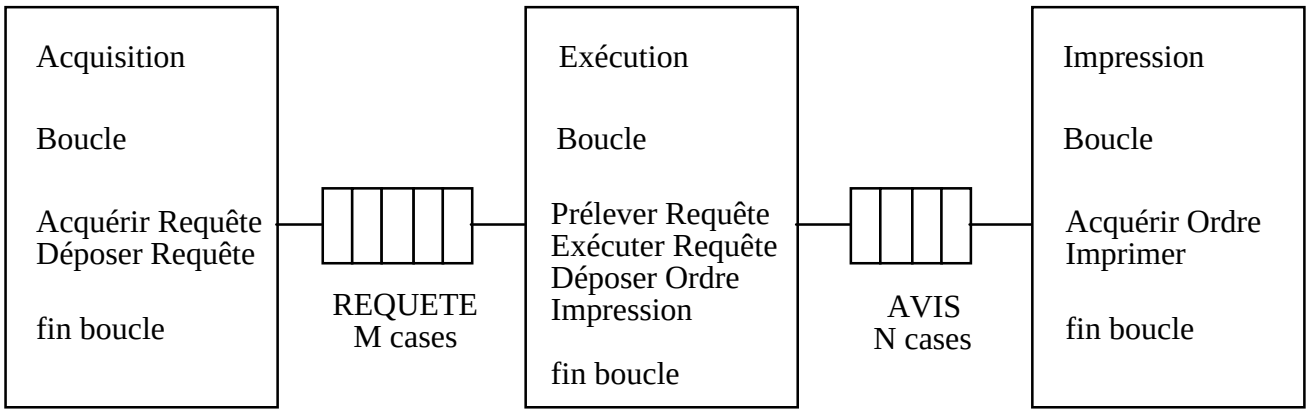
EXERCICES DIRIGES 7 et 8
Synchronisation de processus
CORRECTION

Exercice 1



Question 1

On identifie un schéma producteur/consommateur sur chacun des tampon. On utilise pour chacun de ces schéma un couple de sémaphores :

MVIDE initialisé à M et MPLEIN initialisé à 0 (tampon requête)
NVIDE initialisé à N et NPLEIN initialisé à 0 (tampon avis)

Processus principal
déclarations globales :
requête : tampon (0..M-1) de messages;
avis : tampon (0..N-1) de messages;
M, N : entier;
MVIDE, NVIDE, MPLEIN, NPLEIN : sémaphores;

debut
INIT(MVIDE, M); INIT (NVIDE, N); INIT(MPLEIN, 0); INIT(NPLEIN, 0);
lancer fils_acquisition; lancer fils_exécution; lancer fils_impression;
attendre mes fils
fin

processus acquisition	processus exécution	processus imprimeur
i : index depot requête	j : index retrait requête	l : index retrait avis
mess : message;	k : index depot avis	mess : message;
debut	mess, res : message;	debut
i = 0;	debut	l = 0;
boucle	j = 0; k = 0;	boucle
enregistrer_travail(mess);	boucle	P(NPLEIN);
P(MVIDE);	P(MPLEIN);	mess = avis(l);

requete(i) = mess;
i = i + 1 mod M;
V(MPLEIN);
fin boucle;
fin
	mess = requete(j);
	j = j + 1 mod M;
	V(MVIDE);
	exécuter_travail(mess, res);
	P(NVIDE);
	avis(k) = res;
	k = k + 1 mod N;
	V(NPLEIN);
	fin boucle;
	fin
		l = l + 1 mod N;
		V(NVIDE);
		imprimer_resultat(mess);
		fin boucle;
		fin

Question 2

Il faut maintenant gérer les accès concurrents aux tampons avis et requête. En effet :

- les différents processus Acquisition se partagent l'index i
- les différents processus exécution se partagent l'index j et k
- les différents processus Impression se partagent l'index k

les variables i, j, k, l sont maintenant globales et les accès à ces variables doivent se faire en exclusion mutuelle. On ajoute donc quatre sémaphores d'exclusion mutuelle initailisés à 1 (un sémaphore par index).

Processus principal déclarations globales : requête : tampon (0..M-1) de messages; avis : tampon (0..N-1) de messages; M, N : entier; i, j, k, l : index sur les tampons; MVIDE, NVIDE, MPLEIN, NPLEIN, MUTI, MUTJ, MUTK, MUTL : sémaphores; debut INIT(MVIDE, M); INIT (NVIDE, N); INIT(MPLEIN, 0); INIT(NPLEIN, 0); INIT(MUTI, 1); INIT(MUTJ, 1); INIT(MUTK, 1); INIT(MUTL, 1); i = j= k = l = 0; lancer fils_acquisition; lancer fils_exécution; lancer fils_impression; attendre mes fils fin		

processus acquisition
mess : message;
debut
boucle
enregistrer_travail(mess);
P(MVIDE);
P(MUTI);
requete(i) = mess;
i = i + 1 mod M;
V(MUTI);
V(MPLEIN);
fin boucle;
fin
	processus exécution
	
	mess, res : message;
	debut
	boucle
	P(MPLEIN);
	P(MUTJ)
	mess = requete(j);
	j = j + 1 mod M;
	V(MUTJ);
	V(MVIDE);
	exécuter_travail(mess, res);
	P(NVIDE);
		processus imprimeur
		mess : message;
		debut
		boucle
		P(MUTL);
		P(NPLEIN);
		mess = avis(l);
		l = l + 1 mod N;
		V(NVIDE);
		V(MUTL);
		imprimer_resultat(mess);
		fin boucle;
		fin

	P(MUTK);
	avis(k) = res;
	k = k + 1 mod N;
	V(MUTK);
	V(NPLEIN);
	fin boucle;
	fin	

remarque : l'ordre d'appel des sémaphores d'exclusion mutuelle par rapport à ceux du schéma producteur consommateur n' pas d'importance.

Exercice 2

Question 1

Lorsque l'employé **saisit** une commande, l'écriture sur disque d'une ligne à la fois, implique que, à un instant donné, le disque ne contient qu'une partie de la commande. Si le processus de facturation est lancé, il ne trouvera pas toutes les lignes de la commande pour éditer la facture, qui sera donc partielle. Au moment où on crée une commande, il faut donc interdire au processus de facturation d'accéder aux commandes.

Par ailleurs, l'employé édite les commandes saisies sur la même imprimante que le processus de facturation. Ces éditions se font ligne par ligne, mais toutes les lignes concernant le même bon de commande ou la même facture doivent se trouver regroupées, et non entremêlées. Il faut donc que l'imprimante soit en exclusion mutuelle entre les deux processus.

Question 2 – question 3
La solution proposée garantit bien l'accès en exclusion mutuelle à l'imprimante par les deux processus. Le processus de facturation réserve l'imprimante pendant le traitement des factures d'une période. Ces factures seront donc bien éditées de manière consécutive. Par ailleurs, comme le processus réserve également le fichier des commandes, aucune commande ne peut être en cours de saisie pendant l'édition des factures.

Question 4
L'interblocage est une situation où un ensemble de processus sont bloqués en attente d'une ressource possédée par un autre processus de l'ensemble. Chacun attend qu'un autre veuille bien libérer la ressource qu'il attend. Ceci ne peut se faire sans une intervention extérieure, puisqu'ils sont tous bloqués. Or on ne peut débloquer un processus qu'en lui donnant toutes les ressources nécessaires, et donc en réquisitionnant celle qu'il attend et qui est possédée par un autre processus de l'ensemble.

Dans la solution proposée, on peut imaginer que le processus employé réserve le fichier COM et commence à saisir la commande. A ce moment le processus de facturation est activé, et réserve l'imprimante, puis se bloque en attente du fichier COM. Lorsque l'employé a terminé la saisie, il réserve l'imprimante, mais comme celle-ci est déjà réservée par le processus de facturation, il se bloque en attente de la libération. Nous avons alors deux processus qui attendent mutuellement la libération d'une ressource possédée par un autre processus de l'ensemble : ces deux processus sont en interblocage.

Question 5

Pour ne plus avoir d'interblocage, une des solutions est de réserver les ressources dans le même ordre, puisque, dans ce cas, il ne peut plus y avoir de circularité dans les attentes de ressources. Dans le processus de facturation, il faut donc réserver le fichier COM en premier.

```
processus facturation;
début répéter indéfiniment
        réserver (COM);
        réserver (IMP);
            tant qu'il y a des commandes à facturer dans COM faire
        lire la prochaine commande dans COM; éditer la facture
        correspondante sur IMP; fait;
        libérer(IMP);
        libérer(Com);
        attendre la prochaine période de facturation; fait;
fin;
```

Exercice 3

```
Procedure CredDeb_Compte (Numero_Compte, operation, somme)
debut
/* lecture dans le fichier Fichier_Numero_Compte du solde du compte
Numero_Compte, solde_compte */
      concatener ("Fichier", Numero_Compte);
      Lire (Fichier_Numero_Compte, Numero_Compte, solde_compte);
      Si (operation = "debiter")
      alors
            solde_compte := solde_compte - somme;
      sinon
            solde_compte := solde_compte + somme;
      fsi
/* ecriture dans le fichier Fichier_Numero_Compte du nouveau solde du
compte Numero_Compte */
      Ecrire (Fichier_Numero_Compte, Numero_Compte, solde_compte);
fin


Procedure Donner_Solde (Numero_Compte, var Solde)
debut
/* lecture dans le fichier Fichier_Numero_Compte du solde du compte
Numero_Compte, solde_compte */
      concatener ("Fichier", Numero_Compte);
      Lire (Fichier_Numero_Compte, Numero_Compte, solde_compte);
      Solde := solde_compte;
fin
```

Question 1

Deux processus P1 et P2 exécutent chacun de leur côté, la même opération CredDeb_Compte(10, Crediter,100). le déroulement de ces opérations peut être le suivant. le solde final sera finalement de 1100 à la place de 1200.

P1
lecture du compte 10
le solde est égal à 1000

----------→ P2
lecture compte 10
le solde est égal à 1000

crédit de 100 : solde = 1100
écriture du compte 10
le solde sur le disque est porté à 1100

crédit de 100 : solde = 1100
écriture du compte 10
le solde sur le disque est porté à 1100

Question 2

Il suffit de placer l'exécution de la procédure CredDeb_Compte en exclusion mutuelle, soit avec ACCES sémaphore initialisé à 1.

```
:
Procedure CredDeb_Compte (Numero_Compte, operation, somme)
debut
/* lecture dans le fichier Fichier_Numero_Compte du solde du compte
Numero_Compte, solde_compte */
      concatener ("Fichier", Numero_Compte);
      P(ACCES);
      Lire (Fichier_Numero_Compte, Numero_Compte, solde_compte);
      Si (operation = "debiter")
      alors
            solde_compte := solde_compte - somme;
      sinon
            solde_compte := solde_compte + somme;
      fsi
/* ecriture dans le fichier Fichier_Numero_Compte du nouveau solde du
compte Numero_Compte */
      Ecrire (Fichier_Numero_Compte, Numero_Compte, solde_compte);
      V(ACCES);
fin
```

Question 3

Cette fois, il faut mettre en place un schéma lecteurs/rédacteurs.
La procédure CredDeb_Compte reste comme en Q2.
La procédure Donner_Solde devient :

```
Procedure Donner_Solde (Numero_Compte, var Solde)
debut
/* lecture dans le fichier Fichier_Numero_Compte du solde du compte
Numero_Compte, solde_compte */
      concatener ("Fichier", Numero_Compte);
      P(MUTEX);
      NL = NL + 1;
      Si (NL == 1)
      alors
            P(ACCES)
      Fsi
      V(MUTEX)
      Lire (Fichier_Numero_Compte, Numero_Compte, solde_compte);
      Solde := solde_compte;
      P(MUTEX);
      NL = NL - 1;
      Si (NL == 0)
      alors
            V(ACCES)
      Fsi
V(MUTEX)
fin
```

Exercice 4

Lors de l'utilisation d'une ressource critique par une tâche, il est habituel d'ajouter aux paramètres de description de la tâche ceux supplémentaires suivant :

– $C_i\alpha$: séquence d'instructions précédant l'appel de la ressource,

– $C_i\beta$: séquence d'instruction de la section critique,

– $C_i\gamma$: séquence d'instruction suivant la libération de la ressource avec $C_i = C_i\alpha + C_i\beta + C_i\gamma$.

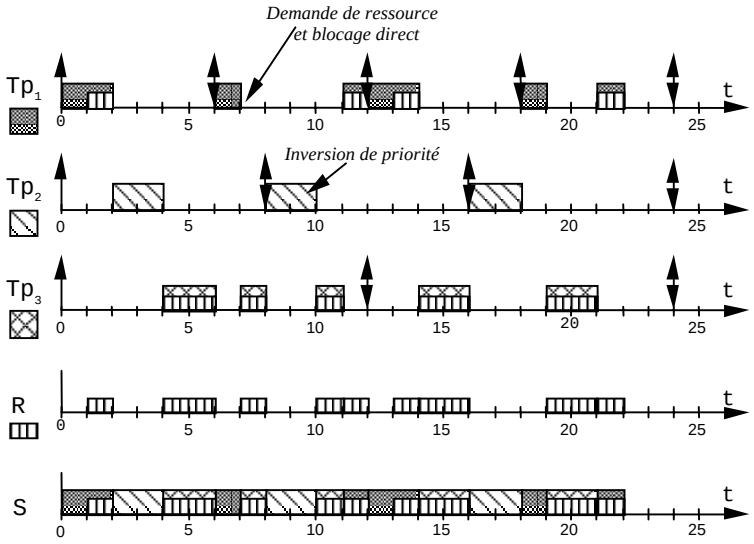
Toute tâche en cours d'utilisation d'une ressource critique peut être préemptée par une tâche plus prioritaire qu'elle, et qui n'a pas besoin de cette ressource.

On considère l'exemple d'une configuration de trois tâches périodiques. Les tâches Tp_1 et Tp_3.

Tâche	$r_{0,i}$	C_i	D_i	P_i	$C_i\alpha$	$C_i\beta$	$C_i\gamma$
Tp_1	0	2	6	6	1	1	0
Tp_2	0	2	8	8	2	0	0
Tp_3	0	4	12	12	0	4	0

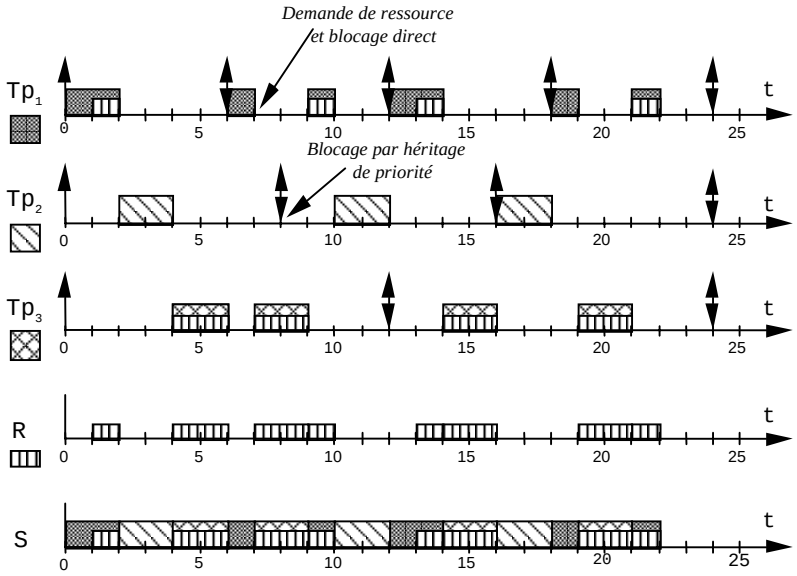
Question 1

Comme le montre la figure, à l'instant t=8, la tâche 3 est bloquée par la tâche 2 parce qu'elle est plus prioritaire qu'elle. Mais comme la tâche 1 est en attente de la ressource critique (occupée par la tâche 3) depuis l'instant t=7, on observe donc que la tâche 2 est exécutée avant la tâche 1 : c'est le phénomène d'inversion de priorité.



Question 2

Pour éviter ce problème d'inversion de priorité, à l'instant t=7, lorsque la tâche 1 se met en attente de la ressource occupée par la tâche 3, la tâche 3 prend la priorité de la tâche 1. Par conséquent, à l'instant t=8, la tâche 3 est plus prioritaire que la tâche 2 et continue à s'exécuter. L'exécution de la tâche 2 se trouve ainsi repoussée à l'instant t=10 après la tâche 1 (cf. figure 3.15).


Demande de ressource
et blocage direct
Blocage par héritage
de priorité
Tp1
Tp2
Tp3
R
S
t
0
5
10
15
20
25