Региони

Региони

- **Програмска парадигме за приступ критичној секцији**
- **Увођење посебне синтаксе за експлицитно означавање критичних секција**
- **Обезбеђивање међусобног искључивање процеса**
- **Условни критични регион је критични регион који поред обезбеђивања међусобног искључивања има и механизам за синхронизацију процеса преко (опционих) *await* наредби**

Региони

Декларација ресурса одређеног типа се обавља као и декларација сваке друге дељене променљиве:

res: **shared** *type*;

Условни критични регион се синтаксно дефинише на следећи начин (***await*** наредба је опциона и ако се изостави добијамо обичан критични регион):

region res **do**

begin

...

[**await**(condition);]

...

end;

Задаци

Условна синхронизација

Дат је упоредни програм на проширеном Pascal-у:

```
procedure makepoints;
var i: integer;
begin
        for i := 1 to n do
                x := i;
                y := i*i;
        end
end;
```

```
procedure printpoints;
var i: integer;
begin
        for i := 0 to n do
                write('(', x, ', ', y, ')');
        end
end;
```

Жељени излаз програма је низ парова облика:

(0,0) (1,1) (2,4) ... (n,n^2)

Отклонити временску зависност у датом програму употребом условних критичних области

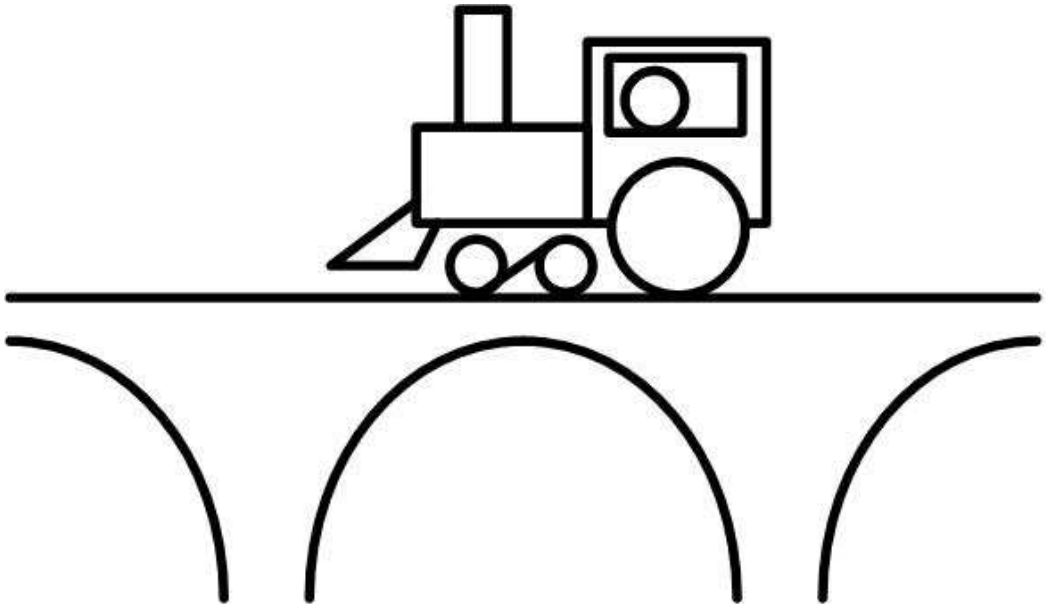
Условна синхронизација

```
program graph;
const n = ...;
type point = record
            x, y:integer;
             full: boolean
    end;
var p: shared point;

procedure makepoints;
var i: integer;
begin
    for i := 1 to n do
            region p do
                    begin
                            await(not p.full);
                            p.x := i;
                            p.y := i*i;
                            p.full := true
            end
end;
```

Условна синхронизација

```
procedure printpoints;
var i: integer;
begin
    for i := 0 to n do
        region p do
            begin
                await(p.full);
                write('(',p.x,',',p.y,')');
                p.full := false
            end
end;


begin
    p.x := 0; p.y := 0; p.full := true;
    cobegin
        makepoints;
        printpoints;
    coend
end.
```

One-lane bridge problem



One-lane bridge problem

Аутомобили који долазе са севера и југа морају да пређу реку преко моста. На мосту, на жалост, постоји само једна возна трака. Значи, у било ком тренутку мостом може да прође један или више аутомобила који долазе из истог смера (али не и из супротног смера). Написати алгоритам за аутомобил са севера и аутомобил са југа који долазе на мост, прелазе га и напуштају га са друге стране.

One-lane bridge problem

```
var most: shared record
            juzni, severni: integer
    end
            "u pocetku oba su nula"

"automobil sa juga"
begin
    region most do
            begin
                        await (severni = 0);
                        juzni := juzni + 1;
            end

    predji most;

    region most do
            juzni := juzni - 1;
end
```

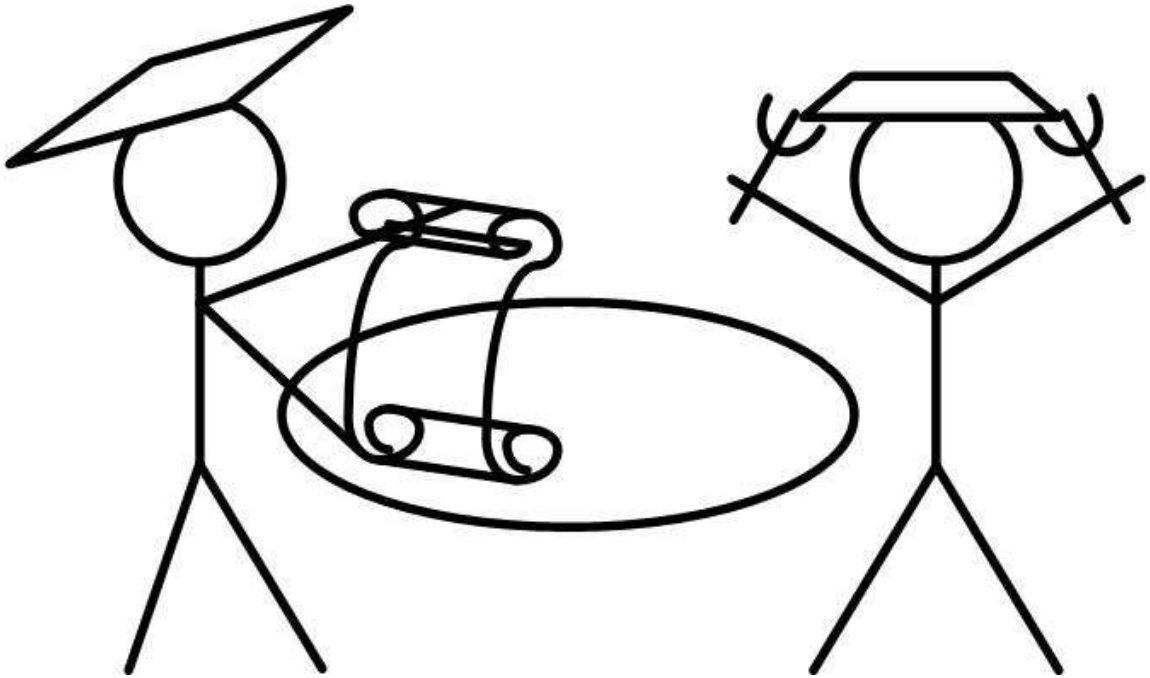
One-lane bridge problem

Усавршити решење претходног задатка тако да се смер саобраћаја мења сваки пут након што га пређе 10 аутомобила из истог смера, ако су за то време један или више аутомобила чекали да га пређу из супротног смера.

One-lane bridge problem

```
type smer =          record; cekaju, prelaze, ispred: integer; end
var most:            shared record juzni, severni: smer; end
"automobil sa juga"
begin
    region most do
        with juzni do
        begin
            cekaju := cekaju + 1;
            await (severni.prelaze = 0 AND ispred < 10);
            cekaju := cekaju - 1;
            prelaze := prelaze + 1;
            if (severni.cekaju > 0) then ispred := ispred + 1;
        end;
    predji most;
    region most do
        with juzni do
        begin
            prelaze := prelaze - 1;
            if (prelaze = 0) then severni.ispred := 0;
        end
end
```

Dining philosophers problem



Dining philosophers problem

Пет филозофа седи око стола. Сваки филозоф наизменично једе и размишља. Испред сваког филозофа је тањир шпагета. Када филозоф пожели да једе, он узима две виљушке које се налазе уз његов тањир. На столу, међутим, има само пет виљушки. Значи, филозоф може да једе само када ниједан од његових суседа не једе. Написати алгоритам за филозофа (0<=i<=4).

Dining philosophers problem - 1

```
var         viljuske: shared array [0..4] of 0..2;
procedure filozof (i:0..4);
var         levi, desni: 0..4;
begin
    levi := (i-1) mod 5;
    desni := (i+1) mod 5;
    repeat
            razmisljaj;
            region viljuske do
            begin
                    await (viljuske [i] = 2);
                    viljuske [levi] := viljuske [levi] - 1;
                    viljuske [desni] := viljuske [desni] - 1;
            end;
            jedi;
            region viljuske do
            begin
                    viljuske [levi] := viljuske [levi] + 1;
                    viljuske [desni] := viljuske [desni] + 1;
            end;
    forever;
end;
```

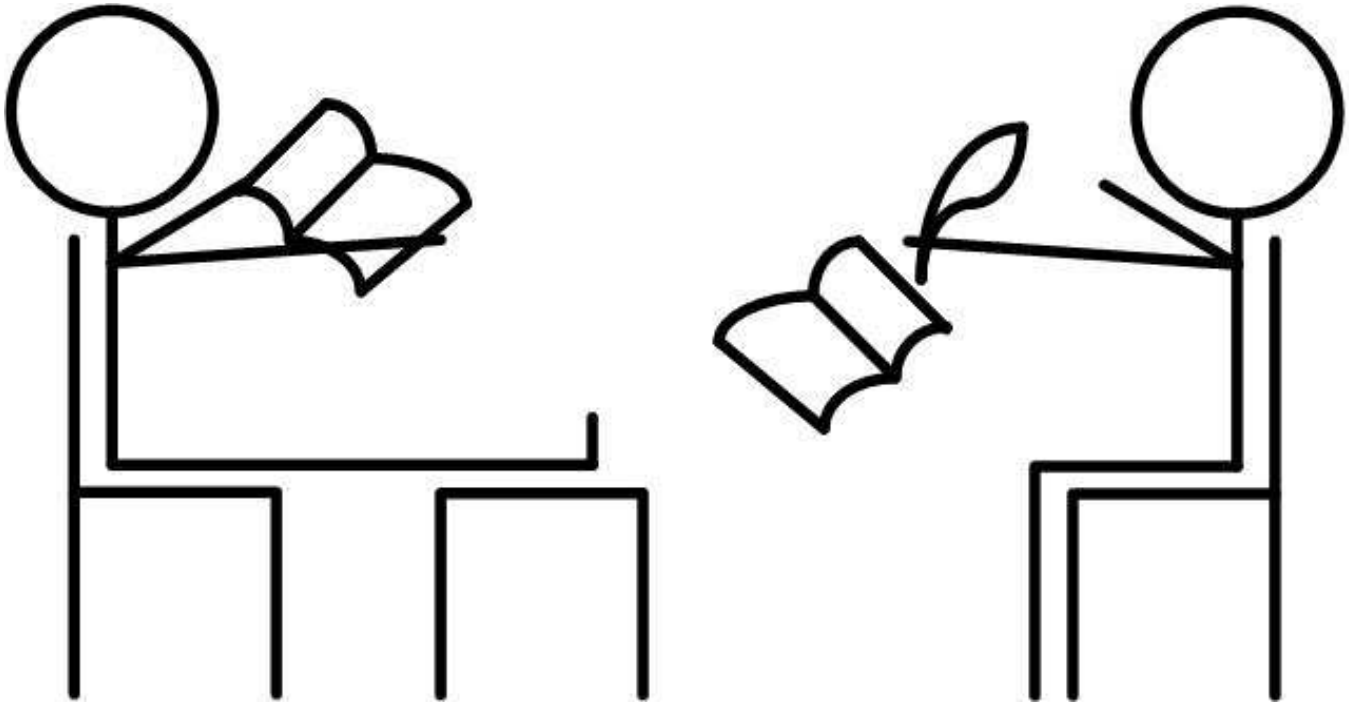
Dining philosophers problem - 2

```
var viljuska: array [0..4] of shared boolean;
"filozof i"
repeat
    razmisljaj;
    region viljuska [i] do
        region viljuska [(i+1) mod 5] do jedi;
forever
```

Dining philosophers problem - 3

```
var         razmisljanje: shared array [0..4] of boolean;
            "u pocetku sve je tacno“

"filozof i"
repeat
    razmisljaj;
    region razmisljanje do
    begin
            await (razmisljanje [(i-1) mod 5] AND razmisljanje [(i+1) mod 5]);
            razmisljanje [i] := false;
    end;
    jedi;
    region razmisljanje do razmisljanje [i] := true;
forever;
```

Readers – Writers problem



Readers – Writers problem

Група упоредих процеса који приступају заједничком средству састоји се од читалаца Ri, i = 1,... ,m, и писаца Wj, j = 1,...,n.

```
v: shared record r,w: integer end;
v1: shared integer;
begin
    v.r := 0; v.w := 0;
    cobegin R1; ... Rm; W1; ... Wn coend
end
```

За сва предложена решења одговорити да ли је:

- **Међусобно искључење осигурано.**
- **Могуће узајамно блокирање читалаца и писаца.**
- **Могуће узајамно блокирање писаца (при r=0).**
- **Могуће 'изгладњивање' читалаца.**
- **Могуће 'изгладњивање' писаца.**

Readers – Writers problem - 1

```
"Ri"
repeat
    region v do
    begin
        await (w = 0);
        r := r + 1
    end;
    read;
    region v do r := r - 1;
    nekritične operacije;
forever
"Wi"
repeat
    region v do
    begin
        w := w + 1;
        await (r = 0)
    end;
    write;
    region v do w := w - 1;
    nekritične operacije;
forever
```

Readers – Writers problem - 2

```
"Ri"
repeat
    region v do
    begin
                    await (w = 0);
                    r := r + 1
    end;
    read;
    region v do r := r - 1;
    nekritične operacije;
forever
"Wi"
repeat
    region v do
    begin
                    w := w + 1;
                    await ((r = 0) and (w = 1))
    end;
    write;
    region v do w := w - 1;
    nekritične operacije;
forever
```

Readers – Writers problem - 3

```
var v: shared record
            r,w: integer;
            rturn: boolean
    end;
    begin
    v.r := 0;
    v.w := 0;
    v.rturn := false;
    cobegin
            R1;
             ...
            Rm;
            W1;
             ...
            Wn
    coend
end;
```

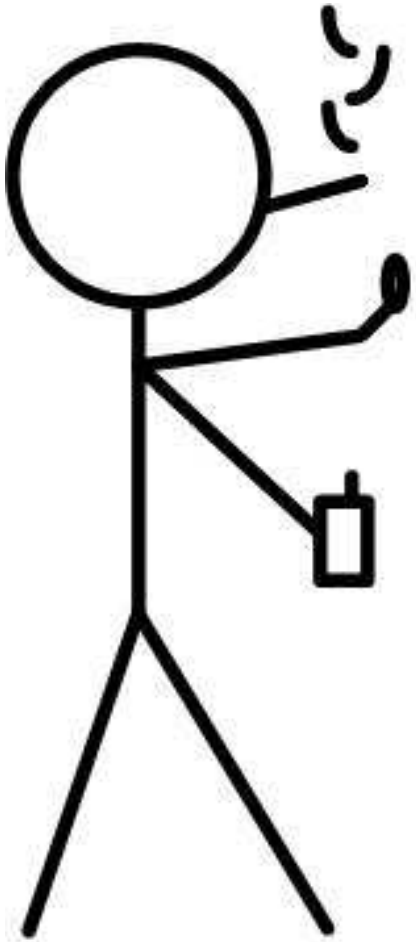
Readers – Writers problem - 3

```
"Ri"
repeat
    region v do
    begin
        if (rturn) then
            begin
                r := r + 1; await (w = 0)
            end
        else
            begin
                await (w = 0); r := r + 1
            end;
    end;
    read;
    region v do
    begin
            r := r - 1;  rturn := false
    end;
    nekritične operacije;
forever
```

Readers – Writers problem - 3

```
"Wi"
repeat
    region v do
    begin
        if (rturn) then
            begin
                await (r = 0); w := w + 1
            end
        else
            begin
                w := w + 1; await (r = 0)
            end;
    end;
    write;
    region v do
    begin
            w := w - 1; rturn := true
    end;
    nekritične operacije;
forever
```

Cigarette Smokers' problem



Cigarette Smokers' problem

Користећи условне критичне регионе написати програм који решава проблем и симулира систем "нервозних пушача" (*Cigarette Smokers' problem*). Постоји један агент и три нервозна пушача. Агент поседује резерве три неопходна предмета за лечење нервозе: папир, дуван и шибице. Један од пушача има бесконачне залихе папира, други – дувана, а трећи - шибица. Агент почиње тако што два различита предмета ставља на сто, један по један. Пушач, коме баш та два предмета фале, узима их, завија и пали цигарету и ужива. Након тога обавештава агента да је завршио, а агент онда ставља два нова предмета на сто, итд.

Cigarette Smokers' problem

```
program CigaretteSmokers(input, output);
type table = record
                    paper, tobacco, matches : boolean;
                    ok : boolean;
          end;
var p: shared table;
```

Cigarette Smokers' problem

```
procedure Agent;
var n : integer;
begin
    while (true) do begin
            n := RANDOM(0, 2);
            region p do
            begin
              case n of
                0: begin p.paper := false; p.tobacco := true;  p.matches := true;  end;
               1: begin p.paper := true;  p.tobacco := false; p.matches := true; end;
               2: begin p.paper := true; p.tobacco := true; p.matches := false; end;
               else ;
             await(p.ok);
             p.ok := false;
            end;
    end;
end;
```

Cigarette Smokers' problem

```
procedure smocker_with_Matches;
begin
    while (true) do
        region p do
        begin
            await(p.paper and p.tobacco);
            p.paper := false;
            p.tobacco := false;
            enjoy;
            p.ok := true;
        end;
    end;
end;
```

Cigarette Smokers' problem

```
procedure smocker_with_Tobacco;
begin
    while (true) do
    begin
        region p do
        begin
            await(p.paper and p.matches);
            p.paper := false;
            p.matches := false;
            enjoy;
            p.ok := true;
        end;
    end;
end;
```

Cigarette Smokers' problem

```
procedure smocker_with_Paper;
begin
    while (true) do
    begin
        region p do
        begin
            await(p.matches and p.tobacco);
            p.matches := false;
            p.tobacco := false;
            enjoy;
            p.ok := true;
        end;
    end;
end;
```

Cigarette Smokers' problem

```
begin
    p.paper := false;
    p.tobacco := false;
    p.matches := false;
    p.ok := false;
    cobegin
        Agent;
        smocker_with_Paper;
        smocker_with_Tobacco;
        smocker_with_Matches;
    coend;
end.
```

Питања?

Захарије Радивојевић
Електротехнички Факултет
Универзитет у Београду
zaki@etf.rs