AMIARD Landry
CROQUEFER Gaẽtan

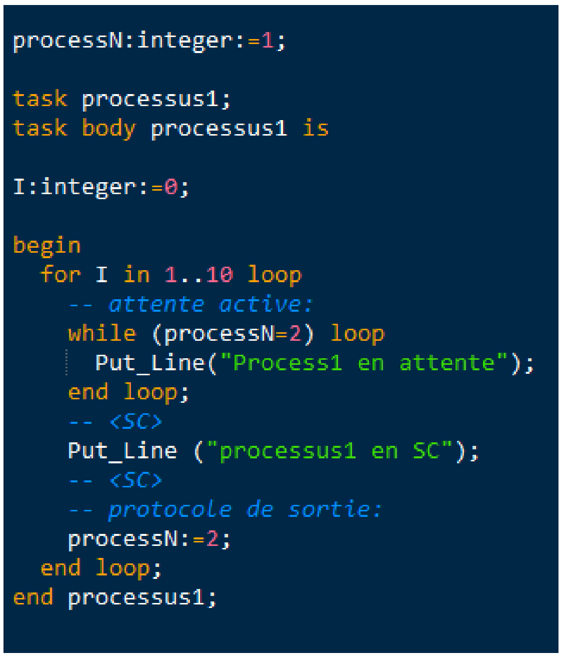
2018/2019

Rapport TP sur les Processus Concurrents

UE. Paradigmes de programmation

I – Algorithmes de Dekker et Peterson

Dekker 1 :

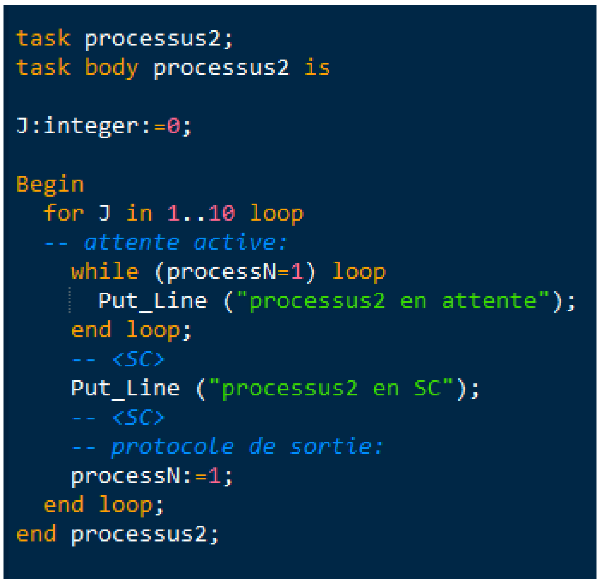


```
processN:integer:=1;

task processus1;
task body processus1 is

I:integer:=0;

begin
  for I in 1..10 loop
    -- attente active:
    while (processN=2) loop
      Put_Line("Process1 en attente");
    end loop;
    -- <SC>
    Put_Line ("processus1 en SC");
    -- <SC>
    -- protocole de sortie:
    processN:=2;
  end loop;
end processus1;
```

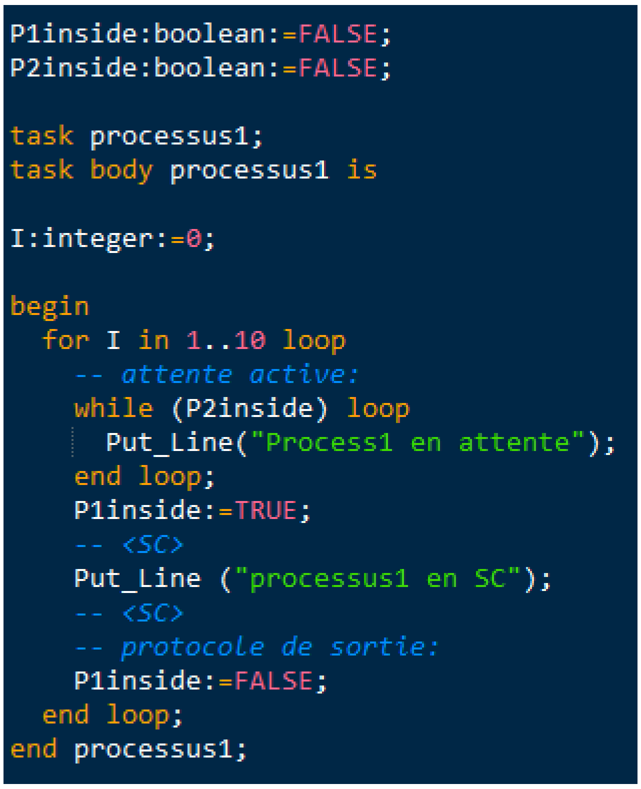


```
task processus2;
task body processus2 is

J:integer:=0;

Begin
  for J in 1..10 loop
  -- attente active:
    while (processN=1) loop
      Put_Line ("processus2 en attente");
    end loop;
    -- <SC>
    Put_Line ("processus2 en SC");
    -- <SC>
    -- protocole de sortie:
    processN:=1;
  end loop;
end processus2;
```

Dekker 2 :

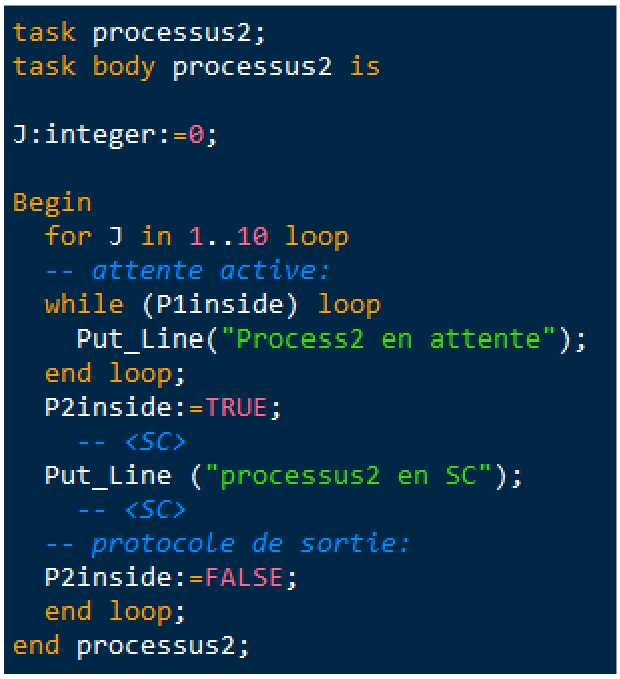


```
P1inside:boolean:=FALSE;
P2inside:boolean:=FALSE;

task processus1;
task body processus1 is

I:integer:=0;

begin
  for I in 1..10 loop
    -- attente active:
    while (P2inside) loop
      Put_Line("Process1 en attente");
    end loop;
    P1inside:=TRUE;
    -- <SC>
    Put_Line ("processus1 en SC");
    -- <SC>
    -- protocole de sortie:
    P1inside:=FALSE;
  end loop;
end processus1;
```

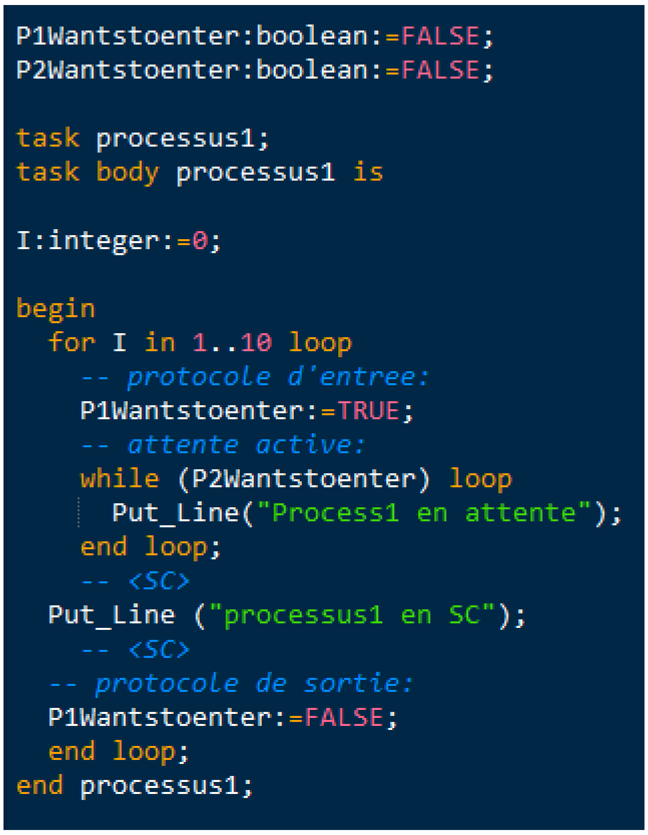


```
task processus2;
task body processus2 is

J:integer:=0;

Begin
  for J in 1..10 loop
  -- attente active:
  while (P1inside) loop
    Put_Line("Process2 en attente");
  end loop;
  P2inside:=TRUE;
    -- <SC>
  Put_Line ("processus2 en SC");
    -- <SC>
  -- protocole de sortie:
  P2inside:=FALSE;
  end loop;
end processus2;
```

Dekker 3 :

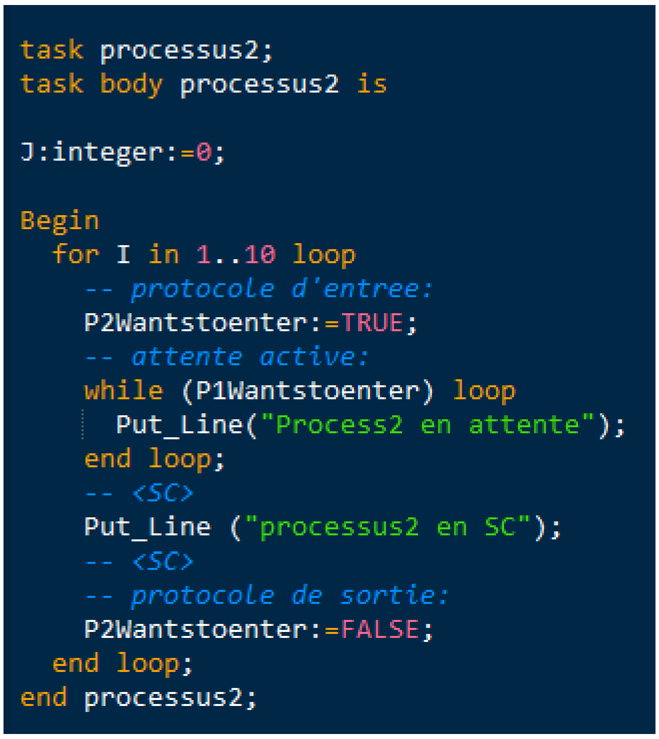


```
P1Wantstoenter:boolean:=FALSE;
P2Wantstoenter:boolean:=FALSE;

task processus1;
task body processus1 is

I:integer:=0;

begin
  for I in 1..10 loop
    -- protocole d'entree:
    P1Wantstoenter:=TRUE;
    -- attente active:
    while (P2Wantstoenter) loop
      Put_Line("Process1 en attente");
    end loop;
    -- <SC>
  Put_Line ("processus1 en SC");
    -- <SC>
  -- protocole de sortie:
  P1Wantstoenter:=FALSE;
  end loop;
end processus1;
```



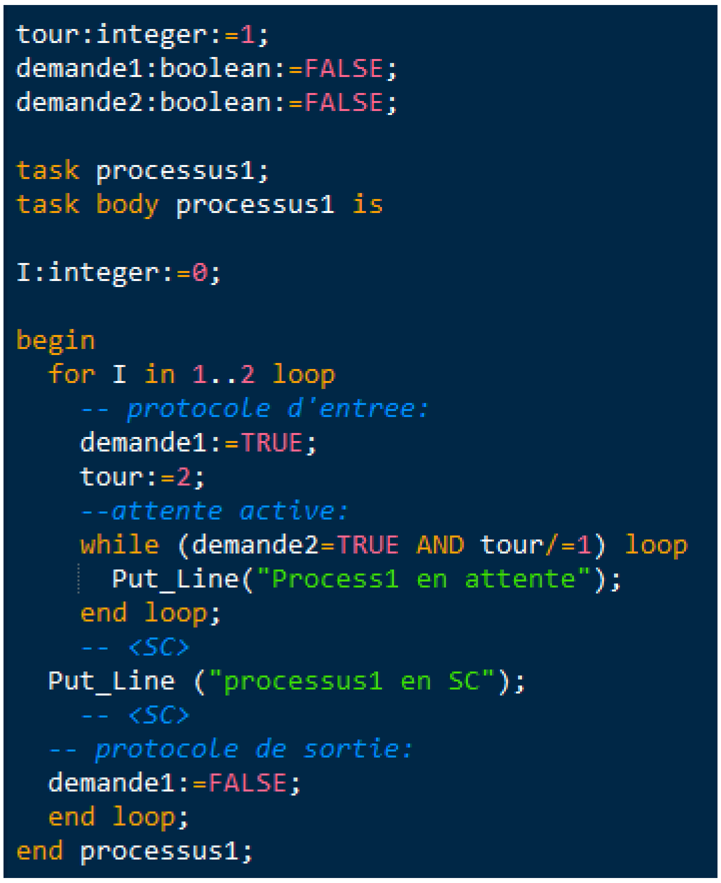
```
task processus2;
task body processus2 is

J:integer:=0;

Begin
  for I in 1..10 loop
    -- protocole d'entree:
    P2Wantstoenter:=TRUE;
    -- attente active:
    while (P1Wantstoenter) loop
      Put_Line("Process2 en attente");
    end loop;
    -- <SC>
    Put_Line ("processus2 en SC");
    -- <SC>
    -- protocole de sortie:
    P2Wantstoenter:=FALSE;
  end loop;
end processus2;
```

- Dekker 4 étant quasi équivalent à Dekker 3, nous ne l'avons pas traité-

Peterson :

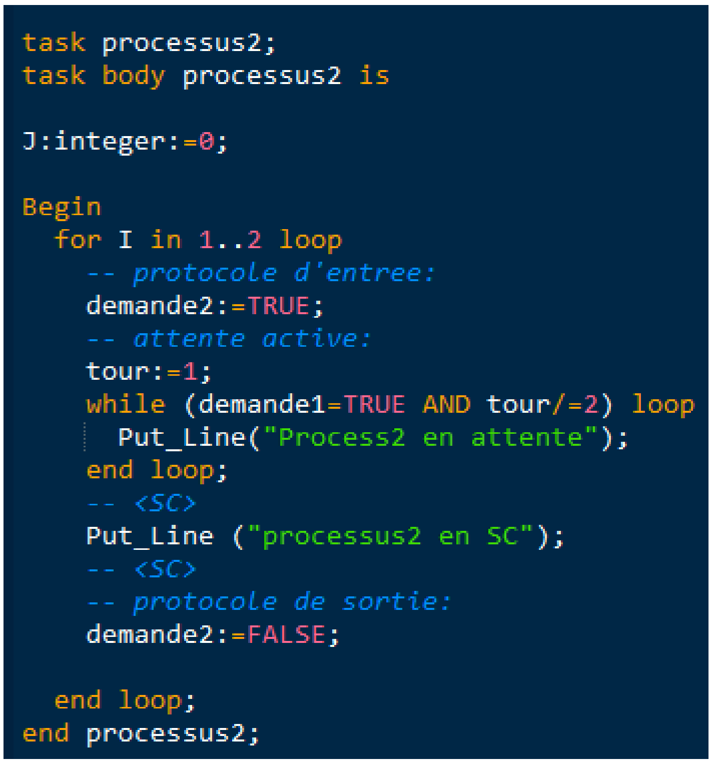


```
tour:integer:=1;
demande1:boolean:=FALSE;
demande2:boolean:=FALSE;

task processus1;
task body processus1 is

I:integer:=0;

begin
  for I in 1..2 loop
    -- protocole d'entree:
    demande1:=TRUE;
    tour:=2;
    --attente active:
    while (demande2=TRUE AND tour/=1) loop
      Put_Line("Process1 en attente");
    end loop;
    -- <SC>
  Put_Line ("processus1 en SC");
    -- <SC>
  -- protocole de sortie:
  demande1:=FALSE;
  end loop;
end processus1;
```



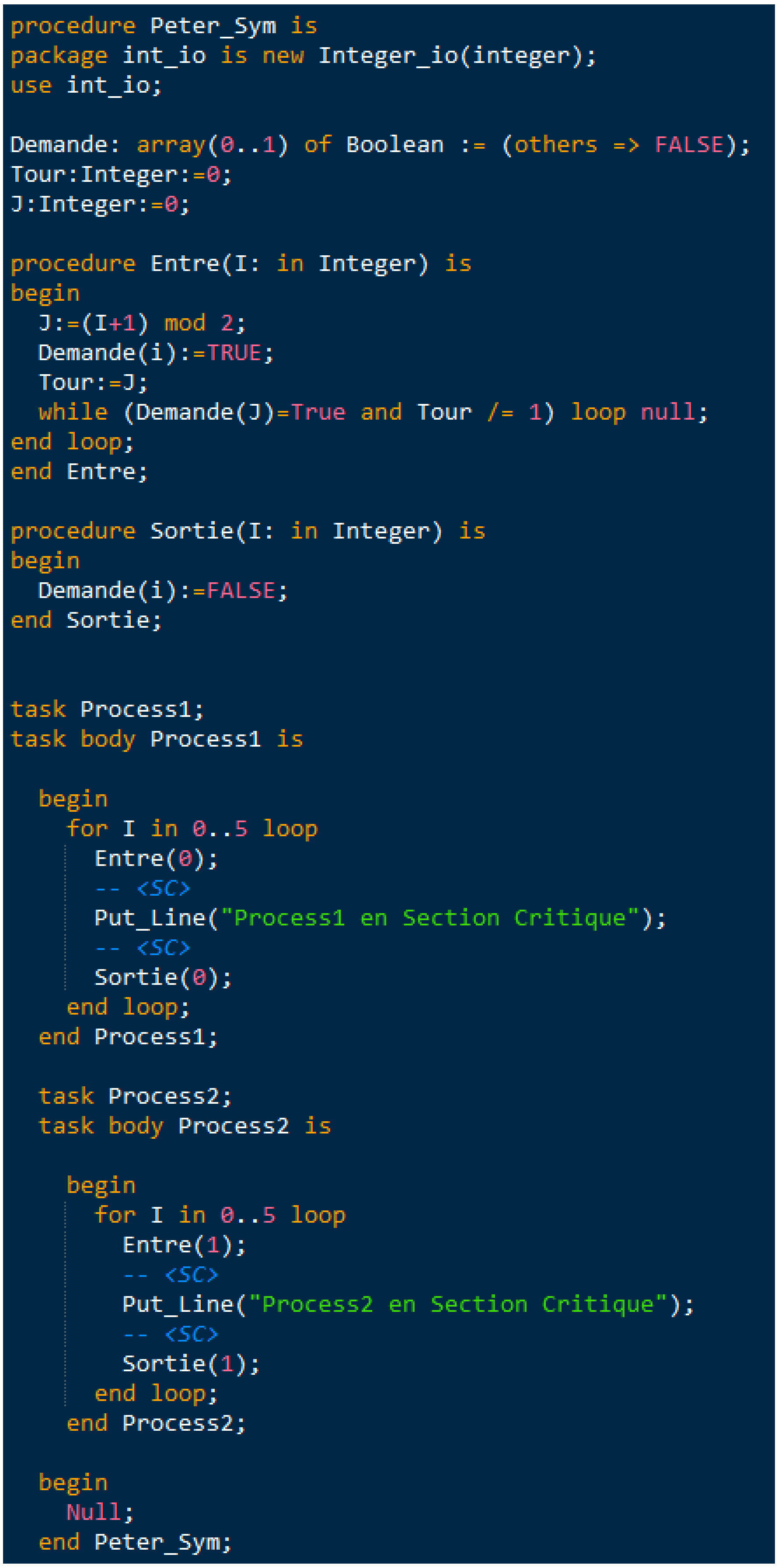
```
task processus2;
task body processus2 is

J:integer:=0;

Begin
  for I in 1..2 loop
    -- protocole d'entree:
    demande2:=TRUE;
    -- attente active:
    tour:=1;
    while (demande1=TRUE AND tour/=2) loop
      Put_Line("Process2 en attente");
    end loop;
    -- <SC>
    Put_Line ("processus2 en SC");
    -- <SC>
    -- protocole de sortie:
    demande2:=FALSE;

  end loop;
end processus2;
```

Peterson symétrique :



```
procedure Peter_Sym is
package int_io is new Integer_io(integer);
use int_io;

Demande: array(0..1) of Boolean := (others => FALSE);
Tour:Integer:=0;
J:Integer:=0;

procedure Entre(I: in Integer) is
begin
  J:=(I+1) mod 2;
  Demande(i):=TRUE;
  Tour:=J;
  while (Demande(J)=True and Tour /= 1) loop null;
end loop;
end Entre;

procedure Sortie(I: in Integer) is
begin
  Demande(i):=FALSE;
end Sortie;


task Process1;
task body Process1 is

  begin
    for I in 0..5 loop
      Entre(0);
      -- <SC>
      Put_Line("Process1 en Section Critique");
      -- <SC>
      Sortie(0);
    end loop;
  end Process1;

  task Process2;
  task body Process2 is

    begin
      for I in 0..5 loop
        Entre(1);
        -- <SC>
        Put_Line("Process2 en Section Critique");
        -- <SC>
        Sortie(1);
      end loop;
    end Process2;

  begin
    Null;
  end Peter_Sym;
```