USTHB
Faculté d'Electronique et Informatique
Département Informatique
Epreuve Finale – Durée : 1h15
(Systèmes d'exploitation- M1: IV) Le 22/01/2019 - Année 19/20

Exercice n°1 : (6 pts= 2*3)

Répondre aux questions suivantes :

A- Pourquoi la primitive *signal ()* n'est pas nécessaire dans un moniteur avec conditions de kessels ?

B- Citer les opérations minimales nécessaires pour les boites aux lettres communes.

C- Donner la définition de la fragmentation interne dans un SGF.

Exercice 2 (7 pts= 1,5+1,5+2+2)

On considère un système de fichiers à accès direct avec table d'indexes. On suppose que la taille d'un bloc logique est de 512 octets et un fichier de taille 598 272 octets et la taille d'un enregistrement est de 256 octets :

A/ Quel est le facteur de blocage ?

B/ Donner la taille de la fragmentation interne.

C/ Si on considère une fonction de hachage modulo 750, donner la taille de la zone de débordement.

D/ En moyenne, combien de clés donnent la même entrée dans la table d'adresses ?

Exercice 3 : (7 pts=4.5+2.5)

On considère le modèle du producteur / consommateur avec un tampon de taille *N*, défini comme suit :

- $N=C*k$ cases où *C* et *k* sont deux entiers naturels et k est pair,
- Le producteur dépose au même moment et à chaque fois dans *k* cases du tampon, mais le consommateur prélève à chaque fois deux cases.

A- Donner une solution à ce problème à l'aide des moniteurs classiques en considérant un seul processus producteur et un seul consommateur,

B- Généraliser la solution à plusieurs producteurs et plusieurs consommateurs.

Bon Courage

Correction de l'Epreuve Finale

Exercice 1 : (6 pts= 2*3)

D- La primitive *signal ()* n'est pas nécessaire dans un moniteur avec conditions de kessels car quand un processus termine une procédure du moniteur ou se bloque dernière une condition, tous les processus bloqués sur des conditions réévaluent automatiquement leurs conditions et un est réveillé automatiquement parmi ceux dont la condition est devenue vrai.

E- Les opérations minimales nécessaires pour les boites aux lettres communes sont : la création, la destruction, l'envoi de message, réception de message.

F- La fragmentation interne dans un SGF est celle engendrée dans le dernier bloc physique alloué à un fichier s'il ne l'occupe pas en entier.

Exercice 2 : (7 pts= 1,5+1,5+2+2)

C/ Si on considère une fonction de hachage modulo 750, donner la taille de la zone de débordement.

On considère un système de fichiers à accès direct avec table d'indexes. On suppose que la taille d'un bloc logique est de 512 octets et un fichier de taille 598 272 octets et la taille d'un enregistrement est de 256 octets :

A/ Le facteur de blocage est : 512/256=2 enregistrements par bloc logique.

B/ Taille de la fragmentation interne : Il s'agit de la taille non occupée du dernier bloc du fichier, notée Fg.

Fg= Taille bloc – Taille occupée du dernier bloc du fichier= 512- (598 272 Mod 512)= 512-256=256 octets.

C/ Taille zone débordement = Taille de la table d'adresses en nombre d'entrées (ie. nombre d'enregistrements)- taille de la table d'adresses de base

= Taille fichier/ Taille enregistrement – 750

= 598 272/256 -750

= 2337 – 750

= 1587.

D/ D/ En moyenne, le nombre de clés qui donnent la même entrée dans la table d'adresses est : nombre de clés / taille table d'adresses de base = 2337/750= 3,116

Exercice n°3 : (7 pts=4.5+2.5)

A/ Il s'agit du modèle d'un producteur et d'un consommateur, donc on a besoin de deux conditions *cp* et *cc*. Cependant le modèle possède la particularité suivante :

Le producteur dépose *k* cases à la fois et puisque *N=C*k* donc il ne peut déposer que *C* fois avant de remplir le tampon, donc *nv* doit être initialisé à *C*. Une fois qu'il dépose *k* articles, le producteur incrémente *nb_art* de *k* et réveille le consommateur s'il est bloqué. Par conséquent et sachant qu'il prélève deux cases à la fois, le consommateur incrémente *cpt* de *2* et quand *cpt* devient *nul*, il est sûr qu'il a vidé *k* cases donc il incrémente *nv* de *1* et réveille le producteur s'il est bloqué.

Remarque : D'autres solutions plus simples existent aussi.

Forme générale d'un processus
Type Tart :.... ;
Var T : Tableau [0..N-1] de Tart ; t, q : entier :=0 ;

Processus Prod() ;
Début
Répeter
Produite (Art) ;
M. Dem_dep() ;
<Déposer (Art_k, T)>
M. Fin_Dep() ;
Jusqu'à Faux
Fin.

Processus Cons() ;
Début
Répeter
M. Dem_Prel() ;
<Prélever(Art_2, T)>
M. Fin_Prel()
Consomer(Art)
Jusqu'à Faux ;
Fin.

M : Moniteur ;
Const k =... ; C =....
Var cc, cp : condition ; np, nv, cpt : entier ;

Entry procédure Dem_Dep() ;
Début Si (nv=0) Alors cp.wait() fsi ; nv :=nv-1 Fin ;

Entry procedure Fin_Dep() ;
Début np :=np+k ; cc.signal() Fin ;

Entry procédure Dem_Prel () ;
Début Si (np<2) Alors cc.wait() Fsi ; np :=np-2 ; Fin ;

Entry procédure Fin_Prel() ;
Début cpt :=(cpt+2) Mod k ; Si (cpt=0) Alors nv :=nv+1 ; cp.signal () Fsi Fin ;
Init Début np :=0 ; nv :=C ; cpt :=0 ; Fin.

2/ Généralisation

M : Moniteur ;
Const k =... ; C =....
Var cc, cp, c_encours, p_encours : condition ; np, nv, cpt : entier ;
bprod, bcons : booléen ;

Entry procédure Dem_Dep() ;
Début Si (nv=0) Alors cp.wait() fsi ; nv :=nv-1 ;
Si bprod Alors p_encours.wait() Fsi
bprod :=vraie ;
Fin ;

Entry procedure Fin_Dep() ;
Début np :=np+k ; cc.signal() ; bprod :=faux ; p_encours.signal() Fin ;

Entry procédure Dem_Prel () ;
Début Si (np<2) Alors cc.wait() Fsi ; np:=np-2 ;
Si bcons Alors c_encours.wait() Fsi ; bcons :=vrai
Fin ;

Entry procédure Fin_Prel() ;
Début cpt :=(cpt+2) Mod k ; Si (cpt=0) Alors nv :=nv+1 ; cp.signal () Fsi ;
bcons :=faux ; c_encours.signal() Fin ;

Init Début np :=0 ; nv :=C ; cpt :=0 ; bprod :=faux ; bcons :=faux Fin.