INF203 - Partiel
Année 2022-2023

Durée : 1h30.
Une feuille A4 manuscrite recto/verso autorisée, dictionnaire papier (non annoté) autorisé pour les étudiants étrangers uniquement.
Tout autre document, calculatrices et appareils électroniques interdits.
Pour chaque question, une partie des points peut tenir compte de la présentation.
Le barème est indicatif.

Toute réponse même incomplète sera valorisée à partir du moment où elle réalise au moins une partie de ce qui est demandé. Les questions sont relativement indépendantes, et dans tous les cas vous pouvez utiliser les scripts et les fonctions demandées dans les questions précédentes même si vous n'avez pas réussi à les écrire.

Par ailleurs, tout code ou algorithme *élégant*, en plus d'être correct, sera bonifié.

Préliminaire

1. (1 point) Par quel moyen pouvez-vous, depuis un interpréteur Bash, obtenir des informations sur une commande donnée ?

   Une fois que vous avez répondu à cette question, vous avez le droit d'utiliser l'annexe en dernière page de ce sujet.

1 Langage C

Calculer la moyenne olympique d'une suite de notes $\{n_1, ..., n_k\}$ consiste à faire la moyenne de cette suite sans tenir compte des deux valeurs les plus éloignées.

Par exemple, la moyenne olympique de $\{3, 5.5, 9, 9, 9, 7.5, 8, 6, 3, 8\}$ est égale à 7 :

— on ne tient pas compte des deux valeurs extrêmes : 3 (une seule fois) et 9 (une seule fois) ;

— on calcule alors $(5.5 + 9 + 9 + 7.5 + 8 + 6 + 3 + 8)/8 = 7$

On suppose ici que `k` notes ($\texttt{k} \geq 3$) sont stockées dans un tableau de `double` appelé `notes`.

Question :

1. (3 points) Donnez le code en langage C de la fonction :

   `double Olympe(double notes[], int k)`

   qui calcule la moyenne olympique des `k` notes passées en paramètre.

   On ne demande **pas** d'écrire de programme principal.

Solution:

```
#include<stdio.h>

double Olympe(double notes[], int n){
  double min,max,somme;
  int i;
  somme=notes[0];
  min=somme;
  max=somme;
  for(i=1;i<n;i++){
    somme+=notes[i];
    if(notes[i]<min) min=notes[i];
    if(notes[i]>max) max=notes[i];
  }
  return (somme-min-max)/(n-2);
}
```

```
int main(){
  double notes[]={3, 5.5, 9, 9, 9, 7.5, 8, 6, 3, 8};
  printf("%f\n",Olympe(notes,10));
}
```

2 Envoi automatique de mails

Un intervenant de TD/TP souhaite envoyer un mail à ses étudiants afin de les prévenir d'un changement de salle.

Il récupère pour cela le fichier `groupe.csv` sur le site `https://trombi.univ-grenoble-alpes.fr/`.

Ce fichier `groupe.csv` est formatté de la façon suivante :

```
DLST - GRENOBLE - MIN-S2-Groupe 1;

mail;prenom;nom;

Alain.Connu@etu.univ-grenoble-alpes.fr;Alain;Connu;
Anne.Onime@etu.univ-grenoble-alpes.fr;Anne;Onime;
...
```

La suite du fichier suit exactement la même structure, avec un grand nombre d'étudiants.

Questions :

1. (2 points) Considérons la ligne suivante, que nous mettons dans la variable "ligne" :

   ```
   ligne="Alain.Connu@etu.univ-grenoble-alpes.fr;Alain;Connu;"
   ```

   Donnez une ligne de commande qui n'affiche que l'email de l'étudiant à partir de la variable `ligne`.

   Solution:

   ```
   echo $ligne | cut -d ";" -f 1
   ```

2. (1 point) Donnez une ligne de commande qui extrait les lignes qui contiennent un email du fichier `groupe.csv`.

   On s'attend à la sortie suivante :

   ```
   Alain.Connu@etu.univ-grenoble-alpes.fr;Alain;Connu;
   Anne.Onime@etu.univ-grenoble-alpes.fr;Anne;Onime;
   ...
   ```

   Solution:

   ```
   cat groupe.csv | grep etu.univ-grenoble-alpes.fr
   ```

3. (2 points) Écrivez un script Bash qui parcourt le fichier `groupe.csv` et affiche *uniquement les emails* de tous les étudiants répertoriés dans ce fichier.

   On s'attend à la sortie suivante :

   ```
   Alain.Connu@etu.univ-grenoble-alpes.fr
   Anne.Onime@etu.univ-grenoble-alpes.fr
   ...
   ```

   Solution:

   ```
   #!/bin/sh

   mail_lines=$(cat groupe.csv|grep etu.univ-grenoble-alpes.fr)

   for i in $mail_lines
   do
   echo $i | cut -d ";" -f 1
   done
   ```

4. (2 points) Pour envoyer un email à ses étudiants, l'intervenant décide d'utiliser la commande `mail`. Cette commande `mail` s'utilise avec la syntaxe suivante :

```
mail -s "<Objet du Mail>" <destinataire>
```

Une fois cette commande exécutée, le contenu du mail à envoyer est lu depuis l'entrée standard. Par exemple, la commande suivante permet d'envoyer un email avec l'objet `"bonjour"` et le contenu `"vive INF203 !"` au destinataire `romain.couillet@univ-grenoble-alpes.fr` :

```
echo "vive INF203 !" | mail -s "bonjour" romain.couillet@univ-grenoble-alpes.fr
```

Modifiez votre script de la question précédente afin qu'il envoie un email avec pour contenu `"La nouvelle salle est b005"` et pour objet `"changement de salle"` à chacun des étudiants du fichier `groupe.csv`.

Solution:

```
#!/bin/sh

mail_lines=$(cat groupe.csv|grep etu.univ-grenoble-alpes.fr)

for i in $mail_lines
do
email=$(echo $i |cut -d ";" -f 1)
echo "La nouvelle salle est b005" | mail -s "changement de salle" $email
done
```

5. (2 points) Expliquez comment modifier à nouveau votre script de la question précédente afin que :
— le sujet de l'email soit la chaîne de caractères donnée comme premier argument du script ;
— le contenu de l'email soit lu dans un fichier dont le nom est donné en second argument.

Solution: On remplace la ligne `echo ...` par au choix

```
mail -s $1 $email < $2
```

ou

```
cat $2 | mail -s $1 $email
```

6. (1 point) Notre chargé de TD s'aperçoit que le fichier `groupe.csv` est incomplet : sur certaines lignes, on dispose bien du prénom et du nom de l'étudiant, mais l'email n'est pas indiqué.
Écrivez un script Bash qui parcourt le fichier `groupe.csv` et affiche uniquement les lignes de la forme :

`;Anne;Onime;`

dans lesquelles l'email de l'étudiant est manquant.

Solution:

```
cat groupe.csv | tail -n +5 | grep -v etu.univ-grenoble-alpes.fr
```

7. (2 points) Modifiez le script précédent pour que, pour chaque étudiant dont l'email est manquant, il affiche l'adresse email de l'étudiant, construite sous la forme `prenom.nom@etu.univ-grenoble-alpes.fr`.

Solution:

```
nomail_lines=$(cat groupe.csv | tail -n +5 | grep -v etu.univ-grenoble-alpes.fr)

for i in $nomail_lines
do
   prenom=$(echo $i |cut -d ";" -f 2)
   nom=$(echo $i |cut -d ";" -f 3)
   echo $prenom.$nom@etu.univ-grenoble-alpes.fr
done
```

3 Stocker ses photos sur CDROM

Pour une raison qui vous échappe totalement, vos parents ont décidé de stocker toutes les photos de leur disque dur sur un ensemble de ... CDROMs. Bien sûr, ils ont demandé votre aide. L'objectif de l'exercice est d'automatiser cette tâche de copie. Ces photos sont des fichiers d'extension `.jpeg` qui se trouvent dans des dossiers nommés `PhotosXXXX` (par exemple `Photos1972` à `Photos2023`).

On ne demandera pas de conserver ces dossiers, juste de copier les photos (dont on suppose les noms différents). Par contre, la difficulté réside dans le fait que la taille cumulée des fichiers demande un stockage sur plusieurs CDROMs.

Questions :

1. (2 points) Donnez un ensemble d'instructions en SHELL permettant d'évaluer la taille totale, en *octets*, de l'ensemble des photos et qui sera stockée dans la variable `TAILLE`.

Solution:

```
TAILLE=0

for F in Photos*/*.jpeg
do
    TAILLEFICH=$(wc -c < $F)
    TAILLE=$(expr $TAILLE + $TAILLEFICH)
done
```

2. (2 points) Pour évaluer le nombre de CDROMs nécessaires, on établit un seuil `TAILLEMAX=650000000` correspondant à la taille d'un CDROM en octets. Donnez un ensemble d'instructions permettant d'évaluer la variable `N`, initialisée à `N=1`, qui correspond au nombre de CDROMs nécessaires à la copie. La valeur de `N` sera incrémentée à chaque fois qu'est rencontré un fichier dont la copie ne serait plus permise sur le CDROM précédent.

   Par ailleurs, le script devra retourner

   `Le CD numéro X est complet.`

   où `X` vaudra successivement 1, 2, etc., à chaque fois que la taille d'un CDROM est atteinte, ainsi que la mention finale

   `Y CDs utilisés.`

   où `Y` est le nombre total de CDROMs nécessaires.

   Indication. Attention de bien gérer le premier fichier dont la taille engendrera un dépassement de la taille de chaque CDROM.

Solution:

```
TAILLE=0
TAILLEMAX=650000000
N=1

for F in Photos*/*.jpeg
do
    TAILLEFICH=$(wc -c < $F)
    TAILLE=$(expr $TAILLE + $TAILLEFICH)
    if [ $TAILLE -gt $TAILLEMAX ]
    then
        echo "Le CD numéro $N est complet"
        N=$(expr $N + 1)
        TAILLE=$TAILLEFICH
    fi
done
echo $N CDs utilisés
```

3. (2 points) En vous inspirant des deux premières questions, écrire alors un script qui crée à la volée des dossiers `CDROMS/CD1`, `CDROMS/CD2`, etc., dans lesquels on copiera successivement les fichiers d'extension `.jpeg` présents dans les dossiers `Photos1972` à `Photos2023`.

   Pour garder une visibilité sur les erreurs de copies éventuelles (problèmes de noms ou de lecture de certains vieux fichiers), on redirigera la sortie `stderr` de l'instruction de copie de fichier vers un fichier local `erreur.log`. Par ailleurs, afin de suivre le bon déroulé de la copie, tout affichage dans la sortie standard est le bienvenu !

   Indication. Attention à nouveau à bien gérer le premier fichier qui dépasserait la taille maximale de chaque CDROM. Attention également à bien écrire les lignes d'erreurs *successivement* dans le fichier `erreur.log` (et non pas à l'écraser à chaque nouvelle erreur).

Solution:

```
TAILLE=0
TAILLEMAX=650000000
N=1
mkdir CDROMS/CD1

for F in Photos*/*.jpeg
do
    TAILLEFICH=$(wc -c < $F)
    TAILLE=$(expr $TAILLE + $TAILLEFICH)
    if [ $TAILLE -ge $TAILLEMAX ]
    then
        N=$(expr $N + 1)
        mkdir CDROMS/CD$N
        TAILLE=$TAILLEFICH
    fi
    cp $F CDROMS/CD$N 2>> erreur.log
done
```

A Annexe : rappels

On rappelle ici différentes commandes et options qui peuvent être utiles.

cut affiche une partie seulement de chaque ligne de ses entrées.

`-c debut-fin` permet de choisir les caractères affichés ;

`-f debut-fin` permet de choisir des colonnes ;

`-d` permet de choisir le délimiteur.

diff détermine si les contenus de 2 fichiers sont identiques :

— aucune réponse : les 2 fichiers sont identiques.

— la réponse est seulement que les 2 fichiers sont différents : l'un au moins n'est pas un fichier texte.

— la liste des différences entre les 2 fichiers : les 2 fichiers sont des fichiers texte.

expr calcule l'expression décrite par ses arguments.

grep n'affiche que les lignes d'un fichier contenant une chaîne de caractères donnée.

`-c` n'affiche que le nombre total de lignes contenant la chaîne recherchée.

`-v` n'affiche que les lignes ne contenant pas la chaîne donnée.

head et tail

`head` affiche le début de ses entrées, selon l'option choisie :

`-n N :` les `N` premières lignes

`-n -N :` toutes les lignes sauf les `N` dernières

De même la commande `tail` affiche la fin de ses entrées :

`-n N :` les `N` dernières lignes

`-n +N :` les dernières lignes à partir de la N^e

sed affiche les lignes de ses entrées en les filtrant et/ou modifiant selon les instructions données :

`sed /Candide/d` affiche celles ne contenant pas `Candide`

`sed s/Candide/Toto/` remplace le premier `Candide` par `Toto` sur chaque ligne

`sed s/Candide/Toto/g` remplace tout les `Candide` par `Toto`

sort affiche les lignes de ses entrées triées par ordre croissant

tr copie l'entrée standard sur la sortie standard en substituant

`tr chaîne1 chaîne2 :` chaque caractère présent dans `chaîne1` par le caractère de position correspondante dans `chaîne2`

`tr -s chaîne :` les répétitions des caractères présents dans `chaîne` par une occurrence unique

wc compter les caractères, les mots ou les lignes lus sur son entrée standard :

`-c` compte les caractères

`-w` compte les mots

`-l` compte les lignes