R3.05 Programmation Système
Séance pratique 2 : entrées-sorties

Pour vous aider tout au long de ce TP, il est recommandé de vous référer aux supports de cours, ainsi qu'au man de Linux.

Dans les codes fournis, les inclusions de librairies ont été omises

Exercice 1

Nous allons apprendre à utiliser les appels système open, close, read et write.

a) Exécutez ce code et expliquez les erreurs générées

```
int main() {
      int fd = open("toto.txt", O_WRONLY);
      if (fd < 0)
            perror("erreur open");
      char data[] = "Allez les bleus !\n";
      int retour = write(fd, data, strlen(data));
      if (retour < 0)
            perror("erreur write");
      return 0;
}
```

b) Créez maintenant un fichier **toto.txt** dans le même dossier que votre exécutable, et exécutez à nouveau. Y a-t-il des erreurs ? Observez le contenu du fichier toto.txt.

c) Exécutez à nouveau le code, et observez à nouveau le contenu du fichier toto.txt. Expliquez.

d) Remplacez **"Allez"** par **"Super"** dans le code, recompilez, exécutez, et observez à nouveau le contenu. Où écrit-on par défaut après l'ouverture d'un fichier ?

e) Remplacez l'appel à open par celui-ci :

int fd = open("toto.txt", O_RDONLY)

Que se passe-t-il ?

f) Ajoutons maintenant un autre *flag* : remplacez l'appel à open par celui-ci :

int fd = open("toto.txt", O_WRONLY | O_APPEND)

Exécutez plusieurs fois d'affilée le code, et observez le contenu du fichier toto.txt. En déduire le rôle du flag O_APPEND

g) À quoi sert le flag O_TRUNC ? Comment constater son effet ?

h) Même question avec le flag O_CREAT. Attention, lorsque ce flag est indiqué, il faut utiliser la signature de **open** avec un troisième paramètre. À quoi correspond ce paramètre ?

i) Exécutez maintenant ce code :

```
int main() {

      int fd = open("toto.txt", O_RDONLY);
      if (fd < 0)
            perror("erreur open");
      char data;
      while (int retour = read(fd, &data, 1) > 0) {
            write(1, &data, 1);
      }
      return 0;

}
```

À quoi correspond le 1 dans l'appel à **read** ? À quoi correspondent les deux 1 dans l'appel à **write** ? Quand sort-on de la boucle while ?

Exercice 2

Modifiez le programme précédent pour en obtenir un qui a le même comportement que

cat unfichier

où **unfichier** est un fichier texte qui existe. C'est à dire que votre programme devra afficher sur la sortie standard le contenu de ce fichier.
Expliquez avec des mots comment procéder.

Exercice 3

Écrire un programme **monCp** qui prend trois paramètres en ligne de commandes :

- le chemin d'un fichier f1 (qui est supposé exister)
- le chemin d'un fichier f2 (qui n'existe pas forcément : il faut le créer si ce n'est pas le cas)
- un entier T. On rappelle que pour transformer une chaîne de caractères représentant un entier en l'entier correspondant, on peut utiliser la fonction **atoi**.

Votre programme copie tout le contenu de f1 vers f2 par "blocs" de T octets.

Mesurez la différence de temps d'exécution (avec le programme time) en utilisant pour f1 un très gros fichier texte, et en faisant varier la taille de T (essayez avec T qui vaut 1, 100, 1000). Donnez vos résultats et commentez.