université de BORDEAUX

Collège Sciences et technologies

Année	2019-2020	Type	Devoir Surveillé
Licence	Informatique		
Code UE	4TIN503	Épreuve	Programmation Système
Date	24/10/2019	Documents	Non autorisés
Début	8h00	Durée	1h15

Lisez l'intégralité du sujet avant de commencer à répondre aux questions. **La plupart des questions peuvent être traitées même si vous n'avez pas répondu aux précédentes.** *À la fin du sujet, vous trouverez un memento vous rappelant la syntaxe de quelques appels système utiles.*

1 Questions de cours (échauffement)

Question 1 Les entrées-sorties effectuées au moyen des primitives de la bibliothèque standard (e.g. `fread`, `fwrite`) sont-elles généralement plus performantes que les appels systèmes correspondants (e.g. `read`, `write`), ou est-ce le contraire ? Que renferme principalement le type `FILE` ?

Question 2 Dans quelles circonstances un processus peut-il se voir délivrer le signal `SIGSEGV` ? Qui lui envoie ce signal ? Expliquez la succession d'événements ayant conduit à cet envoi. Le signal est-il forcément fatal au destinataire ?

2 Fichiers, processus et tubes

On souhaite disposer d'outils rudimentaires permettant de construire et de consulter des archives de fichiers, c'est-à-dire des fichiers qui contiennent eux-mêmes d'autres fichiers. Une archive est une concaténation de contenus de fichiers, chaque contenu étant précédé d'un entête de trois champs :

— un entier indiquant la longueur totale (en octets) des deux champs suivants et du contenu du fichier ;
— un entier spécifiant la longueur du nom de fichier ;
— les caractères formant le nom du fichier.

La figure 1 illustre la structure d'une archive dans laquelle on a d'abord ajouté un fichier `toto.txt` de 34 octets, puis un second fichier `f.txt` de 25 octets.

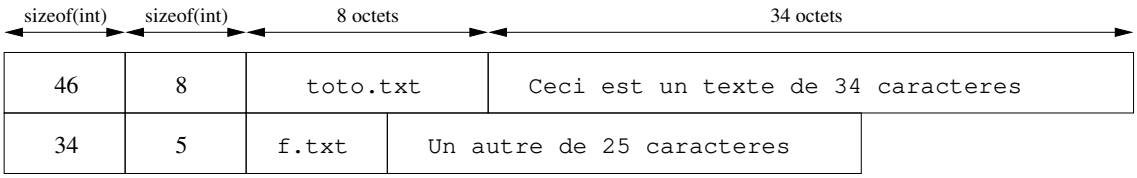

FIGURE 1 – Structure d'une archive contenant deux fichiers, d'une taille totale de $4 + 46 + 4 + 34 = 88$ octets.

On suppose que l'on dispose[1] d'un programme `arch-add` permettant d'ajouter un fichier à une archive (existante ou créée pour l'occasion). Ce programme prend en paramètre le nom de l'archive et le nom du fichier à ajouter. Si le nom du fichier est « - », alors le programme archive les données lues sur l'entrée standard, et le nom sous lequel le contenu va figurer dans l'archive doit être spécifié en troisième paramètre. Voici comment l'archive de la figure 1 a été construite :

```
[my-machine] ls -l
-rw-r--r--  1 rnamyst  staff    34 21 oct 19:25 toto.txt
[my-machine] arch-add arch toto.txt
[my-machine] ls -l arch
-rw-r--r--  1 rnamyst  staff    50 21 oct 20:48 arch
[my-machine] echo -n "Un autre de 25 caracteres" | arch-add arch - f.txt
[my-machine] ls -l arch
-rw-r--r--  1 rnamyst  staff    88 21 oct 20:50 arch
```

1. On ne demande pas le code de `arch-add`

Question 1 Écrivez une fonction `lire_entier` qui retourne la valeur d'un entier lu depuis un fichier dont le descripteur est passé en paramètre :

```
unsigned lire_entier (int fd)
{ ... }
```

On suppose que l'entier est stocké sous forme brute dans le fichier (i.e. sur `sizeof(unsigned)` octets). On suppose également que le fichier est déjà ouvert. En cas de problème, la fonction doit afficher une erreur sur la sortie standard et terminer le programme.

Question 2 On souhaite pouvoir extraire des données depuis une archive à l'aide d'une commande `arch-cat` :

`arch-cat archive_file option file_no`

Cette commande permet d'envoyer sur la sortie standard soit le nom du n-ième fichier d'une archive (option `name`), soit son contenu (option `content`). En supposant l'existence de l'archive « `arch` » décrite précédemment, voici un exemple d'utilisation :

```
[my-machine] arch-cat arch name 2
f.txt
[my-machine] arch-cat arch content 2
Un autre de 25 caracteres
```

Donnez le code du programme `arch-cat`, en veillant à minimiser le nombre d'octets lus depuis l'archive. Le programme doit retourner un code d'erreur $\neq 0$ si le numéro (3e argument) est trop grand.

Question 3 En supposant que l'on dispose uniquement de la version compilée du programme `arch-cat`, nous allons écrire un programme `arch-extract` qui extrait un fichier à partir d'une archive :

`arch-extract archive_file file_no`

Exemple :

```
[my-machine] ls -l
-rw-r--r--  1 rnamyst  staff    88 21 oct 20:50 arch
[my-machine] arch-extract arch 1
[my-machine] ls -l
-rw-r--r--  1 rnamyst  staff    88 21 oct 20:50 arch
-rw-r--r--  1 rnamyst  staff    34 21 oct 19:25 toto.txt
[my-machine] cat toto.txt
Ceci est un texte de 33 caracteres
```

L'idée est d'invoquer le programme `arch-cat` deux fois. Une première fois pour récupérer, au travers d'un tube établi entre `arch-cat` et le programme principal, le nom du fichier. Une seconde fois pour récupérer son contenu. Cette dernière opération peut être réalisée très simplement en pensant à rediriger la sortie de `arch-cat`...

Donnez le code du programme `arch-extract`. On supposera que le nom d'un fichier inclus dans une archive n'excède jamais 1024 caractères. Veillez à bien propager une éventuelle erreur levée par le programme `arch-cat`.

Memento

```
int open(const char *pathname, int flags, mode_t mode);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
off_t lseek(int fd, off_t offset, int whence);
/* whence = SEEK_SET ou SEEK_CUR ou SEEK_END */
int dup2(int oldfd, int newfd);
int execlp(const char *file, const char *arg0, ...);
int execvp(const char *file, char *const argv[]);
int pipe(int fildes[2]);
pid_t wait(int *stat_loc);
```