Rattrapage 2024 : **Système d'exploitation**
Durée : **1h00**
Documents autorisés : **Le poly système et vos codes**
Barème : **Indicatif**
Enseignante : **Fetia Bannour**

Exercice 1 (7 points)

- Partie 1: Shell interactif

  (a) Indiquez ce que fait la commande suivante:

  ```
  sh> find $HOME/tp -type f -name '*.h'
  -exec stat -c '%s' {} \; | sort -n -r | head -n 1
  ```

  (b) Indiquez ce que fait la commande suivante:

  ```
  sh> find $HOME/tp -type f -user louis.dupont
  -exec stat -c '%s' {} \; | { a=0 ; while read x ; do a=$((a + x)) ; done
      ; echo $a ; }
  ```

- Partie 2: Shell script

  Considérons le script `Tools.sh` ci-dessous.

```
e() {
  n="$1"
  m=$(echo "$n" | sed -e '1s/[+-]\?[0-9]\+//')
  test -z "$m"
}

f() {
  a=$1
  if test $a -le 1 ; then
      echo 1
  else
      n=$((a-1))
      echo $(( a * $(f $n) ))
  fi
}
```

Tools.sh

Q-1 Indiquez ce que fait la builtin commande e.

Q-2 Indiquez ce que fait la builtin commande f.

Q-3 En utilisant les deux fonctions du script `Tools.sh`, écrivez un script shell qui prend deux arguments entiers n et m (s'il n'a pas 2 arguments entiers, affiche un message d'erreur standard (*stderr*) et se termine avec un statut d'erreur) et qui écrit sur le flux standard de sortie $n!/m!$.
Le script suivra les usages standards d'*Unix*.

Exercice 2 (6,5 points)

Les questions de cet exercice portent sur le programme `signaux.c` présenté ci-dessous. Compilez et exécutez ce programme.

Q-1 Expliquez brièvement ce que fait le programme `signaux.c`. Citez les processus crées et décrivez leurs états suite au lancement de ce programme.

Q-2 Expliquez l'utilisation de la fonction `fork()`?

Q-3 A quoi sert la fonction `alarm(int sec)` ? Expliquez.

Q-4 Quel est le signal transmis au processus (qui appelle la fonction `alarm()`) après un certain délais que vous spécifierez ? Quel est le comportement par défaut de ce signal ? Quel est son numéro ? Est-ce possible de l'attraper ?

Q-5 Proposez une solution pour que le processus père puisse attraper le signal en question et tuer le processus fils. Donnez le code associé dans les parties "A COMPLÉTER" du programme.

```
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>

pid_t pid;

/*** fonction gestionnaire (qu'on nommera "onalarm") À COMPLÉTER ***/

int main()
{
    int status;
    pid = fork();
    if (pid == -1)
        printf ("Erreur de création de processus\n");
    else
        {
            if (pid == 0)
                {
```

```
                printf("Valeur du fork dans le fils: %d\n", pid);
                printf("Je suis le fils, mon pid:%d\n",getpid());
                fflush(stdout);
                while(1);
            }
        else
            {
                printf("Valeur du fork dans le père: %d\n", pid);
                printf("Je suis le père, mon pid:%d\n",getpid());
                fflush(stdout);

                /***** À COMPLÉTER ******/

                alarm(5);
                wait(&status);
            }
    }
}
```

signaux.c

Exercice 3 (6,5 points)

Q-1 A quoi servent les tubes de communication ?

Q-2 Qu'appelle t-on tube de communication nommé ? Donnez deux petites différences de fonctionnement entre les tubes nommés (objet de cet exercice) et les tubes non nommés (ou anonymes) (`pipe()`).

Q-3 Rappelez brièvement la différence entre les appels système et les appels de la bibliothèque standard du C (*libc*).

Q-4 Donnez le prototype de l'appel système (la fonction noyau) qui permet de créer un tube nommé.

Q-5 Donnez la commande Shell qui permet de créer un tube nommé.

On appelle généralement *écrivain* le programme chargé d'écrire dans le tube et *lecteur* celui qui lit son contenu. Supposons que l'on ouvre un premier terminal pour le processus *écrivain* (qui écrit le mot "yes" en permanence) et un deuxième terminal pour le processus *lecteur* (qui lit en permanence le contenu du tube).

On suppose que l'on lance dans le premier terminal le script `écrivain.sh` puis dans le deuxième terminal le script `lecteur.sh`.

Q-6 Décrivez ce qui se passe au niveau des deux scripts. Rappelez les spécificités des opérations d'ouverture, de lecture et d'écriture dans le tube.

Q-7 Donnez la commande Shell qui permet de terminer le processus *lecteur*. On suppose que l'on exécute cette commande dans un troisième terminal. Quel signal a t-il été envoyé à ce processus provoquant sa terminaison ? Peut-il être attrapé ?

Q-8 Que se passe t-il si vous tentez d'écrire un message dans la fenêtre de l'*écrivain* ? Quel est le signal reçu ? et au niveau de quel processus ? Rappelez le comportement par défaut de ce signal.