CORRIGÉ de l'Examen du SRIB 17 juin 2000 partie systèmes
Indications de solution

PREMIER PROBLÈME : **LE JEU SIBER** (14 points)

1. ALLOCATION DYNAMIQUE DES CASES DU TABLEAU T

Question 1 (1pt)

Montrer qu'il peut y avoir interblocage entre les processus serveurs qui se partagent dynamiquement les 200 cases du tableau T.

Interblocage si toutes les cases sont allouées et tous les serveurs attendent une nouvelle case.
Par exemple, les 100 serveurs ont acquis chacun 2 cases.

Question 2 (2pts)

Rappeler le principe de l'algorithme simplifié. Montrer une situation où cet algorithme ne laisse qu'un seul des 100 serveurs s'exécuter et en fait attendre 99.

Il suffit de ne faire une allocation, que si on peut garder pour le processus le mieux pourvu le nombre suffisant de cases pour atteindre 4 cases. Alors, au moins ce processus pourra s'exécuter, puis libérer ses quatre cases.
Si 99 serveurs ont reçu 2 cases, que le centième n'a reçu aucune case, et s'ils demandent tous une case, on allouera une case à l'un des 99 serveurs et on réservera 1 case pour ce serveur et les autres attendront la fin de son exécution.

Question 3(2pts)

Montrer qu'il n'y a pas de danger d'interblocage et que parmi les 60 serveurs en session, 20 ne sont jamais bloqués.

Si on limite à 60 le nombre de serveurs concurrents en session, si ces serveurs ont obtenu 3 cases, au moins 20 pourront obtenir 4 cases. Il n'y a pas de danger d'interblocage.

2. PROGRAMMATION DU CONTRÔLE DE SESSION

Question 4(1pt)

Complétez les procédures `entree` *et* `sortie` *pour garantir le contrôle de session.*

```
package body session is
Session : semaphore;
  procedure entree is
  begin P(Session); end entree;
  procedure sortie is
  begin V(Session); end sortie;
begin E0(Session,60);
end session;
```

3. PROGRAMMATION DE L'ALLOCATION DYNAMIQUE DES CASES
Question 5 (3pts).

Programmer avec des sémaphores le maintien de la cohérence dans le paquetage Casier.

```
package body Casier is
type Case is record
   Nom : IdServeur;
   Obj : Objet;
   Co : Coup;
end record;
-- Une case C est libre si C.nom=0, occupée sinon
T : array(1..200) of Case;
S_Nbcase, Mutex_T : semaphore;
   procedure Inserer(UnObjet:in Objet;S:in IdServeur;K:in Coup);
   C:=integer; -- numero de case;
   begin
   P(S_Nbcase);
   P(Mutex_T );
   for i in 1..200 loop
     if T(i).nom=0 then C:=i; exit;
     endloop;
   T(C).nom:=S; T(C).Obj:=UnObjet;T(C).Co:=K;
   V(Mutex_T );
   end Inserer;

   procedure Recuperer(UnObjet:out Objet;S:in IdServeur;K:in Coup) is
   C:=integer; -- numero de case;
   begin
   P(Mutex_T );
   for i in 1..200 loop
     if T(i).nom:=S and T(i).Co:=K then C:=i; exit;
     end loop;
   UnObjet:=T(C).Obj;
   T(C).Nom:=0;
   V(Mutex_T );
   V(S_Nbcase);
   end Recuperer;

begin for i in 1..200 loop T(i).Nom:=0; end loop;
E0(S_Nbcase,200);E0(Mutex_T,1);
end Casier;
```

4. PROGRAMMATION DU STOCKAGE SUR DISQUE

Question 6 (3pts)

Programmer la coopération des processus par le paquetage Tampon et la gestion du tableau Tamp.

```
package body Tampon is
Tamp : array (0..5) of Message;
Tete, Queue : Integer;
Nplein,Nvide,Mutexprod : semaphore;
  procedure DebutSequence is
  begin
  P(Mutex_prod);
  end DebutSequence;

  procedure FinSequence is
  begin
  V(Mutex_prod);
  end FinSequence;

  procedure Deposer(M:in Message) is
  begin
  P(Nvide); Tamp(Queue):=M; Queue:=(Queue+1) mod 6;
  V(Nplein);
  end Deposer;
  procedure Retirer(M:out Message) is
  begin
  P(Nplein);
  M:=Tamp(Tete); tete:= (tete+1) mod 6:
  V(Nvide);
  end Retirer;
begin
Tete:= Queue:=0;
E0(Nplein,0); E0(Nvide,6); E0(Mutexprod,1);
end Tampon;
```

Question 7 (2pts)

Modifier en conséquence les procédures FinSequence et Deposer.

```
  procedure FinSequence is
  begin
  for i in 1..4 loop V(Nplein);end loop;
  V(Mutex_prod);
  end FinSequence;

  procedure Deposer(M:in Message) is
  begin
  P(Nvide); Tamp(Queue):=M; Queue:=Queue+1 mod 6;
  end Deposer;
```

DEUXIÈME PROBLÈME : **GESTION D'UN CACHE** (6 points)

Question 8 (2pts)

Rappeler le principe de fonctionnement d'un cache. Supposant que pour toute opération une image transite par le cache, expliquer pour chaque opération l'utilisation du cache.

Indications de réponse.
Fonctionnement du cache : si l'image est présente dans le cache, effectuer l'opération dans le cache; sinon rechercher une case vide du cache, s'il n'y a pas de case vide remplacer une image du cache.
Ajouter_image recherche une case dans le cache, écrit l'image dans la case
Retirer_image Si l'image est dans le cache, libérer la case correspondante
En fait, c'est plus compliqué : il faudrait maintenir une liste des images accessibles, ou marquer les images qui deviennent inaccessibles
Voir aussi question 9 où il est malheureusement dit que retirer_image provoque au préalable une lecture de l'image...
Visualiser_image Si l'image est dans le cache lire l'image, sinon rechercher une case et lire l'image à partir du disque dans le cache.

Question 9 (1pt)

Quels sont à votre avis les avantages et inconvénients de cette politique de remplacement?

Indications de réponse.
Avantages algorithme de remplacement très simple, et donc très rapide
Inconvénients S'il y a peu de libération de cases, on remplacera en général la page la plus récemment chargée;

Question 10 (1pt)

Proposer une amélioration de l'algorithme de remplacement.

Indications de réponse.
Lors de remplacement d'image, on ne réécrira l'image sur disque que si c'est une nouvelle image.

Question 11 (1pt)

Proposer un autre algorithme de remplacement utilisant cette date. Quels en sont à votre avis les avantages et les inconvénients?

Indications de réponse.
On peut proposer un algorithme de type LRU.
Avantages Evite des transferts disque. Valable si on a une localité des références aux images, si on consulte davantage les images récemment ajoutées par exemple.
Inconvénients Complexité de l'algorithme, structures de données à gérer etc...

Question 12 (1pt)

Quel algorithme de remplacement choisiriez-vous? Justifier votre réponse en considérant par exemple la fréquence des défauts de cache, la complexité des algorithmes.

Indications de réponse.
Si on double la taille du cache, il y aura moins de défauts de cache. La performance des algorithmes de remplacement aura moins d'incidence.