CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

U.B.S. - I.U.T. de Vannes

Département Informatique
Contrôle de 2e année
Date : 17/10/2016

M3101 - Principe des systèmes d'exploitation

Devoir surveillé

Responsable(s) et auteur(s) : F. Merciol – M. Le Lain

Documents personnels et calculatrice autorisés
Téléphone interdit — Durée : 2h
Les barèmes sont donnés à titre indicatif — 9 page(s) de texte

---

Conseils :

- Indiquez **votre nom** sur chaque feuille à rendre **dès qu'elles vous sont données**.
- Afin d'éviter la copie, toute réponse non justifiée sera considérée comme nulle.
- **Les parties sont indépendantes** et corrigé par des enseignants différents (commencez par la plus simple pour vous).
- Rendez les parties séparément.
- Il est demandé des réponses à la fois claires et concises.
- **Lisez en entier** le contrôle avant de commencer à répondre.
- **Ne restez pas bloqué**, vous pourrez revenir sur une question difficile par la suite.
- **Ne brûlez pas toute votre énergie**, des calculs longs peuvent vous rapporter moins de points que la réponse à des questions de réflexion.
- **Conservez du temps pour chaque partie**
  (120 min / 20 pts => pas plus de 6 minutes par point)

Le devoir se compose de 3 parties :

1) [4 pts] questions de cours (QCM : Questions à Choix Multiple)
2) [8 pts] résolution de problème sur les tâches (en java)
3) [8 pts] Expressions régulières

Ce corrigé est donné à titre indicatif. D'autres réponses peuvent être considérés comme justes.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

1.) [4 pts] question de cours (QCM : Questions à Choix Multiple)

✗ Cochez la case réponse

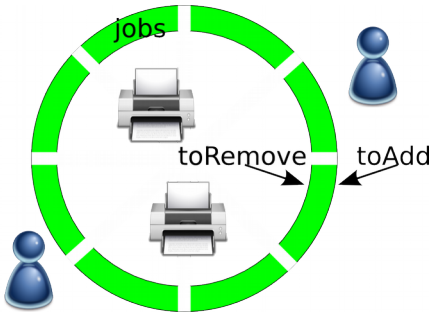
Question	✾	✱	❀	✲
Un processus démon ne peut jamais être : ✾) créé par un moldu ✾) défunt ❀) suspendu ✲) tué		○		
Le SFINX est ✾) un routeur aléatoire ✾) un animal mythique ❀) l'interconnexion des réseaux en France ✲) le protocole d'échange égyptien			○	
La mémoire flash ✾) est une forme d'EEPROM ✾) permet de modifier un octet ❀) ne dure pas longtemps ✲) sert à étaler sa vie privée	○			
L'algorithme de suppression de page LRU : ✾) tient compte du dernier utilisé ✾) prive de ressources ❀) prend en file ✲) vide le cache	○			
Un processus terminé dont le père n'a pas encore lu le testament se nomme : ✾) zombie ✾) orphelin ❀) en attente ✲) accidenté	○			
MBR veux dire ✾) Mon Bien Réutilisable ✾) Master Boot Record ❀) Manual Before Repair ✲) Memory Boot Recover		○		
Concernant l'emploi hors État-Unis d'Amérique, l'Internet ✾) détruit des emplois ✾) développe le commerce local ❀) créer des libraires ✲) repeuple les campagnes	○			
Le PID désigne : ✾) la Pile Interne des Données ✾) un processus ❀) un programme ✲) un processeur		○		
Laquelle de ces adresses réseaux est une classe C privée ? ✾) 10.128.192.0/24 ✾) 192.128.168.0/24 ❀) 172.16.197.0/24 ✲) 192.168.128.0/24				○
StringTokenizer découpe en ✾) paragraphes ✾) lignes ❀) anneaux magiques ✲) mots				○
Pour éviter les réseaux commerciaux de collecte de données personnelles (comme FaceBook), il faut ✾) ne pas consulter les pages avec un f ✾) interdire les cookies ❀) créer un profil restreint ✲) faire un vœux	○			
Quel est le cycle énergétique le plus court ? ✾) Le nucléaire ✾) Le pétrole ❀) Le papier ✲) Celui de la comète de Halley			○	
Quel est le marquage de fin de ligne sous Unix ? ✾) \n ✾) \r\n ❀) \n\r ✲) \r	○			
En informatique, l'acronyme RAID désigne : ✾) un regroupement de disques ✾) une unité de police ❀) un algorithme de recherche ✲) une épreuve sportive	○			
On qualifiera une application de "temps réel" si des actions y peuvent être : ✾) en moins d'1 ms ✾) abandonnées ❀) en moins d'1 ns ✲) en moins d'1 s		○		
L'électricité est un vecteur de transport d'énergie. Mais dans le monde, quelle source d'énergie est la moins utilisée par l'Internet ? ✾) Le charbon ✾) Le renouvelable ❀) Le nucléaire ✲) Les gaz de schiste		○		
Un programme Java qui ouvre une fenêtre se termine ✾) à la fin du main ✾) par un banquet ❀) dans un temps déterminé ✲) par appel à System.exit				○
Les premiers systèmes de gestion de fichiers sont apparus dans les années : ✾) 90 ✾) 70 ❀) 60 ✲) 80			○	
Les objets Java ou C++ créés dans un processus sont placés dans quelle mémoire ? ✾) données ✾) tas ❀) pile ✲) code		○		
Les premières mémoires électroniques (sans consommation d'énergie une fois programmées), utilisaient des ✾) éléphants ✾) rayons de lumière ❀) anneaux magnétiques ✲) petites réserves d'eau			○	

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

2.) [8 pts] Résolution de problème sur les tâches (en java)

«Chacun son tour »



Nous nous proposons de gérer une file d'impression en réseaux. Il y a plusieurs utilisateurs et plusieurs imprimantes. Chaque utilisateur peut déposer un document à imprimer et reçoit un numéro de traitement. Chaque imprimante retire un numéro de traitement et imprime le document correspondant.

Pour l'exercice nous ne gérons pas les documents, mais uniquement les numéros de traitement (« job »). La file d'attente d'impression est limitée en taille (« maxToPrint »). Nous représentons le tableau des derniers numéros de traitement dans un anneau.

Un utilisateur déposera son document dans la case pointé par « toAdd » et incrémentera ce pointeur.

Une imprimante recherchera à imprimer le document pointé par « toRemove » et incrémentera ce pointeur.

Il n'est pas possible d'imprimer un document si « toAdd » et « toRemove » pointent vers la même case (phénomène de file vide : aucun document n'est à imprimer).

Il n'est pas possible d'envoyer un document à imprimer si « toAdd » se trouve juste avant « toRemove » (phénomène de file pleine : trop de document sont en attente).

Le caractère unique du numéro de traitement est garanti par la variable « nextPrinterJob ».

Voici la classe Java qui vous est fournie (l'algorithme vous est offert) :

```
  public class PrinterPool {

    private int nextPrinterJob, maxToPrint;
    private int[] jobs;
5   private int toRemove, toAdd;

    public PrinterPool (int maxToPrint) {
      this.maxToPrint = maxToPrint;
      jobs = new int[maxToPrint];
10    for (int i = 0; i < maxToPrint; i++)
        jobs[i] = -1;
    }

    public int sendDocument () {
15    int job = nextPrinterJob++;
      jobs[toAdd++%maxToPrint] = job;
      return job;
    }

20  public int printDocument () {
      int job = jobs[toRemove++%maxToPrint];
      return job;
    }
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

```
    public class Printer extends Thread {
      public Printer (String name) {
        super (name);
        run ();
      }
      public void run () {
        for (int i = 0; i < 10; i++) {
          System.out.println ("job-"+printDocument ()+" printed by "+getName ());
          pause (.5);
        }
      }
    }
```

```
    public class User extends Thread {
      public User (String name) {
        super (name);
        run ();
      }
      public void run () {
        for (int i = 0; i < 10; i++) {
          System.out.println (getName ()+" wait to print job-"+sendDocument ());
          pause (.5);
        }
      }
    }
```

```
    public void pause (double secondes) {
      try {
        Thread.sleep ((int)(secondes*1000));
      } catch (InterruptedException e) {
      }
    }
    public static void main (String[] arg) {
      PrinterPool pp = new PrinterPool (10);
      for (int i = 0; i < 3; i++) {
        pp.new Printer ("printer-"+i);
        pp.new User ("user-"+i);
      }
    }
  }
```

La classe principale contient deux classe interne « Printer » et « User ». Pour pouvoir créer des objets des sous objets dans un objet englobant « pp », nous utilisons l'instruction new dans le contexte de l'objet qui contiendra ces nouveaux éléments. La syntaxe est la suivante :

```
        pp.new Maclasse();
```

Corrigé (Les réponses apparaissent avec ce fond de caractère)

Compréhension du code.

2.A) [½ pt] A quoi sert la méthode « pause » ? Pourquoi peut-on l'utiliser dans un objet « Printer » ?

La méthode pause permet d'attendre un nombre de secondes en paramètre. Un objet Printer peut invoquer toutes les méthodes défini dans une classe englobante.

2.B) [½ pt] Où est définie la méthode « getName » de la classe « Printer » ? Quelle valeur est afficher ?

La méthode est héritée de la classe Thread. La valeur affichée est celle fournie au constructeur Printer.

On exécute ce code en ne prenant que les 15 premières lignes triées par ordre alphabétique de la façon suivante :

```
java printerPool | sort | head -10
```

On obtient la trace :

```
job--1 printed by printer-0
job--1 printed by printer-0
job--1 printed by printer-1
job--1 printed by printer-2
job-0 printed by printer-1
job-1 printed by printer-2
job-11 printed by printer-0
job-12 printed by printer-1
job-14 printed by printer-0
job-14 printed by printer-2
```

2.C) [½ pt] Pourquoi les premiers documents imprimés ont-ils le numéro « -1 » ?

Les imprimantes ont démaré alors que la file était vide

2.D) [½ pt] Pourquoi le document « 14 » est-il imprimé 2 fois ?

Les imprimantes « 0 » et « 2 » ont utilisé la variable « toRemove » en même temps avant de l'incrémenter.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

Pour pouvoir fonctionner, ce code a été corrigé. Dans la version que vous lisez les objets « Printer » et « User » s'exécutaient entièrement successivement sans être entrelacé.

2.E) [1 pt] Quelles lignes faut-il corriger pour une exécution en parallèle ? Pourquoi ? Par quelle instruction faut-il corriger ?

Il faut corriger les lignes 27 et 39.

La méthode « run » ne crée pas de tâches séparées.

Il faut remplacer « run » par « start ».

2.F) [1 pt] De quelles autres manières peut-on créer des tâches en Java ?

Ici la création s'est faite par héritage. On peut également procéder par héritage de classe anonyme. On peut aussi créer une tâche par implentation de l'interface « Runnable ».

2.G) [1 pt] Comment modifier le code pour que l'ajout et le retrait de document soit en exclusion mutuelle ?

Il faut définir les méthodes « sendDocument » et « printDocument » comme synchronisées.

```
public synchronized int sendDocument () {
public synchronized int printDocument () {
```

2.H) [1 pt] Dans « sendDocument », ajouter le code nécessaire pour attendre que la pile ne soit pas pleine (indiquez la ligne où insérer ce code).

Il faut attendre qu'au moins une case soit libre. Dans le cas contraire on fait « wait ». Le code doit être ajouter en ligne 15 ou 16, avant la lecture du tableau.

```
while (toAdd-toRemove == maxToPrint-1)
    try {
      wait ();
    } catch (InterruptedException e) {
    }
```

CORRIGÉ (LES RÉPONSES APPARAISSENT AVEC CE FOND DE CARACTÈRE)

2.I) [1 pt] Dans « printDocument », ajouter le code nécessaire pour attendre que la pile ne soit pas vide (indiquez la ligne où insérer ce code).

Le code doit être ajouter en ligne 16 avant le « return »

```
while (toAdd-toRemove == maxToPrint-1)
    try {
       wait ();
    } catch (InterruptedException e) {
    }
```

2.J) [1 pt] A quel moment faut-il réexaminer les condition pile vide / pile pleine ?Quel code faut-il ajouter pour prévenir de cet évènement (indiquer les instructions et à quelle ligne les insérer).

L'état change lorque l'on ajoute (pile devenant non vide) ou que l'on retire (pile devenant non pleine) un document.

il faut ajouter un notifyAll dans chaque méthodes synchronizées.

avant les « returns » lignes 17 et 22.

Corrigé (Les réponses apparaissent avec ce fond de caractère)

Corrigé (Les réponses apparaissent avec ce fond de caractère)

NOM : _____ PRENOM : _____ GROUPE TD : ____ Note :

3.) [8 pts] Expressions régulières