EXERCICES DIRIGES
Synchronisation de processus
CORRECTION

Exercice 1

Question 1

Une seule écriture de nb_octets de données placées dans le tableau `tampon` est réalisée à la fois.

Question 2

```
Fonction ECRIRE_DISQUE (tampon, nb_octets)
char tampon[] ;
int nb_octets;
objet verrou := libre;

{
Verrouiller (verrou)
i = 0 ;
Tant que (nb_octets > 0)
Faire
-- l'écriture physique s'effectue par blocs de 512 octets
     Ecrire_blocdisque (tampon[i, i + 512], 512) ;  -- on écrit
physiquement 512 octets
     nb_octets = nb_octets – 512 ; -- on décrémente le nombre total
d'octets à écrire
     i = i + 512 ; -- on avance l'index dans le tableau des octets à
écrire
Fait
Deverrouiller (verrou)
Return ;
}
```

Question 3

La section critique peut être réalisée en masquant et démasquant les interruptions.

Exercice 3

Question 1

On identifie un schéma producteur/consommateur sur le tampon. On utilise un couple de sémaphores :

NVIDE initialisé à N et NPLEIN initialisé à 0 (tampon avis)

<table>
<tr><td colspan="3">Processus principal
déclarations globales :

avis : tampon (0..N-1) de messages;
N entier;
NVIDE, NPLEIN : sémaphores;

debut
INIT(NPLEIN, 0); INIT (NVIDE, N); lancer thread_exécution; lancer thread_impression;
attendre mes fils
fin</td></tr>
<tr><td>n</td><td>thread exécution (requete)

i : index depot avis
res : message;
debut
i = 0;
boucle
exécuter_travail(requete, res);
P(NVIDE);
avis(i) = res;
i = i + 1 mod N;
V(NPLEIN);
fin boucle;
fin</td><td>thread imprimeur

j : index retrait avis
mess : message;
debut
j = 0;
boucle
P(NPLEIN);
mess = avis(j);
j = j + 1 mod N;
V(NVIDE);
imprimer_resultat(mess);
fin boucle;
fin</td></tr>
</table>

Question 2

Il faut maintenant gérer les accès concurrents au tampon avis e. En effet :

- les différents threads exécution se partagent l'index i
- les différents threads Impression se partagent l'index j

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

Processus principal
déclarations globales :

avis : tampon (0..N-1) de messages;
N entier;
NVIDE, NPLEIN, MUTI, MUTJ : sémaphores;
i : index depot avis
j : index retrait avis

debut
INIT(NPLEIN, 0); INIT (NVIDE, N);
INIT (MUTI, 1); INIT (MUTJ,1);
lancer thread_exécution; lancer thread_impression;
attendre mes fils
fin

n	thread exécution (requete)	thread imprimeur

	res : message;
	debut
	
	boucle
	exécuter_travail(requete, res);
	P(MUTI) ;
	P(NVIDE);
	avis(i) = res;
	i = i + 1 mod N;
	V(NPLEIN);
	V(MUTI) ;
	fin boucle;
	fin
		mess : message;
		debut
		
		boucle
		P(MUTJ) ;
		P(NPLEIN);
		mess = avis(j);
		j = j + 1 mod N;
		V(NVIDE);
		V(MUTJ) ;
		imprimer_resultat(mess);
		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 3

Question 1

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 2

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;
```

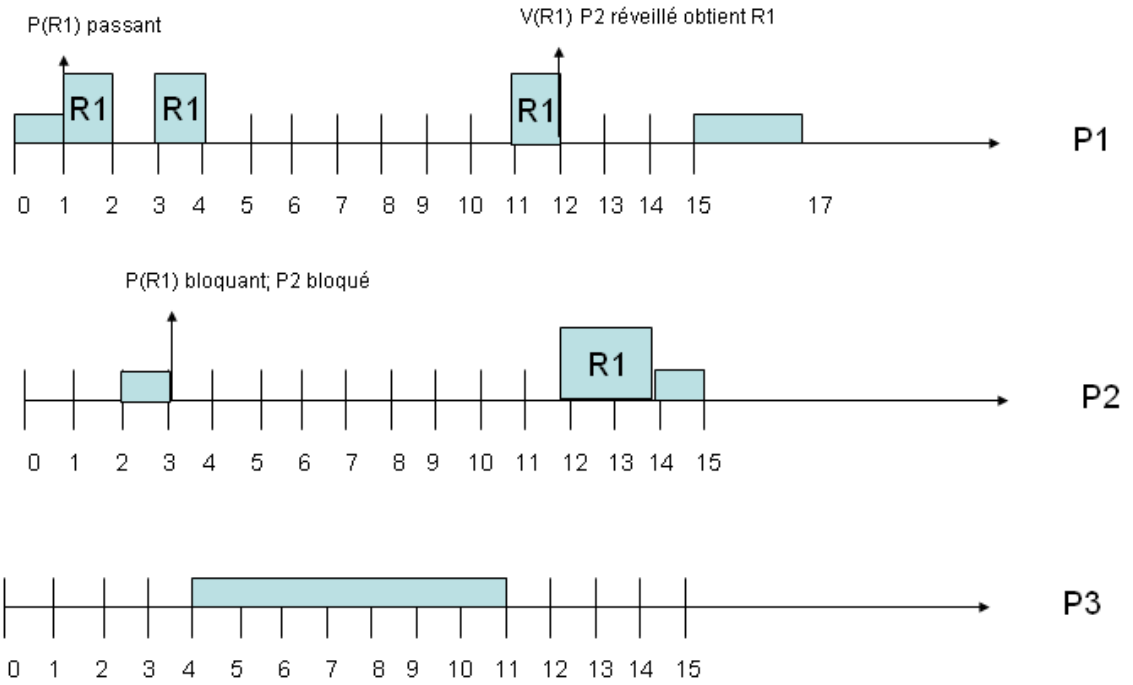
Exercice 3

Question 1

Traduisez les opérations

INIT (MutexR1, 1) ; Prendre (R1) → P (MutexR1) ; ; Rendre (R1) → V (MutexR1)

Question 2



Du fait du conflit de ressources entre R1 et R2, le processus P2, plus prioritaire des trois processus est retardé. C'est un processus de priorité intermédiaire P3, qui prend la main.