**SYSTÈMES ET RÉSEAUX INFORMATIQUES**

# CORRIGÉ DE L'EXAMEN DU 18 JUIN 2005

**portant sur l'enseignement des SYSTÈMES INFORMATIQUES**

## Gestion de cases pour pagination à la demande

### *Réponses aux questions*

**RÉPONSE 1.** Tous ces paquetages ont un seul producteur (le Disque) et un seul consommateur (Serveur ou Videur) et sont donc sur le même modèle simple donné en cours, permettant la concurrence d'accès entre Deposer et Retirer. Il vient alors pour **Depot_Videur :**

```
package body Depot_Videur is
    Nplein, Nvide : Semaphore ;
    N: Integer := 10 ; -- on a mis 10 car pour une instruction on peut avoir plusieurs pages manquante
    Tete, Queue : Mod N := 0 ; -- on anticipe aussi le cas où un Serveur contiendra plusieurs Threads
    T : array(0.. Max – 1) of Descripteur_De_Case;
    procedure Deposer(X : in Descripteur_De_Case) is
    begin
        P(Nvide) ; T(Queue) := X ; Queue := Queue + 1 ; V(Nplein) ;
    end Deposer;
    procedure Retirer(X : out Descripteur_De_Case) is
    :begin
        P(Nplein) ; X := T(Tete) ; Tete:= Tete + 1 ; V(Nvide) ;
    end Retirer;
begin
    E0(Nvide, N) ; E0(Nplein, 0) ;
end Depot_Videur;
```

**RÉPONSE 2.** Cette fois il y a plusieurs producteurs et un consommateur. D'autre part il faut noter la longueur pour pouvoir la lire. C'est un autre schéma du cours. Voici une solution possible avec concurrence d'accès entre Deposer et Retirer. (attention (Tete – Queue) mod 10 = 0 si nombre = 10)

```
package bodyDepot_Disque is
    Nplein, Nvide, Mutex_Prod, Mutex_Nombre : Semaphore ;
    N: Integer := 10 ;
    Tete, Queue : Mod N := 0 ;
    T : array(0.. Max – 1) of Descripteur_De_Case;
    Nombre : Integer := 0 ;
    procedure Deposer(X : in Descripteur_De_Case) is
    begin
        P(Nvide) ;
        P(Mutex_Prod) ; -- entre producteurs
        T(Queue) := X ; Queue := Queue + 1 ;
        P(Mutex_Nombre) ; Nombre := Nombre + 1 ; V(Mutex_Nombre) ; -- cohérence de Nombre
        V(Mutex_Prod) ;
        V(Nplein) ;
    end Deposer;
    procedure Retirer(X : out Descripteur_De_Case) is
    :begin
        P(Nplein) ; X := T(Tete) ; Tete:= Tete + 1 ; -- un consommateur au plus
        P(Mutex_Nombre) ; Nombre := Nombre -1 ; V(Mutex_Nombre) ;
        V(Nvide) ;
    end Retirer;
    procedure  Combien_De_Messages(Compte : out Integer)
    begin
        P(Mutex_Nombre) ; Compte := Nombre ; V(Mutex_Nombre) ;
```

```
    end Combien_De_Messages ;
begin
    E0(Nvide, N) ; E0(Nplein, 0) ; E0(Mutex_Prod, 1) ; E0(Mutex_Nombre, 1) ;
end Depot_Disque ;
```

**RÉPONSE 3.**

```
package body Signal is
    Present : Boolean := False ; -- indique si le signal est déjà arrivé
    S, Mutex : Semaphore ; -- S contrôle l'attente de réception du signal, Mutex pour la section critique
    procedure Envoyer is   -- enregistre une fois un signal provenant d'un ou plusieurs Serveurs
    begin
        P(Mutex) ;
        if not Present then Present := True ; V(S) ; end if ; -- test et écriture en section critique
        V(Mutex) ;
    end Envoyer ;
    procedure Attendre is   -- bloque l'appelant en attente du signal
    begin
        P(S) ;
        P(Mutex) ; Present := False ; V(Mutex) ; -- écriture atomique, exclusion mutuelle non nécessaire
    end Attendre ;
begin
    E0(S, 0) ; E0(Mutex, 1) ; -- initialisation des sémaphores
end Signal ;
```

**RÉPONSE 4.**

Le Stock_Global contient deux lots de cases, un lot de cases pures (ou non écrites) et un lot de cases écrites. Ces deux lots sont rangés dans un tableau Stock dans deux piles ; les cases pures sont empilées au début du tableau, les cases écrites sont empilées en fin du tableau.

| 1 | 2 | 3 | Tete_Pure | | | | | Tete_Ecrite | | Max-1 | Max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Pure | Pure | Pure | Pure | ---> | | | <--- | Ecrite | Ecrite | Ecrite | Ecrite |

On donne ci-après les procédures de manipulation du Stock.

```
Package body Stock_Global is
    La_Case : Descripteur_De_Case;
    Max : Integer := 530; -- nombre total des cases allouables dans le système
    Stock : array(1 .. Max ) of Descripteur_De_Case; -- gestion des cases
    Tete_Pure : Integer := Max ; -- initialement Stock est rempli de cases pures
    Tete_Ecrite,: Integer := Max + 1; -- initialement Stock ne contient pas de case écrite
    -- nombre  de cases pures = Tete_Pure ; -- taille de pile vers le haut
    -- nombre de cases écrites = Max + 1 – Tete_Ecrite ; -- taille de pile vers le bas

    Mutex_Pures, Mutex_Ecrites,  Sem_Pures, Sem_Ecrites : Semaphore ;

    procedure Demander_Pure(X : out Descripteur_De_Case; Y : in Integer) is
    -- fournit au processus Y une case non écrite et la sort du stock
    begin
        -- contrôler  qu'il y a bien une case pure disponible, sinon bloquer l'appelant
        P(Sem_Pures) ;
        -- accès contrôlé à la gestion des cases pures
        P(Mutex_Pures) ;
        X := Stock(Tete_Pure) ; Tete_Pure:= Tete_Pure - 1; -- on dépile  la pile du haut
        V(Mutex_Pures) ;
        X. Hote:= Y ;
    end Demander_Pure;

    procedure Demander_Ecrite(X: out Descripteur_De_Case; Y : in Integer) is
    -- fournit au processus  Y une case écrite et la sort du stock
    begin
```

```
        -- contrôler  qu'il y a bien une case écrite disponible, sinon bloquer l'appelant
        P(Sem_Ecrites) ;
        -- accès contrôlé à la gestion des cases écrites
        P(Mutex_Ecrites) ;
        X := Stock(Tete_Ecrite) ; Tete_ Ecrite:= Tete_ Ecrite + 1; -- on dépile  la pile du bas
        V(Mutex_Ecrites) ;
        X. Hote:= Y ;
    end Demander_ Ecrite;

    procedure Rendre (X : in Descripteur_De_Case) is
    begin
        X. Hote:= 0 ;
        If X.Ecrite then
            -- accès à la gestion des cases écrites
            P(Mutex_Ecrites) ;
            Tete_ Ecrite:= Tete_ Ecrite - 1; Stock(Tete_Ecrite) := X ;
            V(Mutex_Ecrites) ;
            -- indiquer au contrôle qu'il y a une case écrite de plus
            V(Sem_Ecrites) ;
        else
            -- accès contrôlé à la gestion des cases pures
            P(Mutex_Pures) ;
            Tete_Pure:= Tete_Pure + 1; Stock(Tete_Pure) := X ;
            V(Mutex_Pures) ;
            -- indiquer au contrôle qu'il y a une case pure de plus
            V(Sem_Pures) ;
        end if ;
    end Rendre;

    procedure Compter_Ecrites (Compte : out Integer) is
    begin
        -- accès contrôlé à la gestion des cases écrites
        P(Mutex_Ecrites) ;
        Compte := Max + 1 – Tete_Ecrite ;
        V(Mutex_Ecrites) ;
    end Compter_Ecrites;

begin
    for I in 1 .. Max loop
        La_Case.Info := -- mise à jour de l'adresse physique de la prochaine case allouable
        Stock(I) := La_Case ; -- les valeurs par défaut sont bonnes à conserver
    end loop ;
    -- à l'initialisation, le stock est plein de cases pures et ne contient aucune case écrite
    E0(Mutex_Pures, 1) ; E0(Mutex_Ecrites, 1) ; E0(Sem_Pures, Max) ; E0(Sem_Ecrites , 0) ;
end Stock_Global ;
```

**RÉPONSE 5**. Placer Stock_Local dans l'espace virtuel utilisateur accessible en mode privilégié présente
a) comme avantages : de protéger les stocks entre eux et contre une mauvaise utilisation par un processus non maître des cases, de permettre de sauvegarder le stock sur disque en même temps que le processus, ce qui permet d'économiser de la place de mémoire centrale.
b) comme inconvénient : de ne pas toujours être accessible en particulier quand le processus vient d'être élu et que le premier traitement du défaut de page a besoin de ce stock qui, n'étant pas résidant en mémoire centrale, peut encore être sur le disque de va et vient. Cela entraîne un nouveau défaut de page pendant le traitement du premier défaut de page.

Pour pallier cet inconvénient de défaut de page en cascade, il faut soit prévoir d'empiler les défauts de page et de les traiter dans le bon ordre, soit faire un préchargement du Stock_Local quand le processus participe à nouveau à la multiprogrammation (et donc redevient allocataire dans la stratégie globale de gestion conjointe de mémoire centrale et d'unité centrale).

**RÉPONSE 6.** Dans chaque partition associée à un Serveur, la politique de remplacement se fait à

l'ancienneté du chargement de page (FIFO). On peut améliorer cela par la méthode de l'horloge ou de la seconde chance telle qu'on l'a trouvée dans Multics dès 1968 et qu'on trouve aujourd'hui dans certains Unix, Solaris, Linux, OS9 de MacIntosh (voir cours chapitre 5).

**RÉPONSE 7.** Calculs

a) 9 rafales de taille 1 (position de départ : cylindre 100)

| ancienneté | | cylindre le plus proche | | ascenseur | |
|---|---|---|---|---|---|
| cylindre suivant | traversés | cylindre suivant | traversés | cylindre suivant | |
| 55 | 45 | 55 | 45 | 55 | 45 |
| 58 | 3 | 58 | 3 | 58 | 3 |
| 39 | 19 | 39 | 19 | 39 | 19 |
| 18 | 21 | 18 | 21 | 18 | 21 |
| 90 | 72 | 90 | 72 | 90 | 72 |
| 160 | 70 | 160 | 70 | 160 | 70 |
| 150 | 10 | 150 | 10 | 150 | 10 |
| 38 | 112 | 38 | 112 | 38 | 112 |
| 184 | 146 | 184 | 146 | 184 | 146 |
| total traversés | 498 | total traversés | 498 | total traversés | 498 |

b) 3 rafales de taille 3 (position de départ : cylindre 100)

| ancienneté | | cylindre le plus proche | | ascenseur | |
|---|---|---|---|---|---|
| cylindre suivant | traversés | cylindre suivant | traversés | cylindre suivant | |
| 55 | 45 | 58 | 42 | bas 58 | 42 |
| 58 | 3 | 55 | 3 | 55 | 3 |
| 39 | 19 | 39 | 16 | 39 | 16 |
| fin rafale1 | total rafale : 67 | fin rafale1 | total rafale : 61 | fin rafale1 | total rafale : 61 |
| | | | | | |
| 18 | 21 | 18 | 21 | bas 18 | 21 |
| 90 | 72 | 90 | 72 | haut 90 | 72 |
| 160 | 70 | 160 | 70 | 160 | 70 |
| fin rafale2 | total rafale : 163 | fin rafale2 | total rafale : 163 | fin rafale2 | total rafale : 163 |
| | | | | | |
| 150 | 10 | 150 | 10 | 184 | 24 |
| 38 | 112 | 184 | 34 | bas 150 | 34 |
| 184 | 146 | 38 | 146 | 38 | 112 |
| fin rafale3 | total rafale: 268 | fin rafale3 | total rafale : 190 | fin rafale3 | total rafale : 170 |
| | | | | | |
| rafale1 | 67 | rafale1 | 61 | rafale1 | 61 |
| rafale 2 | 163 | rafale 2 | 163 | rafale 2 | 163 |
| rafale 3 | 268 | rafale 3 | 190 | rafale 3 | 170 |
| total global | 498 | total global | 414 | total global | 394 |

c) 1 rafale de taille 9 (position de départ : cylindre 100)

| ancienneté | | cylindre le plus proche | | ascenseur | |
|---|---|---|---|---|---|
| cylindre suivant | traversés | cylindre suivant | traversés | cylindre suivant | traversés |
| 55 | 45 | 90 | 10 | haut 150 | 50 |
| 58 | 3 | 58 | 32 | 160 | 10 |
| 39 | 19 | 55 | 3 | 184 | 24 |
| 18 | 21 | 39 | 16 | bas 90 | 94 |
| 90 | 72 | 38 | 1 | 58 | 32 |
| 160 | 70 | 18 | 20 | 55 | 3 |
| 150 | 10 | 150 | 132 | 39 | 16 |
| 38 | 112 | 160 | 10 | 38 | 1 |
| 184 | 146 | 184 | 24 | 18 | 20 |
| total traversés | 498 | total traversés | 248 | total traversés | 250 |

Les calculs donnent les résultats suivants
a) 9 rafales de taille 1 : 498 cylindres traversés quelle que soit la politique (comme il n'y a jamais de requêtes en attente, il ne peut y avoir d'optimisation : toutes les politiques se valent)
b) 3 rafales de taille 3 : cylindres traversés , toujours 498 pour FIFO, mais 414 pour CPP et 394 pour l'ascenseur. On a un gain de 100/498 soit 20%.
c) 1 rafale de taille 9 : cylindres traversés , toujours 498 pour FIFO, mais 248 pour CPP et 250 pour l'ascenseur. On a un gain de 248/498 soit 50%.

Conclusion : En déduire qu'il faut que la taille d'une rafale soit assez grande pour que l'optimisation puisse jouer.
Si on fait une rafale de 5 suivie d'une rafale de 4, on obtient le même résultat qu'avec une rafale de 9. on a déjà un gain de 50% . Une rafale de 5 est déjà performant pour cet échantillon (attention, ce n'est qu'un échantillon favorable et non un résultat statistique).

d) 1 rafales de taille 5 puis une rafale de taille 4 (position de départ : cylindre 100)

| ancienneté | | cylindre le plus proche | | ascenseur | |
|---|---|---|---|---|---|
| cylindre suivant | traversés | cylindre suivant | traversés | cylindre suivant | traversés |
| 55 | 45 | 90 | 10 | haut bas 90 | 10 |
| 58 | 3 | 58 | 32 | 58 | 32 |
| 39 | 19 | 55 | 3 | 55 | 3 |
| 18 | 21 | 39 | 16 | 39 | 16 |
| 90 | 72 | 18 | 21 | bas 18 | 21 |
| fin rafale 1 | total rafale | fin rafale 1 | total rafale : 82 | fin rafale 1 | total rafale : 82 |
| 160 | 70 | 38 | 20 | haut 38 | 20 |
| 150 | 10 | 150 | 112 | 150 | 112 |
| 38 | 112 | 160 | 10 | 160 | 10 |
| 184 | 146 | 184 | 24 | 184 | 24 |
| fin rafale 2 | total rafale | fin rafale 2 | total rafale : 166 | fin rafale 2 | total rafale : 166 |
| rafale1 | | rafale1 | 82 | rafale1 | 82 |
| rafale 2 | | rafale 2 | 166 | rafale 2 | 166 |
| total traversés | 498 | total traversés | 248 | total traversés | 248 |