical section

一个程序片段的集合，这些程序片段分散在不同的进程中，对某个共享的数据结构（共享资源）进行操作.

在进程中涉及到临界资源的程序段叫临界区.

多个进程的临界区称为相关临界区.

使用互斥区的原则

(1)当有若干个进程要求进入临界区时，应使一个进程进入临界区，它们不应相互等待而使谁都不能进入，即进程不能无限地停留在等待临界资源的状态。

(2)一次只允许一个进程进入临界区中，即各进程互斥访问临界资源。

(3)各进程使用临界资源的时间是有限的，即任何一个进程都必须在有限的时间内释放所占资源。

**信号量及P.V操作**

信号量（Semaphore）是表示资源的实体，是一个与队列有关的整型变量，其值仅能由P、V操作改变。

信号量分为：公用信号量和私用信号量。

公用信号量:用于实现进程间的互斥，初值通常设为1，它所联系的一组并行进程均可对它实施P、V操作；

私用信号量用于实现进程间的同步，初始值通常设为0或n，允许拥有它的进程对其实施P操作。

**P(s)**{

s.value = s.value - 1;

if (s.value < 0) {

该进程状态置为等待状态;

将该进程的PCB插入相应的等待队列末尾s.queue;

}

}

**V(s)**{

s.value = s.value + 1;

if (s.value < = 0) {

唤醒相应等待队列s.queue中等待的一个进程

改变其状态为就绪态

并将其插入就绪队列

}

信号量值的含义：

s.value >= 0时，其值表示还有可用的资源数；

s.value < 0 时，其绝对值表示有多少个进程因申请该信号量表示的资源，得不到而进入阻塞态；

进程间的同步

对P，V操作的使用应注意：

P，V操作都是成对出现的：互斥操作时，它们在同一进程中；同步操作时，它们处于不同的进程。

在进程中，P操作的位置和次序至关重要。一般情况下，对互斥信号量的P操作在后。而V操作没有特别的限制。

P，V操作的优点是：原语完备，表达能力强，可以解决任何同步和互斥问题；缺点是不够安全，实现复杂。

用P，V操作实现司机－售票员同步：设置信号量S车，S门，初值均为0

生产者-消费者问题

读者、写者问题

理发师问题

**读者写者问题**

读者进程

begin

P(mutex);申请readnum使用权

readnum=readnum+1;

if (readnum==1) P(write);

V(mutex);

read file;

P(mutex);

readnum=readnum－1;

if (readnum==0) V(write);

V(mutex);

End

写者进程

begin

P(write);

write file;

V(write);

end

使用readnum对读者计数，初值为0；

mutex是对readnum进行互斥操作的信号量，初值为1；

write是写信号量，初值为1。

**1读2写3读4读5读-------n读**

读者来，write=0；

写者来，P（write）阻塞

读者来，readnum+1继续读

读者来，readnum+1继续读

对于写者不公平。需要等所有读者读完才能写。写者会出现“饥饿”现象

当有进程读文件时，**如果有写进程请求写**，那么新的读进程被拒绝，待现有读进程读完后，立即让写进程开始运行，当无写进程运行时才让读进程运行。

设置信号量S实现读者与写者或写者之间的互斥，初值为1；用信号量Sn限制系统中最多n个进程，初值为n

读者进程Pi(i=1,2,...,n)

begin

P(S);

P(Sn);

V(S);

read file;

V(Sn);

end

写者进程Pj (j=1,2,...,n)

begin

P(S);

for i=1 to n do P(Sn);

write file;

for i=1 to n do V(Sn);

V(S);

end

读者来

P(S); s=0

P(Sn); sn=n-1

V(S); s=1

read file;

V(Sn);

写者来

P(S); s=0

for i=1:n doP(Sn)sn=-1阻塞

write file;

for i=1 to n do V(Sn);

V(S);

写者来

P(S); s=-1阻塞

读者来

P(S); s=-2阻塞

多个读者可以同时读，

读者后面来写者后就不能读了

理发师问题

Customers记录等待理发师的顾客数

=waiting是变量

为啥设置变量，信号量不能进行判定操作，仅能进行PV操作。

V(barbers);//一个理发师开始理发（有顾客后，立即释放barbers,其他人可以申请，提前准备）

V(mutex);//释放等候

P(mutex);//（申请waiting的使用权）

V(customers);//如果必要，唤醒理发师（先告诉理发师可以理发了，）

V(mutex);//释放访问等候

**思路**

**1分析进程的关系：同步或者互斥。**

* **能不能同时操作；不能同时操作，设置互斥信号量。（写者）**
* **多个进程使用同一临界资源是互斥（缓冲池）。**

**设置互斥信号量。**

* **按照固定顺序执行是同步。（先生产再取）。**

**在前趋进程加V操作唤醒后继进程，后继进程加P操作，**

**保证是在之后进行。**

**2设置信号量，变量以及初值。**

**3写代码。**

**生产者-消费者问题**

生产者和消费者**同步**：设S缓，S产，初值分别为n和0

多个生产者或者消费者**互斥访问缓冲区**：互斥信号量mutex初值1

**读者、写者问题**

1多个读进程可以同时共享资源，但不能和写进程共享。

当第一个读进程开始进行访问时执行P操作，当最后一个读进程结束访问时执行V操作

使用readnum对读者计数，初值为0；

mutex是对readnum进行互斥操作的信号量，初值为1；

2写进程之间互斥，访问时必须独占资源。互斥信号量write初值1

**理发师问题**

只有一个理发师，理发或者休息

**理发师与顾客**，**同步**，customers barbers

customers用来记录等待理发师的顾客数（不包括正在理发的顾客），初值为0；

barbers，记录正在等候顾客的理发师数，为0或1，初值为0

**顾客与顾客**，没有同步关系（谁先来谁理），没有互斥关系，不能理发就等待。

n个等待，等待人数，互斥信号量对计数器进行互斥。

管程（monitor）：解决互斥问题的抽象数据类型。进程可以调用管程中的过程，但不能在管程外的过程中直接访问管程内的数据结构。基本思想是集中管理各进程临界区，按不同的管理方式定义模块的类型和结构，用数据表示抽象系统资源，增加模块的相对独立性。

3.5 进程通信

低级通信和高级通信：根据交换信息量的多少和效率的高低，如P、V操作属于低级通信；高级通信包括管道通信和信箱通信。

低级通信只传递状态和整数值，信息量小，效率低，传递较多信息需要多次通信，编程复杂，不易理解。

高级通信能够传递大批量数据，减轻程序编制的复杂度。包括共享内存模式、消息传递模式和共享文件模式（管道）。

直接通信：信息由发送方直接传递给接收方，如管道。

间接通信：将收发双方进程之外的共享数据结构作为通信中转。

(1)共享内存模式：一种最快捷高效的方式，在UNIX系统中被使用。

系统在内存中指定一个区域作为共享存储区，建立一张段表进行管理，各进程可以申请其中一个存储段，并在申请时提供关键字。若申请的存储区已经被其它进程占有，系统会向申请进程返回关键字，该存储区就链接到了进程的逻辑地址空间，此后进程就可以直接存取共享存储区中的数据。

(2)消息传递方式：

消息由发送方形成，通过一定的机制传递给接收方。长度可以固定，也可以变化。

每个消息都由消息头和消息体组成，其主要结构包含：

指向发送进程的指针：Sptr

指向下一消息缓冲区的指针：Nptr

消息长度：Size

消息正文：Text

(3)信箱

是一个用来对一定数量的消息进行缓存的地方。是一段存储区，每一个信箱用标识符加以区分，由信箱头和信箱体两部分组成。信箱头存放控制信息，信箱体存放消息内容。一个信箱可以被多个进程共享，就实现了消息的广播发送。

Send(X,mail)：邮件mail发送到信箱X中

Receive(X,mail)：接收信箱X中的邮件mail

(4)管道(pipe)：是一种共享文件模式，基于文件系统，连接于两个进程之间，以先进先出的方式实现消息的单向传送。

在UNIX系统中，管道的创建用函数pipe()实现。该函数返回用于读、写操作的文件描述符fd[0],fd[1]。读管道时调用read()函数，利用参数fd[0]从管道中读取字节。写管道时调用write()函数，利用参数fd[1]向管道写信息。

命名管道: 命名管道可以在不相关的进程之间和不同计算机之间使用，服务器建立命名管道时给**它指定一个名字**，任何进程都可以通过该名字打开管道的另一端，根据给定的权限和服务器进程通信。

无名管道:只能用于具有亲缘关系的进程之间，父子进程，兄弟进程之间通信，因为只有子进程才能继续父进程的文件描述符。

3.6 线程

进程作为调度基本单位的问题：

调度工作量大，耗费系统资源

进程间通信延迟大，频率高的通信过程效率低下。

没有达到理想的并行度。

线程（thread）：也叫轻型进程，是可执行的实体单元，可代替以往的进程，是处理机调度的基本单位。

多线程：单个进程中执行多个线程。典型操作系统有Windows NT、Solaris、Mach和OS/2。

在多线程环境中，进程被定义为保护单位和资源分配单位，在一个进程内部可以有多个线程。

3.7 unix 进程管理

1 有5个进程Pa、Pb、Pc、Pd和Pe，它们依次进入就绪队列，它们的优先级和需要的处理机时间：

进程 处理机时间 优先级  
Pa 10 3  
Pb 1 1  
Pc 2 3  
Pd 1 4  
Pe 5 2  
忽略进程调度等所花费的时间，请回答下列问题：  
(1) 分别写出采用“先来先服务”和“非抢占式的优先数(优先数大优先级低)”调度算法选中进程执行的次序。  
(2) 分别计算出上述两种算法使各进程在就绪队列中的等待时间及两种算法下的平均等待时间。  
2 某寺庙,有小和尚和老和尚若干,有一个水缸,由小和尚提水入缸供老和尚饮用.水缸可以容纳10桶水,水取自同一口井中,由于水井口窄,每次只能容纳一个水桶取水.水桶总数为3个.每次入水、取水仅为一桶,且不可同时进行.试给出有关取水、入水的算法描述.