L3

Examen final de Programmation Unix – session 1 – 5 janvier 2015

Michel SOTO

AUCUN DOCUMENT AUTORISÉ

Durée: 1 H 30

Le barème est indicatif - Nombre de pages: 2 sur 2 feuilles

La concision de vos réponses et la propreté de votre copie seront prises en compte.

PARTIE I : CONNAISSANCE DU COURS

Question 1 (4 points)

Répondez aux affirmations suivantes uniquement par "VRAI", ou "FAUX" ou "NE SAIS PAS".
Barème : **réponse exacte : +1 point, réponse fausse : -0,5 point sur la copie, "ne sais pas" : 0 point**

a) Tout thread se termine quand le processus qui l'a créé se termine.

b) Par défaut, plusieurs instances d'un même signal peuvent être pendantes.

c) La pose, par un processus P, d'un verrou exclusif en mode *advisory* sur une zone de fichier interdit l'accès à cette zone à tout autre processus que P.

d) Le nommage d'une socket est une opération qui associe une chaîne de caractère à une socket.

Question 2 (4 points)

Soit le programme suivant :

```
main(void) {
        int fdA, sd;
  /* 1 */ fdA = open("fichA", O_WRONLY);
  /* 2 */ dup2 (fdA,1);
  /* 3 */ printf("coucou\n");
  /* 4 */ sd=socket(AF_INET, SOCK_STREAM, 0);
  /* 5 */ dup2 (sd, 0);
  /* 6 */ sleep (2);
} /* main */
```

a) Représentez par un schéma l'état de la u_ofile **avant** l'exécution de la ligne 1.

b) Représentez par un schéma l'état de la u_ofile **après** l'exécution des lignes 1 à 5. Justifiez votre réponse

c) Quel est le résultat obtenu après de l'exécution de ce programme. Justifiez votre réponse.

PARTIE II : APPLICATION DU COURS

Dans les questions suivantes, le code devra être proprement indenté. Vous êtes dispensé des vérifications du nombre de paramètres éventuels, des valeurs de retour des primitives système et des `includes`.

Question 3 (4 points)

Ecrire le programme `reverse.c` qui affiche le contenu d'un fichier, passé en paramètre, du dernier au premier caractère. Par exemple, si le fichier `f` contient "Bonne année\n2015" `reverse f` affiche :

5012
eénna ennoB

Question 4 (8 points)

La méthode du tri rapide d'un tableau consiste à placer un élément du tableau (appelé pivot) à sa place définitive, en permutant tous les éléments de telle sorte que tous ceux qui sont inférieurs au pivot soient à sa gauche et que tous ceux qui sont supérieurs au pivot soient à sa droite. Cette opération s'appelle le partitionnement. Pour chacun des sous-tableaux, gauche et droite, on définit un nouveau pivot et on recommence l'opération de partitionnement. Ce processus est répété récursivement, jusqu'à ce que l'ensemble des éléments du tableau soit trié.
La fonction `quicksort` ci-dessous implémente une version séquentielle du trie rapide d'un tableau `tab` s'étendant de l'indice `premier` à l'indice `dernier` inclus. La fonction `partition` réalise le partitionnement du tableau `tab`

entre de l'indice `premier` et l'indice `dernier` inclus. Elle retourne l'indice de du pivot dans la zone du tableau partitionnée. Vous n'avez pas besoin de connaître le code de cette fonction.
Vous devez modifiez le code ci-dessous pour en faire en une version parallèle fondée sur les threads. Dans cette version parallèle, **chaque appel à la fonction `quicksort` sera exécuté par un thread**.

a) Ecrivez le code de la ligne 2 définissant la structure contenant les paramètres passés à chaque thread exécutant la fonction `quicksort`.
b) Ecrivez le code de la ligne 4 redéfinissant le prototype de la fonction `quicksort` afin qu'elle soit exécutable par un thread.
c) Ecrivez le code de la ligne 6 déclarant les variables locales nécessaires à la fonction `quicksort` pour son exécution par un thread.
d) Ecrivez le code des lignes 9, 10 et 20 remplaçant chaque appel à la fonction `quicksort` par la création d'un thread. Précisez à chaque fois le n° de la ligne concernée (9, 10 ou 20).
e) Ecrivez le code de la ligne 11 permettant de joindre (attendre la fin) les des deux threads créés en remplacement des lignes 9 et 10. Vous considérerez que les threads sont joignables par défaut.
f) Le thread créé en replacement de la ligne 20 doit-il ou pas être joint afin que le tableau trié soit correctement affiché ? Justifiez votre réponse.

```
1.   #include <stdio.h>
2.
//= Prototype de la fonction de partitionnement =
3.  int partition (int tab[], int premier,
                              int dernier);
// ============================================

// ============================================
4.   void quickSort (int tab[], int premier,
                                  int dernier){
// ============================================
5.   int j;
6.
7.   if (premier < dernier){
 // Partionnement du tableau
8.  j = partition (tab, premier, dernier);

// Tri de la partie à gauche du pivot
9.    quickSort (tab, premier, j-1);

// Tri de la partie à droite du pivot
10.   quickSort (tab, j+1, dernier);
11.
   }// if
 }// quickSort
```

```
// ========================================
12. void main () {
// ========================================
13. int i, tab[] = { 79, 17, 14, 65, 89, 4, 95};
14. int indice_max;

15. indice_max=(sizeof (tab)/sizeof (int)) - 1;

16. printf ("\nTableau non trié:\t");
17. for (i = 0; i <= indice_max; ++i)
18.      printf (" %d ", tab[i]);
19. printf ("\n");

20. quickSort (tab, 0, indice_max);

21. printf ("Tableau trié:\t\t");
22. for (i = 0; i <= indice_max; ++i)
23.      printf (" %d ", tab[i]);
24. printf ("\n");

 }// main
```

ANNEXE

```
int pthread_create (pthread_t *thread, pthread_attr_t *attr,
                    void * (*start_routine)(void *), void *arg);

int pthread_join(pthread_t thread, void **thread_return);

                                                ===
off_t lseek(int fd, off_t offset, int whence);
- fd : descripteur du fichier
- whence : SEEK_SET : déplacement du pointeur par rapport au début du fichier
           SEEK_CUR : déplacement du pointeur par rapport à la position courante
           SEEK_END : déplacement du pointeur par rapport à la fin du fichier
- offset : position par rapport à whence

Retourne : la nouvelle position du pointeur de lecture/écriture du fichier, exprimée
           en octets, depuis le début du fichier ou -1 en cas d'échec
```