L2 informatique - Université Gustave Eiffel

Architecture des systèmes informatiques, examen première session 2022-2023

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 : Questions relatives aux cours (4 points)

1) Citez des exemples de primitives dans une UAL de processeur ?
2) Quelles informations peut-on trouver dans l'inode d'un fichier ?
3) Dans quel organe d'un ordinateur se trouve la mémoire de plus grande capacité ? quelles sont les caractéristiques de cette mémoire ?
4) Comment fonctionnent les priorités des processus sous Unix ?

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 l'auteur. Soyez précis, les intentions ce n'est pas relire et décortiquer la commande, c'est deviner et expliquer les besoins finaux de l'utilisateur.

1) `chown root ???.cfg`
2) `ls -l | grep -e "-..x..x..x" | wc -l`
3) `find . -name "*.sh" | xargs chmod o-w`
4) `ls -R ~/Images/ | grep -e ".png$"| wc -l`

Exercice 3 : Changements de base (4 points)

Un programmeur de niveau 1 n'a pas réussi à ouvrir en mode texte la sortie standard de son processus. Le programme s'est bien éxécuté mais la seule chose qu'il a pu récupérer est un flux binaire décrit par la séquence hexadécimale suivante.

49 6E 63 72 6F 79 61 62 6C 65 21

Qu'essayait donc d'écrire le programme éxécuté ?

hint : le nom de l'exercice est Changements de base. Il y a des choses dans la feuille de pompe intégrée de l'épreuve. Encore et toujours, tout résultat non justifié n'octroye aucun point.

Exercice 4 : Maximum sur trois valeurs (4 points)

Le but de cet exercice est de faire tourner un programme dans un simulateur de processeur à 5 registres (type little thinker ou mini-brain). Voici un rappel de toutes les commandes de ce CPU virtuel.

```
Cheat sheet CPU :
=================

-147 : interpreted directly as a constant value
$12  : value at address 12 inside the central memory
a    : the accumulator
#3   : value stored at register 3
$#2  : value in central memory at address indexed by value of register 2
#$47 : value stored at register indexed by value in central memory at address 47

* ld [$? or value] [#?-a]: load value inside the CPU
* st [#?-a or value] [$?]: store value to central memory
* mv [#?-a] [#?-a]: internal fast copy inside the CPU

* inc [#?]: increment targeted register (up the flag if zero)
* dec [#?]: decrement targeted register (up the flag if zero)
* add [#?-a or value or $?]: add targeted value to the accumulator (up the flag if zero)
* sub [#?-a or value or $?]: subtract targeted value to the accumulator (up the flag if zero)
* mul [#?-a or value or $?]: multiply by targeted value the accumulator (up the flag if zero)
* div [#?-a or value or $?]: divide the accumulator by targeted value
                             (up the flag if zero if exact division)
* mod [#?-a or value or $?]: replace the accumulator by its remainder when divide by targeted value
                             (up the flag if zero)
* cmp [#?-a or value or $?]: do nothing (up the flag if the accumulator is smaller than the argument)

* bfup [#?-a or value or $?]: jump to instruction indexed by the value if the flag is up
* bfdn [#?-a or value or $?]: jump to instruction indexed by the value if the flag is down
* bnow [#?-a or value or $?]: jump in all cases to instruction indexed by the value

* exit [#?-a or value or $?]: stop the program with return code the targeted value
* - empty instruction - : produce the behavior of exit a
```

Pour chaque ligne, il y a le numéro d'instruction puis l'instruction ou la valeur du programme.

```
1  | #votre programme
2  | #votre programme
3  | #votre programme
...
19 |
20 | 34
21 | 546
22 | 21
23 |
...
```

Écrire un programme qui recherche la plus grande des trois valeurs entre les trois cases mémoires 20, 21 et 22. Cette valeur devra alors être ré-écrite sur la ligne 23. Les trois valeurs au dessus sont données à titre d'exemple. Votre programme devra être fonctionnel quelque soient ces valeurs. Installez votre programme à partir de la ligne 1. Les lignes de 20 à 23 sont réservées pour les données, les 19 premières lignes sont largement suffisantes pour un tel programme.

Exercice 5 : Damier binaire (4 points)

Voici une table de vérité associée à une condition logique portant sur quatres variables A, B, C et D. Cette table de vérité, affichée sous la forme d'un tableau de karnaugh ressemble étrangemant à un damier.

CD\AB	00	01	11	10
00	0	**1**	0	**1**
01	**1**	0	**1**	0
11	0	**1**	0	**1**
10	**1**	0	**1**	0

1) Trouver une expression logique algébrique (avec possiblement des `not`, `and`, `or`, `nand`, `nor`, `xor`), la plus simple possible, pour cette table de vérité portant sur quatres variables A, B C et D.
2) Proposer un circuit logique simple réalisant l'opération logique associée à cette table de vérité.

Exercice 6 : Statistiques sur Premier Langage Platon (6 points)

Voici un apperçu de la table principal derrière les éxécutions dans

USER	PL	GRADE	SEED	DATE	ACTIVITY
[redacted]	64695 : doctest_hello	0	87	13 May 2023 09:49	1834 doctest_me_bro
[redacted]	64695 : doctest_hello	-	75	13 May 2023 09:49	-
[redacted]	64695 : doctest_hello	0	42	13 May 2023 09:49	1834 doctest_me_bro
[redacted]	64917 : diagram_to_Karnaugh	-	14	13 May 2023 09:48	-
[redacted]	64916 : diagram_to_Karnaugh	100	11	13 May 2023 09:48	1849 logical_circuit
[redacted]	64695 : doctest_hello	-	96	13 May 2023 09:48	-
[redacted]	64652 : meta_caractere	100	29	13 May 2023 09:47	1829 terminal_meta_regex
[redacted]	64652 : meta_caractere	-	82	13 May 2023 09:46	-
[redacted]	64651 : meta_caractere	100	80	13 May 2023 09:46	1829 terminal_meta_regex
[redacted]	64651 : meta_caractere	-	47	13 May 2023 09:45	-

Il s'agit d'un affichage d'une table de base de données relationnelles. Cette même table peut être exportée au format CSV donnant un fichier `log.csv` structuré par ligne comme il suit :

```
[redacted]rie, 64695, doctest_hello, 100, 87, 1684005989, 1834, doctest_me_bro
nborie, 64695, doctest_hello, 50, 67, 1684005945, 1834, doctest_me_bro
nborie, 64695, doctest_hello, , 75, 1684005906, ,
nborie, 64695, doctest_hello, 0, 42, 1684005842, 1834, doctest_me_bro
drevuz, 64917, diagram_to_Karnaugh, 100, 11, 1684005811, 1849, logical_circuit
drevuz, 64917, diagram_to_Karnaugh, , 96, 1684005645, ,
```

Chaque ligne est ainsi structurée de la manière suivante :

```
login, exo_id, exo_name, note, graine_aléa, timestamp, tp_id, tp_name
```

1) Proposez une commande qui affiche le nombre de fois que l'utilisateur `nborie` a obtenu la note maximale de 100 sur 100.
2) Proposez une commande qui affiche la liste des exercices, sans répétitions, pour lesquels l'utilisateur `drevuz` a obtenu la note de 0 sur 100.
3) Proposez une commande comptant le nombre d'utilisateurs ayant travaillé sur l'exo dont l'id est `64695`.
4) Proposez une commande donnant le nom de l'exerice réussi avec la note maximale de 100 le plus souvent sur Platon.
5) Proposez une commande donnant le nom de l'exercice ayant été pratiqué par le plus d'utilisateurs différents.
6) Proposez une commande qui affiche, de manière ordonnée et pour chaque utilisateur, le nombre de tp différents sur lesquels chaque utilisateur à travaillé.

Cheat Sheet : L'examen avec feuille de pompe intégrée

chmod droit fichier : changer les droits d'un fichier

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

ls : liste le contenu du répertoire courant (-l pour les détails dont les droits, -a pour les fichiers commençant par point)

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 : rechercher les lignes du fichier en argument contenant le motif en argument

grep (-e) motif : rechercher les lignes de l'entrée standard contenant le motif en argument
-v : option pour inverser la mise en correspondance, pour sélectionner les lignes ne correspondant pas au motif.

find rep *conditions* : rechercher dans rep et ses sous répertoires les éléments (fichiers ou répertoires) vérifiant les *conditions*
(-type pour préciser un type de fichiers, -name pour imposer un nom de fichier)

cut -d *délimiteur* -f *champs* : couper un fichier organisé par lignes selon des champs délimités.
Les champs commencent au premier 1 et pas zéro 0.

sort : trier les lignes d'un fichier ou de l'entrée standard
-n : option pour que le trie soit numérique

uniq : supprimer les lignes doublons consécutives d'un fichier ou de l'entrée standard
-c : option pour préfixer les lignes par le nombre d'occurrences

sed -e "s/REGEX_avant/MOTIF_apres/g" : remplacer partout toutes les occurences de `REGEX_avant` par `MOTIF_apres`

head -n : affiche les n premières ligne du fichier en argument ou bien de l'entrée standard.

tail -n : affiche les n dernières ligne du fichier en argument ou bien de l'entrée standard.

| **xargs** : utiliser la sortie standard de la commande précédante non pas comme entrée standard mais comme liste d'arguments pour la commande suivante

`Code ASCII (man 7 ascii)`

```
  30 40 50 60 70 80 90 100 110 120
  --------------------------------
0:    (  2  <  F  P  Z  d   n   x
1:    )  3  =  G  Q  [  e   o   y
2:    *  4  >  H  R  \  f   p   z
3: !  +  5  ?  I  S  ]  g   q   {
4: "  ,  6  @  J  T  ^  h   r   |
5: #  -  7  A  K  U  _  i   s   }
6: $  .  8  B  L  V  `  j   t   ~
7: %  /  9  C  M  W  a  k   u   DEL
8: &  0  :  D  N  X  b  l   v
9: '  1  ;  E  O  Y  c  m   w
```