université de BORDEAUX

Collège Sciences et technologies

Année	2020-2021	**Type**	Entrainement
Master	Informatique		
Code UE	4TIN503EX	**Épreuve**	Programmation Système
Date	10/11/2020		
Début	15h30	**Durée**	1h20

1 Questions de cours

Question 1 Dans un système d'exploitation interactif tel qu'Unix, rappelez précisément pourquoi un processus exécutant une boucle infinie ne monopolise pas pour autant le processeur s'il existe d'autres processus prêts dans le système.

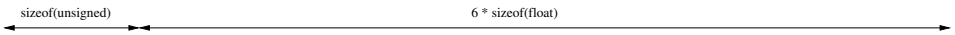
Question 2 Expliquez ce qu'est un signal *masqué*. Que se passe-t-il lorsque plusieurs occurrences d'un signal sont envoyées vers un processus qui a masqué le signal ?

2 Fichiers

On se place dans le contexte d'un programme de dessin qui manipule des figures géométriques en deux dimensions : des lignes, des carrés, des ronds et des triangles. Chaque figure est définie par son type (le caractère 'L', 'C', R' ou 'T'), sa couleur (codée sur un entier 32 bits non signé) et :

— dans le cas d'une ligne, les coordonnées des deux extrêmités (x_1, y_1) et (x_2, y_2) stockées au moyen de 4 nombres réels (`float`) ;
— dans le cas d'un carré, des coordonnées du coin inférieur gauche (x, y) et de la longueur du côté (3 nombres réels au total) ;
— dans le cas d'un rond, des coordonnées du centre (x, y) et du rayon (3 nombres réels au total) ;
— dans le cas d'un triangle, des coordonnées des trois sommets (6 nombres réels au total.

Le programme est doté d'une fonction de sauvegarde qui enregistre dans un même fichier les figures dans l'ordre où elles ont été dessinées par l'utilisateur. Le fichier contient, pour chaque figure, les informations décrites précédemment sous forme binaire. Voici un exemple de fichier contenant 4 figures : un rond, deux triangles et un carré. La taille de ce fichier est donc de 92 octets (4 × sizeof(char) + 4 × sizeof(unsigned) + 18 × sizeof(float)).



T	0xFF0000	1.200	4.300	1.500	1.890	3.000	2.000
R	0xFFFF00	6.200	8.100	2.500			
T	0x008000	0.000	0.000	3.000	3.000	6.000	0.000
C	0xF080FF	5.000	12.000	3.000			

FIGURE 1 – Exemple de fichier de sauvegarde « `figs` » contenant quatre figures.

Question 1 Écrivez un programme `afficher` qui prend en arguments un nom de fichier et un caractère spécifiant le type de figure que l'on souhaite afficher. Voici à quoi ressemble la sortie standard de ce programme appliqué au fichier d'exemple précédent :

```
[my-machine] afficher figs T
Triangle 1 : couleur 0xFF0000, (1.200, 4.300) (1.500, 1.890) (3.000, 2.000)
Triangle 2 : couleur 0x008000, (0.000, 0.000) (3.000, 3.000) (6.000, 0.000)
```

Question 2 Pour accélérer le parcours du fichier lorsque l'on ne s'intéresse qu'à un seul type de figure, on se propose de créer un fichier d'index pour chaque type de figure. Ces fichiers d'index (de noms respectifs `L.idx`, `C.idx`, `R.idx` et `T.idx`) contiendront la liste des positions des figures du type voulu dans le fichier de sauvegarde. Toujours à titre d'illustration, en reprenant la sauvegarde `figs` précédente, le fichier `T.idx`, qui indexe donc les triangles, devrait contenir les deux entiers 0 et 46.

Écrivez un programme `indexer` générant les quatre fichiers d'index à partir du fichier contenant les figures dont le nom est passé en paramètre. Exemple d'appel :

```
[my-machine] indexer figs
```

Les positions devront être écrites en binaire sur `sizeof(ssize_t)` octets (et non comme une chaîne de caractères représentant l'entier en base 10). Si le fichier `index` existe, il devra être tronqué. Sinon, il devra être créé.

Question 4 Réécrivez le programme `afficher` en utilisant le fichier d'index adéquat pour accélérer le parcours dans le fichier d'entrée.

3 Gestion de processus

Question 1 Écrire une fonction de prototype

```
int pipe3 (char *cmd1, char *arg1, char *cmd2, char *arg2, char *cmd3, char *arg3);
```

où `cmd1`, `cmd2` et `cmd3` désignent des noms de commandes ayant chacune exactement un argument, et qui lance les processus correspondant à la commande shell :

```
$ cmd1 arg1 | cmd2 arg2 | cmd3 arg3
```

On demande que le processus qui exécute la fonction `pipe3` crée 3 processus fils, le premier exécutant `cmd1`, le second exécutant `cmd2`, et le 3^{e} exécutant `cmd3`.

Question 2 Modifiez le profil de la fonction `pipe3` de manière à permettre le lancement de commandes à nombre arbitraire d'arguments. Indiquez les modifications à apporter au code de la fonction.

Question 3 Modifier le programme pour que le processus père affiche sur la sortie erreur standard un message si l'une ou plusieurs des trois commandes se terminent anormalement, en précisant le `pid` des processus fautifs.

Memento

```
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int open(const char *pathname, int flags, mode_t mode);
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 *arg, ...);
int execvp(const char *file, char *const argv[]);
size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
size_t fwrite(void *ptr, size_t size, size_t nitems, FILE *stream);
int fscanf(FILE *stream, char *format, ...);
```