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

Architecture des systèmes informatiques : Examen seconde session 2019-2020

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 10 minutes... (4 points)

1) Avec les processeurs actuels, de plus en plus multi-coeurs, les ordonnanceurs de taches sont-ils devenus plus ou moins importants qu'avant ?
2) Pourquoi les CPU modernes gèrent-ils un aussi grand nombre d'interruptions ? À quoi servent ces interruptions ?
3) Dans le système de fichiers Unix, peut-on créer une infinité de fichiers vides (de taille zéro octet) ? Comment ?
4) Le cours (et plusieurs TP) montrent que les pipes font que les programmes Unix enchaînés sont exécutés en parallèle. Pourquoi les experts en algorithmique disent-ils que la commande `sort` casse le parallélisme ?

Note : Seules les justifications sont notées. Ainsi le fait de réponse oui, non, vrai ou faux rapporte systématiquement zéro point à la question. Une bonne justification avec une réponse erronée peut même rapporter quelques points.

Exercice 2 : Lire dans les pensées (4 points)

Pour chacune des commandes qui suivent, tentez de déterminer les intentions précises de son auteur. Il s'agit ici de deviner quelles informations ou actions l'auteur de la commande souhaite obtenir. Il est inutile de décrire et de décomposer la commande, il faut en deviner la finalité.

1) `ps -ef | grep "root" | wc -l`
2) `find / -name "stdio.h" 2> /dev/null`
3) `ls -al | grep "-rwxrwxrwx"`
4) `find -name ".py" | xargs chmod +x`

Exercice 3 : changements de base (4 points)

Traduire les chiffres suivants de la base 10 vers la base 8 :

83, 63, 219, 650

Les calculs étant plus importants que les résultats. Veuillez faire apparaître toutes les étapes de calcul vous permettant d'opérer le changement de base.

Exercice 4 : Statistiques sur logiciel de gestion de versions (8 points)

Voici un exemple, vie réelle, d'extraction d'informations à coup de longues commandes pipées avec Unix. Les enseignants de l'UPEM et de l'université Paris Scalay conçoivent des exercices sous la plateforme PLaTon. Pour cela, tous les enseignants éditent des fichiers code-source d'exercices sur un même dépôt de gestions de versions. C'est comme si tous les enseignants partageaient un même disque dur virtuel dans le cloud.

L'activité d'édition peut être visualisée sur le fichiers log `Ygg_log` du gestionnaire de version. En voici un extrait du fichier `Ygg_log` (le vrai fichier cumule 18543 lignes…) :

```
1547753407|plgitlogin|A|/readme.md
1547757065|Dominique Revuz|M|/readme.md
1547851259|Dominique Revuz|A|/default/default.pl
1547851339|Dominique Revuz|A|/template/dd.pl
1547851391|Dominique Revuz|D|/template/dd.pl
1547851391|Dominique Revuz|A|/template/default.pl
1548077198|Coumes Quentin|M|/template/default.pl
1549021981|Mamadou Cisse|M|/Chemistry/alcen.pl
1549462018|Dominique Revuz|A|/utils/basic.py
1549532556|Dominique Revuz|M|/Language/conjugaison.pl
1550149544|David DOYEN|A|/MathLive/MathLive.pl
1550149544|David DOYEN|A|/MathLive/form_MathLive.html
1550149860|David DOYEN|M|/MathLive/MathLive.pl
1550510492|Hugo Mlodecki|A|/hugos/builder/build.py
1550510492|Hugo Mlodecki|A|/hugos/template/code.pl
1550510492|Hugo Mlodecki|A|/hugos/template/graderCpp.py
...
...
```

Chaque ligne se découpe ainsi :
`temps en secondes depuis l'epoch|user|action|fichier`

`temps en secondes depuis l'epoch` : nombre de secondes écoulées depuis le premier janvier 1970 à 0h00m00s
`user` : login de l'enseignant éditeur sous PLaTon
`action` : une majuscule (A : ajout d'un fichier, M : modification du fichier, D : délétion du fichier)
`fichier` : Le chemin absolu du fichier dans ce disque dur virtuel partagé

1) Proposez une commande Unix qui compte le nombre d'effacements de fichiers dans le dépôt.
2) Proposez une commande Unix qui détermine combien de fichiers ont été ajoutés dans le dépôt par l'utilisateur `Dominique Revuz`.
3) Chaque ligne étant une action d'édition, proposez une commande Unix qui affiche de manière trié (du plus actif au moins actif), pour chaque utilisateur, le nombre d'actions d'édition effectuées.
4) Proposez une commande Unix qui compte le nombre total de fichiers sur lesquels l'utilisateur `David DOYEN` a travaillé. *Attention, un utilisateur peut par exemple créer et modifier plusieurs fois un même fichier.*
5) Proposez une commande Unix qui affiche les 5 utilisateurs ayant le plus créé de fichier (c'est l'action `A`).
6) Bonus : proposer une commande Unix qui affiche tous les éditeurs de ressources, de celui qui a travaillé sur le plus de fichiers différents à celui qui a travaillé sur le moins de fichier différents. Votre commande pourra laisser les nombres de fichiers en préfixe de lignes. Ainsi, elle devra à peu près afficher une des deux sorties suivante.

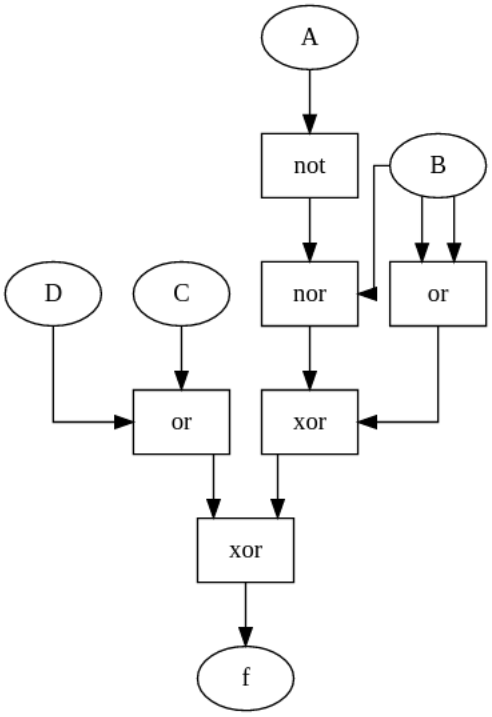
```
1103  Dominique Revuz
 613  David DOYEN
 280  Nicolas Borie
etc ...
```

ou bien

```
Dominique Revuz
David DOYEN
Nicolas Borie
etc ...
```

Exercice 5 : Circuit (4 points)

Le circuit logique suivant prend 4 bits en argument dans ses entrées A, B, C et D. Ce circuit possède une unique binaire sortie f.



Proposez une table de vérité pour ce circuit logique qui décrit donc la sortie f en fonctions des quatre entrées A, B, C et D du circuit.

Feuille de pompe intégrée :

ps -ef : affiche tous les processus actifs sur la machine avec leurs informations détaillés, un par ligne

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 : découpe et sélectionne des données structurées par ligne.
(le premier champ est le champ 1, il n'existe pas de champ 0.)

- option -d pour préciser le caractère de séparation.
- option -f pour sélectionner un ou plusieurs champs.
  (-f 2,3 garde les champs 2 et 3 séparés par l'ancien caractère délimiteur)

sort : trie les lignes de manière croissante dans l'ordre lexico-ascii.

- option -r pour renverser l'ordre du tri (tri décroissant).
- option -n pour utiliser un ordre numérique.
- option -u pour supprimer les doublons en triant.

uniq : supprime les lignes répétées.

- option -c préfixe les lignes par leur nombres d'occurrences

wc : comptage textuel sur le flux et/ou fichier(s).

- option -l pour compter seulement le nombre de lignes.
- option -w pour compter seulement le nombre de mots.
- option -c pour compter seulement le nombre de caractères.

sed `-e "s/OLD_reg_exp/NEW_reg_exp/g"` : remplace toutes les occurrences de l'ancienne expression régulière par la nouvelle.

head : permet de visualiser les premières lignes d'un flux et/ou fichier(s).

- option -NOMBRE pour afficher seulement les NOMBRE premières lignes
  (`head -12 plop` affiche les 12 premières lignes du fichier plop)

symbole	opérateur logique
and	ET
or	OU
xor	OU EXCLUSIF
not	NON
nand	NON ET
nor	NON OU