Faculté de physique et ingénierie
Université de Strasbourg

Licence SPI ingénierie L2 S3
TP n°3 Architecture des ordinateurs

Nous allons travailler sur la « programmation système ». Il s'agit d'accéder aux fonctions de base du système d'exploitation. Vous me mettrez vos sources (c.à.d les programmes en C, pas les exécutables), dans leur état actuel, sur moodle, dès la fin de la première demi-séance (impérativement, le logiciel me donnant la date de votre dernière modification, n'y touchez plus après). Vous me rendrez aussi un rapport m'expliquant ce que vous avez compris, mais aussi les difficultés que vous avez rencontrées, dè la fin de la seconde demi-séance, sur moodle impérativement.

1) parcours d'un répertoire

Enregistrez le programme ci-contre dans le fichier « dir.c ». Ce programme permet de récupérer la liste des fichiers d'un répertoire ("." est le répertoire actuel). De plus il indique son type : 8=fichier, 4=répertoire (vous ne vous occuperez pas des autres cas). Essayez le.
Puis vous améliorerez ce programme en le décomposant en une fonction, qui reçoit en argument un nom de répertoire, et affiche son contenu. Le main demandera le nom du répertoire (une chaîne de caractères) et appellera votre fonction. Attention, le format "%s" de scanf n'accepte pas les espaces, contrairement au format "%[^\n]".

```
#include <stdio.h>
#include <dirent.h>
int main()
{
 struct dirent *l;
 DIR *rep;
 rep = opendir("." );
 while (l = readdir(rep)) {
   printf("%X : %s\n", l->d_type,l->d_name);
 }
 closedir(rep);
}
```

Il serait intéressant de permettre un affichage récursif : ne changez rien pour les fichiers normaux (4), mais si c'est un répertoire (8), rappelez votre fonction qui listera donc le contenu du sous répertoire. Peut-être serait-ce plus simple de ne pas traiter tous ceux dont le nom commence par « . » (je vous rappelle qu'une chaîne de caractères est un tableau, il est facile d'accéder à chaque caractère, y compris le premier). Autres petites remarques : strcpy permet de copier une chaîne dans une autre, strcat de la recopier à la suite. Si vous passez une chaîne en argument, c'est par adresse, donc il vaut mieux ne pas la modifier (par strcat par exemple) si vous voulez la retrouver intacte à la sortie. Opendir retourne NULL si le répertoire n'existe pas ou si vous n'y avez pas accès.

Si vous êtes fort et avez encore du temps, il serait aussi amusant de faire un programme qui compte le nombre de répertoires et de fichiers de votre pc. Vous commencez dans votre répertoire perso. Si c'est un fichier, ajouter 1. Si c'est un répertoire, rechercher (récursivement) son contenu. Comptez aussi la profondeur. Si elle dépasse 100 il y a peut-être un problème ?

2) droits d'accès à un fichier

Enregistrez le programme ci-contre dans le fichier « mode.c ». Ce programme récupère diverses informations sur le fichier « mode.c » (ça marche pour tout autre fichier, bien sûr), et les affiche. Le nombre en lui même n'a pas d'intérêt, mais ses trois bits de poids faible indiquent les droits d'accès au fichier pour tout le monde : bit 0 accès lecture (noté R), bit 1 droit d'écriture (noté W), bit 2 droit d'exécution (noté X).

```
#include <stdio.h>
#include <sys/stat.h>
int main()
{
 struct stat s;
  char nomfic[256]="mode.c";
  stat(nomfic,&s);
  printf("%d\n",s.st_mode);
}
```

Les 3 suivants (dans le même ordre) sont les droits du groupe d'utilisateurs (tous les L2 SPI sont dans un même groupe), les 3 suivants ceux du propriétaire (vous). Modifiez ce programme pour qu'à l'aide de masques et décalages il affiche en clair ces 9 droits d'accès. Puis vous améliorerez ce programme pour qu'il demande le nom du fichier. Si vous avez le temps, vous pourrez le combiner à la question 1 pour afficher les droits de tous les fichiers d'un répertoire.

3) multitâches avec dialogue entre tâches

Vous savez ce qu'est un système d'exploitation multitâche. Sous Linux, vous pouvez plusieurs terminaux et lancer des programmes dans chacun d'eux. Mais peuvent-ils se transmettre des informations ? Oui. Il y a plusieurs moyens pour cela, dont les « tubes ». Un « tube nommé » permet à deux tâches tournant simultanément de se transmettre des informations. L'une (l'écrivain) peut écrire des choses dans le tube (des entiers, des caractères…) et l'autre (le lecteur) peut récupérer ces informations, dans le même ordre. Copiez ces deux programmes et testez les en les lançant dans deux terminaux différents (lancez l'écrivain en premier!)

```
//écrivain
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#define TAILLE_MESSAGE  256

int main(void)
{
 int tube;
 char nomTube[] = "essai.fifo";
 char c[TAILLE_MESSAGE] = "Bonjour";
 mkfifo(nomTube, 0644); //creation du tube
 tube = open(nomTube, O_WRONLY); //en écriture
 write(tube, c, TAILLE_MESSAGE);
 close(entreeDuTube); //fermeture du tube
 unlink(nomTube);  //detruction du tube
 return 0;
}
```

```
//lecteur
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define TAILLE_MESSAGE  256

int main(void)
{
 int tube;
 char nomTube[] = "essai.fifo";
 char c[TAILLE_MESSAGE];
 tube=open ("essai.fifo", O_RDONLY);
 read(tube, c, TAILLE_MESSAGE);
 printf("%s\n", c);
 close(tube);
 return 0;
}
```

Essayez de programmer le jeu du « plus ou moins », en DEUX programmes qui dialoguent entre eux. Dans l'un (appelé le « serveur de jeu ») vous imaginez un nombre (aléatoire) à découvrir, et à chaque fois que l'autre programme vous propose un nombre (dans un tube) vous lui dites (printf) si sa proposition est trop grande ou trop petite (ou s'il a trouvé). L'autre programme (appelé « client ») sert à faire les propositions (c'est à dire les envoyer dans un tube). Ci-dessous, un petit programme qui peut vous servir de base au serveur

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
 {
  int sol;
  srand(time(NULL));
  sol=(rand()%100)+1;
  printf("%d\n",sol);
}
```

pour rand, srand et time

initialisation du générateur de nombres

le reste de la division (%) par 100 est entre 0

Peut-être aurez-vous le temps de tester si plus d'un client peut jouer simultanément avec le même serveur ? Peut-être que le fait de mettre le « mkfifo » et « unlink » dans l'autre programme donnerait un résultat différent ?
Si l'on voulait, en plus, que le serveur réponde au client (trop petit ou trop grand), il faudrait créer un second tube (dans l'autre sens). Dans ce cas le serveur n'aurait même pas besoin d'un terminal.