Operativni sistemi 1

Konkurentno programiranje

Komunikacija i sinhronizacija

Predmetni asistenti:

Mihailo Obrenović, Jelica Vasiljević

Semafori

- Objekti operativnog sistema, služe za zaštitu kritičnog regiona i sinhronizaciju operacija
- Podaci semafora:
  - Semafor varijabla $s \in \{0, 1, \ldots, N\}$
  - Procesna lista B – lista procesa blokiranih na semaforu
- Operacije semafora:
  - P – dekrementiranje semafor varijable
  - V – inkrementiranje semafor varijable

Kako radi semafor?

- Početno stanje
  - Varijabla s može da se inicijalizuje na bilo koju vrednost između 0 i N
  - N uglavnom predstavlja broj raspoloživih jedinica nekog resursa
  - Procesna lista B – prazna
- Operacije:

P ::

```
if s > 0 then s := s - 1;
else
  {
    A -> B
    R -> A
  }
```

V ::

```
s := s + 1;
if not empty(B) then
  {
    A -> R
    B -> R
    R -> A
  }
```

- A – Active
- R – Ready
- B - Blocked

Problem kritičnog regiona


```
P ::          Q ::

s11;          s21;

cr1;          cr2;

s12;          s22;
```

Problem kritičnog regiona - rešenje

P ::

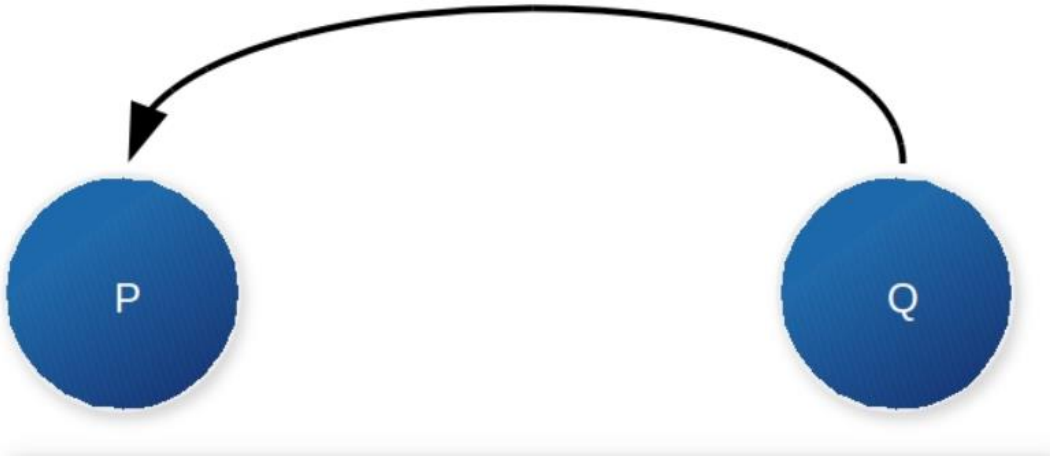
```
s11;
p(sem);
cr1;
v(sem);
s12;
```

Q ::

```
s21;
p(sem);
cr2;
v(sem);
s22;
```

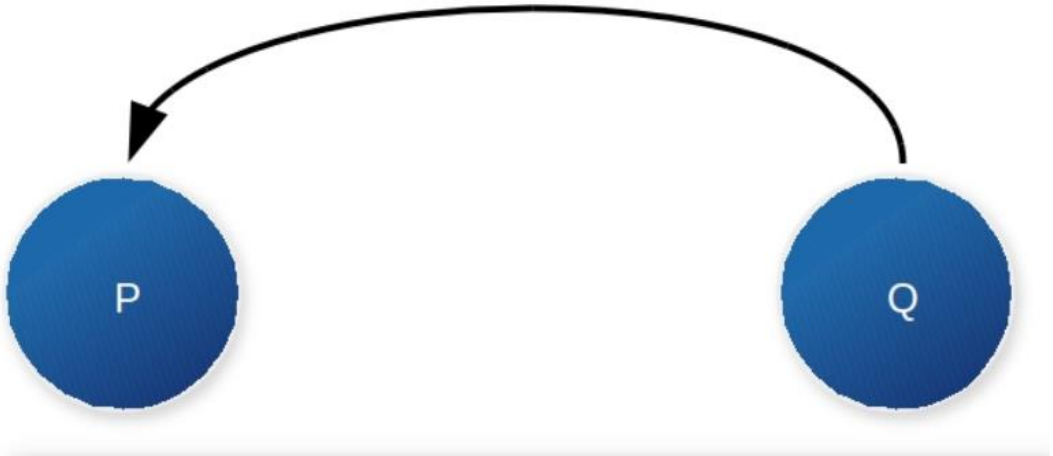
- Binarni semafor, $N = 1, s \in \{0, 1\}$
- Početno stanje: $s = 1$
- *sem* – ime semafora

Problem signalizacije



P ::	Q ::
s11;	s21;
Čekaj na signal iz Q;	Signaliziraj P da može da nastavi;
s12;	s22;

Problem signalizacije - rešenje



P ::

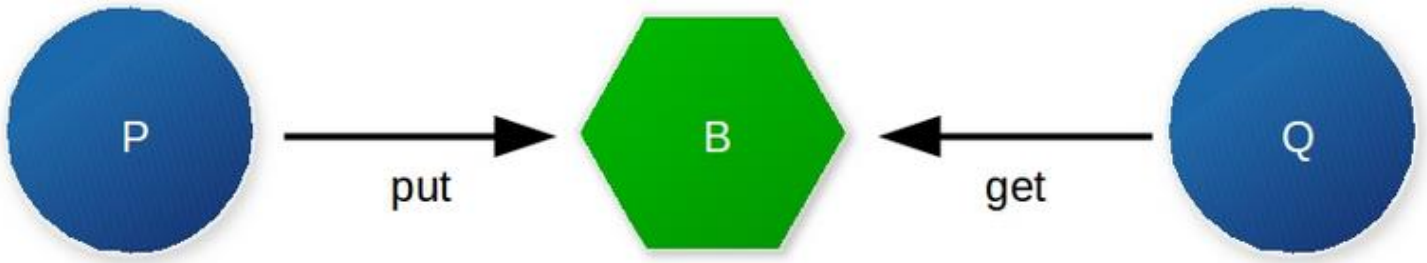
```
s11;
p(sem);
s12;
```

Q ::

```
s21;
v(sem);
s22;
```

- Binarni semafor, $N = 1, s \in \{0, 1\}$
- Početno stanje: $\boldsymbol{s = 0}$
- Ako je proces *P* prvi počeo sa radom izvršiće naredbu *s11* i preći će u blokirano stanje.
- Čekaće sve dok mu proces *Q* ne pošalje signal preko semafora *sem*.

Problem primopredaje podataka – jednostruki bafer



- P – proces Producer
- Q – proces Consumer
- B – bafer za razmenu podataka
- Sihronizacija proizvodnje i utroška

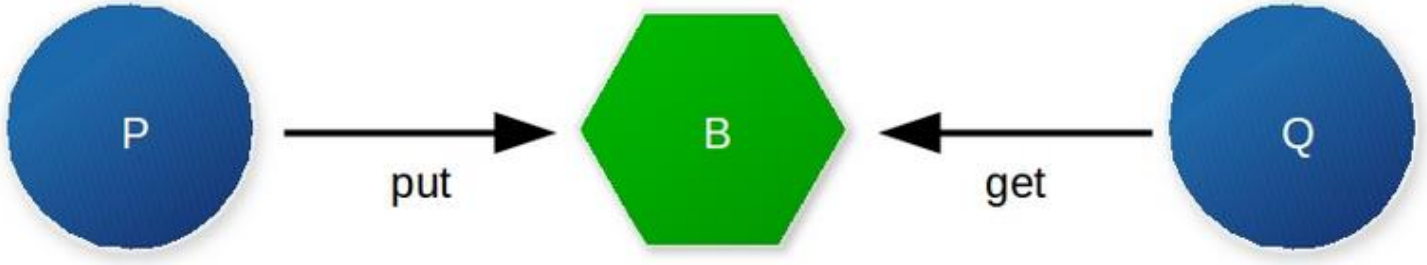
P ::

```
var x : slot;
while true do
begin
  proizvedi x;
  put(x);
end;
```

Q ::

```
var y: slot;
while true do
begin
  get(y);
  utrosi y;
end;
```

Problem primopredaje podataka – jednostruki bafer



P ::

```
var x : slot;
while true do
begin
  proizvodi x;
  p(sem1);
  put(x);
  v(sem2);
end;
```

Q ::

```
var y : slot;
while true do
begin
  p(sem2);
  get(y);
  v(sem1);
  utrosi y;
end;
```

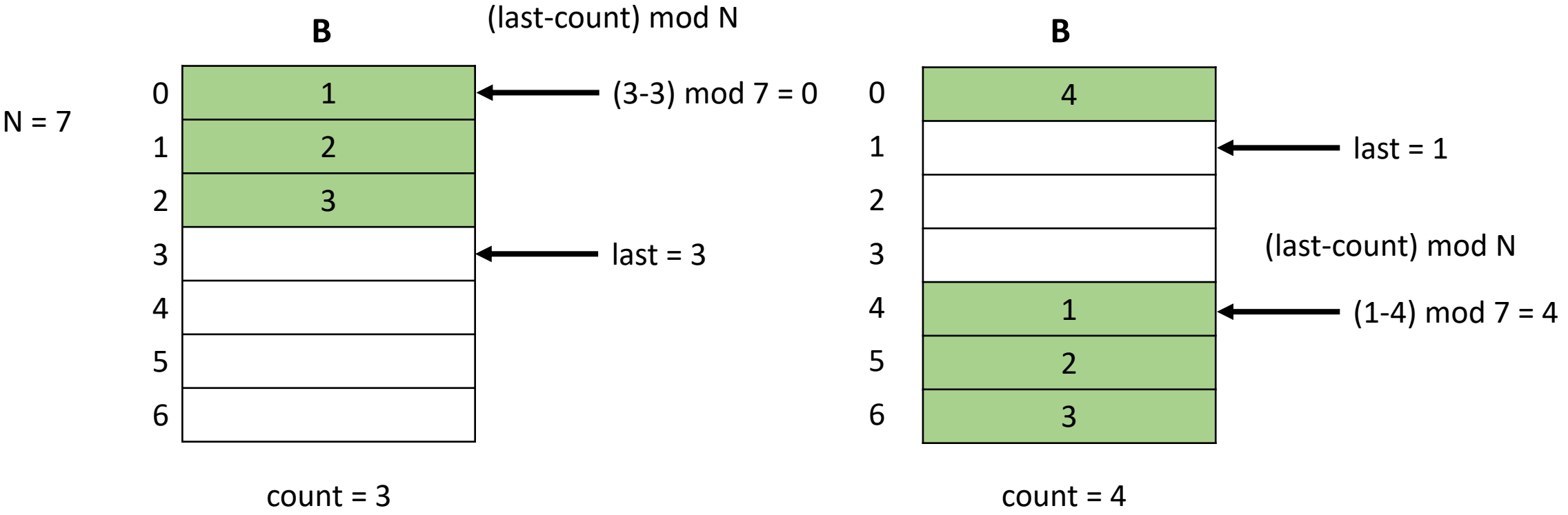
Rešenje

- Dva binarna semafora
- *sem1*
  - $s = 1$ – bafer prazan
  - $s = 0$ – bafer pun
  - blokira producer-a kada bafer nije ispražnjen
- *sem2*
  - $s = 1$ – bafer pun
  - $s = 0$ – bafer prazan
  - blokira consumer-a kada bafer nije napunjen

Početno stanje

- *sem1* - $s = 1$
- *sem2* - $s = 0$

Problem višestrukog (konačnog) bafera



- Proces P proizvodi podatke za proces Q
- Bafer se popunjava kružno

- Q treba da obavesti P da je ispraznio bafer
- P treba da obavesti Q da je napunio bafer

Konačni bafer - rešenje

P ::

```
var x : slot;
while true do
begin
  proizvedi x;
  p(space);
  p(mutex);
  put(x);
  v(mutex);
  v(count);
end;
```

Q ::

```
var y : slot;
while true do
begin
  p(count);
  p(mutex);
  get(y);
  v(mutex);
  v(space);
  utrosi y;
end;
```

- 3 semafora
  - *mutex* – binarni semafor za kontrolu pristupa baferu, proverava da li je kritični region zauzet
  - *space* – celobrojni semafor, broji prazna mesta u baferu
  - *count* – celobrojni semafor, broji zauzeta mesta u baferu
  - Kada vrednost varijable space padne na 0, bafer je pun, ne može da se piše u njega
  - Kada vrednost varijable count padne na 0, bafer je prazan, ne može da se čita iz njega
- Inicijalizacija
  - *mutex* -> 1
  - *space* -> N
  - *count* -> 0