Examen final de système d'exploitation OS302

31 mai 2017 – O. Aktouf

Remarques :

1. Il ne sera répondu à aucune question sur le sujet. Si un énoncé vous paraît flou, vous pouvez considérer des hypothèses supplémentaires pour le traiter, en les indiquant clairement sur votre copie.
2. Les solutions doivent être décrites **brièvement** et **clairement** avant d'être codées.
3. Le barème est donné à titre indicatif

Exercice 1 (6 points)

On souhaite trouver toutes les occurrences d'un élément `val` dans un tableau donné `Tab` de `N` éléments entiers. Pour cela, si le tableau est long (sa taille est supérieure à une constante `TAILLE_MIN`), on le coupe en deux et on effectue la recherche indépendamment dans les deux moitiés, en les confiant à deux processus différents travaillant en parallèle.

Écrire un programme effectuant la recherche de cette manière. Le programme devra afficher un message pour chaque occurrence trouvée indiquant le processus responsable de la recherche, puis afficher le nombre total d'occurrences de l'élément dans le tableau. Pour cela, on utilisera les valeurs de retour des processus fils.

Exercice 2 (7 points)

Soit le programme suivant.

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int var1 = 100 ;
int p[2] ;
char ch[7];

int main () {

int var2 = 50 ;

/* processus père, point 1 */
printf("Adresse de var1 dans le processus pere:%p\n", &var1) ;
printf("Adresse de var2 dans le processus pere :%p\n", &var2) ;
printf("1 : Valeur de var1 et var2 dans le processus pere --> %d %d \n",
var1, var2) ;
if (pipe(p) == -1) {
   printf("Erreur de creation de tube \n");
```

```
    exit(1) ;
}

switch (fork ()) {
case (pid_t)-1 :
    printf("Erreur de creation de processus \n") ;
    exit(2) ;
case (pid_t) 0 :    /* processus fils*/
    close(p[1]) ;
    /*point 2*/
    read(p[0], ch, 7) ;
    printf("Chaine recuperee dans le fils :%s \n", ch) ;
    printf("Adresse de var1 dans le processus fils :%p\n", &var1) ;
    printf("Adresse de var2 dans le processus fils :%p\n", &var2) ;
    printf("2 : Valeur de var1 et var2 dans le processus fils --> %d %d \n",
    var1, var2) ;
    var2 = var2 * 2;
    printf("3 : Valeur de var1 et var2 dans le processus fils -->%d %d \n",
    var1, var2) ;
    sleep(3) ;
    printf("6 : Valeur de var1 et var2 dans le processus fils --> %d %d \n",
    var1, var2) ;
    exit(0) ;
default : /* processus pere */
    close(p[0]) ;
    /*point 3*/
    write(p[1], "Je suis le pere", 15);
    sleep(2) ;
    printf("4 : Valeur de var1 et var2 dans le processus pere --> %d %d \n",
    var1, var2) ;
    var1 = var1 * 3 ; var2 = var2 * 3 ;
    printf("5 : Valeur de var1 et var2 dans le processus pere --> %d %d \n",
    var1, var2) ;
    sleep(2) ;
    exit(0) ;
}
}
```

1. Donner l'état des tables du système (tables des inoeuds, des fichiers ouverts, des descripteurs) aux points d'exécution suivants, indiqués en commentaires :

    a. point 1

    b. point 2

    c. point 3

2. Le résultat d'exécution suivant de ce programme comporte des erreurs. Indiquer lesquelles en justifiant la réponse.

```
Adresse de var1 dans le processus pere:211d8
Adresse de var2 dans le processus pere :ffbef894
1 : Valeur de var1 et var2 dans le processus pere --> 100 50
Chaine recuperee dans le fils : Je suis le pere
Adresse de var1 dans le processus fils :211d8
Adresse de var2 dans le processus fils :ffd212cb
2 : Valeur de var1 et var2 dans le processus fils --> 100 50
3 : Valeur de var1 et var2 dans le processus fils -->100 100
4 : Valeur de var1 et var2 dans le processus pere --> 100 100
5 : Valeur de var1 et var2 dans le processus pere --> 300 300
6 : Valeur de var1 et var2 dans le processus fils --> 300 300
```

Exercice 3 (7 points)

Ecrire un programme serveur qui effectue les opérations suivantes :

1. Le serveur boucle et écoute les messages des clients sur une file de messages. La clé de création de la file de messages est obtenue avec le fichier « `$HOME/.serveur` ».

2. Les messages lus sont composés de deux éléments, le `PID` du client émetteur du message et le nom de l'exécutable du client émetteur du message.

3. Après lecture d'un message le serveur envoie au client un message « `Bonjour !` » pour lequel il utilise le `PID` du client comme type de message. Dans ce message il communique au client son propre `PID` (`PID` du serveur).

4. Les clients répondent au message du serveur en lui envoyant un signal `SIGUSR1`.

5. Le serveur compte le nombre de signaux `SIGUSR1` reçus.

6. Le serveur est arrêté avec la frappe des touches `Ctrl-C`. Il affiche alors le nombre de messages initiaux reçus des clients ainsi que le nombre de signaux `SIGUSR1` reçus, et il détruit l'`IPC` utilisé.

Attention : il ne vous est pas demandé d'écrire le code du client.