Université de Chlef Janvier 2018

Département Informatique
Filière : 1 ère Année Master - Ingénierie des Logiciels

Examen semestriel

Module de Systèmes d'exploitation

Durée : 1H30

Exercice 1 : ***(7 points)***

On considère le système de gestion de fichiers de Unix. La taille d'un bloc de données est de 2 ko (kilo-octets). Chaque pointeur (numéro de bloc) occupe 4 octets. Chaque inode comprend 10 liens directs, 1 lien indirect simple, 1 lien indirect double et 1 lien indirect triple.

1/ Quel est le rôle de l'inode dans ce système ? .

Réponse :
L'inode est une structure de données permettant d'accéder aux détails d'un fichier sur le système Unix. Chaque fichier a son propre inode qui contient des métadonnées pour la description du fichier (-l'identifiant du propriétaire , l'identifiant du groupe propriétaire , -le type de l'inode , -les droits d'accès, la date de dernière modification du fichier, ...etc,) mais surtout des liens vers les blocs de données composant le fichier.
(1 point)

2/ Combien de blocs de données et de blocs de liens sont nécessaires pour représenter un fichier ayant une taille de 600 Ko ? Justifiez avec un schéma.

Réponse :

Nombre de blocs de données : 300
(0.5 point)

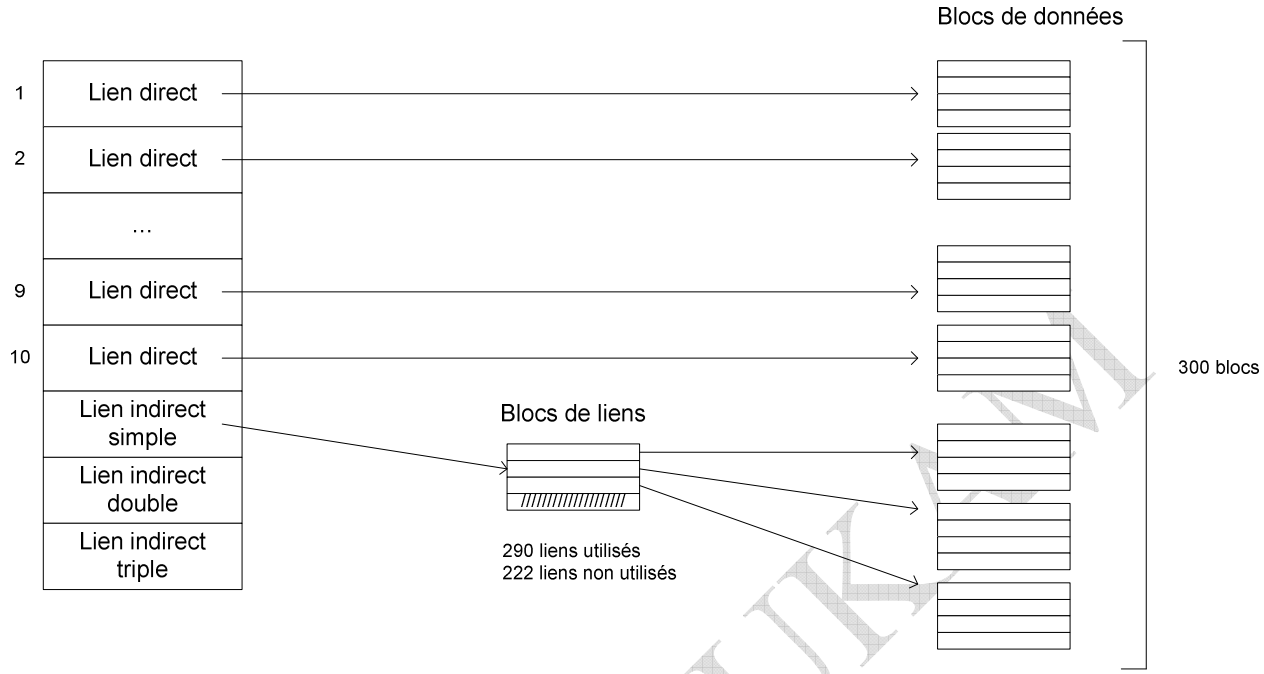
Nombre de blocs de liens : 1
(0.5 point)

Justification :

Chaque bloc occupe 2 Ko. Pour sauvegarder les données d'un fichier de 600 Ko, il faut donc 600/2 soit 300 blocs.
Chaque bloc de liens contient 2048/4, soit 512 liens.

Pour assurer les 300 liens nécessaires à ce fichier, on aura besoin de :
10 liens directs(aucun bloc de liens n'est utilisé)
300-10 soit 290 liens qu'on puisera dans le seul bloc d'indirection simple.

Dr Mourad LOUKAM



(0.5 point)

3/ Même question pour un fichier de 500.000 Ko ?.

Réponse :

Nombre de blocs de données : 250.000

(0.5 point)

Nombre de blocs de liens : 490

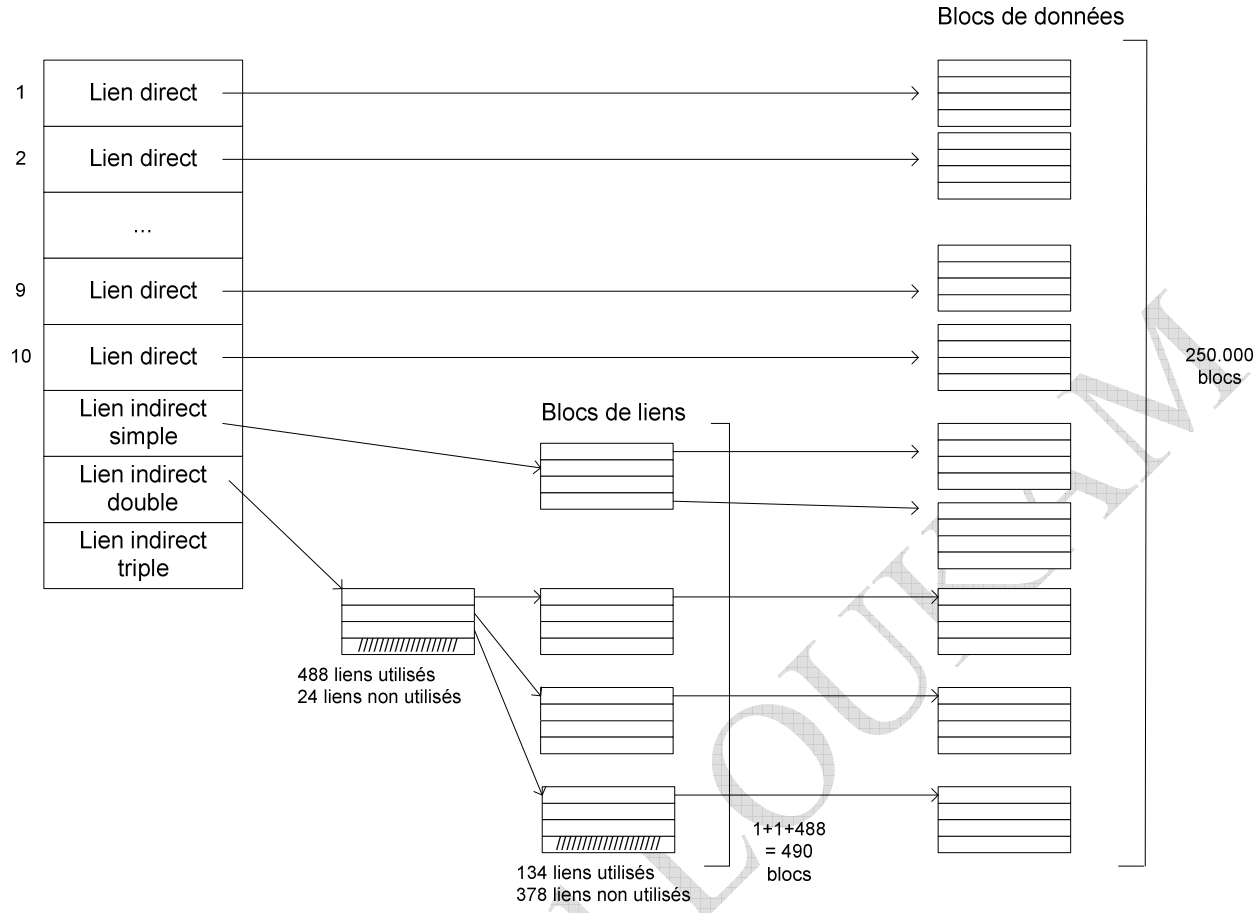
(0.5 point)

Justification :

Pour sauvegarder les données d'un fichier de 500.000 Ko, il faut 500.000/2 soit 250.000 blocs.

Pour assurer les 250.000 liens nécessaires à ce fichier, on aura besoin de :
10 liens directs(aucun bloc de liens n'est utilisé)
512 liens présents sur le seul bloc de liens indirects simples
249.478 liens présents sur 488 blocs de liens d'indirection double.

Dr Mourad LOUKAM



(0.5 point)

4/ Quelle est la taille minimale que doit avoir un fichier pour qu'on soit obligé d'utiliser le lien indirect triple ?. Justifiez.

Réponse :

Taille minimale que doit avoir le fichier : 537.939.969 octets.

(0.5 point)

Justification :

Pour qu'on soit obligé d'utiliser le lien indirect triple, il faut que les liens directs, indirects simples et doubles soient saturés. Cela donne :
10 liens directs → 10 blocs de données
512 liens indirects simple → 512 blocs de données
512 x 512 liens indirects double → 262.144 blocs de données

Pour qu'on soit obligé d'utiliser le lien indirect triple, la taille du fichier doit donc être au minimum égale à :

(10 + 512 + 262.144) x 2048 +1 , soit 537.939.969 octets.

(1 point)

Dr Mourad LOUKAM

5/ Quelle est la taille maximale d'un fichier qu'on peut représenter avec cette organisation ? Quel serait le nombre de blocs de données et de blocs de liens ?. Justifiez.

Réponse :

Taille max du fichier : 268.960.788 Ko
Nombre de bloc de données : 134.480.394
Nombre de blocs de liens : 263.171

(0.5 point)

Justification :

La taille max d'un fichier est obtenue lorsque tous les liens (directs et indirects) sont saturés. Cela donne :
10 liens directs → 10 blocs de données
512 liens indirects simple → 512 blocs de données
512 x 512 liens indirects double → 262.144 blocs de données
512 x 512 x 512 liens indirects triples → 134.217.728 blocs de données

Au total , nous avons :
(10 + 512 + 262.144 + 134.217.728) soit 134.480.394 blocs de données
(1 + 1 + 512 + 1 + 512 + 512 x 512) soit : 263.171 blocs de liens
La taille max du fichier est égale à :134.480.394 x 2, soit 268.960.788 Ko.

(1 point)

Exercice 2 : *(7 points)*

Résoudre le problème des Lecteurs-Rédacteurs en utilisant les moniteurs de Hoare et en donnant une priorité aux rédacteurs.

Réponse :

```
Processus Lecteur_i
Debut
        Fichier.DebutLire

        Lire

        Fichier.FinLire
Fin.
```

(0.5 point)

```
Processus Redacteur_j
Debut
        Fichier.DebutEcrire

        Ecrire

        Fichier.FinEcrire
Fin.
```

(0.5 point)

Dr Mourad LOUKAM

```
Moniteur = Fichier

Var Ecrire : Boolean;          //Variable logique pour indiquer si le fichier est en écriture
    Lire  : Boolean ;          //Variable logique pour indiquer si le fichier est en lecture
    NbLecteur : Integer;       //Variable entière pour compter le nombre de lecteurs
    NbRedacteur : Integer ; //Variable entière pour compter le nombre de rédacteurs
    CEcrire , Clire : Condition ; //Variables conditionnelles pour bloquer les rédacteurs et les lecteurs.

Procedure Entry DebutLire
Begin

If Ecrire Or NBRedacteur>0 then CLire.wait;
NbLecteur := NbLecteur +1 ;
Lire := True ;
CLire.signal

End ;

Procedure Entry FinLire
Begin
NBLecteur := NBLecteur – 1 ;
If NBLecteur = 0 then Begin Lire :=False ; CEcrire.Signal
                       End

End

Procedure Entry DebutEcrire
Begin
NBRedacteur := NBRedacteur + 1 ;
If (Ecrire)  or (Lire) Then CEcrire.wait
Ecrire := True
End

Procedure Entry FinEcrire
Begin
Ecrire :=False
NBRedacteur := NBRedacteur - 1
If NBRedacteur>0 then CEcrire.Signal
                 Else CLire.Signal
End

Begin /* Initialisation */
      Ecrire :=False ; Lire := False ;
      NBLecteur := 0 ; NBRedacteur :=0
End.
```

(06 Points)

Exercice 3 : ***(6 points)***

Ecrire en java le code de deux threads simultanés qui affichent chacun les entiers de 1 à 100. L'affichage doit être alterné, comme suit :

Thread 1 : 1
Thread 2 : 1

Dr Mourad LOUKAM

Thread 1 : 2
Thread 2 : 2
Thread 1 : 3
Thread 2 : 3
Thread 1 : 4
Thread 2 : 4
.....

Soignez votre code et vos déclarations.

Réponse :

```
package java_threads_afficheur;

/**
 *
 * @Dr Mourad LOUKAM exam, January 2018
 */
import java.util.concurrent.Semaphore;



public class Java_Threads_Exam2018 {

   static Semaphore S1 = new Semaphore(1);
   static Semaphore S2 = new Semaphore(0);

   static class Thread1 extends Thread  {

      // On redéfinit la méthode run()
      public void run () {
         for (int i=1; i<=100 ; i++) {
            try {
               S1.acquire();
               System.out.println("Thread 1 : " +i);
               S2.release();
            } catch (InterruptedException e) { return; }
         }
      }
   }


   static class Thread2 extends Thread  {

      // On redéfinit la méthode run()
      public void run () {
         for (int i=1; i<=100 ; i++) {
            try {
               S2.acquire();
               System.out.println("Thread 2 : " +i);
               S1.release();
            } catch (InterruptedException e) { return; }
         }
      }
   }
```

Dr Mourad LOUKAM

```
    public static void main(String[] args) {
     // On instancie les threads
        Thread1 th1 = new Thread1 ();
        Thread2 th2 = new Thread2 ();

        // On démarre les deux threads
        th1.start();
        th2.start();
    }
}
```

(06 Points)

Dr Mourad LOUKAM