IUT de Montpellier, Dép. Informatique
M2101 - Archi 2 V. Poupet

Examen (1 heure)

Les documents, calculatrices, téléphones etc. ne sont pas autorisés.
Le sujet comporte 2 pages.

Remarque : Lorsqu'il vous est demandé d'écrire un programme, vous pouvez ignorer les inclusions de bibliothèques. Vous pouvez négliger les erreurs qu'il n'est pas explicitement demandé de traiter.

Exercice 1. *Lecture et écriture*

1 Écrivez une fonction `void strToFile(char *s, int fd)` qui écrit le contenu de la chaîne de caractères `s` dans le fichier désigné par le descripteur `fd` (voir la suite du sujet pour un exemple d'utilisation). Si l'écriture échoue, la fonction doit afficher un message d'erreur et faire terminer le programme.

Remarque : Vous pouvez utiliser la fonction `int strlen(char *s)` qui renvoie la longueur d'une chaîne de caractères passée en argument.

On considère maintenant le programme (incomplet) suivant

```
1  int main(int agc, char **argv) {
2    int fd; // descripteur de fichier pour l'écriture, puis la lecture
3
4    // [1] Ouverture du fichier en écriture (avec gestion des erreurs)
5
6    int i;
7    for (i=0; i < 5000; i++) {
8      strToFile("All work and no play makes Jack a dull boy\n", fd);
9    }
10
11   // [2] Fermeture puis ouverture du fichier en lecture
12
13   // [3] Affichage du contenu du fichier dans la sortie standard
14
15   return 0;
16 }
```

2 Écrivez les lignes nécessaires pour ouvrir en écriture au programme (bloc `[1]`). Faites en sorte que le programme s'interrompe en donnant un message d'erreur si l'ouverture du fichier a échoué.

Remarque : Si vous ne connaissez pas les noms exacts des *flags* à passer à la fonction `open` expliquez en français les arguments qu'il faudrait donner.

3 Écrivez les lignes permettant de fermer le fichier ouvert en écriture puis de le rouvrir en lecture avec le même descripteur de fichier (bloc `[2]`).

4 Écrivez les lignes qui permettent d'afficher le contenu du fichier dans la sortie standard (bloc `[3]`).

5 Le programme complet ouvre le fichier, écrit des données, ferme le fichier puis le rouvre pour lire les données. Puisque l'on écrit et lit dans le même fichier, on aurait pu l'ouvrir une seule fois en mode lecture et écriture (`O_RDWR`). Quelle action supplémentaire doit-on effectuer dans ce cas entre l'écriture des données et la lecture ?

Exercice 2. *Processus*

On considère le programme suivant

```
1  int main (int argc, char const *argv[])
2  {
3    int pid;
4    pid = fork();
5    if (pid !=0) {
6      printf("1\n");
7      waitpid(pid, NULL, 0);
8      printf("5\n");
9    } else {
10     printf("2\n");
11     pid = fork();
12     if (pid == 0) {
13       printf("3\n");
14     } else {
15       printf("4\n");
16     }
17   }
18   return 0;
19 }
```

1 Représentez à l'aide d'un schéma l'ensemble des processus créés lors de l'exécution de ce programme.

2 Décrivez brièvement (2-3 lignes max) ce qui se passe lorsque le programme exécute la fonction `fork`. À quoi correspond la valeur renvoyée ?

3 Étant donné que la création d'un nouveau processus est une opération complexe, après un appel à la fonction `fork` le processus père a en général le temps de continuer son exécution avant que le fils puisse commencer. Selon ce principe, quel est l'ordre le plus probable d'affichage des lignes 1 à 5 ? Justifiez votre réponse.

4 On considère maintenant qu'après un appel à la fonction `fork` les deux processus s'exécutent totalement en parallèle et donc le processus fils peut tout à fait continuer son exécution avant le père (on ignore l'argument de la question précédente).
Parmi les permutations suivantes indiquez celles qui ne sont pas des sorties possibles du programme (justifiez en une phrase pourquoi elles ne sont pas possibles)

a. `1, 2, 3, 5, 4`
b. `1, 2, 4, 5, 3`
c. `1, 2, 3, 4, 5`
d. `2, 4, 3, 1, 5`
e. `1, 4, 3, 2, 5`
f. `1, 3, 2, 4, 5`