L2 informatique - Université Paris Est à Marne-La-Vallée

Architecture des systèmes informatiques : Examen première session 2018-2019

Durée de l'épreuve : 2 heures

Les notes de cours, de TD et de TP ne sont pas autorisées. Les ordinateurs, calculatrices ainsi que les téléphones sont interdits et éteints durant l'épreuve. Il est possible à tout moment d'admettre la réponse (algorithme, fonction, programme, etc.) à une question non traitée à condition de le préciser explicitement. Il est très fortement préconisé de lire exhaustivement l'énoncé avant de commencer à répondre aux exercices.

Les justifications et explications sont toujours plus importantes que les résultats attendus aux questions. Ainsi, la qualité de la rédaction de vos réponses aura un impact très important sur la notation. Fournir un résultat correct sans aucune explication ne donne aucun point. Fournir une explication valide avec un résultat erroné peut donner un grand nombre de points sur les questions.

Le barème est juste donné à titre indicatif, il donne un indice sur la longueur et la difficulté des exercices.

Exercice 1 : Ouais, j'ai lu le cours 5 minutes... (4 points)

1) Dans quels endroits peut-on trouver de la mémoire dans un ordinateur moderne ?
2) Citez des spécificités qui caractérisent l'essence des systèmes d'exploitation de type Unix ?
3) À quoi sert le shell dans les systèmes de types Unix ? Que peut-on faire avec ?
4) À quoi sert le pipe | en bash ? Comment cela fonctionne ?

Exercice 2 : Lire dans les pensées (4 points)

Dans cet exercice, on vous propose des commandes Unix évoluées et fonctionnelles. Pour chacune de ces commandes, vous devez deviner précisément les intentions de auteur.

Bonus : Devinez les intentions de l'auteur de la commande et précisez s'il fait des erreurs (lesquelles, pourquoi…)

1) `ls -al | grep "^d" | wc -l`
2) `find my_code -name "*.py" | xargs cat | grep "^[ ]*for"`
3) On a un fichier personnes.csv dont les lignes ont la structure suivante : identifiant numérique, nom, prénom, age, mail. La commande est alors la suivante : `cat personnes.csv | grep "Martin" | wc -l`
4) `cat personnes.csv | cut -f 3 -d ',' | uniq | sort | wc -l`

Exercice 3 : changements de base (4 points)

1) Qu'affiche le programme suivant sur sa sortie standard ?

```c
#include <stdio.h>
int main(int argc, char* argv[]){
  printf("%X %X %X %X %X\n", 23, 54, 123, 241, 93);
  return 0;
}
```

1) Écrire en binaire la donnée hexadécimale suivante : 33 AA 3F D4 99 ?

Exercice 4 : Manipulations de fichiers (4 points)

En partant des trois fichiers et d'un répertoire suivants :

```
nborie@bayer:$ ls -lR
.:
total 4
drwxr-xr-x 2 nborie nborie 4096 mai   14 07:19 d
-rw-r--r-- 1 nborie nborie    0 mai   14 07:19 f1
-rw-r--r-- 1 nborie nborie    0 mai   14 07:19 f2
-rw-r--r-- 1 nborie nborie    0 mai   14 07:19 f3

./d:
total 0
```

Sachant que l'utilisateur **nborie** connaît le mot de passe super-utilisateur, quelles commandes doit-il faire pour obtenir la configuration suivante:

```
nborie@bayer:$ ls -lR
.:
total 4
drwxr-xr-x 2 nborie nborie 4096 mai   14 07:19 d
-rwxr-xr-x 1 nborie nborie    0 mai   14 07:19 f1
-rw-r--r-- 1 root   nborie    0 mai   14 07:19 f2

./d:
total 0
-rwxrwxrwx 1 nborie nborie 0 mai   14 07:19 f3
```

Exercice 5 : Fabricant de composants (6 points)

Une U.A.L. utilise le circuit suivant dans une de ses primitives processeurs. Ce circuit est composé de 6 postes XOR (ou exclusif). Il prend en argument un mot binaire sur 4 bits et a pour sortie un autre mot binaire sur 4 bits.

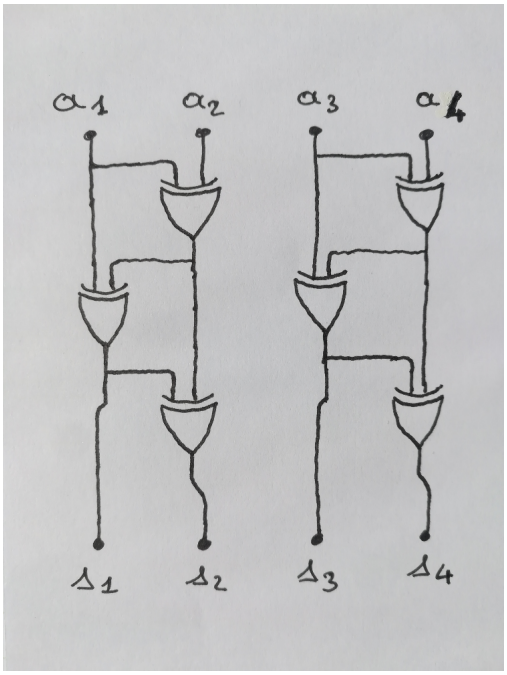

Figure 1: opération sur 4 bits

Décrire son effet et proposer un autre circuit produisant exactement les mêmes calculs avec moins de portes logiques.

Exercice 6 : Faites le planning (4 points)

On possède une machine à un seul coeur, elle ne peut exécuter qu'une seule tâche en même temps. Cette machine exécute plusieurs programmes simultanément (speudo-simultanément car un seul processus est élu à chaque quantum de temps pour exécution).

L'ordonnanceur de cette machine suit les règles suivantes :

- Il y a trois niveaux de priorité : fort - normal - faible. Tous les processus de niveau fort doivent être exécutés avant les processus de niveau normal et tous les processus de niveau normal doivent être exécutés avant les processus de niveau faible.
- À même niveau de priorités, les processus en attentes sont traités de manières équitables. L'ordonnanceur fait ainsi le tourniquet (algorithme de Round-Robin) parmi ces tâches pour distribuer les quantums de temps.
- Les nouvelles taches s'insèrent en dernier dans leur file de priorité associée.

À partir d'une liste de tâches à exécuter, vous devez proposer un planning quantum par quantum valide avec les règles données au dessus.

Chaque tache à un nom, un niveau de priorité, une durée et un moment où elle est crée (elle est crée quand l'opérateur appuie sur la touche entrée de son terminal…).

Listes des taches:

Nom de la tache	Priorité	durée	création
iA : init_A	fort	2	0
cA : corps_A	normal	3	0
dA : display_A	faible	1	0
iB : init_B	fort	2	3
cB : corps_B	normal	4	3
dB : display_B	faible	1	3
iC : init_C	fort	4	9
cC : corps_C	normal	1	9
dC : display_C	faible	1	9

Proposez un planning valide pour exécuter ces tâches à partir du quantum de temps numéro 0. Donner les dates (des numéros de quantums) de fin d'exécution des trois processus de type display_X. Comme suggestion de présentation, vous pouvez proposer une flèche du temps, qui commence au quantum zéro et qui indique, quantum par quantum, qui est exécuté.

Exercice 7 : Traçage d'exécution (4 points)

Prenons le programme suivant :

```c
#include <stdio.h>

int factorial(int n){
  if (n <= 1)
    return 1;
  return n*factorial(n-1);
}

int main(int argc, char* argv[]){
  printf("Factorielle(5) vaut : %d\n", factorial(5));
  return 0;
}
```

Si on regarde la pile d'appels (appels des fonctions avec leurs arguments), pour l'exécution de ce programme au dessus, on peut produire la trace suivante :

```
main
  factorial 5
    factorial 4
      factorial 2
        factorial 1
```

Il y a eu 5 appels de fonction et on remarque que le main a appelé factorial(5) qui a appelé factorial(4), etc… L'indentation donne une idée de qui appelle qui.

À partir de l'exemple donné au dessus, afficher la trace donnant l'évolution de la pile d'appel du programme suivant :

```c
#include <stdio.h>

int binomial(int n, int p){
  if ((n == 0) || (p == 0) || (n == p))
    return 1;
  return binomial(n-1,p) + binomial(n-1,p-1);
}

int main(int argc, char* argv[]){
  printf("Binomial de 5, 3 vaut : %d\n", binomial(5, 3));
  return 0;
}
```

Cheat Sheet : L'examen avec feuille de pompe intégrée

La table de 16 :

×	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
16	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	256

(printf("%x", i), c'est pour afficher une donnée entière i en hexadécimal avec des lettres en minuscules)

(printf("%X", i), c'est pour afficher une donnée entière i en hexadécimal avec des lettres en majuscules)

chmod droit fichier : changer les droits d'un fichier

chown user fichier : donner la propriété du fichier à l'utilisateur spécifié

mv source cible : déplacer ou renomer

mkdir nom : créer un nouveau répertoire

rmdir cible : effacer un répertoire vide

rm cible(s) : effacer des fichiers et des répertoires

grep -e motif fichier : recherche les lignes du fichier en argument contenant le motif en argument

grep motif : recherche les lignes de l'entrée standard contenant le motif en argument

find rep *conditions* : recherche dans rep et ses sous répertoires les éléments (fichier ou répertoire) vérifiant les *conditions*

cut -d *délimiteur* -f *champs* : Pour couper un fichier organisé par lignes selon des champs délimités

sort : trie les lignes d'un fichier ou de l'entrée standard

uniq : supprime les lignes doublons consécutives d'un fichier ou de l'entrée standard

Définition de ou exclusif : $XOR(a, b) := (a \ et \ \overline{b}) \ ou \ (\overline{a} \ et \ b)$