Mardi 31 mai 2005
Durée : 3 heures
Documents autorisés

Examen : systèmes d'exploitation
L3, parcours informatique, 2004-2005

Examen

Les valeurs de retour des primitives ne sont pas systématiquement testées dans les programmes de l'énoncé. On supposera donc que les primitives ne renvoient jamais un code d'erreur.

Les appels système

Exercice 1. Les deux programmes suivants implémentent de façon élémentaire la commande `cat` :

cat1.c

```
 1 : #include <fcntl.h>
 2 : #include <unistd.h>
 3 :
 4 : int main(int argc, char *argv[])
 5 : {
 6 :   char c;
 7 :   int df = open(argv[1], O_RDONLY);
 8 :
 9 :   while (read(df, &c, 1) > 0)
10 :     write(STDOUT_FILENO, &c, 1);
11 :
12 :   _exit(0);
13 : }
```

cat2.c

```
 1 : #include <stdio.h>
 2 : #include <stdlib.h>
 3 :
 4 : int main(int argc, char *argv[])
 5 : {
 6 :   char c;
 7 :   FILE *df = fopen(argv[1], "r");
 8 :
 9 :   while (fread(&c, sizeof(char), 1, df) > 0)
10 :     fwrite(&c, sizeof(char), 1, stdout);
11 :
12 :   exit(0);
13 : }
```

Quel est, à votre avis, le programme plus performant ? Expliquez votre réponse.

Les processus

Exercice 2.

1. Dessiner l'arbre généalogique des processus engendrés par l'évaluation de la commande C

   ```
   fork() || ( fork() && fork() ) ;
   ```

2. Écrire un petit programme en C et une ligne de commande shell permettant de vérifier votre réponse.

Exercice 3.

1. Quand un processus est appelé *orphelin* ? Quand un processus est dans l'état *zombie* ?
2. Écrire un programme qui engendre deux fils, dont l'un deviendra zombie, et l'autre deviendra orphelin.

La communication par signaux

Exercice 4.

1. Commenter, ligne par ligne, le programme suivant :

signaux.c

```
 1 : #include <stdio.h>
 2 : #include <unistd.h>
 3 : #include <signal.h>
 4 :
 5 : void interruption(int signum)
 6 : {
 7 :   printf("%d\n", getpid());
 8 : }
 9 :
10 : int main(void)
11 : {
12 :   struct sigaction action;
13 :   action.sa_handler = interruption;
14 :   action.sa_flags = 0;
15 :   sigemptyset(&action.sa_mask);
16 :   sigaction(SIGUSR1, &action, NULL);
17 :
18 :   if (fork())
19 :   {
20 :     sigset_t ens;
21 :     sigemptyset(&ens);
22 :     sigaddset(&ens, SIGUSR1);
23 :     sigprocmask(SIG_SETMASK, &ens, NULL);
24 :     sleep(5);
25 :     sigemptyset(&ens);
26 :     sigprocmask(SIG_SETMASK, &ens, NULL);
27 :   } else
28 :   {
29 :     sleep(1);
30 :     kill(getppid(), SIGUSR1);
31 :     sleep(1);
32 :     interruption(0);
33 :   }
34 :   return 0;
35 : }
```

2. On suppose que le PID du processus est 1000 et que le PID du fils est 1001 : dire ce qui est affiché à l'écran pendant l'exécution du programme `signaux.c`. Expliquez votre réponse.
3. Que se passe-t'il si l'on remplace la ligne `24` par

   ```
   pause();
   ```

   dans le programme `signaux.c` ?
4. Est il possible de réécrire le programme `signaux.c` en utilisant l'interface de programmation du System V ? Expliquez votre réponse.

La communication par tubes

Exercice 5.

1. Expliquer la différence entre un fichier régulier et un fichier de type tube.
2. Expliquer la différence entre un tube nommé et un tube anonyme.
3. On peut appeler les primitives `read` et `write` avec des fichiers réguliers et des tubes. Donner un exemple d'un appel système qu'on utilise avec les fichiers réguliers mais qui retournera toujours un code d'erreur si utilisé avec un tube. Justifiez votre exemple.

Exercice 6. Considérez le programme suivant :

tubes.c

```
 1 : #include <stdio.h>
 2 : #include <stdlib.h>
 3 : #include <unistd.h>
 4 : #include <fcntl.h>
 5 : #include <sys/stat.h>
 6 :
 7 : void fils(int d_lecture)
 8 : {
 9 :   ssize_t no_lu, i;
10 :   char tampon[100];
11 :
12 :   while ((no_lu = read(d_lecture, tampon, sizeof(tampon))) > 0)
13 :   {
14 :     for (i = 0; i < no_lu; i++)
15 :       if ('a' <= tampon[i] && tampon[i] <= 'z')
16 :         tampon[i] += 'A' - 'a';
17 :     write(STDOUT_FILENO, tampon, no_lu);
18 :   }
19 : }
20 :
21 : void pere(d_ecriture)
22 : {
23 :   char *message = "abc";
24 :   write(d_ecriture, message, sizeof(message));
25 : }
26 :
27 : int main(void)
28 : {
29 :   int d_ecriture, d_lecture;
30 :
31 :   unlink("tube");
32 :   mkfifo("tube", S_IRWXU);
33 :   chmod("tube", S_IRWXU);
34 :
35 :   d_ecriture = open("tube", O_WRONLY);
36 :   d_lecture = open("tube", O_RDONLY);
37 :
38 :   if (fork())
39 :     pere(d_ecriture);
40 :   else
41 :     fils(d_lecture);
42 :
43 :   unlink("tube");
44 :   exit(0);
45 : }
```

1. Expliquez la raison des lignes `31--33`.
2. Dans ce programme on y trouve deux erreurs de traitement des tubes nommés. Trouvez les et, pour chacune d'eux, expliquez pourquoi il s'agit d'une erreur et proposez une correction.
3. Expliquez de façon concise ce qui est fait par le programme `tubes.c` une fois corrigé.

Le shell et les scripts

Exercice 7. Écrire un script shell permettant de chercher de façon récursive tous les fichiers ayant un nom donné dans un répertoire. Le script affichera les chemins complets des tous les fichiers ayant ce nom. Le nom à chercher sera passé en paramètre au script. Le nom du répertoire où chercher, passé comme deuxième paramètre, est optionnel : s'il manque, le répertoire courant est cherché.