1:1

Programmation des Systèmes

Exemple de QCM

Consignes

Le non-respect des consignes suivantes pourra bien entendu être pénalisé.

1. L'utilisation du mini-man, des notes de cours, de TD et de TP est autorisée.
2. **Les réponses se font sur la feuille de réponses séparée** que vous pourrez détâcher **avec soin** pour la remplir plus facilement.
3. **Vérifiez que toutes les pages du sujet soient lisibles et correctement imprimées dès le début de l'épreuve**.
4. **Le crayon de papier est interdit** : utiliser un stylo **bleu** ou **noir**.
5. Pour cocher une case, remplissez plus de 50% de sa surface (■ ou ● mais pas ☒ ou ☑) **sans dépasser sur une autre case**.
6. En cas d'erreur, **effacez totalement** (au blanc correcteur) **toute trace** sur une case qui doit rester blanche (effacez au passage son cadre, pour être sûr).
7. Les **réponses fausses** font **perdre des points** ; prenez donc bien le temps nécessaire avant de répondre.
8. Les questions **sans** le symbole « ★ » sont à choix **unique** :
   - cochez une seule réponse sinon la question est ignorée.
9. Les questions **avec** le symbole « ★ » sont à choix **multiple** :
   - chaque réponse proposée peut être bonne ou mauvaise ; il peut y compris y avoir 0 (ou 1 seule) réponse correcte ; elles peuvent aussi être toutes correctes,
   - chaque bonne réponse rapporte des points, chaque mauvaise réponse en fait perdre.
10. Une question laissée sans réponse ne donne ni point ni pénalité.
11. Les extraits de code donnés dans l'énoncé ne mentionnent pas les `#include` : vous supposerez qu'ils sont toujours renseignés correctement (en particulier, ils ne causent aucun bug)[1].

[1] Cela explique que les numéros de ligne ne commencent pas forcément à 1, par exemple.

1 Échauffement

Question 1 Un chemin absolu...

- [A] doit commencer par « . » ou « .. »
- [B] peut commencer par « . » ou « .. »
- [C] doit commencer par « / »
- [D] peut commencer par « / »

Question 2 Un chemin relatif...

- [A] doit commencer par « . » ou « .. »
- [B] peut commencer par « . » ou « .. »
- [C] doit commencer par « / »
- [D] peut commencer par « / »

Question 3 ★ On réalise un appel à `access("./toto.txt", R_OK | W_OK)`, la fonction retourne -1, et `errno` est positionné à `EACCES`. Quelles sont les explications possibles ?

- [A] le fichier `toto.txt` existe mais n'est pas accessible en écriture
- [B] le fichier `toto.txt` n'existe pas
- [C] le fichier `toto.txt` existe mais n'est pas accessible en lecture

Question 4 On exécute les commandes suivantes :

```
$ ls
$ ln -s file file
$ cat file
cat: file: Trop de niveaux de liens symboliques
```

Que peut-on en conclure ?

- [A] un cycle empêche de trouver une cible de lien symbolique
- [B] `file` est un mot réservé, ne peut être utilisé pour un lien symbolique
- [C] il n'y a plus de place sur le disque

Question 5 ★ Que peut-on conclure de :

```
$ mkdir rep/sousrep
mkdir: impossible de créer le répertoire « rep/sousrep »: Aucun fichier ou dossier de ce type
```

- [A] le répertoire `rep/sousrep` existe déjà
- [B] le disque est plein
- [C] je n'ai pas les droits nécessaires pour créer un répertoire
- [D] le répertoire `rep` n'existe pas
- [E] le répertoire `rep` existe déjà

Question 6 ★ On considère la session suivante :

```
$ ls -l file42
---x------ 1 ray  mond    8496 oct 11 09:32 file42
```

Quelles sont les affirmations correctes :

- [A] le fichier `file42` est un lien symbolique
- [B] le propriétaire du fichier `file42` a le droit de l'exécuter
- [C] la première ligne du fichier `file42` contient la sous-chaîne de caractères `"42"`
- [D] le propriétaire du fichier `file42` a le droit d'en lire le contenu

2 Données d'Alice

Dans les questions de cette section, on considère l'arborescence suivante.

```
                              st_size   st_blocks
/home/alice                      4096           8
├── fic                             1           8
└── rep                          4096           8
    ├── fic                         2           8
    ├── liensym -> fic              3           0
    ├── sousrep                  4096           8
    │   ├── fic                     4           8
    │   ├── soussous             4096           8
    │   │   └── fic                 5           8
    │   └── sym -> ../sousrep2     11           0
    └── sousrep2                 4096           8
        └── rome                    6           8
```

Question 7 ★ On suppose que le répertoire courant est `/home/alice/rep`. Parmi les chemins suivants, lesquels menent à « rome » ?

- A `/home/alice/sousrep/rome`
- B `sousrep2/rome`
- C `../rep/sousrep2/rome`
- D `fic/../sousrep2/rome`
- E `liensym/sousrep2/rome`
- F `/home/alice/rep/sousrep2/rome`
- G `sousrep/sym/rome`
- H `sousrep/../rome`

Question 8 ★ On suppose que le répertoire courant est `/home/alice/rep/sousrep`. Quelles sont les affirmations correctes parmi les suivantes ?

- A après un appel à `stat("sym", &st)`, `st.st_size` vaut 11
- B après un appel à `lstat("sym", &st)`, `st.st_size` vaut 11
- C après un appel à `stat("sym", &st)`, `st.st_size` vaut 4096
- D après un appel à `lstat("sym", &st)`, `st.st_size` vaut 4096

Question 9 On suppose que l'on est dans le répertoire `/home/alice`. Quel est le résultat de `du -b -L rep/liensym` (taille apparente, en suivant les liens) ?

A 0 B 1 C 2 D 3 E 4 F 5

Question 10 On utilise ici la commande `mdu` écrite en TP. Quelle formule correspond au résultat que donne `./mdu -b /home/alice`[a] ?

- A $1+2+4+5+6+5\times4096$
- B 10×8
- C $1+2+3+4+5+6+11+5\times4096$
- D $10\times8+2\times8$
- E $1+2\times2+3+4+5+6+11+6\times4096$

[a] On supposera qu'il n'y a pas deux liens physiques vers un même i-nœud dans le répertoire `/home/alice`.

Question 11 ★ On suppose que le répertoire courant est `/home/alice/rep`. Après l'exécution de `rm fic`, `stat("liensym", &st)`...

- A ... positionne `errno` à `EACCES`
- B ... positionne `errno` à `ENOENT`
- C ... retourne -1
- D ... retourne 0

Question 12 ★ Toujours dans le répertoire `/home/alice/rep`, après l'exécution de `rm fic`, `lstat("liensym", &st)`...

A ... positionne `errno` à `ENOENT`
B ... retourne 0
C ... positionne `errno` à `EACCES`
D ... retourne -1

3 Compter les sous-répertoires

On souhaite écrire un outil qui calcule la distance du répertoire courant à la racine, c'est-à-dire le nombre de sous-répertoires de la racine jusqu'au répertoire courant.

Par exemple :

```
$ echo $PWD
/a/b/c/d/e
$ count
4
```

On propose le code suivant :

```
10 int main() {
11     struct stat pathname_stat, root_stat;
12     int status, count = -1;
13     char pathname[PATH_MAX];
14     char *pwd;
15
16     pwd = getcwd((char *) NULL, 0);
17     /* assert ? */
18     snprintf(pathname, PATH_MAX, "%s", pwd);
19     /* assert ? */
20
21     status = stat("/", &root_stat);
22     /* assert ? */
23     status = stat(pathname, &pathname_stat);
24     /* assert ? */
25
26     while (pathname_stat.st_ino != root_stat.st_ino) {
27         /* assert ? */
28         strcat(pathname, "/..");
29         /* assert ? */
30         count++;
31         status = stat(pathname, &pathname_stat);
32         /* assert ? */
33     }
34
35     printf("%d\n", count);
36     return 0;
37 }
```

Question 13 ★ Pour que le code soit plus sûr, il faut ajouter des assertions aux lignes...

A 17 B 19 C 22 D 32 E 29 F 24 G 27

Question 14 L'implantation est réalisée en utilisant `stat`. Si on utilise `lstat` à la place, cela change-t-il quelque chose ?

A cela dépend de l'arborescence B non C oui

Question 15 ★ On propose les correctifs de bugs suivants. Lesquels sont judicieux ?

- A s'assurer que la longueur de `pathname` ne dépasse pas `PATH_MAX` ligne 20
- B initialiser `count` à 0
- C remplacer la condition ligne 26 par « `pathname_stat.st_dev != root_stat.st_dev` »
- D s'assurer que la longueur de `pathname` ne dépasse pas `PATH_MAX` ligne 29
- E ajouter « `&& pathname_stat.st_dev != root_stat.st_dev` » à la condition ligne 26

4 Liste des fichiers

Les questions de cette section porte sur l'arborescence du répertoire `examen` décrite par la figure 1 à la page 9.

Question 16 Considérez la fonction `list_files` donnée figure 2 (p. 9).
Supposez que la fonction `list_files()` est appelée avec l'argument `"sub"` et que le répertoire courant est `examen`. Quel sera l'affichage sur la sortie standard ?

- A
```
sub/reponses.txt
sub/../pds.txt
sub/contenu.txt
sub/logo.jpg
sub/contenu.txt
```
- B
```
sub/reponses.txt
```
- C
```
sub/reponses.txt
```
- D
```
sub/reponses.txt
```

Question 17 On voudrait modifier la fonction `list_files` pour lister les fichiers dans les sous-répertoires de manière récursive. Quel code faut-il écrire ligne 24 ?

- A
```
if (strcmp(dirp->d_name, ".") != 0 && strcmp(dirp->d_name, "..") != 0)
  list_files(filename);
```
- B
```
if (strcmp(dirp->d_name, ".") != 0 && strcmp(dirp->d_name, "..") != 0)
  list_files(dirp->d_name);
```
- C Ce n'est pas possible.
- D `list_files(filename);`

Question 18 Considérez encore une fois la fonction `list_files()` modifiée comme à la question précédente pour explorer récursivement les sous-répertoires.
Le lien symbolique `examen/sub/tp1` pointe vers un répertoire en dehors de l'arborescence. Supposez que le répertoire courant est `sub`. Qu'est-ce qui s'affiche si on appelle `list_files("tp1")` ?

- A le contenu complet du répertoire cible et de ses sous-repertoires
- B les noms de tous les fichiers dans l'arborescence `../../../TP1`
- C rien
- D `tp1`

5 Questions diverses sur le système de fichiers

Question 19 ★ On veut recopier l'entrée standard sur la sortie standard. Considérons l'implémentation suivante.

```
3  #define TAILLE 8
4
5  int main() {
6      int r;
7      char tampon[TAILLE];
8
9      while((r = read(0, tampon, TAILLE)) != -1) {
10         write(STDOUT_FILENO, tampon, TAILLE);
11     }
12
13     return 0;
14 }
```

Indiquez s'il y a des bugs dans le code de main.

- A 1 erreur ligne 7
- B 1 erreur ligne 9
- C 2 erreurs ligne 9
- D 3 erreurs ligne 9
- E 1 erreur ligne 10
- F 2 erreurs ligne 10
- G 3 erreurs ligne 10

Question 20 ★ On dispose d'un fichier qui fait 100 octets ouvert en lecture sur le descripteur `fd`. On exécute l'instruction suivante `lseek(fd,-50,SEEK_END)` et on suppose que la variable `buf` est un tableau de 100 octets. Parmi les résultats suivants, lesquels sont possibles[a] :

- A l'instruction `read(fd,&buf,10)` retourne 10
- B l'instruction `read(fd,&buf,60)` retourne 50
- C l'instruction `read(fd,&buf,60)` retourne 0
- D l'instruction `read(fd,&buf,50)` retourne 50

[a] On supposera que le fichier n'est pas modifié simultanément par un autre processus.

Question 21 Sur un fichier donné, on voudrait se déplacer à la fin du fichier. Quelle fonction faut-il appeler ?

- A `of = read(fd, buf, sizeof(fd));`
- B `of = lseek(fd, 0, SEEK_END);`
- C `of = lseek(fd, 0, SEEK_CUR);`
- D `of = read(fd, buf, sizeof(buf));`

Question 22 ★ La fonction suivante compte le nombre de lignes dans les prochains `BUF_SIZE` octets d'un fichier. La fonction contient un (ou plusieurs) bug(s), trouvez-les :

```
4  int count_lines(int fd) {
5      char buf[BUF_SIZE];
6      int s, i;
7      int cnt = 0;
8
9      s = read(fd, buf, BUF_SIZE * sizeof(char));
10
11     for (i = 0; i <= BUF_SIZE; i++)
12         if (buf[i] == '\n')
13             cnt++;
14
15     return cnt;
16 }
```

A La borne de la boucle `for` doit être `i < BUF_SIZE`.

B La condition ligne 12 devrait être `strcmp(buf[i], "\n") == 0`.

C La borne de la boucle `for` doit être `i < s`.

D On ne vérifie pas la valeur de retour de `read`.

6 Processus

Question 23 ★ Considérons le programme suivant :

```
6  int main() {
7      int i = 0;
8
9      switch (fork()) {
10         case -1:
11             printf("échec ");
12             exit(EXIT_FAILURE);
13         case 0:
14             i++;
15             printf("%d ", i);
16             exit(EXIT_SUCCESS);
17         default:
18             i--;
19             printf("%d ", i);
20             exit(EXIT_SUCCESS);
21     }
22 }
```

Quels affichages ce programme peut générer parmi les choix suivants ?

A échec 1

B 0 0

C 1 0

D échec

E -1 1

F 0 1

G 1 -1

H -1 échec

Question 24 ★ Considérez le programme suivant :

```c
int main() {
    pid_t child;
    int i;

    for (i = 0; i < 2; i++) {
        child = fork();
        if (child == 0) {
            printf("%d, ", i);
            return 0;
        } else if (child < 0)
            exit(EXIT_FAILURE);
    }
    wait(NULL);
    wait(NULL);
    printf("%d\n", i);
    return EXIT_SUCCESS;
}
```

Quels sont les affichages possibles sur la sortie standard ?

A 1, 0, 2
B 1, 2, 0
C 2, 0, 1
D 2, 1, 0
E 0, 2, 1
F 0, 1, 2

```
$ tree examen
examen/
├── data.bin
├── panorama.jpg
├── pds.txt
├── questions.txt
└── sub
    ├── contenu.txt -> ../pds.txt
    ├── logo.jpg -> ../panorama.jpg
    ├── reponses.txt
    └── tp1 -> ../../../TP1/

$ ls -l -R examen
examen:
total 52
-rw-rw-r-- 1 lipari lipari   17 oct.   2 18:15 data.bin
-rw-rw-r-- 1 lipari lipari 1024 oct.   2 18:28 panorama.jpg
-rw-rw-r-- 1 lipari lipari   34 oct.   2 18:15 pds.txt
-rw-rw-r-- 1 lipari lipari   32 oct.   2 18:15 questions.txt
drwxrwxr-x 2 lipari lipari 4096 oct.   2 18:11 sub

examen/sub:
total 32
lrwxrwxrwx 1 lipari lipari   10 oct.   2 18:11 contenu.txt -> ../pds.txt
lrwxrwxrwx 1 lipari lipari   15 oct.   2 18:10 logo.jpg -> ../panorama.jpg
-rw-rw-r-- 1 lipari lipari   17 oct.   2 18:16 reponses.txt
lrwxrwxrwx 1 lipari lipari   13 oct.   2 20:05 tp1 -> ../../../TP1/
```

Figure 1: Exploration du répertoire `examen`

```c
void list_files(const char *dirname) {
    DIR *dir;
    struct dirent *dirp;
    struct stat st;
    char filename[PATH_MAX];

    dir = opendir(dirname);
    assert(dir != NULL);

    while ((dirp = readdir(dir)) != 0) {
        snprintf(filename, PATH_MAX, "%s/%s", dirname, dirp->d_name);
        assert(lstat(filename, &st) == 0);

        if (S_ISDIR(st.st_mode)) {
            /* code à insérer */
        } else if (!S_ISLNK(st.st_mode)) {
            printf("%s\n", filename);
        }
    }

    closedir(dir);
}
```

Figure 2: Fonction `list_files`

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

⟵ Veuillez coder votre numéro d'étudiant ci-contre et écrire vos nom et prénom dans la case ci-dessous.

Nom et prénom :
....
....

---

1 Échauffement

Question 1 : ☐A ☐B ☐C ☐D

Question 2 : ☐A ☐B ☐C ☐D

Question 3 : ☐A ☐B ☐C

Question 4 : ☐A ☐B ☐C

Question 5 : ☐A ☐B ☐C ☐D ☐E

Question 6 : ☐A ☐B ☐C ☐D

2 Données d'Alice

Question 7 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 8 : ☐A ☐B ☐C ☐D

Question 9 : ☐A ☐B ☐C ☐D ☐E ☐F

Question 10 : ☐A ☐B ☐C ☐D ☐E

Question 11 : ☐A ☐B ☐C ☐D

Question 12 : ☐A ☐B ☐C ☐D

3 Compter les sous-répertoires

Question 13 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G

Question 14 : ☐A ☐B ☐C

Question 15 : ☐A ☐B ☐C ☐D ☐E

4 Liste des fichiers

Question 16 : ☐A ☐B ☐C ☐D

Question 17 : ☐A ☐B ☐C ☐D

Question 18 : ☐A ☐B ☐C ☐D

5 Questions diverses sur le système de fichiers

Question 19 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G

Question 20 : ☐A ☐B ☐C ☐D

Question 21 : ☐A ☐B ☐C ☐D

Question 22 : ☐A ☐B ☐C ☐D

6 Processus

Question 23 : ☐A ☐B ☐C ☐D ☐E ☐F ☐G ☐H

Question 24 : ☐A ☐B ☐C ☐D ☐E ☐F