Master RSD – M1 **Le 11/01/2014**
Département d'Informatique, FEI, USTHB.

Examen de Système d'Exploitation

Exercice 1 :

Nous considérons un bus de transport en commun et les codes suivants:

Procédure AccèsBus()	**Procédure Bus()**
Début	Début
<Aller-vers-station> ;	<arrêt-station> ;
<Monter sur le bus> ;	<ouvrir-porte> ;
Fin	Tant que (Non (Délai Ecoulé))
	faire <Attente>; fait;
	<Fermer-Porte> ;
	<démarrer> ;
	Fin.

Par soucis de sécurité des clients,

- une seule porte est réservée à la montée des clients,
- un seul client peut monter à un instant donné,
- la porte ne doit pas gêner un client qui essaye de monter.

Afin de maintenir ces règles de sécurité, un mécanisme de synchronisation est nécessaire.

1. Définir les sections critiques (SC).
2. Donner les procédures d'accès et de sortie de ces SC en utilisant des sémaphores.

Une fois le délai d'attente du bus écoulé, l'accès au bus doit être bloqué pour permettre au bus de fermer ses portes et de repartir. Le bus fait passer alors un flag lumineux au rouge pour indiquer que le bus ne prend plus de passagers. Un client ne doit donc essayer de monter que si le flag n'est pas rouge.

3. Compléter les codes de sorte à prendre en en compte ces consignes supplémentaires.

Pour assurer le confort des passagers du bus et pour plus de sécurité,

- le nombre de passagers est limité à 20,
- et, pendant toute la durée d'attente du bus, la porte ne doit rester ouverte à un moment donné que s'il y a des clients qui veulent monter. Elle doit se fermer dès qu'il n'y a pas de clients qui veulent monter. Mais, lorsqu'au moins un client se présente de nouveau, elle doit encore s'ouvrir,...

4. Donner les codes modifiés à exécuter par les processus clients et bus.
5. Remplacer le code de la procédure AccèsBus() donnée ci-dessus par deux codes AccèsBusAdulte() et AccèsBusEnfants() de sorte que les enfants soient prioritaires sur les adultes pour l'accès au bus.

Exercice 2:

Nous considérons le SGF UNIX, nous supposons une version modifiée de l'adressage avec un adressage indexé à plusieurs niveaux, des blocs disque de taille 400 octets et des adresses disque sur 4 octets. Sure cette structure nous avons les adresses :

- a0 qui peut pointe 10 blocs contigus (certains de ces blocs peuvent être libres),
- a1 qui peut pointer une table d'indexes à 1 niveau,
- a2 qui peut pointer la racine d'une table d'indexes à deux niveaux,
- a3 qui peut pointer la racine d'une table d'indexes à trois niveaux,
- a4 qui peut pointer la racine d'une table d'indexes à quatre niveaux,

a0
a1
a2
a3
a4

1. Donner la taille maximale d'un fichier dans ce cas.
2. Calculer la taille maximale de l'espace réservé à l'adressage. Critiquer le résultat obtenu. Proposer une solution pour réduire cet espace.
3. Nous supposons un système de gestion de fichier avec buffer cache.
   - 3. 1. Donner les différents accès effectués pour accéder au bloc 1207. Représenter ces accès sur un schéma de la structure.
   - 3. 2. Quelle serait la taille minimale du buffer cache pour avoir un nombre d'accès disque optimal dans le cas d'un fichier de taille 1530 blocs.

Bon courage.

Corrigé Examen de Système d'Exploitation

Exercice 1 : (10pts)

Nous considérons un bus de transport en commun et les codes suivants:

```
Procédure AccèsBus()
Début
 <Aller-vers-station> ;
 <Monter sur le bus> ;
Fin
```

```
Procédure Bus()
Début
  <arrêt-station> ;
 <ouvrir-porte> ;
 Tant que (Non (Délai Ecoulé))
  faire        <Attente>;  fait;
 <Fermer-Porte> ;
 <démarrer> ;
Fin.
```

Par soucis de sécurité des clients,

- une seule porte est réservée à la montée des clients,
- un seul client peut monter à un instant donné,
- la porte ne doit pas gêner un client qui essaye de monter.

Afin de maintenir ces règles de sécurité, un mécanisme de synchronisation est nécessaire.

1. Définir les sections critiques (SC). → Sol : Voir procédure ci-dessous. // 1pt
2. Donner les procédures d'accès et de sortie de ces SC en utilisant des sémaphores. // 2pts

```
m : sémaphore binaire;
init(m ,1) ;

Procédure AccèsBus()
Début
 <Aller-vers-station> ;
P(m) ;      // AccèsSC
      <Monter sur le bus> ; //SC
V(m) ;     // SortieSC1() ;

Fin
```

```
Procédure Bus()
Début
  <arrêt-station> ;
 <ouvrir-porte> ;
 Tant que (Non (Délai Ecoulé))
  faire        <Attente>;  fait;
 P(m) ;  //Accès SC
   <Fermer-Porte> ;  //SC
 V(m) ;   //Sortie SC
 <démarrer> ;
Fin.
```

Une fois le délai d'attente du bus écoulé, l'accès au bus doit être bloqué pour permettre au bus de fermer ses portes et de repartir. Le bus fait passer alors un flag lumineux au rouge pour indiquer que le bus ne prend plus de passagers. Un client ne doit donc essayer de monter que si le flag n'est pas rouge.

3. Compléter les codes de sorte à prendre en en compte ces consignes supplémentaires. // 3pts

```
flag : sémaphore privé;
init (flag ,0) ;
Procédure AccèsBus()
Début
 <Aller-vers-station> ;
P(flag) ;     // attend  flag vert

   P(m) ; //un seul monte à la fois
   <Monter sur le bus> ;
   V(m) ;

V(flag) ; // libère accès si flag pas rouge
Fin
```

```
Procédure Bus()
Début
  <arrêt-station> ;
 <ouvrir-porte> ;
 V(flag);  // Flag Vert

 Tant que (Non (Délai Ecoulé)) faire  <Attente>;  fait;

 P(flag) ;  // Flag Rouge pas d'accès jusqu'à prochaine ouverture
   <Fermer-Porte> ;

   <démarrer> ;
Fin.
```

Pour assurer le confort des passagers du bus et pour plus de sécurité,

- le nombre de passagers est limité à 20,
- et, pendant toute la durée d'attente du bus, la porte ne doit rester ouverte à un moment donné que s'il y a des clients qui veulent monter. Elle doit se fermer dès qu'il n'y a pas de clients qui veulent monter. Mais, lorsqu'au moins un client se présente de nouveau, elle doit encore s'ouvrir,...

4. Donner les codes modifiés à exécuter par les processus clients et bus. // 2pts

```
n :sémaphore compteur ; e:sémaphore binaire;
flag : sémaphore privé; i :entier ;
init (flag ,0) ; init (n,20);   i=0 ;
Procédure AccèsBus()
Début
  <Aller-vers-station> ;

  P(n) ;

  P(e) ;  // ouverture si 1er d'un ensemble de clients
    i++ ; Si (i=1) alors V(open) fsi ;
  V(e) ;

  P(flag) ;
            <Monter sur le bus> ;
  V(flag) ;

  P(e) ; //libérer  fermeture  par le dernier du groupe
    i-- ; Si (i=0) alors V(close) ; fsi ;
  V(e) ;
Fin
```

```
Procédure Bus()
Début
  <arrêt-station> ;
Tant que (Non (Délai Ecoulé))   faire

  P(open) ;   // attendre clients pour ouvrir porte

       <ouvrir-porte> ;

  V(flag);



  P(close) ;  // attente montée  groupe clients avant de Fermer

  P(flag) ;
  <Fermer-Porte> ;
fait;

<démarrer> ;
Fin.
```

5. Remplacer le code de la procédure AccèsBus() donnée ci-dessus par deux codes AccèsBusAdulte() et AccèsBusEnfants() de sorte que les enfants soient prioritaires sur les adultes pour l'accès au bus. // 2pts

```
n :sémaphore compteur ; e:sémaphore binaire;
flag : sémaphore privé; i :entier ;
init (flag ,0) ; init (n,20);   i=0 ; nChild=0;
Procédure AccèsBusEnfants()
Début
  <Aller-vers-station> ;

  P(n) ;

  P(e) ;
    i++ ; Si (i=1) alors V(open) ; fsi ;
  V(e) ;

  P(m); //1er enfant bloque adultes si pas adulte sinon attend
   nChild++ ;    if nChild==1 then P(adulte) ;
  V(m)

  P(flag) ;
              <Monter sur le bus> ;
  V(flag) ;

  P(m);          //dernier enfant libère les adultes
   nChild--;   if nChild==0 then V(adulte) ;
  V(m)

  P(e) ;
  i-- ; Si (i=0) alors V(close);fsi;//libérerFermeturePorte
  V(e) ;
  Fin.
```

```
Procédure AccèsBusAdultes()
Début
  <Aller-vers-station> ;

  P(n) ;

  P(e) ;
    i++ ; Si (i=1) alors V(open) ; fsi ;
  V(e) ;

  P(adulte) ;



  P(flag) ;   //ici, on peut supprimer flag mais elle peuvent être
nécessaire si elle permettent l'affichage des signaux lumineux
            <Monter sur le bus> ;
  V(flag) ;

   V(adulte) ;



  P(e) ;
    i-- ; Si (i=0) alors V(close) ; fsi ; //libérer     fermeture porte
  V(e) ;
  Fin.
```

Exercice 2:

Nous considérons le SGF UNIX, nous supposons une version modifiée de l'adressage avec un adressage indexé à plusieurs niveaux, des blocs disque de taille 400 octets et des adresses disque sur 4 octets. Sur cette structure nous avons les adresses :

- a0 qui peut pointe 10 blocs contigus (certains de ces blocs peuvent être libres),
- a1 qui peut pointer une table d'indexes à 1 niveau,
- a2 qui peut pointer la racine d'une table d'indexes à deux niveaux,
- a3 qui peut pointer la racine d'une table d'indexes à trois niveaux,
- a4 qui peut pointer la racine d'une table d'indexes à quatre niveaux,

a0
a1
a2
a3
a4

1. Donner la taille maximale d'un fichier dans ce cas.
   (1pt) Taille= $10+100+100^2+100^3+100^4$
2. Calculer la taille maximale de l'espace réservé à l'adressage. Critiquer le résultat obtenu. Proposer une solution pour réduire cet espace.
   (1pt)TailleAdressage=$1+(1+100)+(1+100+100^{2})+(1+100+100^2+100^{3})+(1+100+100^2+100^3+100^{4})$
   (1pt) Critique : taille importante des adresses ; elle correspond pratiquement au nb de bloc du fichier
   (2pts) Solution : une est de faire que chaque adresse d'une table d'indexe pointe une série de blocs du fichier (qui peuvent être contigües et alors les extensions seraient de même ou une chaîne de blocs du fichier)
3. Nous supposons un système de gestion de fichier avec buffer cache.
   3. 1. Donner les différents accès effectués pour accéder au bloc 1207. Représenter ces accès sur un schéma de la structure.
   (2pts)
   1207 -10- 100 = 1097 ce qui est < à 10110 donc, on aura un adressage à 2 niveaux
   - accès répertoire,
   - accès inode modifié
   - accès T0 table Index racine de l'adressage indirect à 2 niveau,
   - récupère 11èmme entrée de T0, accès T1 table index pointée par cette adresse,
   - récupérer 97èmme entrée de T1, bloc 1207.

   Donc : 5 (2+ 2 + 1) accès
   (1pt) Schéma structure .....
   3. 2. Quelle serait la taille minimale du buffer cache pour avoir un nombre d'accès disque optimal dans le cas d'un fichier de taille 1530 blocs.
   (2pts)
   - Répertoire
   - Inodé
   - Table d'index au nb max de 3 à la fois (pour l'adressage indexé 2 niveaux)
   - Donc 5= 2+2+ 1 (1 pour le bloc à lire si on veut faire plusieurs accès au même bloc).