**Université Aboubekr BELKAID Tlemcen**



**Faculté des Sciences**

**Département d’informatique**

**Classe LMD S5 Informatique**

**Année Universitaire : 2021 / 2022**

**TD N° 6 : Les Moniteurs**

**Exercice N°1**

Vous avez le programme de moniteur suivant :

moniteur myMoniteur   
{  
Variable cv : condition   
void procedure 1(int x) {  
printf("A ");  
wait(cv);  
printf("B ");  
}  
void procedure2(int x) {  
printf("C ");  
signal(cv);  
printf("D ");  
}  
}

Supposons que le thread T1 appelle le procedure1 avant que le thread T2 appelle

le procédure 2.

Q1 : Si le système implémente des moniteurs, quelle sera la sortie de ce programme? Expliquer votre réponse.

Q2 : modifier le programme en utilisant les moniteurs pour avoir l’affichage de message suivant : **CADB**

**Exercice N°2**

**Q1 :** Vous avez un programme composé par 4 threads suivants : **T1, T2, T3 et T4**

**T1** : affiche le mot **«née »**

**T2** : affiche le mot **«ne an »**

**T3** : affiche le mot **«2022 »**

**T4** : affiche le mot **«Bon»**

1-Synchronisez les 4 threads : T1, T2, T3 et T4 en utilisant les primitives de moniteurs wait et signal pour que le programme affiche obligatoirement la phrase suivante  **: « Bonne année 2022»**

**Exercice N°3**

Dans un lavomatique, on cherche une solution pour permettre de repartir les M machines à laver entre les clients. Chaque machines à laver est identifiée par un numéro. . Pour obtenir une machine, chaque client doit utiliser la fonction allouer-machine(). Après usage de la machine, il doit utiliser liberer-machine().

Deux procédures sont utilisées :

**Allouer -machine (var num-machine : 1….M)**

**liberer-machine (var num-machine :1….M)**

**Q1**-donnez les conditions de blocage des clients

**Q2**-resoudre seproblème avec les **moniteurs.**

**Exercice N°4**

Nous nous intéressons au problème d’allocation des pistes dans un disque dur. Nous supposons disposer de plusieurs pistes dans le disque dur, dont un nombre quelconque de pistes peut être demandé par un processus à un moment donné. Pour simplifier.

Tout processus actif qui demande un nombre quelconque de pistes, doit exécuter

Les deux procédures suivantes :

**Demande (k)** : pour demander l’allocation de **k** pistes de disque dure.

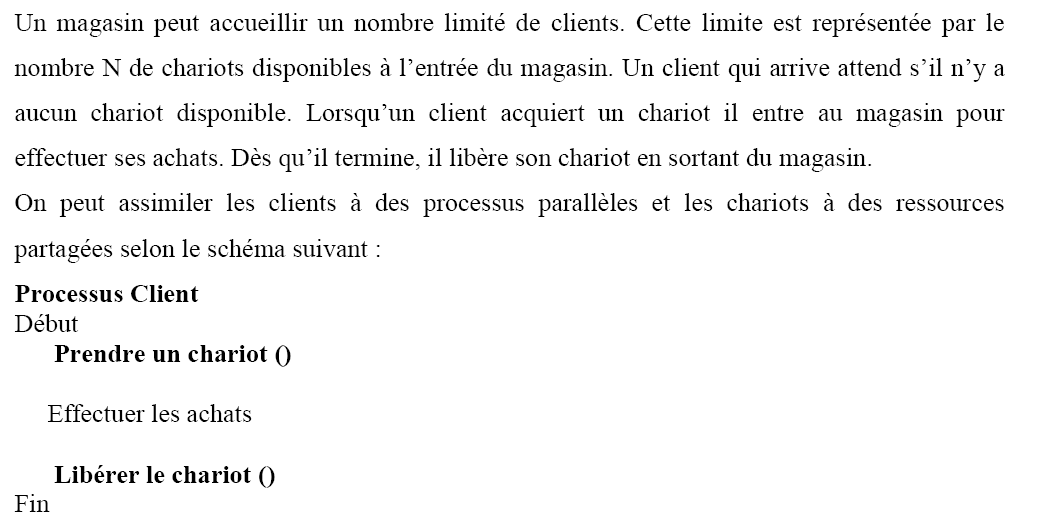
**Libérer (k)** : pour libérer les k pistes de disque dur. K varié entre 1 à 3

On a N pistes dans le disque dure telque N> k.

**Q1 :** indiquez la condition de blocage des processus.

**Q2-** Résoudre ce problème par **le moniteur** dans le cas ou chaque processus demande plusieurs pistes (k = 1 à 3) de cette ressource.

**Exercice N°5**



**Q1 : Donner la solution de ce problème avec les moniteurs.**

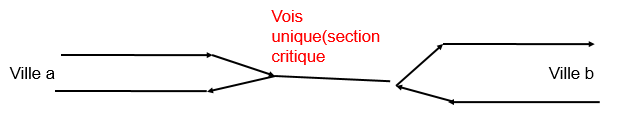
**Exercice N° 6**

Une ligne de chemin de fer reliant deux villes A et B comporte une section à voie unique. On représente les trains par des processus. Deux types de processus existent : processus Avers B et processus B vers A.

- Deux trains arrivant de sens opposés ne peuvent pas traverser la section au même

moment.

**Q1 :** résoudre le problème en utilisant les moniteurs



**Solution TD N° 6 : Les Moniteurs**

**Exercice N°1**

Vous avez le programme de moniteur suivant :

moniteur myMoniteur   
{  
Variable cv : condition   
void procedure 1(int x) {  
printf("A ");  
wait(cv);  
printf("B ");  
}  
void procedure2(int x) {  
printf("C ");  
signal(cv);  
printf("D ");  
}  
}

Supposons que le thread T1 appelle le procedure1 avant que le thread T2 appelle

le procédure 2.

Q1 : Si le système implémente des moniteurs, quelle sera la sortie de ce programme? Expliquer votre réponse.

Q2 : modifier le programme en utilisant les moniteurs pour avoir l’affichage de message suivant : **CADB**

**Solution exercice N°1**

**Q1** : Si le système implémente des moniteurs, quelle sera la sortie de ce programme? Expliquer votre réponse.

L’affichage est : ACBD (il y a une synchronisation entre les deux processus en utilisant les moniteurs

**Q2** : modifier le programme en utilisant les moniteurs pour avoir l’affichage de message suivant : **CADB**



**Exercice N°2**

**Q1 :** Vous avez un programme composé par 4 threads suivants : **T1, T2, T3 et T4**

**T1** : affiche le mot **«née »**

**T2** : affiche le mot **«ne an »**

**T3** : affiche le mot **«2022 »**

**T4** : affiche le mot **«Bon»**

1-Synchronisez les 4 threads : T1, T2, T3 et T4 en utilisant les primitives de moniteurs wait et signal pour que le programme affiche obligatoirement la phrase suivante  **: « Bonne année 2022»**



**Solution exercice N°2**

**T1** : affiche le mot **«née »**

**T2** : affiche le mot **«ne an »**

**T3** : affiche le mot **«2022 »**

**T4** : affiche le mot **«Bon»**

synchronisation par les moniteurs :

**wait(c2) T1 signal(c3)**

**wait(c1) T2 signal(c2)**

**wait(c3) T3**

**T4 signal(c1)**

**Exercice N°3**

Dans un lavomatique, on cherche une solution pour permettre de repartir les M machines à laver entre les clients. Chaque machines à laver est identifiée par un numéro. . Pour obtenir une machine, chaque client doit utiliser la fonction allouer-machine(). Après usage de la machine, il doit utiliser liberer-machine().

Deux procédures sont utilisées :

**Allouer -machine (var num-machine : 1….M)**

**liberer-machine (var num-machine :1….M)**

**Q1**-donnez les conditions de blocage des clients

**Q2**-resoudre seproblème avec les **moniteurs.**

**Solution exercice N°3**

Moniteur machine-laver ()

Occupé : tableau [1..M] : booléen (initialisé a faux)

M : constante le nombre de machine laver M=4

Client : condition ile d’attente

Num : entier

Cpt=0 //compteur

**Solution 1**

**Procedure allouer-machine ()**

Debut

Si cpt >= M alors wait(client)

Cpt++

Num :=1 ;

Tant que occupé[Num] {

alors faire Num :=Num + 1 }

Occupé[Num] :=vrai ;

fin

**Procedure liberer-machine (int Num)**

Debut

Cpt--

Occupé[Num] :=faux

Signal(client)

fin

**Solution 2**

Moniteur machine-laver ()

Occupé : tableau [1..M] : booléen (initialisé a faux)

M : constante le nombre de machine laver M=4

Client : condition ile d’attente

Num : entier

**cpt=M=4** //compteur

**Procedure allouer-machine ()**

Debut

Si cpt = 0 alors wait(client)

 Cpt=cpt-1

Num :=1 ;

Tant que occupé[Num] {

alors faire Num :=Num + 1 }

Occupé[Num] :=vrai ;

fin

**Procedure liberer-machine (int Num)**

Debut

**cpt++**

Occupé[Num] :=faux

Signal(client)

fin

**Exercice N°4**

Nous nous intéressons au problème d’allocation des pistes dans un disque dur. Nous supposons disposer de plusieurs pistes dans le disque dur, dont un nombre quelconque de pistes peut être demandé par un processus à un moment donné. Pour simplifier.

Tout processus actif qui demande un nombre quelconque de pistes, doit exécuter

Les deux procédures suivantes :

**Demande (k)** : pour demander l’allocation de **k** pistes de disque dure.

**Libérer (k)** : pour libérer les k pistes de disque dur. K varié entre 1 à 3

On a N pistes dans le disque dure telque N> k.

**Q1 :** indiquez la condition de blocage des processus.

**Q2-** Résoudre ce problème par **le moniteur** dans le cas ou chaque processus demande plusieurs pistes (k = 1 à 3) de cette ressource.

**Solution exercice N°4**

**Q1 :** indiquez la condition de blocage des processus.

**Les processus seront bloqués lorsqu’il n’y a pas des pises disponible dans le disque dur**

**Q2-** Résoudre ce problème par **le moniteur** dans le cas ou chaque processus demande plusieurs pistes (k = 1 à 3) de cette ressource.

**Monieur disque ()**

**Var**

**Allocation : tableau [1..3] : de type condition**

**Nbr\_disponible : entier (nombre de piste disponible initialisé a N)**

**Procedure demande(k : entier )**

**Debut**

**Si (k > nbr\_disponible) alors wait (allocation[k])**

**Nbr\_disponible = nbr\_disponible – k**

**Fin**

**Procedure liberer (k : entier)**

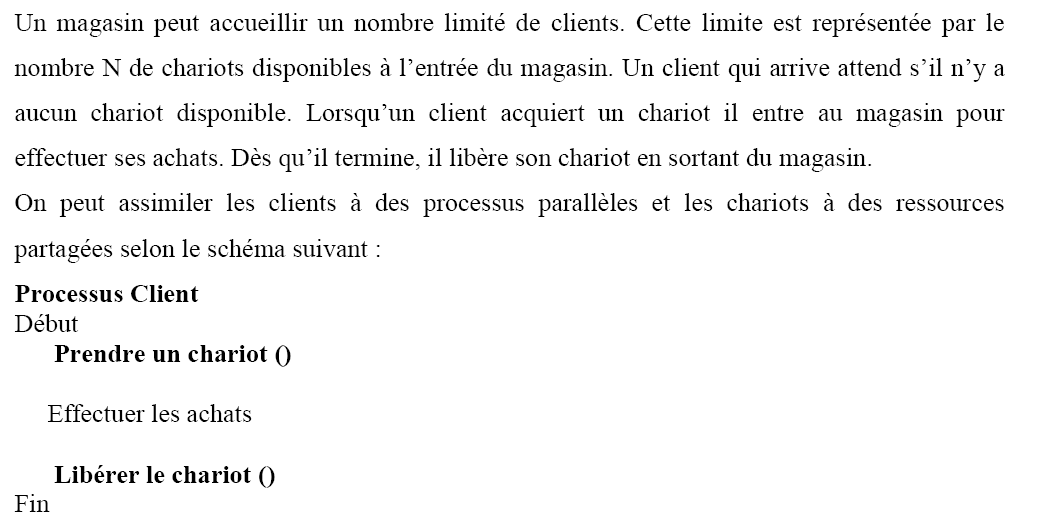
**Debut**

**Nbr\_diponible = nbr\_disponible + k**

**Signal (allocation [k]) ;**

**fin**

**Exercice N°5**



**Q1 : Donner la solution de ce problème avec les moniteurs.**

**Solution exercice N°5**

**Monieur chariot()**

**Var**

**client: condition //file d’atte**

**Nbr-chariot: N (nombre de chariots disponible)**

**Procedure prend chariot()**

**Debut**

**si nbr-chariot= 0 alors wait(client)**

**nbr-chariot = nbr-chariot -1**

**Fin**

**Procedure liberer chariot ()**

**Debut**

**nbr-chariot = nbr-chariot +1**

**Signal (client) ;**

**fin**

**Solution avec sémaphore**

**Processus client{**

**e(chariot)=N**

**Debut**

**p(chariot) //entrer magazin**

**Section critique MAGAZIN**

**V(chariot)// sortie magazin**

**Fin**

}

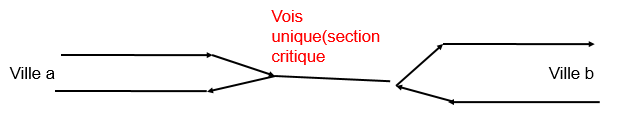
**Exercice N° 6**

Une ligne de chemin de fer reliant deux villes A et B comporte une section à voie unique. On représente les trains par des processus. Deux types de processus existent : processus Avers B et processus B vers A.

- Deux trains arrivant de sens opposés ne peuvent pas traverser la section au même

moment.

**Q1 :** résoudre le problème en utilisant les moniteurs



**Solution exercice N°6**

**Moniteur train ()**

**Var**

**Sense-AB: condition //file d’attente**

**Sense-BA : condition //file d’attente**

**Nbre-ab :nombre de train sense AB**

**Nbre-ba : nombre de train sense BA**

**Procedure entre\_train\_AB()**

**Debut**

**si nbr-ba > 0 alors wait(sense-AB)**

**Nbr-ab++**

**Signal (sense-AB)**

**Fin**

**Procedure sortie\_train\_AB()**

**Debut**

**Nbr-ab=nbr-ab -1**

**Si nbr-ab= 0 alors signal(sense-BA)**

**fin**

**Procedure entre\_train\_BA()**

**Debut**

**si nbr-ab > 0 alors wait(sense-BA)**

**Nbr-ba++**

**Signal (sense-BA)**

**Fin**

**Procedure sortie\_train\_BA()**

**Debut**

**Nbr-ab=nbr-ba -1**

**Si nbr-ba= 0 alors signal(sense-AB)**

**fin**