



# Synchronizacja cz. II

## Semaforey

Semaforem nazywamy zmienną chronioną, na ogół będącą nieujemną zmienną typu integer, do której dostęp (zapis i odczyt) możliwy jest tylko poprzez wywołanie specjalnych funkcji (operacji) dostępu i inicjacji.

Wyróżnia się semaforey:

- ☐ binarne  
przyjmujące tylko wartość 0 lub 1,
- ☐ ogólne (licznikowe)  
mogą przyjąć nieujemną wartość całkowitoliczbową.



## Instrukcja testandset

Załóżmy, że w systemie dostępna jest instrukcja typu testandset (a, b), która w sposób atomowy (niepodzielny) dokonuje odczytu zmiennej b, zapamiętania wartości tej zmiennej w zmiennej a oraz przypisania zmiennej b wartości true.

testandset(a, b) is equivalent to :

```
-----  
a := b;  
b := True;  
-----
```

nottestandset(a, b) is equivalent to :

```
-----  
a := b;  
b := False;  
-----
```



## Operacje P(S) i V(S)

Operacje P i V (Dijkstra)

Oznaczenie:

- P – pochodzi od holenderskiego proberen (testuj),
- V – pochodzi od holenderskiego od verhogen (inkrementuj)



Operacja P(S) na semaforze S działa w sposób następujący:

```
if S > 0  
then S := S - 1  
else (wait on S)
```

Operacja V(S) na semaforze S działa następująco:

```
if (one or more processes are waiting on S)  
then (let one of these processes proceed)  
else S := S + 1
```

## Przykład - testandset

```
1. program TESTANDSET EXAMPLE;  
2. var active: BOOLEAN;  
3. procedure PROCESSEONE;  
4. var oneCannotEnter: BOOLEAN;  
5. begin  
6.   while True do  
7.     begin  
8.       oneCannotEnter:=True;  
9.       while oneCannotEnter do  
10.        testandset  
11.          (oneCannotEnter, active);  
12.        criticalSectionOne;  
13.        active:=False;  
14.        otherStuffOne;  
15.      end  
16.    end;  
17.  end;  
18. procedure PROCESSTWO;  
19. var twoCannotEnter: BOOLEAN;  
20. begin  
21.   while True do  
22.     begin  
23.       twoCannotEnter:=True;  
24.       while twoCannotEnter do  
25.        testandset  
26.          (twoCannotEnter, active);  
27.        criticalSectionTwo;  
28.        active:=False;  
29.        otherStuffTwo;  
30.      end  
31.    end;  
32.  end;  
33. begin  
34.   active:=False;  
35.   parbegin  
36.     PROCESSEONE;  
37.     PROCESSTWO;  
38.   parend  
39. end.
```



## Przykład - semaforey

```
1. program SEMAPHOREEXAMPLE;  
2. var active: SEMAPHORE;  
3. procedure PROCESSEONE;  
4. begin  
5.   while True do  
6.     begin  
7.       P(active);  
8.       criticalSectionOne;  
9.       V(active);  
10.      otherStuffOne;  
11.    end  
12.  end;  
13. begin  
14.   semaphore_initialize(active,1);  
15.   parbegin  
16.     PROCESSEONE;  
17.     ...  
18.     PROCESSEONE;  
19.   parend  
20. end.
```



## Problem producenta-konsumenta

```

1. program PRODUCERCONSUMER;
2. var emptyBuffers, fullBuffers, active: SEMAPHORE;

3. procedure PRODUCER;
4. begin
5.   while True do
6.     begin
7.       produceNextRecord;
8.       P(emptyBuffers);
9.       P(active);
10.      addToBuffer;
11.      V(active);
12.      V(fullBuffers);
13.    end
14.  end;

15. procedure CONSUMER;
16. begin
17.   while True do
18.     begin
19.       P(fullBuffers);
20.       P(active);
21.       takeFromBuffer;
22.       V(active);
23.       V(emptyBuffers);
24.       processNextRecord;
25.     end
26.   end;
27. begin
28.   semaphore_initialize(active, 1);
29.   semaphore_initialize(emptyBuffers, N);
30.   semaphore_initialize(fullBuffers, 0);
31.   parbegin
32.     PRODUCER;
33.     CONSUMER;
34.   parend
35. end.

```



Slajd 8

## Specyfikacja implementacji z aktywnym czekaniem (ang. busy wait)

```

procedure P_S;
begin
  while S ≤ 0 do;
    S:=S-1
  end;

procedure V_S;
begin
  S:=S+1
end;

```



(c) Zakład Systemów Informatycznych

Slajd 11

## Semaforey binarne

*Semaforey binarne*  $S_b$  mogą przyjmować tylko dwie wartości: 0 i 1.

Przez  $P_b$  i  $V_b$  oznaczone są operacje na semaforach binarnych odpowiadające operacjom  $P$  i  $V$ .

Definicja  $P_b$  jest taka sama jak  $P$ , natomiast definicja  $V_b$  różni się od  $V$  tylko tym, że  $V_b$  nie zmienia wartości semafora binarnego jeśli miał on wartość 1.

(c) Zakład Systemów Informatycznych

Slajd 9

## Implementacja z aktywnym czekaniem

```

1. program PV_IMPLEMENTATION;
2. var active, delay: BOOLEAN;
3. var NS: INTEGER;
4. procedure IMPLEMENTATION;
5. var pActive, pDelay: BOOLEAN;
6. begin
7.   pActive:=True;
8.   while pActive do
9.     testandset(pActive, active);
10.    NS:=NS-1;
11.    if NS ≥ 0 then
12.      begin
13.        S:=S-1;
14.        active:=False;
15.      end
16.    else
17.      begin
18.        active:=False;
19.        pDelay:=True;
20.        while pDelay do
21.          testandset(pDelay, delay);
22.        end
23.      end;
24. procedure VIMPLEMENTATION;
25. var vActive: BOOLEAN;
26. begin
27.   vActive:=True;
28.   while vActive do
29.     testandset(vActive, active);
30.    NS:=NS+1;
31.    if NS > 0 then
32.      S:=S+1
33.    else
34.      delay:=False;
35.      active:=False
36.    end;
37. begin
38.   active:=False;
39.   delay:=True;
40. end.

```



(c) Zakład Systemów Informatycznych

Slajd 12

## Implementacja binarnych operacji semaforowych z aktywnym czekaniem

$P_b(S_b)$  is equivalent to:

```

repeat
  nottestandset(pActive, Sb) /* Sb:=False */
until (pActive);

```

$V_b(S_b)$  is equivalent to :

```

Sb := True ;

```



(c) Zakład Systemów Informatycznych

Slajd 10