SYSTÈMES ET RÉSEAUX INFORMATIQUES

COURS B4 : HTO (19339) et ICPJ (21937)
CYCLE PROBATOIRE INFORMATIQUE
(Conception et développement informatique)

EXAMEN DU 18 JUIN 2005

portant sur l'enseignement des SYSTÈMES INFORMATIQUES

Date : samedi 18 juin 2004
Heure : 14h30 à 17h30
Lieu : au CNAM, rue Conté, salle 30.-1.02 et 30.-1.04

TOUS DOCUMENTS PAPIERS AUTORISES

(Les portables téléphoniques, les ordinateurs et les calculatrices ne sont pas autorisés)

****

Énoncé de 5 pages et une annexe.

L'examen comprend :
quatre questions sur la synchronisation
trois question sur la gestion des ressources

Chaque question peut être traitée indépendamment.

Ne pas écrire au crayon ni à l'encre rouge
(Sous peine de nullité, selon le règlement des examens)

****

Barème indicatif

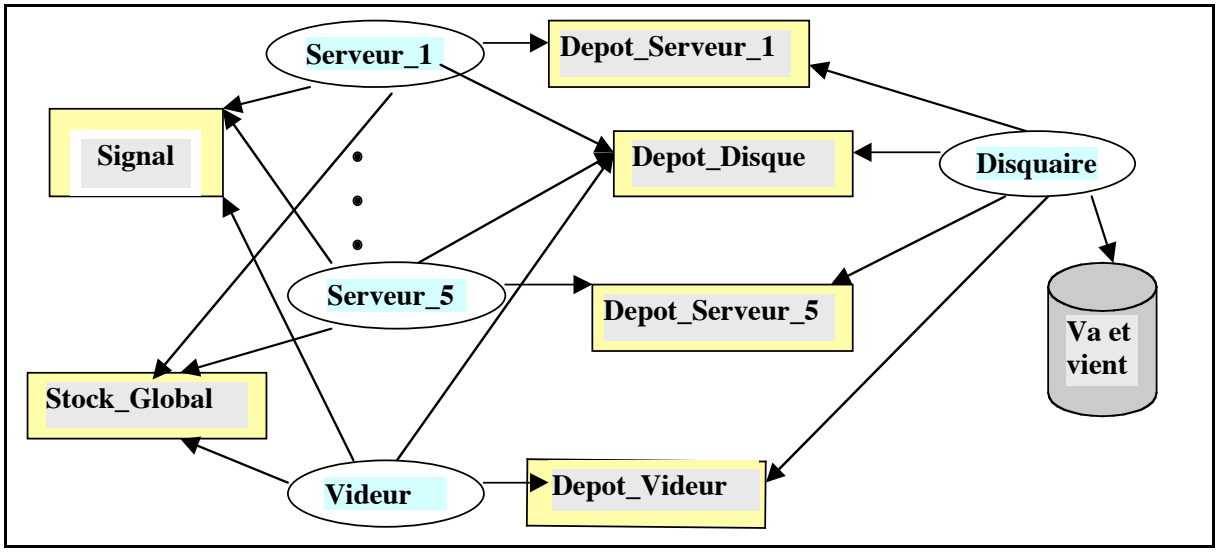
Question	1	2	3	4	5	6	7
Notation	3	3	3	3	2	3	4

Gestion de cases pour pagination à la demande

A. Synchronisation

1. Présentation et schéma des processus

On veut étudier une gestion simplifiée de cases pour un système de 5 processus Serveurs utilisant la pagination à la demande. Ce système organise la gestion des cases en faisant appel à un processus Videur et à un processus Disquaire. L'architecture logique des processus est la suivante.



Chaque **Serveur_I** peut faire des défauts de page. Pour traiter un défaut de page, le serveur utilise une case non-écrite qu'il obtient d'un Stock_Global de cases ou qu'il récupère par remplacement dans son Stock_Local de cases (il y a un Stock_Local associé à chaque serveur) et échange avec une page non-écrite du Stock_Global. Le Serveur_I envoie alors au processus **Disquaire** une demande de chargement de case avec la page manquante depuis le disque de va et vient. Cette demande est déposée dans la boîte à lettres **Depot_Disque** et la réponse disant que ce chargement a été fait est retirée de la boîte à lettres **Depot_Serveur_I**.

Le Stock_Global contient des cases écrites et des cases non-écrites (pures). Les Serveurs ne lui demandent que des cases non-écrites et ils lui rendent des cases écrites et des cases non-écrites. Pour maintenir un lot suffisant de cases non-écrites, un processus **Videur** est amené à demander au processus Disquaire de sauvegarder sur disque de va et vient le contenu de cases écrites et ainsi les rendre non-écrites. Le Videur dépose ses demandes de sauvegarde dans la boîte à lettres **Depot_Disque** et la réponse disant que la sauvegarde a été faite est retirée de la boîte à lettres **Depot_Videur**.

Le processus **Disquaire** gère le disque de va et vient ; il reçoit les demandes de transfert dans la boîte à lettres **Depot_Disque** et répond dans les boîtes à lettres **Depot_Videur** et **Depot_Serveur_I**.

Chaque case de mémoire de ce système est décrite par

```
type Descripteur_De_Case is record
    Ecrite : Boolean := False ; -- initialement non écrite
    Hote: Integer := 0 ; -- le videur est l'hôte 0, les serveurs sont les hôtes 1 à 5
    Case_Id : adresse physique de la case en mémoire centrale
    Les_Info : Info ;
    -- Info contient tout ce qui sert à gérer la case, les contenus de la table des pages de Hote,
    -- la page d'adresse virtuelle qui est mappée avec elle,
    -- l'adresse disque de l'information contenue dans la case, etc..
end record ; -- Descripteur_De_Case
```

2. Communication de messages entre processus

2.1. Les processus Serveurs, Videur et Disquaire

a) **Les Serveurs**. Suite à un défaut de page, un processus Serveur_I dépose dans Depot_Disque des demandes de chargement de cases en utilisant :

Depot_Disque.Deposer(Une_Case) ; -- Une_Case est un descripteur de case du Serveur_I.

Puis il attend le résultat du chargement en lisant sa boîte à lettres avec :

Depot_Serveur_I.Retirer(Une_Case).

b) **Le Videur**. Le processus Videur peut demander la sauvegarde sur le disque de va et vient du contenu de cases. Il fait cela en utilisant :

Depot_Disque.Deposer(Une_Case) ;

Puis il attend le résultat de chaque sauvegarde en lisant sa boîte à lettres avec :

Depot_Videur.Retirer(Une_Case).

c) Le Disquaire. Le processus Disquaire lit les demandes de transfert en utilisant :

Depot_Disque.Retirer(Une_Case) ;

Puis il gère les transferts disque en les optimisant éventuellement, et ensuite il renvoie la réponse dans la boîte à lettres du demandeur en utilisant l'un des appels de procédure :

Depot_Serveur_I.Deposer(Une_Case) ;

Ou :

Depot_Videur.Deposer(Une_Case).

Le choix entre ces deux appels est guidé par la valeur du champ Hote de Une_case.

2.2. Les paquetages implémentant les paradigmes

a) Les paquetages **Depot_Serveur_I** et **Depot_Videur** peuvent être réalisés selon le même paradigme et le même modèle d'implémentation. On se limite ici à Depot_Videur.

```
package Depot_Videur is
    procedure Deposer(X : in Descripteur_De_Case) ; -- le processus appelant dépose X
    procedure Retirer(X : out Descripteur_De_Case) : -- le processus appelant retire X
-- ce paquetage autorise la concurrence simultanée d'un accès pour Deposer et d'un accès pour Retirer
end Depot_Videur;
```

QUESTION 1. Expliquer comment ce modèle peut autoriser la concurrence simultanée d'un accès pour Deposer et d'un accès pour Retirer et pourquoi on peut n'avoir que les sémaphores de gestion de flux. On vous demande en conséquence de programmer le package body de Depot_Videur en assurant la synchronisation avec des sémaphores, en utilisant le moins possible de sémaphores, et en faisant en sorte que l'on puisse avoir la concurrence simultanée d'un accès pour Deposer et d'un accès pour Retirer. On utilise un tampon interne de taille 10. Ne pas oublier d'initialiser les sémaphores utilisés, sinon la solution est fausse.

b) Le paquetage **Depot_Disque**, par contre, doit être programmé différemment.

```
package Depot_Disque is
    procedure Deposer(X : in Descripteur_De_Case) ;
    procedure Retirer(X : out Descripteur_De_Case) ;
    -- pour indiquer le nombre de messages en attente dans Depot_Disque, on utilise :
    procedure  Combien_De_Messages(Compte : out Integer) ;
end Depot_Disque;
```

QUESTION 2. Expliquer pourquoi le modèle du package Depot_Videur n'est pas valable pour Depot_Disque. On vous demande de programmer le package body de Depot_Disque en assurant la synchronisation avec des sémaphores. On utilise un tampon interne de taille 10. Ne pas oublier d'initialiser les sémaphores utilisés, sinon la solution est fausse. Ne pas oublier la troisième procédure Combien_De_Messages. Peut-on faire en sorte que l'on puisse avoir la concurrence simultanée d'un accès pour Deposer et d'un accès pour Retirer?

3. Synchronisation entre les Serveurs et le Videur

3.1. Les processus Serveurs et Videur

Le vidage des pages est à la charge du processus **Videur** qui fonctionne en concurrence asynchrone et qui est déclenché par les **Serveurs**. Chaque **Serveur_I** utilise pour cela :

Signal.Envoyer ;

Le Videur attend le déclenchement et utilise :

Signal.Attendre ;

Si le Videur est déjà déclenché , il n'est pas déclenché une nouvelle fois.
À chaque déclenchement, le Videur envoie vers le processus Disquaire des cases écrites à sauvegarder.

3.2. Le paquetage Signal implémentant le paradigme

package **Signal** is
    procedure **Envoyer** ; -- enregistre une fois un signal provenant d'un ou de plusieurs Serveurs
    procedure **Attendre** ; -- bloque l'appelant (le Videur) en attente du signal
end **Signal** ;

La procédure Envoyer enregistre un signal. Si le signal est déjà enregistré, cette procédure ne fait rien. La procédure Attendre bloque l'appelant tant que le signal n'est pas pas arrivé. Si le signal est déjà enregistré, Attendre n'est pas bloquant. Quand Attendre s'exécute cela a pour effet d'annuler le signal enregistré.

QUESTION 3. On vous demande de programmer le package body en assurant la synchronisation avec des sémaphores. Ne pas oublier d'initialiser les sémaphores utilisés, sinon la solution est fausse.

4. Coopération pour gérer les stocks de cases

4.1. Les processus Serveurs et Videur

a) Les **Serveurs**. Un Serveur_I, pour gérer le défaut de pages, demande une case non écrite en utilisant

Stock_Global.Demander_Pure(Ma_Case, **Id_Serveur) ;** -- Id_Serveur est son nom unique

En fin d'exécution, il rend chaque case qu'il a utilisée, qu'elle soit écrite ou non écrite en appelant:

Stock_Global.Rendre(Ma_Case) ;

b) Le **Videur**. Le Stock_Global est surveillé par le Videur qui demande une case écrite par :

Stock_Global.Demander_Ecrite(Ma_Case, 0) ; -- 0 désigne le Videur

Pour ne pas être bloqué, il a demandé le compte de pages écrites par :

Stock_Global.Compter_Ecrites(Compte) ;

Il envoie les demandes de sauvegarde au Disquaire, puis une fois celles-ci faites, il remet chaque case (devenue pure) dans le Stock_Global en utilisant :

Stock_Global.Rendre(Ma_Case) ;

4.2. Le paquetage Stock_Global implémentant le paradigme

Package **Stock_Global** is
    procedure **Demander_Pure**(X : out Descripteur_De_Case; Y : in Integer) ;
    -- fournit au processus Serveur _Y une case non écrite et la sort du stock
    procedure **Demander_Ecrite**(X: out Descripteur_De_Case; Y : in Integer) ;
    -- fournit au processus Y une case écrite et la sort du stock ; Y est en fait le processus Videur
    procedure **Rendre**(X : in Descripteur_De_Case) ; -- un processus rend une case écrite ou non
    procedure **Compter_Ecrites**(Compte : out Integer) ; -- indique le nombre de cases écrites du stock
end **Stock_Global** ;

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

    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
        -- accès contrôlé à la gestion des cases pures
        X := Stock(Tete_Pure) ; Tete_Pure:= Tete_Pure - 1; -- on dépile  la pile du haut
        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
        -- accès contrôlé à la gestion des cases écrites
        X := Stock(Tete_Ecrite) ; Tete_ Ecrite:= Tete_ Ecrite + 1; -- on dépile  la pile du bas
        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
            Tete_ Ecrite:= Tete_ Ecrite - 1; Stock(Tete_Ecrite) := X ;
            -- indiquer au contrôle qu'il y a une case écrite de plus
        else
            -- accès contrôlé à la gestion des cases pures
            Tete_Pure:= Tete_Pure + 1; Stock(Tete_Pure) := X ;
            -- indiquer au contrôle qu'il y a une case pure de plus
        end if ;
    end Rendre;

    procedure Compter_Ecrites (Compte : out Integer) is
    begin
        -- accès contrôlé à la gestion des cases écrites
        Compte := …. ;
    end Compter_Ecrites;

begin
    for I in 1 .. Max loop
        La_Case.Case_Id:= -- 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
end **Stock_Global** ;

QUESTION 4 . On vous demande de programmer avec des sémaphores le contrôle d'utilisation concurrente des procédures de Stock_Global. Penser à programmer la possibilité de deux accès simultanés, l'un au lot des cases pures et l'autre au lot des cases écrites. Ne pas oublier d'initialiser les sémaphores utilisés, sinon la solution est fausse.

B. Gestion des ressources

1. Placement en mémoire virtuelle

QUESTION 5. Dans le contexte de pagination à la demande, on veut installer ces programmes concurrents dans un système de la famille Unix-Linux (voir chapitre 1 du cours). Les processus Videur et Disque sont dans l'espace du système résidant en mémoire centrale. Les processus Serveur s'exécutent en mémoire virtuelle dans l'espace utilisateur. On vous demande d'examiner si on peut placer aussi chaque Stock_Local dans l'espace utilisateur de chaque processus Serveur. Donner les avantages et les inconvénients de cette solution et indiquer des mécanismes à ajouter pour pallier ces inconvénients.

. Politique de remplacement de pages

QUESTION 6. Chaque processus Serveur gère le lot de cases de son Stock_Local avec un tampon circulaire géré par un pointeur de tête et un pointeur de queue. Il met en queue les cases reçues et, s'il faut faire du remplacement, choisit comme victime la case désignée par le pointeur de tête. Si cette case victime est une case écrite, elle est échangée contre une case non-écrite du Stock_Global. Quelle politique de remplacement de pages a-t-on programmé dans le Serveur? Comment pourrait-on l'améliorer sans ajouter de mécanisme câblé supplémentaire, en s'inspirant des méthodes utilisées dans des systèmes courants.

3. Politique de vidage et transfert disque.

Le processus Videur demande la recopie sur disque de cases écrites (vidage de cases). On sait que la performance de l'accès disque dépend de la politique choisie pour la gestion du bras et du nombre de requêtes en attente de transfert. On peut donc choisir le nombre de cases à demander à sauvegarder en fonction des performances potentielles de l'algorithme du disque. On a relevé une suite de références aux cylindres de disque, soit :

55, 58, 39, 18, 90, 160, 150, 38, 184

On veut l'utiliser pour comparer les transferts par différentes quantités appelées rafales, et on suppose qu'entre chaque rafale, le disque a pu traiter entièrement la rafale précédente demandée. On compare les rafales de taille 1 puis 3 puis 9. On le fait chaque fois avec les trois politiques FIFO (ancienneté), CPP (Cylindre le plus proche), Ascenseur (essayant de partir d'abord dans le dernier sens mémorisé ; le sens initial est vers les cylindres croissants). La position initiale du disque est le cylindre 100.

QUESTION 7. Faire les calculs en utilisant les tableaux donnés en annexe (remettre ces tableaux avec votre copie). Comparer les résultats et en particulier le nombre total de cylindres traversés dans les trois cas. On donnera dans chacun des trois cas le gain par rapport à la politique FIFO. Qu'en déduire comme valeur à choisir pour la taille d'une rafale?

Si vous en avez le temps, faites aussi les calculs pour une rafale de taille 5 suivie d'une rafale de taille 4. Vous verrez que c'est étonnant.

ANNEXE POUR LA QUESTION 7

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	
160	70	160	70	160	
150	10	150		150	
38	112	38		38	
184	146	184		184	
total traversés	498	total traversés		total traversés	

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			haut __	
160	70				
fin rafale2	total rafale : 163	fin rafale2	total rafale :___	fin rafale2	total rafale : ___
150	10				
38	112				
184	146				
fin rafale3	total rafale: 268	fin rafale3	total rafale: ___	fin rafale3	total rafale : ___
rafale1	67	rafale1	61	rafale1	61
rafale 2	163	rafale 2		rafale 2	
rafale 3	268	rafale 3		rafale 3	
total global	498	total global		total global	

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		90	10	haut 150	50
58		58	32	160	10
39		55	3	184	24
18					
90					
160					
150					
38					
184					
total traversés		total traversés		total traversés	

FEUILLE À REMPLIR POUR LA QUESTION 7 **numéro d'inscription :**
numéro de copie :

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	
160	70	160	70	160	
150	10	150		150	
38	112	38		38	
184	146	184		184	
total traversés	498	total traversés		total traversés	

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			haut __	
160	70				
fin rafale2	total rafale : 163	fin rafale2	total rafale :___	fin rafale2	total rafale : ___
150	10				
38	112				
184	146				
fin rafale3	total rafale: 268	fin rafale3	total rafale: ___	fin rafale3	total rafale : ___
rafale1	67	rafale1	61	rafale1	61
rafale 2	163	rafale 2		rafale 2	
rafale 3	268	rafale 3		rafale 3	
total global	498	total global		total global	

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		90	10	haut 150	50
58		58	32	160	10
39		55	3	184	24
18					
90					
160					
150					
38					
184					
total traversés		total traversés		total traversés	