



LIFSE Contrôle

Contrôle du mardi 28/02/23 - 15 minutes

Nom : ADAM  
Prénom : OUMAR Adam  
No. étu. : 12110686

Numéro d'étudiant :

0 0 0 0 0 0 0 0  
1 1 1 1 1 1 1 1  
2 2 2 2 2 2 2 2  
3 3 3 3 3 3 3 3  
4 4 4 4 4 4 4 4  
5 5 5 5 5 5 5 5  
6 6 6 6 6 6 6 6  
7 7 7 7 7 7 7 7  
8 8 8 8 8 8 8 8  
9 9 9 9 9 9 9 9

Utilisez un stylo noir (pas au crayon de bois), et répondez uniquement dans les cadres prévus à cet effet.

Aucun document n'est autorisé. Les téléphones, ordinateurs, et toutes communication avec les autres étudiants sont interdits. Une seule réponse par question.

Question 1 Le chemin ././ désigne

- ☐ le répertoire enfant  
☒ le répertoire courant  
☒ le répertoire parent

Question 2 En ligne de commandes, quel est le nom de la variable d'environnement qui contient la liste de tous les répertoires où vont être cherchés les exécutables ?

- ☒ PATH  
☒ HOME  
☐ USER  
☐ PWD

Question 3 D'après la page de manuel, l'en-tête de l'appel système read() est ssize\_t read(int fd, void \*buf, size\_t count). On précise que le fichier /dev/random existe et peut être lu pour obtenir des octets aléatoires. On considère l'extrait de code suivant :

```
1 int f = open("/dev/random", O_RDONLY);  
2 char t;  
3 int nbrd = read(f, &t, 1);  
4 close(f);
```

Ce morceau de code

- ☐ va provoquer une erreur à la compilation  
☒ va s'exécuter normalement, et charger une octet (aléatoire) dans t  
☐ risque de provoquer une erreur de segmentation à l'exécution

Question 4 Dans le contexte des systèmes d'exploitation, NFS est l'acronyme d'un

- ☐ nouveau fichier système  
☒ système de fichiers réseaux  
☐ système de chiffrement de fichiers

Question 5 Dans le contexte des fichiers, on parle de typage fort quand

- ☒ le type des fichiers est déterminé par leur contenu  
☒ le type des fichiers est déterminé par leur nom  
☐ le type des fichiers est déterminé par leur intention



## 1 Exercice

On considère le code suivant, qui permet de réaliser une commande `mycat` qui ressemble à `cat` (en incluant les bons fichiers, ce programme pourrait être compilé sans problème) :

```
1  #define LEN 16
2  int main(int argc, char *argv[]) {
3      char buf[LEN];
4      int fd = open(argv[1], O_RDONLY);
5      int nbrd = read(fd, buf, LEN);
6      while(nbrd > 0) {
7          int nrem = nbrd;
8          int nbwr = 0;
9          while(nrem > 0) {
10             int t = write(STDOUT_FILENO, buf+nbwr, nrem);
11             nrem -= t;
12             nbwr += t;
13         }
14         nbrd = read(fd, buf, LEN);
15     }
16     close(fd);
17     return 0;
18 }
```

**Question 6** On lance le programme avec `./mycat tutu.txt` dans le shell ; quelle commande permet d'afficher l'entier retourné par le programme ?

- ☐ `echo $!`
- ☐ `echo $return`
- ☒ `echo $?`

**Question 7** Si on appelle `mycat` avec `./mycat toto.txt titi.txt` alors

- ☒ le programme se termine sur un échec
- ☐ dans le programme `argv[1]` désigne la chaîne "titi.txt"
- ☒ dans le programme `argv[1]` désigne la chaîne "toto.txt"

**Question 8** Dans ce programme, `STDOUT_FILENO` est

- ☐ le descripteur de fichier du fichier d'entrée
- ☐ le descripteur de fichier de l'entrée standard
- ☒ le descripteur de fichier de la sortie standard

**Question 9** L'appel à `write()` de la ligne 10 permet d'écrire

- ☐ au moins `nrem` octets
- ☐ exactement `nrem` octets
- ☒ au plus `nrem` octets

**Question 10** Lors de chaque entrée dans la boucle `while(nbrem > 0)` à la ligne 9, `nrem` donne

- ☐ le nombre d'octets présents dans le buffer `buf` déjà écrits
- ☒ le nombre d'octets présents dans le buffer `buf` restant à écrire
- ☒ le nombre d'octets qui vont être lus sur `STDOUT_FILENO` par l'appel

**Question 11** On suppose que chaque appel à `read()` lit toujours le plus d'octets possible. Le fichier passé au programme contient exactement 54 octets. Lors de la dernière entrée dans la boucle `while(nbrem > 0)`, que vaut `nbrd` ?

- ☒ 6
- ☒ on ne peut pas savoir
- ☐ 5