**Université Paul Sabatier** **UE : 1L3INA1**
**L2 Informatique** **Systèmes 1**

**Contrôle terminal**
**Vendredi 11 janvier 2008**
**2h00 - sans document**

---

L'épreuve comporte quatre exercices indépendants. L'écriture des scripts en Shell de Bourne sous-entend l'écriture de vérifications appropriées sur les paramètres, ainsi qu'une gestion cohérente des codes de retour. Il est conseillé de commencer par lire l'ensemble de l'énoncé et de tenir compte du barème. La propreté de la copie sera prise en compte, en particulier en ce qui concerne la lisibilité des caractères ' et \.

---

## Exercice 1 (3 points)

Réaliser chacune des tâches suivantes à l'aide d'une seule ligne de commande :

1. Afficher page par page les informations détaillées concernant les répertoires, cachés ou non, du répertoire courant.
2. Écrire dans le fichier `prem.txt` le premier mot de la première ligne du texte contenu dans le fichier `fich.txt`. On suppose que le texte est constitué de mots séparés par des espaces ou des retours-chariot sans ponctuation. Si le fichier `prem.txt` existe déjà, son contenu ne doit pas être écrasé et le mot doit être ajouté à la fin du fichier.

## Exercice 2 (4 points)

Écrire le script `debut.sh` dont la syntaxe d'appel est la suivante :

```
debut.sh expr_reg nom_fichier
```

qui affiche sur `stdout` le contenu du fichier dont le nom est passé en second paramètre jusqu'à la première ligne correspondant à l'expression régulière passée en premier paramètre. Par exemple, si le fichier `fich1.txt` contient :

```
quelque chose de joli
quelque chose de simple
quelque chose de beau
quelque chose d'utile
pour l'oiseau
```

alors la commande suivante :

```
$ debut.sh 'e$' fich1.txt
```

doit afficher à l'écran les deux lignes suivantes :

```
quelque chose de joli
quelque chose de simple
```

## Exercice 3 (8,5 points)

1. Écrire, *sans* utiliser la commande `sed`, le script `remplacer.sh` dont la syntaxe d'appel est la suivante :
   `remplacer.sh ligne_avant ligne_après nom_fichier`
   qui affiche sur `stdout` le contenu du fichier dont le nom est passé en troisième paramètre dans lequel toutes les lignes égales à la chaîne de caractères passée en premier paramètre sont remplacées par la chaîne de caractères passée en deuxième paramètre. Les autres lignes doivent être laissées intactes. Par exemple, si le fichier `fich2.txt` contient :
   ```
   il s'est levé
   je vois
   son chapeau sur sa tête
   je vois
   son manteau de pluie
   ```
   alors la commande suivante :
   ```
   $ remplacer.sh "je vois" "il a mis" fich2.txt
   ```

doit afficher à l'écran les cinq lignes suivantes :

```
il s'est levé
il a mis
son chapeau sur sa tête
il a mis
son manteau de pluie
```

Il est conseillé pour cela de parcourir le fichier ligne par ligne. Pour chaque ligne du fichier, si elle est identique à la ligne recherchée, afficher la nouvelle ligne, sinon afficher la ligne du fichier.

2. Écrire, en utilisant le script précédent et *sans* utiliser la commande `sed`, le script `statmots.sh` dont la syntaxe d'appel est la suivante :

   ```
   statmots.sh nom_fichier
   ```

   qui, à partir d'un fichier contenant un texte, affiche sur `stdout` chaque mot du texte suivi du nombre de fois où il apparaît dans le texte. On suppose que le texte est constitué de mots en minuscules séparés par des espaces ou des retours-chariot sans ponctuation. Par exemple, si le fichier `fich3.txt` contient :

   ```
   ils sont à table
   ils ne mangent pas
   ils ne sont pas dans leur assiette
   ```

   alors la commande suivante :

   ```
   $ statmots.sh fich3.txt
   ```

   doit afficher à l'écran les lignes suivantes :

   ```
   ils 3
   sont 2
   à 1
   table 1
   ne 2
   mangent 1
   pas 2
   dans 1
   leur 1
   assiette 1
   ```

   Il est conseillé pour cela d'utiliser deux fichiers temporaires. C'est dans le premier fichier temporaire que le résultat va être construit petit à petit. Le second fichier temporaire ne sera utilisé que quand une ligne du premier fichier temporaire devra être mise à jour grâce à un appel au script `remplacer.sh`. Il est également conseillé de parcourir le fichier ligne par ligne et chaque ligne mot par mot. Pour chaque mot, il faut vérifier sa présence dans le premier fichier temporaire. S'il est déjà présent, il faut mettre à jour son nombre d'apparitions. S'il n'est pas présent, alors il faut rajouter une nouvelle ligne à la fin du fichier. Après avoir affiché le résultat, les deux fichiers temporaires doivent être effacés.

## Exercice 4 (4,5 points)

Écrire le script `rechercher.sh` dont la syntaxe d'appel est la suivante :

```
rechercher.sh expr_reg nom_répertoire
```

qui recherche le nombre de lignes qui correspondent à l'expression régulière passée en premier paramètre dans les fichiers contenus dans le répertoire dont le nom est passé en second paramètre, quel que soit le niveau des fichiers dans l'arborescence. Pour chaque fichier de la sous-arborescence examinée, s'il contient au moins une ligne qui correspond, alors la désignation du fichier et le nombre de lignes qui correspondent doivent être affichées sur `stdout`. Si le fichier n'est pas accessible, un message doit être affiché sur `stderr`, mais l'exécution du script ne doit pas être interrompue. Par exemple, si le répertoire `REP` contient un fichier `f1` avec 12 lignes commençant par `if` et un répertoire `SREP` contenant lui-même un fichier `f3` avec 3 lignes commençant par `if` et si aucun autre fichier ne contient de ligne commençant par `if`, alors la commande suivante :

```
$ rechercher.sh '^if' REP
```

doit afficher à l'écran les deux lignes suivantes :

```
Le fichier REP/f2 contient 12 lignes qui correspondent.
Le fichier REP/SREP/f3 contient 3 lignes qui correspondent.
```

# RAPPELS

| | |
|---|---|
| **cat** | `cat [fichier]` affiche sur `stdout` le contenu du fichier dont le nom est passé en paramètre (ou de `stdin` par défaut). |
| **cut** | `cut -f liste -d délimiteur` ne retient sur chaque ligne de `stdin` que les champs spécifiés par `liste` séparés par `délimiteur`. Le numéro du premier champ sur la ligne est 1. |
| **echo** | `echo $var` affiche la valeur de la variable `var` suivie d'un caractère `'\n'`.<br>`echo "$var\c"` affiche la valeur de la variable `var`. |
| **expr** | La commande `expr` permet de réaliser des opérations entières grâce aux opérateurs `+`, `-`, `\*`, `/`, `%`. |
| **grep** | `grep expression [fichier ...]` affiche sur `stdout` les lignes des fichiers passés en paramètres (les lignes de `stdin` par défaut) qui correspondent au modèle décrit par l'expression régulière `expression`. Dans cette dernière, `^` désigne le début de la ligne, `$` la fin de la ligne, `.` un caractère quelconque, `*` une répétition (de longueur quelconque éventuellement vide) du caractère qui précède et `\` "protège" le caractère qui suit. Le code de retour de `grep` est nul si au moins une occurrence est trouvée.<br>Avec l'option `-n`, chaque ligne qui correspond à l'expression régulière est précédée de son numéro (la première ligne a le numéro 1), suivi du caractère `:`<br>Avec l'option `-c`, la commande grep affiche uniquement sur `stdout` le nombre de lignes qui correspondent à l'expression régulière. |
| **head** | Affichage sur `stdout` du début du fichier passé en paramètre (de `stdin` par défaut). `head -5 fich` produit l'affichage des cinq premières lignes de `fich` (si elles existent). |
| **ls** | La commande `ls REP` affiche sur `stdout` la liste des noms de fichiers et de répertoires contenus dans le répertoire `REP`.<br>La commande `ls -al REP` affiche sur `stdout` des renseignements détaillés concernant le contenu complet du répertoire `REP`. Sans le nom du répertoire, c'est le répertoire courant qui est considéré. Sans l'option `-a`, les fichiers cachés sont ignorés. |
| **more** | `more [fichier...]` permet d'afficher sur `stdout` page par page le contenu des fichiers dont les noms sont passés en paramètres ou de `stdin` s'il n'y a aucun paramètre. |
| **mv** | `mv ancien_nom nouveau_nom` permet de renommer le fichier `ancien_nom` en l'appelant `nouveau_nom`. |
| **read** | La commande `read a` permet d'affecter à la variable `a` une ligne provenant de `stdin`. |
| **rm** | `rm nom_fichier [...]` permet de supprimer un ou plusieurs fichiers passés en paramètres. |
| **test** | À la suite de la commande `test`, peuvent apparaître les opérateurs logiques `-a`, `-o`, `!`, les comparateurs d'entiers `-eq`, `-ne`, `-gt`, `-ge`, `-lt`, `-le`, les comparateurs de chaînes de caractères `=`, `!=`, les opérateurs sur un fichier ou un répertoire `-f`, `-d`, `-r`, `-w`, `-x`. |

*Remarque :* les informations données ci-dessus en guise de rappel n'impliquent pas que les commandes indiquées doivent obligatoirement apparaître dans les réponses aux exercices, ni que ce sont les seules qui doivent être utilisées.