Corrigé du TD4 de Systèmes d'exploitation
La synchronisation

Exercice4 : Producteur-Consommateurs

b)

```
type Exercice ;
Exercices  tableau[0..M-1] de Exercice;
Cpt   tableau[0..M-1] de entier init (0);
tserie  tableau[0..M-1] de entier init (1) ;
Mutex  tableau[0..M-1] de Sémaphore init(1);

Sémaphore   plein init(0);
Sémaphore   vide init(M);
Sémaphore   fins init(0);
```

Programme Enseignant	**Programme Etudiant$_i$**
Entier i init(0); Exercice exo; **Début** Répéter *Rédiger_exercice(exo);* P(vide) ; Exercices[i] := exo; i := (i + 1) mod M; V(plein) ; Jusqu'à fin; **Fin du Programme enseignant;**	Entier j init(0), serie init(1); Exercice exo; **Début** Répéter P(Mutex[j]); Si tserie[j]< serie alors V(mutex[j]) ; P(fins) ; V(fins) ; P(mutex[j]) ; finsi Si Cpt[j] = 0 alors P(plein); /*premier*/ exo := Exercices[j]; Cpt[j] :=(Cpt[j]+1) mod N; Si Cpt[j] = 0 alors /*dernier*/ V(vide); tserie[j] := tserie[j]+1 ; V(Mutex[j]); V(fins) ; P(fins) ; Sinon

	V(Mutex[j]); Finsi j := (j + 1) mod M; Si j=0 alors serie := serie+ 1; *Faire_exercice(exo);* Jusqu'à fin; **Fin du Programme étudiant$_i$;**

Exercice5 : Le modèle des lecteurs/rédacteurs (Courtois et al, 1971)

1) Priorité aux lecteurs

Pour donner la priorité aux lecteurs on utilise un sémaphore r initialisé à 1 ; ce sémaphore est utilisé, seulement, par les rédacteurs.

Sémaphore f init (1) ; Entier nl init(0); Sémaphore mutex init (1); Sémaphore r init (1) ;	
Lecteurs	Rédacteurs

Début
/* demande de lecture */
p(mutex);
nl=nl+1;
si nl=1 alors p(f);
v(mutex);
< *lire* >
/* fin de lecture */
p(mutex);
nl=nl-1;
si nl=0 alors v(f);
v(mutex);
fin;
	Début
	/* demande d'écriture */
	p(r)
	p(f);
	<*écrire*>
	/* fin d'écriture */
	v(f);
	v(r) ;
	fin;

2) Priorité aux Rédacteurs

Pour donner la priorité aux rédacteurs on utilise deux autres sémaphores lect et lr initialisés à 1 ;

- lec : utilisé par les lecteurs et permet de faire passer un seul lecteur à la fois ;
- lr : utilisé par les lecteurs et les rédacteurs, le premier rédacteur « d'une série » bloque les lecteurs qui arrivent après lui(c'est à dire si le fichier est en cours de lecture, les lecteurs ne peuvent plus y accéder jusqu'à la fin de toutes les écritures).

On utilise une variable nr qui compte le nombre d'écritures en cours ou en attente. Cette variable n'est utilisée que par les rédacteurs. La variable nr doit être utilisée en exclusion mutuelle; Soit sr un sémaphore, initialisé à 1, qui protégera cette variable.

Sémaphore f init (1) ;
Sémaphore lr init (1) ;
Sémaphore lec init (1) ;
Entier nl init(0); Sémaphore mutex init (1);
entier nr init(0); Sémaphore sr init (1);

Lecteurs	Rédacteurs

```
Début
/* demande de lecture */
  p(lec) ;
     p(lr) ;
        p(mutex);
           nl=nl+1;
           si nl=1 alors p(f);
        v(mutex);
     v(lr) ;
  v(lec) ;
  < lire >
  /* fin de lecture */
  p(mutex);
     nl=nl-1;
     si nl=0 alors v(f);
  v(mutex);
fin;
```

```
Début
 /* demande d'écriture */
    p(sr);
       nr=nr+1;
       si nr=1 alors p(lr);
    v(sr)
    p(f);
       <écrire>
       /* fin d'écriture */
    v(f);
fin;
```

Exercice7 : Rendez-vous par rapport à aux identificateurs des processus

a) **Programmes des 4 processus**

On aura besoin de 4 sémaphores de synchronisation initialisés à zéro.

Sémaphore s0, s1, s2, s3 init (0);

P0	P1	P2	P3
début Opération0 ; V(s1); P(s0); V(s1); Suite0; Fin;	Début P(s1); Opération1 ; V(s2); P(s1); V(s2); Suite1; Fin;	Début P(s2); Opération2 ; V(s3); P(s2); Suite2; Fin;	début P(s3); Opération3 ; V(s0); Suite1; Fin;

b) Programme d'un processus$_i$

On aura besoin de :

- un tableau de **n** sémaphores de synchronisation initialisés à zéro sauf le sémaphore t[0] est initialisé à 1,
- **un** sémaphore d'exclusion mutuelle initialisé à 1 et d'une variable entière initialisée à 0;

Première solution	**Deuxième solution (sans cpt , mutex et s)**
Sémaphore t[0] init 1; Sémaphore t tableau[1..n-1] init(0); Sémaphore s init(0) ; Sémaphore mutex init (1); Entier Cpt init 0;	Sémaphore t[0] init 1; Sémaphore t tableau[1..n-1] init(0);
Processus$_i$ Début P(t[i]); Opérations i; V(t[(i+1) mod n]); /* Début du Rendez-vous */ P(mutex); Cpt++; Si cpt < n alors V(mutex); P(s); V(s); Sinon Cpt=0; V(s); P(s); Finsi; /* Fin du Rendez-vous */ Suitei; Fin;	**Processus$_i$** Début P(t[i]); Opérations i; V(t[(i+1) mod n]); /* Début du Rendez-vous */ P(t[i]); V(t[(i+1) mod n]); /* Fin du Rendez-vous */ Suitei; Fin;

Exercice8 : Crédit bancaire

La procédure *demander_prêt* a comme paramètres le Numéro de compte bancaire du client(*Ncpt_client*) et le montant demandé(*montantd*). On peut considérer le compte bancaire comme identificateur du client.

La procédure *rembourser_prêt* a comme paramètres le numéro de compte bancaire du client (*Ncpt_client*) et le montant à rembourser (*montantr*).

Ces deux procédures doivent être exécutées en exclusion mutuelle, car le compte de la banque (*Ncpt_banque*) ne peut pas être utilisé en même temps par plusieurs procédures *demander_prêt* ou/et *rembourser_prêt*; Soit mutex le sémaphore protégeant ces procédures ; mutex est initialisé à 1.

On utilise une liste pour sauvegarder le numéro de compte du client (*Ncpt_client*), le montant demandé (*montantd*).

On utilise un sémaphore (*sem*) de synchronisation pour tous les clients (processus) ; Ce sémaphore permet de mettre les clients en attente. Ce sémaphore est initialisé à zéro.
On utilise une procédure *enfiler* qui permet d'ajouter un élément dans la liste.

```
Programme d'un client_i
Entier ncpt ;
Réel montant ;
Début
        Ncpt := numéro de compte du client ;
        Montant := montant demandé ;
        Demande_prêt(ncpt, montant) ;
                [ utilisation du montant demandé]
        Rembourser_prêt(ncpt, montant) ;
Fin ;

Liste   Structure
    Num_Cpt : entier;
    Montant : réel ;
    Suivant : *liste ;
fin;
pliste pointeur  sur liste init null;

procédure enfiler(pliste, Ncpt_client, montantd)
```

fonction tête(pliste) : renvoie le montant contenu dans le premier élément de la liste

procédure retirer(pliste, sém, compte, montant) : Affecte le contenu le l'élément pointé par pliste dans sem, compte, montant et supprime cet élément de la liste.

```
 Entier  Ncpt_banque ;
```

Sémaphore sem init 0 ;
Sémaphore mutex init 1;

Demander_prêt(Ncpt_client, montantd)	**Rembourser_prêt(Ncpt_client, montantr)**

Réel mt_banque, mt_client;
Début
P(mutex);
Si pliste # null alors /* ou pliste # vide */
/* enfiler la demande et attendre */
Enfiler(pliste, Ncpt_client, montantd);
V(mutex);
P(sem);
Sinon
Dlire (fichier, Ncpt_banque, mt_banque);
Si montantd > mt_banque alors
/* enfiler la demande et attendre */
Enfiler(pliste, Ncpt_client, montantd);
V(mutex);
P(sem);
Sinon
mt_banque = mt_banque - montantd;
Decrire (fichier,Ncpt_banque,mt_banque)
Dlire (fichier, Ncpt_client, mt_client);
mt_client = mt_client + montantd;
Decrire (fichier, Ncpt_client, mt_client) ;
V(mutex);
Finsi
Finsi
Fin;
	Entier Ncpt ;
	Réel mtd, mt_client, mt_banque;
	Début
	P(mutex);
	Dlire (fichier, Ncpt_client, mt_client);
	Mt_client = mt_client - montantr;
	Decrire(fichier, Ncpt_client, mt_client);
	Dlire (fichier, Ncpt_banque, mt_banque)
	mt_banque = mt_banque + montantr
	/*Safistaire les demandes en attente */
	tantque pliste $\neq$ null et tête $\leq$ mt_banque
	faire
	Retirer(pliste, Ncpt, mtd) ;
	Dlire(fichier,Ncpt, mt_client) ;
	mt_client = mt_client + mtd;
	mt_banque=mt_banque - mtd;
	Decrire(fichier, Ncpt, mt_client);
	V(sem) ;
	fintanque
	Decrire(fichier, Ncpt_banque, mt_banque)
	V(mutex);
	Fin;

Remarque : certains contrôles n'ont pas été effectués sur montantd et montantr pour ne pas encombrer les procédures.

Exercice9 : Salon de coiffure(un seul coiffeur)

Processus : Clients Hommes et Dames
Ressources : les Fauteuils et le Coiffeur.
Programmes de clientes dames et des clients hommes.

Entier nf,attente init 0;
Sémaphore snf, sattente, scoiffeur, sfh,sh init 1;

Clientes dames	Clients hommes

Début
P(sattente)
Si attente = N alors
V(sattennte);
Sinon
attente ++;
V(sattente);
P(snf);
nf ++;
Si nf = 1 alors P(sfh);
V(snf);
P(coiffeur);
P(sattente);
attente--;
V(sattente);
Se coiffer
V(coiffeur);
P(snf);
Nf--;
Si nf = 0 alors v(sfh);
V(snf);
Finsi;
Fin;
	Début
	P(sattente)
	Si attente = N alors
	v(sattennte);
	Sinon
	attente ++;
	V(sattente);
	P(sh)
	P(sfh);
	P(coiffeur);
	P(sattente);
	attente--;
	V(sattente);
	Se coiffer
	V(coiffeur);
	V(sfh);
	V(sh);
	Finsi;
	Fin;

Exercice10 : Pont à voie unique

Solution

Cas 1) : Accès selon l'ordre d'arrivée(FIFO)

Déclaration des variables et des sémaphores

Sémaphore st init (k); Sémaphore sfifo, feu init(1);
Sémaphore mutex1, mutex2 init(1); Entier sens1, sens2 init (0) ;

Automobiliste du Sens1	Automobiliste du Sens2
Début P(sfifo) ; P(mutex1); Sens1:=sens1+1; Si sens1=1 alors p(feu); V(mutex1) ; V(sfifo) ; P(st) ; " *Traverser sens1* ← " V(st) ; P(mutex1) ; Sens1 :=sens1-1 ; Si sens1=0 alors v(feu) ; V(mutex1) ; Fin ;	Début P(sfifo) ; P(mutex2); Sens2:=sens2+1; Si sens2=1 alors p(feu); V(mutex2) ; V(sfifo) ; P(st) ; " *Traverser sens2* → " V(st) ; P(mutex2) ; Sens2 :=sens2-1 ; Si sens2=0 alors v(feu) ; V(mutex2) ; Fin ;

Cas2 : Véhicules prioritaires

Déclaration des variables et des sémaphores

Sémaphore st init (k);
Sémaphore sfifo, feu init(1);
Sémaphore mutex1, mutex2 init(1); Entier sens1, sens2 init (0) ;

Sémaphore mutexp init (1) ; Entier vp init (0) ;
Sémaphore svp init(0);
Sémaphore sfifop init(1);

Sens1 : véhicules non prioritaires	**Sens2 : véhicules non prioritaires**
Début P(sfifo) ; P(mutexp) ; Si vp>0 alors V(mutexp) ; P(svp) ; Sinon V(mutexp) ; Finsi P(mutex1) ; Sens1 :=sens1-1 ; Si sens1=1 alors p(feu) ; V(mutex1) ; V(sfifo) ; P(st) ; " *Traverser sens1* ← " V(st) ; P(mutex1) ; Sens1 :=sens1-1 ; Si sens1=0 alors v(feu) ; V(mutex1) ; Fin ;	Début P(sfifo) ; P(mutexp) ; Si vp>0 alors V(mutexp) ; P(svp) ; Sinon V(mutexp) ; Finsi P(mutex2) ; Sens2 :=sens2-1 ; Si sens2=1 alors p(feu) ; V(mutex2) ; V(sfifo) ; P(st) ; " *Traverser sens2* → " V(st) ; P(mutex2) ; Sens2 :=sens2-1 ; Si sens2=0 alors v(feu) ; V(mutex2) ; Fin ;

Sens1 : véhicules prioritaires	**Sens2 : véhicules prioritaires**
Début P(mutexp) ; vp :=vp+1 ; V(mutexp) ; P(sfifop) ; P(mutex1) ; Sens1 :=sens1+1 ; Si sens1=1 alors p(feu) ; V(mutex1) ; V(sfifop) ; P(st) ; " *Traverser sens1* ← " V(st) ; P(mutex1) ; Sens1 :=sens1-1 ; Si sens1=0 alors v(feu) ; V(mutex1) ; P(mutexp) ; Vp :=vp-1 ; Si vp=0 alors v(svp) ; V(mutexp) ; Fin ;	Début P(mutexp) ; vp :=vp+1 ; V(mutexp) ; P(sfifop) ; P(mutex2) ; Sens2 :=sens2+1 ; Si sens2=1 alors p(feu) ; V(mutex2) ; V(sfifop) ; P(st) ; " *Traverser sens2* → " V(st) ; P(mutex2) ; Sens2 :=sens2-1 ; Si sens2=0 alors v(feu) ; V(mutex2) ; P(mutexp) ; Vp :=vp-1 ; Si vp=0 alors v(svp) ; V(mutexp) ; Fin ;

Transport universitaire :

On dispose d'un bus de N places pour le transport des étudiants. Ce bus fait la navette entre la cité universitaire, lieu de résidence des étudiants, et l'université. Les étudiants ne peuvent commencer à monter dans le bus que s'il est vide (pendant que des étudiants descendent du bus, aucun étudiant ne pourra monter dans le bus). Un seul étudiant peut monter à la fois.

Le conducteur ne peut démarrer son bus que s'il est plein. En utilisant les sémaphores, écrire le programme d'un étudiant et le programme du conducteur du bus.

Solution :

Variables utilisées : **Sup** : semaphore init 0 /* mettre en attente les étudiants qui montent dans le bus*/ **Bus_plein** : semaphore init 0 /* signaler au conducteur que le bus est plein*/ **mutex** : semaphore init 1 **i,j,k** : integer init 0 /* compter les étudiants qui arrivent, qui montent ou qui descendent */ **up** : boolean init true /* les étudiants

Etudiant Var **r** : integer
/*Monter */
P(mutex)
i++
if (not(up)) then
V(mutex)
P(Sup)
Else
i –
V(mutex)
Endif
P(mutex)
j++
<monter>
if j=N then
if (up) then
up :=false
endif
j :=0
V(mutex)
V(bus_plein)
<navette>
Else
V(mutex)
Endif
/* descendre */

```
 P(mutex)
k++
<descendre>
 if k=N then
 r :=0
while (i>0 and r ≤N) do
 i—
r++
V(Sup)
Done
 if r<N then
 up :=true
endif
 k :=0
V(mutex)
V(mutex)
Endif
```