Pale INF422 – Composants d'un système informatique
INF422 Final Exam – Components of a Computing System

2010–2011

- L'examen dure 3 heures.
  The exam lasts 3 hours.
- Tous documents autorisés.
  All documents are authorized.
- Il est impératif de commenter les programmes et de justifier vos réponses.
  Programs must be commented and every answer must be justified.

Sujet en français

1 Questions diverses

Il s'agit d'un questionnaire à choix multiple. Écrivez sur votre copie le numéro de la réponse choisie. Inutile de justifier vos réponses pour cet exercice.

Notation : **1** point par bonne réponse, **0** points en l'absence de réponse, $-\mathbf{0},\mathbf{5}$ points par réponse fausse. La note obtenue entre $-\mathbf{3}$ et **6** est ensuite ramenée au nombre de points accordé à l'exercice dans le barème.

Question 1.1

Trois des commandes suivantes ont pour effet de lister les processus dont le nom contient la sous-chaîne `virus` mais ne contient pas la sous-chaîne `magic`. Laquelle n'en fait pas partie ?

(1) `ps -ef | grep virus | grep -v magic`
(2) `ps -ef > fic && grep virus < fic | grep -v magic`
(3) `ps -ef > fic || grep virus < fic | grep -v magic`
(4) `ps -ef > fic ; grep virus < fic | grep -v magic`

Question 1.2

À quoi sert la variable `PATH` ?

(1) à indiquer au système où chercher des fichiers exécutables ;
(2) à spécifier le répertoire d'accueil de l'utilisateur ;
(3) à indiquer au système où chercher des fichiers texte ;
(4) à indiquer au système où chercher des fichiers de données.

Question 1.3

Que fait le script suivant lors qu'il est exécuté avec l'argument `'ls*'` et que la variable `PATH` vaut `/bin:/sbin` ?

```
#!/bin/sh

for program in "$@"; do
  list=‘echo $PATH | sed -e 's/:/ /g’‘
  for directory in $list; do
    for file in $directory/$program; do
      [ -x "$file" ] && echo $file
    done
  done
done
```

(1) il produit une erreur de syntaxe ;
(2) il affiche dans cet ordre `/bin/ls`, `/bin/lspci`, `/sbin/lsmod` entre autres choses ;
(3) il affiche dans cet ordre `/sbin/lsmod`, `/bin/ls`, `/bin/lspci` entre autres choses ;
(4) il n'affiche rien du tout.

Question 1.4

Qu'est-ce que la projection des entrées-sorties en mémoire (*memory-mapped I/O*) ?

(1) un mécanisme permettant de séparer le code des données des processus ;
(2) un mécanisme permettant de communiquer avec les périphériques ;
(3) un mécanisme pour la traduction des adresses virtuelles en adresses physiques ;
(4) une implémentation des entrées-sorties dans la machine virtuelle Java.

Question 1.5

Qu'est-ce que `resUpdate` dans le fragment de code Java suivant ?

```
final Runnable resUpdate = new Runnable() {
    public void run() {
        textViewResult.setText("Number of primes: " + nbPrimes);
    }
};
```

(1) une instance de l'interface `Runnable`, dont la méthode `run()` a été implémentée ;
(2) un thread, instance de l'interface `Runnable`, dont la méthode `run()` a été implémentée ;
(3) un thread, instance de la classe `Thread`, dont la méthode `run()` a été surchargée ;
(4) une méthode.

Question 1.6

On suppose que le nombre décimal 1000 est codé en représentation *little-endian* sur 16 bits à l'adresse mémoire 42. Quelles sont les valeurs hexadécimales des octets d'adresse 42 et 43 ?

(1) 42 : `0x00` ; 43 : `0x10` ;
(2) 42 : `0x10` ; 43 : `0x00` ;
(3) 42 : `0xE8` ; 43 : `0x03` ;
(4) 42 : `0x03` ; 43 : `0xE8`.

2 Système de fichiers

La commande `ls -al pale_inf422` affiche les lignes suivantes :

```
drwxr-----  3 acohen profs  4096 Nov  6 22:11 .
drwxr-xr-x 17 acohen profs  4096 Nov  6 21:58 ..
-rw-r--r--  1 acohen profs 16224 Nov  6 22:11 pale.tex
drwxr-xr-x  6 acohen profs  4096 Nov  6 22:01 .svn
```

Question 2.1

Listez le ou les sous-répertoire(s) de `pale_inf422`.

Question 2.2

Combien de sous-répertoires le répertoire courant contient il (i.e., le parent de `pale_inf422`) ? Justifiez votre réponse.

Question 2.3

Un autre utilisateur du groupe `profs` peut-il lister le contenu du répertoire `pale_inf422` ? Entrer dans ce répertoire ? Lire le fichier `pale.tex` ? Justifiez votre réponse.

Question 2.4

Un utilisateur du groupe `eleves` peut-il lister le contenu du répertoire `pale_inf422` ? Entrer dans ce répertoire ? Lire le fichier `pale.tex` ? Justifiez votre réponse.

3 Produit scalaire

On souhaite accélérer le calcul du produit scalaire de 2 vecteurs d'entiers `x` et `y` de 10000 éléments. Le processeur ciblé possède 2 coeurs et peut exécuter 2 threads en parallèle.

Question 3.1

Implémentez une méthode Java qui crée 2 threads. Chaque thread est en charge respectivement de la première et de la deuxième moitié des éléments des 2 vecteurs, et calcule le produit scalaire de 2 demi-vecteurs de 5000 éléments. Les vecteurs `x` et `y` seront représentés par des tableaux de 10000 éléments, et chaque demi-vecteurs sera identifié par un intervalle d'indices disjoints (0 à 4999 et 5000 à 9999 respectivement). Le thread principal attend la terminaison des 2 threads de calcul, puis ajoute les 2 résultats partiels et retourne le résultat final.

Question 3.2

On suppose désormais que `x` et `y` sont des vecteurs de nombres à virgule flottante de type `float`. La technique de parallélisation précédente fournit généralement un résultat différent du calcul séquentiel du produit scalaire de 10000 éléments. Expliquez pourquoi, et donnez un exemple ou la somme de 3 nombres à virgule flottante fournit un résultat très différent en fonction de l'ordre des additions.

4 Pointeur de la souris

On étudie le fonctionnement d'un pilote de souris sur port série, le rafraîchissement de la position de la souris à l'écran, et l'interprétation des clicks du bouton (supposé unique).

Le pilote de souris est normalement un module du noyau du système, écrit en langage C et/ou en assembleur. Nous simulerons son fonctionnement en Java.

À chaque changement d'état (pression ou relâchement du bouton, déplacement de la souris), le microcontrôleur dédié de la souris émet deux octets sur le port série à destination de l'ordinateur. Ceux-ci codent en complément à 2 la valeur entre -127 et 127 du déplacement horizontal et vertical de la souris depuis la dernière position enregistrée ; lorsque le premier octet vaut -128 (en binaire 10000000_2), le deuxième octet ne peut prendre que les 3 valeurs suivantes :

- 0 : confirmation périodique de la connexion ;
- 1 : bouton pressé ;
- 2 : bouton relâché.

Coté ordinateur, le port série est géré par un microcontrôleur dédié appelé UART. Celui-ci gère un tampon (*buffer*) de 16 octets pour stocker les données en provenance du port série. Les lectures/écritures sur ce tampon et le contrôle de l'UART sont implémentées par une projection des entrées-sorties en mémoire, et mis en oeuvre dans les programmes Java à travers les variables suivantes :

- `byte buf[16]` : tampon mémorisant jusqu'à 16 octets reçus depuis le port série ;
- `byte cnt` : contient le nombre d'octets valides dans le tampon ;
- `byte ack` : l'assignation de cette variable vide le tampon et autorise la réception de nouvelles données à partir de `buf[0]` ;
- `int millis` : compteur de millisecondes interne à l'UART ; peut être réinitialisé en assignant la valeur de son choix à la variable.

Dès qu'un premier octet est reçu dans le tampon (dans `buf[0]`), l'UART en informe le processeur principal en lui envoyant une interruption. Celle-ci est simulée en Java par l'appel asynchrone de la méthode `onReceive()` dans un thread dédié (mécanisme comparable à la gestion des évènements de l'interface graphique sur Android). Entre le déclenchement de l'interruption et le traitement de celle-ci par la routine dédiée sur le processeur principal, plusieurs octets peuvent s'accumuler dans le tampon sans déclencher d'interruption additionnelle. En revanche, le buffer ne reçoit pas de nouvelles données pendant le traitement de l'interruption. Au delà de 16 octets, les données sont perdues et l'émetteur en est informé ; dans le cas présent, la souris ignore simplement le message de perte de données.

Question 4.1

La paire d'octets $(-128, 0)$ permet à la souris de signaler sa présence à l'ordinateur. Elle est envoyée périodiquement toutes les $10\,ms$, dès lors qu'un changement d'état est intervenu dans cet intervalle de temps.

Que se passe-t-il si la connexion est interrompue entre l'envoi du premier et du deuxième octet sur la liaison série ? Quel effet cela entraine-t-il sur l'interprétation des mouvements ultérieurs par l'ordinateur ? Montrez comment y remédier grace au code périodique de connexion $(-128, 0)$.

Question 4.2

La position courante du pointeur de la souris est mémorisée par l'interface graphique dans les variables `mx` et `my`. Ces variables doivent être mises à jour en ajoutant les valeurs reçues sur le port série lorsque celles-ci sont entre -127 et 127. Implémentez la méthode `onReceive()` pour lire les `cnt` octets reçus et mettre à jour la position de la souris (on ne se souciera pas de borner les valeurs de ces variables pour simuler les bords de l'écran). N'oubliez pas d'informer l'UART de la bonne réception des données et de vider le tampon en écrivant (n'importe quelle valeur) dans la variable `ack`. Vous supposerez que la valeur de `cnt` est paire et qu'il n'y a pas d'erreur de transmission.

Question 4.3

Lorsque le nombre d'octets reçus est impair ou lors d'une interruption de transmission, un décalage peut se produire. Expliquez les différents scénarios possibles et modifiez votre programme pour traiter tous les cas.

Question 4.4

L'affichage du pointeur de la souris à sa nouvelle position dans le framebuffer est réalisé par la méthode `updateMouseState()` de l'interface graphique avant chaque cycle de rafraichissement de l'écran, soit environ 50 fois par seconde. Que risque-t-il de se passer si la méthode `onReceive()` est appelée en même temps que l'interface graphique consulte les variables `mx` et `my` ? Comment y remédier ?

Question 4.5

La pression du bouton, le relâchement du bouton, le click et le double-click sont associés respectivement à quatre méthodes de l'interface graphique : `onPress()`, `onRelease()`, `onClick`, `onDoubleClick()`. Donnez trois raisons pour lesquelles il n'est pas question d'appeler directement ces méthodes directement depuis le code du pilote de la souris.

Question 4.6

Pour signaler les évènements liés au bouton à l'interface graphique, le pilote doit utiliser un handler de callback de l'interface graphique appelé `buttonStateCallback`. Il lui envoie une instance de l'interface `Runnable` dont la méthode `run()` appelle l'une des quatre méthodes de réaction aux évènements liés au bouton.

Écrivez un fragment de programme Java illustrant se mécanisme dans le cas particulier d'un évènement de pression du bouton.

Question 4.7

Modifiez `onReceive()` pour identifier le type d'évènement (inutile de compléter le callback dans les 3 autres cas). Vous supposerez qu'un click est défini par la succession d'une pression et d'un relâchement en moins de 500ms, et qu'un double-click est défini par la succession de deux clicks en moins de 500ms. Si un click est détecté, cet évènement est signalé au lieu de la succession des évènements de pression et de relâchement. Un click ne doit pas être signalé avant d'être certain qu'il ne s'agit pas d'un double-click.