**Peterson算法**

**boolean flag[2];（初始值为假。准备进入临界区时，设置为真，离开临界区后，设置为假）**

**int turn;（执行次序）**

**互斥性：**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **Do{**  **flag[0]=true; turn=1;**  **while (flag[1] && turn==1);**  **临界区（flag[0]=true）**  **flag[0]=false;**  **其余代码**  **}while(1);** | **Do{**  **flag[1]=true; turn=0;**  **while (flag[0] && turn==0);**  **临界区**  **flag[1]=false;**  **其余代码**  **}while(1);** |

**进展性：（单方请求）**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **Do{**  **flag[0]=true; turn=1;**  **while (flag[1] && turn==1);**  **临界区**  **flag[0]=false;**  **其余代码**  **}while(1);** | **Do{**  **flag[1]=true; turn=0;**  **while (flag[0] && turn==0);**  **临界区**  **flag[1]=false;**  **其余代码**  **}while(1);** |

**进展性：（双方请求）**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **Do{**  **flag[0]=true; turn=1;** **①**  **while (flag[1] && turn==1); ③**  **临界区**  **flag[0]=false;**  **其余代码**  **}while(1);** | **Do{**  **flag[1]=true; turn=0; ②**  **while (flag[0] && turn==0); ④**  **临界区**  **flag[1]=false;**  **其余代码**  **}while(1);** |
| **Do{**  **flag[0]=true; turn=1;** **①**  **while (flag[1] && turn==1); ③**  **临界区**  **flag[0]=false;** **⑤**  **其余代码**  **}while(1);** | **Do{**  **flag[1]=true; turn=0; ②**  **while (flag[0] && turn==0); ④⑥**  **临界区**  **flag[1]=false;**  **其余代码**  **}while(1);** |

**有限等待性：**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **Do{**  **flag[0]=true; turn=1;** **①⑥**  **while (flag[1] && turn==1); ③⑦**  **临界区**  **flag[0]=false;** **⑤**  **其余代码**  **}while(1);** | **Do{**  **flag[1]=true; turn=0; ②**  **while (flag[0] && turn==0); ④⑧**  **临界区**  **flag[1]=false;**  **其余代码**  **}while(1);** |

**Dekkel算法**

**int flag[2]; (init 0)**

**int turn; (0 or 1)**

**互斥性：**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **do{**  **flag[0]=1;**  **while(flag[1])**  **if(turn==1){**  **flag[0]=0;**  **while (turn==1)**  **skip;**  **flag[0]=1;**  **}**  **临界区（flag[0]=1，turn=0）**  **turn=1;**  **flag[0]=0;**  **其余代码**  **}while(1);** | **do{**  **flag[1]=1;**  **while(flag[0])** **①**  **if(turn==0){**  **flag[1]=0;**  **while (turn==0) ②**  **skip;**  **flag[1]=1;**  **}**  **临界区**  **turn=0;**  **flag[1]=0;**  **其余代码**  **}while(1);** |

**进展性：（假定turn=0）**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **do{**  **flag[0]=1; ①**  **while(flag[1]) ③…⑧（flag[1]）**  **if(turn==1){ ⑤**  **flag[0]=0;**  **while (turn==1)**  **skip;**  **flag[0]=1;**  **}**  **临界区（flag[0]=1，turn=0）**  **turn=1; ⑨**  **flag[0]=0; ⑩**  **其余代码**  **}while(1);** | **do{**  **flag[1]=1; ②**  **while(flag[0]) ④**  **if(turn==0){**  **flag[1]=0; ⑥**  **while (turn==0)** **⑦**  **skip;**  **flag[1]=1;**  **}**  **临界区**  **turn=0;**  **flag[1]=0;**  **其余代码**  **}while(1);** |
| **do{**  **flag[0]=1; ①**  **while(flag[1]) ③…⑧（flag[1]）**  **if(turn==1){ ⑤**  **flag[0]=0;**  **while (turn==1)**  **skip;**  **flag[0]=1;**  **}**  **临界区（flag[0]=1，turn=0）**  **turn=1; ⑨**  **flag[0]=0; ⑬**  **其余代码**  **}while(1);** | **do{**  **flag[1]=1; ②**  **while(flag[0]) ④⑫…⑭（flag[0]）**  **if(turn==0){**  **flag[1]=0; ⑥**  **while (turn==0)** **⑦⑩**  **skip;**  **flag[1]=1;** **⑪**  **}**  **临界区（flag[0]=0，turn=1）⑮**  **turn=0;**  **flag[1]=0;**  **其余代码**  **}while(1);** |
| **do{**  **flag[0]=1; ①**  **while(flag[1]) ③…⑧（flag[1]）**  **if(turn==1){ ⑤**  **flag[0]=0;**  **while (turn==1)**  **skip;**  **flag[0]=1;**  **}**  **临界区（flag[0]=1，turn=0）**  **turn=1; ⑨**  **flag[0]=0; ⑩**  **其余代码**  **}while(1);** | **do{**  **flag[1]=1; ②**  **while(flag[0]) ④⑬（flag[0]）**  **if(turn==0){**  **flag[1]=0; ⑥**  **while (turn==0)** **⑦⑪**  **skip;**  **flag[1]=1;** **⑫**  **}**  **临界区（flag[0]=0，turn=1）⑭**  **turn=0;**  **flag[1]=0;**  **其余代码**  **}while(1);** |

**有限等待性：**

|  |  |
| --- | --- |
| **P0:** | **P1:** |
| **do{**  **flag[0]=1; ①⑪**  **while(flag[1]) ③…⑧⑫**  **if(turn==1){ ⑤⑬**  **flag[0]=0; ⑭**  **while (turn==1)** **⑮**  **skip;**  **flag[0]=1;**  **}**  **临界区（flag[0]=1，turn=0）**  **turn=1; ⑨**  **flag[0]=0; ⑩**  **其余代码**  **}while(1);** | **do{**  **flag[1]=1; ②**  **while(flag[0]) ④⑰（flag[0]）**  **if(turn==0){**  **flag[1]=0; ⑥**  **while (turn==0)** **⑦⑯**  **skip;**  **flag[1]=1;**  **}**  **临界区（flag[0]=0，turn=1）⑱**  **turn=0;**  **flag[1]=0;**  **其余代码**  **}while(1);** |

**Lamport面包店算法**

|  |  |
| --- | --- |
| **Boolean choosing[0,…,n-1];（false)**  **Int number[0,…,n-1]; (0)**  **Pi 进入：**  **1. choosing[i]=true;**  **2. number[i]=max{number[0],…,number[n-1]}+1;**  **3. choosing[i]=false;**  **4. For(j=0;j<n;j++){**  **5. While (choosing[j]) skip;**  **6. While (number[j]!=0)and**  **7. (number[j],j)<(number[i],i) skip;**  **8. }**  **临界区**  **number[i]=0;** | **时间Tj**  **执行到这里，对于j，后续再发生的抓号一定大于number[i]**  **执行到这里，对于j，其Tj时刻之前抓号并进入和离开临界区的活动，一定已经结束** |

**Eisenberg/Mcguire算法**

**互斥性：**

|  |  |
| --- | --- |
| **Pi:** | **Pk:** |
| **Do{**  **flag[i]=want\_in;**  **j=turn;**  **While (j!=i)**  **If (flag[j] != idle)**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[i]=in\_cs;**  **j=0;**  **While (j<n)and(j==i or flag[j]!=in\_cs) do**  **j++;**  **}while (j!=n);**  **turn=i;**  **临界区(turn=i ,flag[i]=in\_cs)**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[i]=idle;** | **Do{**  **flag[k]=want\_in;**  **j=turn;**  **While (j!=k) （flag[i]=in\_cs）**  **If (flag[j] != idle)**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[k]=in\_cs;**  **j=0;**  **While (j<n)and(j==k or flag[j]!=in\_cs) do**  **j++;**  **}while (j!=n);**  **turn=k;**  **临界区**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[k]=idle;** |

**进展性：**



|  |  |
| --- | --- |
| **Pi(Ti)** | **Pk: (Ti)** |
| **Do{**  **flag[i]=want\_in; ①**  **j=turn;**  **While (j!=i)**  **If (flag[j] != idle)**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[i]=in\_cs; ④**  **j=0;**  **While (j<n)and(j==i or flag[j]!=in\_cs) do**  **j++;**  **}while (j!=n);**  **turn=i; ⑤**  **临界区 (turn=i ,flag[i]=in\_cs)**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[i]=idle;** | **Do{**  **flag[k]=want\_in; ②**  **j=turn;**  **While (j!=k) ③ ⑤**  **If (flag[j] != idle) （flag[i]= want\_in） （flag[i]=in\_cs）**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[k]=in\_cs;**  **j=0;**  **While (j<n)and(j==k or flag[j]!=in\_cs) do**  **j++;**  **}while (j!=n);**  **turn=k;**  **临界区**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[k]=idle;** |



|  |  |  |
| --- | --- | --- |
| **Pi(Ti)** | **Pk: (Ti)** | **Pl: (Tl)** |
| **Do{**  **flag[i]=want\_in; ①**  **j=turn;**  **While (j!=i)** **③ ⑪**  **If (flag[j] != idle) （flag[l]= want\_in）**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[i]=in\_cs; ④**  **j=0;**  **While (j<n)and(j==i or flag[j]!=in\_cs) do ⑨**  **j++; （j=l）**  **}while (j!=n);**  **turn=i;**  **临界区 (turn=i ,flag[i]=in\_cs)**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[i]=idle;** | **Do{**  **flag[k]=want\_in; ②**  **j=turn;**  **While (j!=k) ⑤ ⑧**  **If (flag[j] != idle) （flag[i]= want\_in）（flag[l]=in\_cs）**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[k]=in\_cs;**  **j=0;**  **While (j<n)and(j==k or flag[j]!=in\_cs) do**  **j++;**  **}while (j!=n);**  **turn=k;**  **临界区**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[k]=idle;** | **Do{**  **flag[l]=want\_in; ⑥**  **j=turn;**  **While (j!=l) ⑫**  **If (flag[j] != idle)**  **j=turn；**  **Else**  **j=(j+1)% n;**  **flag[l]=in\_cs; ⑦**  **j=0;**  **While (j<n)and(j==l or flag[j]!=in\_cs) do ⑩**  **j++; （j=i）**  **}while (j!=n);**  **turn=l;** **⑬**  **临界区**  **j=(turn+1)% n;**  **While (flag[j]==idle)**  **j=(j+1)% n;**  **turn=j;**  **flag[l]=idle;** |

基本思想：

两轮排序

* 第一轮，按照从turn出发，距离turn最近的次序处理want\_in（按顺序抓号）
* 第二轮，判断是否存在多个进程同时处于in\_cs状态（由于时间差和进程间执行顺序导致，相当于抓了同一个号码），如果存在多个，重复第一轮过程