Systèmes d'exploitation — Exercices
Fichiers

Seweryn Dynerowicz (Seweryn.DYNEROWICZ@umons.ac.be)

1 Opérations de base

Afin de pouvoir manipuler des fichiers sous Linux, quatres fonctions principales sont disponibles ; ouverture, lecture, écriture et fermeture. Le *listing* ci-dessous reprend le canevas général de l'utilisation de ces fonctions.

```
1  #include <fcntl.h>  // open(..), close(..)
2  #include <unistd.h> // read(..), write(..), lseek(..)
3
4  #define SIZE 1024
5  int buffer[SIZE];
6
7  int main(int argc, char* argv[]) {
8    int fd = open("/home/sdy/tableau.bin", O_RDWR);
9    if (fd == -1) {
10     perror("open");
11     return EXIT_FAILURE;
12   }
13
14   int bytes_read = read(fd, buffer, SIZE * sizeof(int));
15
16   if(bytes_read != -1) {
17     // Work on buffer
18   }
19
20   lseek(fd, 0L, SEEK_SET);
21   write(fd, buffer, bytes_read);
22
23   close(fd);
24
25   return EXIT_SUCCESS;
26 }
```

Figure 1 – Canevas général d'utilisation des opérations de base.

Pour pouvoir accéder à un fichier, il est nécessaire dans un premier temps de demander l'ouverture de celui-ci. La fonction `open(..)` prend toujours au moins deux paramètres ; le chemin (absolu ou relatif) du fichier dont on demande l'ouverture ainsi que les *flags* associés à cette demande d'ouverture. Cette fonction renvoye comme valeur de retour un *file descriptor*[1] ou bien la valeur spéciale `-1` pour signaler que l'appel a échoué[2].

La fonction `read(..)` peut être utilisée pour demander au système d'exploitation de lire le contenu du fichier depuis le support physique de stockage vers la mémoire principale. Les trois paramètres sont ; le *file descriptor* au départ duquel la lecture doit être réalisée, l'adresse à laquelle le contenu lu doit être placé et le nombre maximum d'octets qui doivent être lus. Lorsque la fonction se termine, la valeur de retour stipule combien d'octets ont été effectivement lus (*n.b.* ce nombre est compris entre `0` et le nombre maximum passé en paramètre. Si la valeur de retour est de `-1`, une erreur est survenue et la fonction `perror(..)` peut être utilisée pour obtenir des détails.

La fonction `write(..)` prend les mêmes paramètres et renvoye la même valeur de retour, à ceci prêt que la signification est échangée pour refléter une écriture.

Afin de travailler proprement, il est nécessaire de clôturer une fichier lorsque celui-ci ne sera plus accéder par un processus en utilisant la fonction `close(..)`.

1. Numéro utilisé par le système d'exploitation pour faire le lien avec ce fichier lors des appels systèmes ultérieurs
2. La fonction `perror(..)` pouvant être utilisée pour plus de détails concernant cet échec.

1.1 Position courante

Les opérations de lecture et écriture sous Linux ne prennent pas en paramètre la position à partir de laquelle doit être effectuée l'opération. Le système garde en mémoire l'endroit où le processus est arrivé dans un fichier ouvert. Ceci est représenté par un `offset` qui est associé en interne à un *file descriptor* qui est mis à jour au fur et à mesure que le processus réalise des appels `read(..)` et/ou `write(..)`.

Pour pouvoir déplacer ce marqueur de position courante, la fonction `lseek(..)` peut être utilisée (*cfr.* ligne 20) en stipulant l'*offset* à considérer et comment utiliser ce dernier. Il existe trois manières possibles de mettre à jour la position courante sur base cet *offset*.

— Assignation directe de la valeur de l'*offset* à la position courante (*i.e.* `SEEK_SET`)
— Addition de la valeur de l'*offset* à la position courante (*i.e.* `SEEK_CUR`)
— Assignation de la somme de l'*offset* avec la taille du fichier (*i.e.* `SEEK_END`)

2 Énoncés

Exercice 1

Implémentez un programme qui prend en paramètre le nom d'un fichier texte et qui parcourt le contenu de ce fichier texte en comptabilisant le nombre total de lignes, d'espaces et de caractères alpha-numériques. Le programme affiche ces valeurs avant de se terminer.

Exercice 2

Implémentez un programme qui prend en paramètre le nom d'un fichier texte ainsi qu'un mot-clef. Le programme parcourt le contenu de ce fichier et, pour chaque ligne où le mot clef apparaît, affiche la ligne en question dans le terminal.

Exercice 3

Implémentez un programme qui prend en paramètre un nom de fichier et deux chaînes de caractères. Le programme remplace chaque occurrence de la première chaîne par une occurrence de la seconde dans le fichier d'origine.

Exercice 4

Implémentez un programme qui prend en paramètre deux noms de fichiers texte et affiche l'ensemble des lignes du second fichier qui n'apparaissent pas dans le premier.

Travail de groupe

Implémentez un programme qui permet de gérer une base de données de membres d'une ASBL. Les membres sont identifiés par leur adresse email. La base de données doit enregistrer le nom, prénom, numéro de téléphone et adresse email du membre ainsi que son status (actif/inactif). Le programme doit proposer un menu permettant d'ajouter ou de supprimer un membre, d'afficher la liste des membres actifs ou inactifs et de rechercher les informations d'un membre sur base de son nom.