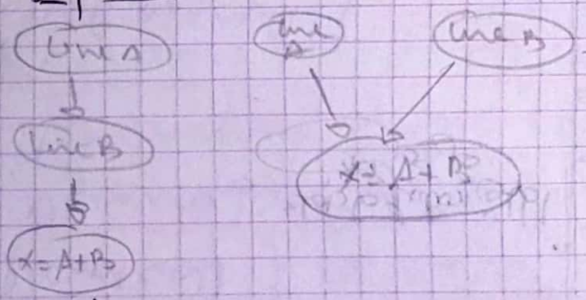
SE

Processeus : programme en cour exécution

tâche et thread :
⇓
ensemble entité traitement elementaire ayant coherence

Processeus
↙ ↘
Sequentiel Parallèl



thread :
defini comme un fils execution gerener par un process. (1)

(A) **Creation thread Java**
peut être crée en etendant classe Thread ou implementer runable (réecrire la methode run)

(B) **creation thread fork**

int id = fork.
↙ ↓ ↘
id=0 Fils
id>0 Père
id<0. Erreur.

Interaction processeus
↙ ↓ ↘

Independant [illegible]
P_1 ne partagent aucun donnée P_2

Coopérant [illegible]
patargent avec eux donnée

concurrent [illegible]
Limité ressource (compétitif)

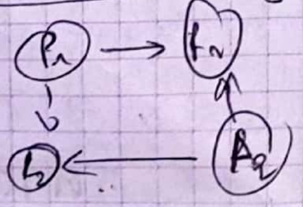
Section Critique et exclusion M

Probletime : Accés Concurrent ressource

SC : suite instruction ou ressoure critique est, accesible par un **seul**

(2) On dit la ressource est en exclusif et les process EM SC (sol EM)

interblockage



(2)

Realisation EM

Proprieté de Djikstro

(A) **Exclusion Mutuel** : ∀ t, nPcsc ≤ 1

(B) **Absence IB** : ∀ t, nPatt>0.

(C) **Progression** : [illegible]

(D) **Absence processus Privielège**

→ blokage P hors Sc ne doit pas empiecher autre entrer sc.

→ sol doit être même pour tous procers.

Sol Attente Passive

Semaphore (sol Sc).

Semaphore mutex = 1 (pour protéger variable).

P : demander jeton (e(s) = e(s) − 1)

V : liberer jeton (e(s) = e(s) + 1).

(4) (**restaurant TD3**)

(3)

I RDV :

```
  int cpt;
  Semaphore S = 0
  Semaphore mutex = 1
    const N = 10;
P(mutex);
 cpt ++;
if (cpt<N) }
  V(mutex);      أطلق futon مادامها
  P(S).          قل من 100
 {
else
 while (cpt<> 1) }
   V(S);         إذا أصبح 11، 12،
                 فانو نحلج
   cpt--;        نقص
 {               دوكي كل زايدين
```

4

II Lect-Red : (priorité lecteur).

~~DL():~~ 1 (Priorité Rédacteur)

```
  int nl = 0;
  int nlatt = nRatt = 0;
  bool E;
  Semaphore SL,
    *1, 1   SR;
```

Semaphore mutex = 1;

```
DL();
P(mutex)
if (E) }
  nlatt++;
  V(mutex);
 P(SL);
{
else
  nl++;
  V(mutex).
```

```
FL()
 P(mutex)
} nl--;

if (nl==0) }
  if (nRatt>0) }
  V(mutex)
  V(SK);
 }
{
V(mutex)
```

Lecture

```
DE();
P(mutex)
if(nl>0 E) }
 nRatt++;
 V(mutex)
 P(SR)
{
 else
  E = true;
  V(mutex);
```

Ecriture

```
FE() (voir Priorité)
 P(mutex)
  E = false;
  if (nLatt>0) }
     nRatt>0
    nlatt--;
      nl++;   E = True
      V(SL)
   {
 else
  if (nRatt>0) }
     nRatt>0
     nRatt--;
      E = true   nl--;
      V(SR)
   {
 else
 V(mutex)
```

5

lect-Red (fifo) : (chaîne Semaphore
1 (fifo))

```
Semaphore mutex, fifo = 1;
Semaphore S = 0;
pbloque = false;
```

2

```
DL():
P(fifo)
P(mutex)
if (E) }
 Pbloque = True
 V(mutex)
 P(S)
{P(mutex)
else nl++;
 V(mutex)
 V(fifo)
```

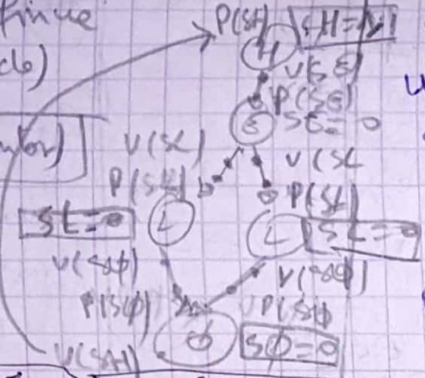
```
FL()
P(mutex)
nl--;
if (nl=0 &&(pbt=f)
 pbloque = false;
 V(S);
}
V(mutex)
```

```
DE();
V(fifo)
P(mutex)
if (E || nl>0) }
  pbloque = true
 V(mutex)
 P(S)
 P(mutex)
{
E = true
V(mute
V(fifo)
```

```
FE();
P(mutex)
E = False
if (PBloque=~~True~~)
P.bloque = False
V(S);
{
V(mutex);
```

6

V Genère Hello Infinie (boucle)



Réception (Moniteur)
tour

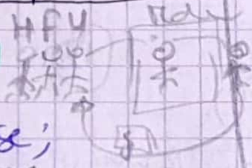
```
SH(){
P(SH);
write("H");
V(SE);
}
```

```
SE(){
 P(SE)
write("E");
 V(SL);
V(SL);
}
```

```
SL(){
P(SL);
P(SL);
write("L");
V(SO);
V(SO);
}
```

```
SO{
 P(SO)
  P(SO)
write("O");
 V(SH);
}
```

VIII médecin : U + F + H



```
bool medOcupé = false;
                 True
    Semaphone SU = 0;
        ''    SF = 0;
        ''    SH = 0;
        ''    mutex = 1;
    Uatt = Fatt = Hatt = 0;
```

O=gestion

EU():   EF()   EH()

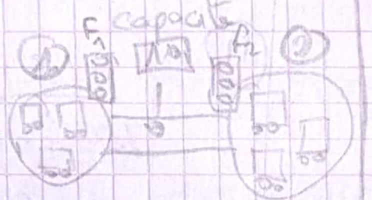
```
  P(mutex)
if(med-Ocupé = True){
               False
  Uatt ++;
  V(mutex);
   P(SU);
}
else
   medOcupé = False
              True
       V(mutex);
```

7

SU() = SF() = SH()

```
P(mutex);
Uatt --;
 if(Uatt>0){
   Uatt --;
   V(SU);
 }
else
 if(nFatt>0){
   Fatt --;
   V(SF);
 }
else if(nHatt>0){
   nHatt --;
     V(SH);
 }
else
  medOupé = false;
              true
  V(mutex)
```

Pont a voie Unique



DT1() : DT2()

```
P(mutex)
 if(fen1 = 'R' | nbVsurpont > N)
   }
 nbAtt1++;
   V(mutex)
   P(DT1);
}
 else{if (nbVsurpont = = 0)
       fen2 = R;
```

Variable

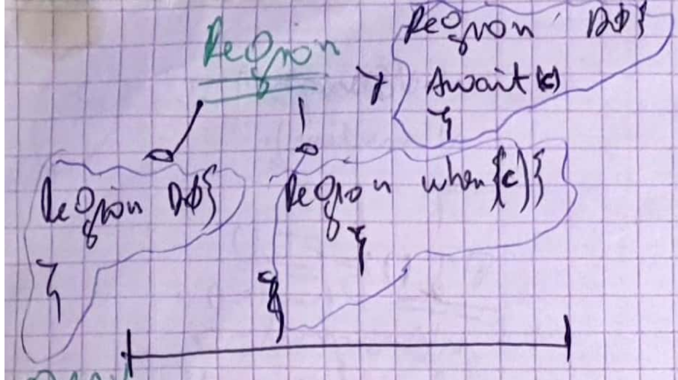
```
fen1 = 'V';
fen2 = 'V';
int nbVsurpont = 0
  semaphone DT1 = 0
      ''    DT2 = 0
      ''    mutex = 1
  const N = N0;
int nbAtt1, nbAtt2 = 0.
```

8

```
nbVsurpont ++;
  V(mutex);
  }
```

```
FT1(); = FT2()
        P(mutex);
nbVsurpont --;
 if (nAtt1 > 0){
   nAtt1 --;
   Vbsurpont ++;
  V(DAtt1);
 }
else if (nbVsurpont == 0)
   fen2 = 'V'
    if (nAtt2 > 0)} fen1 = 'R')
   int k = min (Att2, N);
   while (nbVsurpont ≤ k)
     nAtt2 --;
     nbVsurpont ++;
     V(SAtt2)
   }
 }
V(mutex)
```

9



I RDV

```
int shared cpt=0
Region cpt DO{
    cpt++;
  Await(cpt==N);
}
```

(11)

II lect Red

```
Struct lectRed{
        int nl,nlatt,nRatt
         bool E;
        }

Shared int LR.{0,0,0, false}
  (LR.nl=0    | LR.nRatt=0
   LR.nlatt=0 \ LR.E=false)
```

(21)

```
D(L) :
Region LR where (!E) do
{
  nl++;
}
```

Lecture

```
F(L)
Region LR do
{
 nl--;
}
```

```
D(E) :
Region LR where
(!E && nl==0)
{ E=true}
```

Ecriture

```
F(E) :
Region LR do{
   E=false
}
```

(3) Priorite lecture

(12)

```
D(L)
Region LR DO{
 ~~[illegible]~~ nlatt++;
 ~~A~~ Wait(E=false)
 nlatt--;
 nl++;
}
```

Lecture

```
F(L)
Region LR do{
   nl--;
}
```

```
D(E)
Region LR whe(E=false) &&
(nl=0) && (nlatt=0)
{
 E=true
}
```

Ecriture

```
Region LR DO{
    E=false
```

med-Ocupée = True.

III Cabinet Medical

```
Struct CabMed{
  bool medOcupé;
  int nUatt=nFatt=
         nHatt;
}
Shared CabMed CB
   CB.MedOcupe=false
   CB.nUatt=nFat=nHatt
     =0;
```

EU() = EF() = EH()

(13)

```
Region CB do
 CB.nUatt++;
Await(CB.medOcupé=false)
 CB.nUatt--;
  CB.medOcupé=True;
```

Consulter

FU() = FF() = FH()

```
Region CBDO{
 CB.MedOcupé=false
 }
Region CB DO{ nbHatt++;
Await(Mede Ocupe=false) && (nU=0)
 (nFatt=0)(nH att=0)
```