- **Algorithm 4 :**

However, we now (finally) present a correct solution, due to **Peterson [1981].** This solution is basically a combination of Algorithm 3 and a slight modification of Algorithm 1.

The processes share two variables in common:

var flag : array[0..1] of boolean ;

turn : 0..1;

Initially **flag[0] = flag[1] = false**

P_i :

```
do{
        flag[i] := true;
        turn := j;
 while (flag[j] and turn=j) do skip;
                critical section

                flag[i] := false;

                remainder section
   }while(true);
```

P_j :

```
do{
        flag[j] := true;
        turn := i;
 while (flag[i] and turn=i) do skip;
                critical section

                flag[j] := false;

                remainder section
   }while(true);
```

The processes share two variables in common:

var flag : array[0..1] of boolean ;

turn : 0..1;

Initially **flag[0] = flag[1] = false** .

P_0

```
do{
    flag[0] := true;
        turn := 1;
while (flag[1]==true and turn==1);
            critical section
        flag[0] := false;
            remainder section
}while(true)
```

P_1 :

```
do{
            flag[1] := true;
            turn := 0;
while (flag[0]==true and turn==0) ;
                critical section
        flag[1] := false;
                remainder section
        }while(true);
```

Semaphores

- The previous solution for ME presented not easy to generalize for more complex problems.
- To overcome this difficulty, a new synchronization tool, called a semaphore, was introduced by Dijkstra .
- A semaphore S is an integer variable that, apart from initialization, can be accessed only through two standard atomic operations: P and V.
- The classical definitions of P(wait) and V(signal) are:

P(S): while S <= 0
do skip;
S : = S – 1;

V(S): S: = S + 1;

Binary Semaphores

- A ***binary semaphore*** is initialized to 1
- P() waits until the value is 1
  - Then set it to 0
- V() *sets* the value to 1
  - Wakes up a thread waiting at P(), if any

Two Uses of Semaphores

1. Mutual exclusion
   – Lock was designed to do this

```
lock->acquire();
// critical section
lock->release();
```

Two Uses of Semaphores

1. Mutual exclusion
    1. The lock function can be realized with a binary semaphore:
        - Semaphore has an initial value of 1
        - P() is called before a critical section
        - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

Two Uses of Semaphores

1. Mutual exclusion
   – Semaphore has an initial value of 1
   – P() is called before a critical section
   – V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

litter_box = 1

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

semaphore litter_box = 1;

P(litter_box); // success…

// critical section

V(litter_box);

litter_box = 1 → 0

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

```
semaphore litter_box = 1;
P(litter_box);
// critical section
V(litter_box);
```

litter_box = 0

Two Uses of Semaphores

1. Mutual exclusion
   - Semaphore has an initial value of 1
   - P() is called before a critical section
   - V() is called after the critical section

semaphore litter_box = 1;

P(litter_box); // fail

// critical section

V(litter_box);

litter_box = 0

Two Uses of Semaphores

1. Mutual exclusion
   – Semaphore has an initial value of 1
   – P() is called before a critical section
   – V() is called after the critical section

semaphore litter_box = 1;

P(litter_box);

// critical section

V(litter_box);

litter_box = 0 →1

Two Uses of Semaphores

2. Synchronization: Enforcing some order between threads

T1	**T2**
do X	
	wait for X
	do Y
wait for Y	
do Z	
	wait for Z
……	

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();                   P(wait_left);
    V(wait_left);                   slide_left();
    P(wait_right);                  slide_right();
    slide_right();                  V(wait_right);
}                               }
```

wait_left = 0
wait_right = 0

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();                   P(wait_left);
    V(wait_left);                   slide_left();
    P(wait_right);                  slide_right();
    slide_right();                  V(wait_right);
}                               }
```

wait_left = 0
wait_right = 0

wait

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {
    slide_left();
    V(wait_left);
    P(wait_right);
    slide_right();
}

Right_Paw() {
    P(wait_left);
    slide_left();
    slide_right();
    V(wait_right);
}
```

wait_left = 0
wait_right = 0

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();                   P(wait_left);
    V(wait_left);                   slide_left();
    P(wait_right);                  slide_right();
    slide_right();                  V(wait_right);
}                               }
```

wait_left = 0 → 1
wait_right = 0

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                    Right_Paw() {
    slide_left();                       P(wait_left);
    V(wait_left);                       slide_left();
    P(wait_right);                      slide_right();
    slide_right();                      V(wait_right);
}                                   }
```

wait_left = 1 → 0
wait_right = 0

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();               P(wait_left);
    V(wait_left);               slide_left();
    P(wait_right);              slide_right();
    slide_right();              V(wait_right);
}                           }
```

wait_left = 0
wait_right = 0

Two Uses of Semaphores

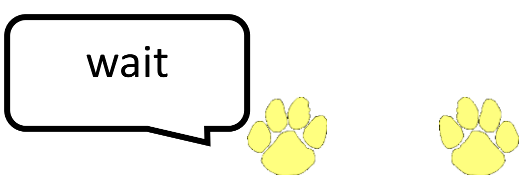
2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();               P(wait_left);
    V(wait_left);               slide_left();
    P(wait_right);              slide_right();
    slide_right();              V(wait_right);
}                           }
```

wait_left = 0
wait_right = 0



Two Uses of Semaphores

2. Scheduling

- Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();                   P(wait_left);
    V(wait_left);                   slide_left();
    P(wait_right);                  slide_right();
    slide_right();                  V(wait_right);
}                                }
```

```
wait_left = 0
wait_right = 0
```

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();                   P(wait_left);
    V(wait_left);                   slide_left();
    P(wait_right);                  slide_right();
    slide_right();                  V(wait_right);
}                               }
```

wait_left = 0
wait_right = 0 → 1

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();                   P(wait_left);
    V(wait_left);                   slide_left();
    P(wait_right);                  slide_right();
    slide_right();                  V(wait_right);
}                               }
```

wait_left = 0
wait_right = 1 → 0

Two Uses of Semaphores

2. Scheduling

– Semaphore usually has an initial value of 0

```
semaphore wait_left = 0;
semaphore wait_right = 0;

Left_Paw() {                Right_Paw() {
    slide_left();               P(wait_left);
    V(wait_left);               slide_left();
    P(wait_right);              slide_right();
    slide_right();              V(wait_right);
}                           }
```

wait_left = 0
wait_right = 0

Counting Semaphore

- Counting Semaphore is defined as a semaphore that contains integer values, and these values have an unrestricted value domain.
- A counting semaphore is helpful to coordinate the resource access, which includes multiple instances.

Problem in this implementation of semaphore

Whenever any process waits then it continuously checks for semaphore value (look at this line while (s<=0); in P operation) and waste CPU cycle. To avoid this another implementation is proposed.

```
P(S): while S <= 0  ;
         S : = S – 1;
```

```
P(Semaphore S):
        S. value := S. value – 1;
        if  S. value < 0
          then begin
                add this process to S.L and block;
           else return;
        end;
  V(Semaphore S):
         S. value := S. value + 1;
        if  S.value ≤ 0
          then begin
                remove this process P from S.L and wakeup(P);
        end;
```

```
P(Semaphore S):
        S. value := S. value – 1;
        if  S. value < 0
          then begin
                add this process to S.L and block;
           else return;
        end;
   V(Semaphore S):
         S. value := S. value + 1;
        if  S.value ≤ 0
          then begin
                remove this process P from S.L and wakeup(P);
        end;
```

P-4

- Current value of Semaphore S is 10, then after we perform 6P operations and 7V operations in the sequence? What will be the final value?