



+88/1/6+

L3 Informatique

Année 2023-2024



SY5 – Systèmes d'exploitation

QCM du 8 novembre 2023

Durée : 30 minutes

Énoncé constitué de 28 questions

Compléter le cadre d'identification ci-contre (numéro étudiant à coder en cochant une case par ligne).

Pour sélectionner une réponse, la case correspondante doit être seulement **cochée et non noircie**.

Pour annuler la sélection d'une réponse, vous pouvez soit **effacer** la coche, soit **noircir** intégralement la case, soit la recouvrir de correcteur blanc; dans ce cas, **ne pas redessiner la case**.

Nom : LIM		
Prénom : Julien		
Groupe : info 3		
Numéro étudiant :		
<input type="checkbox"/> 0 <input type="checkbox"/> 1 <input checked="" type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		
<input type="checkbox"/> 0 <input type="checkbox"/> 1 <input checked="" type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		
<input type="checkbox"/> 0 <input checked="" type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		
<input checked="" type="checkbox"/> 0 <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		
<input type="checkbox"/> 0 <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input checked="" type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		
<input type="checkbox"/> 0 <input checked="" type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		
<input type="checkbox"/> 0 <input type="checkbox"/> 1 <input type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input checked="" type="checkbox"/> 9		
<input type="checkbox"/> 0 <input type="checkbox"/> 1 <input checked="" type="checkbox"/> 2 <input type="checkbox"/> 3 <input type="checkbox"/> 4 <input type="checkbox"/> 5 <input type="checkbox"/> 6 <input type="checkbox"/> 7 <input type="checkbox"/> 8 <input type="checkbox"/> 9		

Dans l'ensemble du QCM, *toto* désigne un fichier ordinaire de contenu "abcdefghijkl" et *buf* un tableau de type `char[32]` rempli d'octets nuls (tous deux réinitialisés au début de chaque question). Les erreurs ne sont pas gérées; on suppose que tous les appels système réussissent.

Question 1 Combien d'appels à `write` le code suivant provoque-t-il, si la sortie standard est un terminal?

```
printf("abc\n"); printf("def");  
printf("ghi\n"); printf("jkl"); _exit(0);
```

☐ 0 ☐ 1 ☒ 2 ☐ 3 ☒ 4 ☐ plus

Question 2 Combien d'appels à `write` le code suivant provoque-t-il, si la sortie standard est redirigée sur un fichier ordinaire?

```
printf("abc\n"); printf("def");  
printf("ghi\n"); printf("jkl"); _exit(0);
```

☒ 0 ☐ 1 ☐ 2 ☐ 3 ☒ 4 ☐ plus

Question 3 Quel est l'affichage obtenu après exécution de :

```
fd = open("toto", O_RDONLY);  
do nb = read(fd, buf, 7); while (nb > 0)  
write(STDOUT_FILENO, buf, 7);
```

☒ abcdefghijkl ☐ hijklab ☐ abcdefg
☐ fghijkl ☒ hijklfg ☐ rien
☐ hijkl ☐ abhijkl ☐ autre chose

Question 4 Quel est l'affichage obtenu après exécution de :

```
fd = open("toto", O_RDONLY); read(fd, buf, 3);  
lseek(fd, 6, SEEK_CUR); read(fd, buf, 3);  
write(STDOUT_FILENO, buf, 6);
```

☐ abcghi ☐ def ☐ ghi
☐ abc ☒ abcjkl ☐ rien
☒ jkl ☐ abcdef ☐ autre chose

Après définition de deux descripteurs `fd1` et `fd2` (éventuellement égaux), on exécute les instructions suivantes :
`nb = read(fd1, buf, 3); write(STDOUT_FILENO, buf, nb); write(fd1, "***", 3);`
`nb = read(fd2, buf, 3); write(STDOUT_FILENO, buf, nb); write(fd2, "...", 3);`
Indiquer dans chaque cas ci-dessous l'affichage obtenu, ainsi que le contenu final du fichier *toto*.

Question 5 `fd1 = fd2 = open("toto", O_RDWR | O_TRUNC);`

☐ abc***ghijkl... ☐ abc ☐ abc***ghi... ☐ abcghi
☐ abcabc ☐ abc...ghijkl*** ☐ abc...ghijkl ☒ ***
☐ abcdefghijkl***... ☐ ... ☒ ***... ☒ rien

Question 6 `fd1 = fd2 = open("toto", O_RDWR | O_APPEND);`

☐ ***... ☒ abc ☐ abc...ghijkl*** ☒ abcdefghijkl***...
☐ abc***ghi... ☐ ... ☐ abcghi ☐ abc***ghijkl...
☐ abcabc ☐ *** ☒ abc...ghijkl ☐ rien



Question 7 `fd1 = open("toto", O_RDWR); fd2 = open("toto", O_RDWR);`

- ☒ abcab ☐ abc ☐ *** ☒ abc...ghijkl
☐ abcg ☐ ***... ☐ ... ☐ abc...ghijkl***
☐ abc***ghi... ☐ abcdefghijkl***... ☐ abc***ghijkl... ☐ rien

Question 8 `fd1 = fd2 = open("toto", O_RDWR);`

- ☐ abcab ☐ ... ☐ abc...ghijkl*** ☐ abc***ghijkl...
☐ *** ☒ abc***ghi... ☐ abc ☒ abcg ☐ abc...ghijkl
☐ abc...ghijkl ☐ abcdefghijkl***... ☐ ***... ☐ rien

Question 9 `fd1 = open("toto", O_RDWR | O_APPEND); fd2 = open("toto", O_RDWR | O_APPEND);`

- ☐ abc...ghijkl*** ☐ ... ☐ abc ☐ ***...
☒ abcdefghijkl***... ☐ abc***ghijkl... ☒ abcab ☐ abc...ghijkl
☐ abc***ghi... ☐ abcg ☐ *** ☐ rien

Question 10 `fd1 = open("toto", O_RDWR | O_APPEND); fd2 = open("toto", O_RDWR);`

- ☐ abcdefghijkl***... ☐ *** ☐ ... ☐ abc***ghi...
☐ abc ☐ ***... ☒ abc...ghijkl*** ☒ abcab
☐ abc...ghijkl ☐ abcg ☐ abc***ghijkl... ☐ rien

Pour chacune des 4 questions suivantes, cocher **exactement** les éléments **nécessaires** (valeur de flags et présence d'un 3^e argument) lors de l'appel `open(fic, flags, ...)` pour pouvoir simuler la redirection indiquée. On supposera que l'option `noclobber` du shell est activée, i.e. que les fichiers existants sont protégés contre l'écrasement par `>`, qui peut être forcé par `>|`.

Question 11 Pour simuler une redirection `< fic` :

- ☒ O_RDONLY ☐ O_RDWR ☐ O_APPEND
☐ O_EXCL ☐ O_CREAT ☐ O_WRONLY
☐ O_TRUNC ☐ le 3^e argument est nécessaire

Question 12 Pour simuler une redirection `> fic` :

- ☒ O_CREAT ☒ O_EXCL ☐ O_RDONLY
☐ O_APPEND ☐ O_TRUNC ☐ O_WRONLY
☐ O_RDWR ☒ le 3^e argument est nécessaire

Question 13 Pour simuler une redirection `>| fic` :

- ☒ O_TRUNC ☐ O_EXCL ☐ O_RDONLY
☒ O_WRONLY ☐ O_RDWR ☐ O_APPEND
☒ O_CREAT ☒ le 3^e argument est nécessaire

Question 14 Pour simuler une redirection `>> fic` :

- ☐ O_RDONLY ☒ O_WRONLY ☐ O_RDWR
☐ O_TRUNC ☐ O_EXCL ☒ O_CREAT
☒ O_APPEND ☒ le 3^e argument est nécessaire

Question 15 Quelle est la plus petite valeur que peut afficher ce programme s'il est exécuté dans un répertoire contenant deux sous-répertoires vides ?

```
cpt = 0; d1 = opendir(".");
while((e1 = readdir(d1)))
  if ((d2 = opendir(e1->d_name))) {
    while((e2 = readdir(d2))) cpt += 1;
    closedir(d2);
  }
printf("%d", cpt);
```

- ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6
☐ 7 ☐ 8 ☐ 9 ☐ 10 ☒ 11 ☐ plus

Question 16 Cocher tous les tests permettant de déterminer si un `mode_t m` est celui d'un fichier ordinaire.

- ☐ (m & S_IFMT) == S_IFREG
☐ (m & S_IFREG) == 0
☐ m != S_IFDIR
☐ (m & S_IFMT) != S_IFDIR
☐ m == S_IFREG
☐ (m & S_IFMT) != 0
☒ (m & S_IFMT) == S_IFREG
☐ (m & S_IFREG) != S_IFDIR
☐ (m & S_IFREG) != 0