进程同步

并发进程间制约关系

- 资源共享关系-间接制约
  - 多个今晨彼此无关，完全不知道或只能间接感知其他今春的存在
  - 系统保证个进程能互斥的访问临界资源
  - 系统资源统一分配，而不允许用户进程直接用
- 相互合作关系-相互制约
  - 系统应保证相互合作的个进程在执行次序上的协调和防止有关时间的差错

临界资源

- 一段时间里只能为某个进程所访问的资源
  - 许多物理设备，变量及表格

临界区

- 每个进程中访问临界资源的那段代码称为临界区，并不是内存中的一段区域
- 保证个今晨互斥的进入自己的临界区是实现他们对临界资源的互斥访问的充要条件

访问临界资源的循环进程描述

```
进程Pi
begin
    repeat
    ......
    进入区
    临界区
    退出区
    ......
    until false;
end
```

进程同步机制准则

进程同步机制基本准则

- 空闲让进
  - 当无进程处于临界区时，允许请求进入临界区的进程立即进入自己的临界区
- 忙则等待
  - 当已有进程进入自己的临界区时，所有企图进入进入临界区的进程必须等待
- 有限等待
  - 对要求访问临界资源的进程，应保证该进程能在有限时间内进入自己的临界区
- 让权等待
  - 当进程不能进入自己的临界区时，应释放处理机

进程互斥访问临界资源的软件解决方案

- 有两个进程Pi，Pj共享某临界资源R
- 主程序

```
begin
    perbegin //并行执行
    Pi;
    Pj;
    perend;
end
```

算法#1

设置全局授权访问编号

设置一个全局的整型变量,如果该变量等于本进程的pid号，则可访问临界区，如果不等于，程序进入循环，执行空操作等待其他进程释放。当进程使用完临界区代码后，全局变量指向另一个进程，使其可以访问临界区

global

```
global flag:integer:=NULL;
```

P_i

```
begin
repeat
    ...
    while flag do
        no_op;
    flag:=PID_i;
    临界区
    flag:=NULL;
    ...
until false;
end
```

违背了空闲让进:

上面的算法属于强制交替访问临界资源，不能灵活考虑进程的实际需求

算法#2

为每个进程设置全局访问标志

充分的考虑进程对临界资源访问的需求，在访问之前表示自己正在访问临界资源，每个程序都有一个全局变量用于标识自己否在访问临界资源，访问临界资源时，本程序的访问标志设置为真，访问结束后才将标志设置为假

global

```
glbal flag_i,flag_j:boolean:=false,false;
```

P_i

```
begin
repeat
    ...
    while flag_j do
        no_opt;
    flag_i:=true;
    临界区
    flag_i:=false;
    ...
until false;
end
```

P_j

```
begin
repeat
    ...
    while flag_j do
        no_opt;
    flag_i:=true;
    临界区
    flag_i:=false;
    ...
until false;
end
```

违背了忙则等待：

当i进程检查j的标志位时，发现j的标志位还没有设置为真，循环判定为假，若此时调度到j程序，j进程检查i的标志位为假，则循环判定为假，进入临界区，此时调度程序重新调度到i进程，把自己的标志位设置为真，也要进入临界区，此时两个程序都要访问临界区。

算法#3

设置欲访问标志位

每次有访问临界区的需求时，先把自己的欲访问标志置成真，再判断其他软件的欲访问标志位，如果检查到其他软件的访问欲标志位为真，则等待其他软件的欲访问标志位为假，

global

```
global flag_i,flag_j:boolean:=false,false;
```

P_i

```
begin
repeat
    ...
    flag_i:=true;
    while flag_j do
        no_op;
    临界区
    flag_i:=false;
    ...
until false;
end
```

P_j

```
begin
repeat
    ...
    flag_j:=true;
    while flag_i do
        no_op;
    临界区
    flag_j:=false;
    ...
until false;
end
```

违背了空闲让进，有限等待：

当程序i刚刚把欲访问标志设置为真的时候，调度程序进行了重新调度，调度到了j程序，j程序也设置了自己的欲访问权限，然后检查i程序的欲访问标志，发现i的欲访问标志为真，所以开始死等，调度到i进程的时候，他会开始检查j的欲访问标志位，发现j的欲访问标志为也为真，所以也开始了等待，因为没有进入临界区，所以不会重设欲访问标志位为假，两个进程陷入了死等状态。

算法#4

编号+标志-Peterson算法

global

```
global flag_i,flag_j:=false;
global turn:=integer;
```

P_i

```
begin
repeat
    ...
    flag_i:true; turn:=j;
    while(flag_j and turn==j)
        do no_op;
    临界区
    flag_i=false;
    ...
until false;
end
```

P_j

```
begin
repeat
    ...
    flag_j:=true; turn:=i;
    while(flag_i and turn==i)
        do no_op;
    临界区
    flag_j=false;
    ...
until false;
end
```

谦让政策：

先让对方访问临界区，如果对方不想访问临界区，那么他的标志一定不为真，跳出循环，进入临界区。

当另一个程序想访问临界区，它的循环判定一定为真，不会干扰到另一个程序的运行

信号量

两种操作

P/V：wait/signal

整型信号量

`s:=integer;`

wait(): while s<0 do no_op; s:=s-1;

signal():s:=s+1;

没有实现让权等待