TD1 : Les entrées/sorties

La bibliothèque standard C

Un premier exemple

Exercice 1.

qat.c

```
 1 : #include <stdlib.h>
 2 : #include <stdio.h>
 3 :
 4 : int main(int argc, char *argv[])
 5 : {
 6 :   FILE *fic;
 7 :   char tampon[100];
 8 :
 9 :   /*  nombre suffisant d'arguments */
10 :   if (argc <= 1)
11 :     return EXIT_FAILURE;
12 :   printf("Le nom du fichier est : %s\n",argv[1]);
13 :
14 :   fic = fopen(argv[1],"r");
15 :   if(fic == NULL )
16 :     return EXIT_FAILURE;
17 :
18 :   while(fgets(tampon,100,fic) != NULL)
19 :     printf("%s",tampon);
20 :
21 :   fclose(fic);
22 :
23 :   return EXIT_SUCCESS;
24 : }
```

Décrire le comportement du programme `qat.c`. A quelle commande du shell le programme `qat.c` vous fait penser ?

Fichier textes

Exercice 2. Écrire une fonction `prendreChaine` similaire à `fgets` mais qui supprime le caractère fin de ligne (`\n`). On utilisera la fonction `fgetc`. Le prototype des deux fonctions est le suivant :

```
int fgetc(FILE *fic) ;
char *prendreChaine(char *s, int n, FILE *fic) ;
```

Exercice 3. On suppose que l'on a un fichier appelé `mots5.txt` contenant dix mots de cinq lettres. Écrire une fonction tirant un mot au hasard dans le fichier `mots5.txt`. On utilisera la fonction `fseek` d'accès direct :

```
int fseek(FILE *fic, long deplacement, int origine) ;
```

Exercice 4. Écrire une fonction qui crée un nouveau fichier texte à partir de deux autres en entrelaçant les lignes des deux premiers fichiers.

Exercice 5. Écrire des fonctions `arrierefopen`, `arrieregetc`, `arrierefgets` qui se comportent exactement comme la fonctions `fopen`, `getc`, `fgets` sauf qui lisent un fichier de la fin vers son début.

Exercice 6. Modifier le programme `qat.c` pour qu'il affiche le nombre de fois où le fichier (texte) a été lu par `qat`. On pourra, par exemple, inclure dans le fichier une ligne contenant le nombre d'ouvertures. (Suggestion : ajouter cette ligne à la fin et se servir des fonctions `arrierefopen`, `arrieregetc`, `arrierefgets`).

Fichier binaires

Exercice 7. Écrire un programme qui copie un fichier texte ou binaire. On utilisera les fonctions `fread` et `fwrite` de lecture et d'écriture dans un fichier binaire :

```
int fread(void *dest, int taille, int nombre, FILE *fic) ;
int fwrite(void *srce, int taille, int nombre, FILE *fic) ;
```

Exercice 8. On suppose que l'on a déclaré le type fiche suivant :

```
typedef struct fiche
{
  char nom[20] ;
  int age ;
  int jour-naissance ;
} Fiche ;
```

On veut écrire un bibliothèque de fonctions pour gérer un fichier contenant des donnés de type `fiche`. En particulier, on veut écrire les fonctions `ajouter` pour ajouter une fiche, `chercherNom` pour obtenir la fiche dont le `nom` correspond à un nom donné et `chercherNum` pour obtenir la n-ième fiche où n est le paramètre.

Les primitives Posix

Exercice 9. Analyser le programme `autreqat.c` :

autreqat.c

```
 1 : #include <stdlib.h>
 2 : #include <unistd.h>
 3 : #include <fcntl.h>
 4 :
 5 : int main(int argc, char *argv[])
 6 : {
 7 :   int desc;
 8 :   char tampon[TAILLETAMPON];
 9 :   size_t no_lu;
10 :
11 :   if (argc <= 1)
12 :     return EXIT_FAILURE;
13 :
14 :   if((desc = open(argv[1],O_RDONLY)) == -1 )
15 :     return EXIT_FAILURE;
16 :
17 :   while((no_lu = read(desc,tampon,TAILLETAMPON)) >0)
18 :     write(STDOUT_FILENO,tampon,no_lu);
19 :
20 :   close(desc);
21 :
22 :   return EXIT_SUCCESS;
23 : }
```

Commenter la suite de commandes qui suivent, et en donner le résultat :

```
[utilisateur@localhost repcourant]$ ls --block-size=1 -s longfichier.ps
3158016 longfichier.ps
[utilisateur@localhost repcourant]$ for i in 1 2 4 10 50 100 1000 ;do \
echo ; \
gcc -Wall -pedantic -DTAILLETAMPON=$i autreqat.c ; \
time a.out longfichier.ps > /dev/null ;\
done
```