TP N° 5 - Shell scripts, Codage, compression, archivage

EXERCICE 1 - On reprend les deux dernières questions du TP précédent, mais on réalise cette fois les opérations en écrivant des scripts Python

Rappel : on un répertoire nommé `REP` dans lequel on a créé 250 fichiers vides nommés "`fich-N.Z`" où `N` est un nombre entier variant de 0 à 249.

1- Créer un second répertoire vide, `REP2`, et y copier les 50 derniers fichiers de `REP` (numéros 200 à 249), en changeant leur le suffixe de `.Z` en `.txt` (on utilisera cette fois un prog. Python).

2- Copier les 10 premiers fichiers de REP (numéros 0 à 9) vers REP2, en ajoutant en même temps à leur contenu une ligne de la forme "fichier N".

EXERCICE 2 - Évaluation des expressions par le shell (csh)

1- Afficher `$HELLO` (avec le caractère $).

2- `"HELLO"` (avec les guillemets).

3- La commande `hostname` affiche le nom de votre machine. Dans votre `.cshrc`, créer une variable d'environnement `MACHINE` qui contienne le nom de votre machine.

4- Créer une variable `N` qui soit égale au nombre de fichiers et répertoires dans `/etc` (note : `ls -1` permet de forcer l'affichage de `ls` sur une colonne).

1 Codage des fichiers

Sous UNIX, le contenu d'un *fichier*, quel que soit son type, peut être considéré comme une suite d'octets prenant des valeurs quelconques.

1.1 Rappels sur les fichiers textes

On appelle *fichier texte* un fichier dans lequel les informations sont structurées en *lignes*, séparées par des marques de fin de ligne.

Le marquage des fins de lignes est d'ailleurs différents sur les systèmes UNIX et les systèmes DOS (et dérivés) : UNIX utilise un caractère de fin de ligne, `\n`, de code ascii 10, tandis que les lignes des fichiers texte DOS sont séparées par deux caractères `\r\n` (codes ascii 13 et 10). A l'origine, le caractère `\r` indiquait un retour chariot (souvenir des machines à écrire), et `\n` un passage à la ligne suivante.

C'est pourquoi l'on doit préciser le type du fichier (texte ou binaire) lors de son ouverture sous DOS : `open( nomfichier, "wb" )` (ou `fopen` en langage C) pour un fichier binaire (les données sont écrites sans transformation), et `fopen( nomfichier, "wt" )` pour un fichier texte (le système d'exploitation (DOS ou Windows) remplace les caractères `\n` par la séquence `\r\n`.

Cette distinction n'a pas lieu d'être sous UNIX. Toutefois, il est préférable d'indiquer dans vos programmes le type (`b` ou `t`) des fichiers ouverts en écriture, afin que le code soit portable.

1.2 Codage ASCII 7 bits

Normalement, un fichier texte ne contient que des caractères ASCII (codes entre 0 et 127, le bit 7 de chaque octet étant toujours zéro). En dehors des pays anglo-saxons, on utilise un code ASCII "étendu" qui permet de coder quelques accents (attention, il existe plusieurs codages incompatibles...).

Les premiers protocoles de transport de mail (courrier électronique) (par exemple uucp) étaient prévus pour transmettre des messages constitué de texte ASCII 7 bits. Certains de ces composants sont encore en service sur l'Internet (de plus en plus rarement). Lorsque l'on envoie en message électronique, il est donc recommandé de n'utiliser que des caractères ASCII 7 bits.

Que faire si l'on veut transmettre des données binaires (image, fichier de données, ...) ? On doit *coder* les données, en n'utilisant que des octets entre 0 et 127. Une suite de N octets ($8N$ bits) sera codée par une suite d'approximativement $8N/7$ octets ASCII.

Deux codages standard existent :

1. **uuencode** : standard sous UNIX (système "uucp", *unix-to-unix copy*), via les commandes `uuencode` et `uudecode`.
2. **base64** : utilisé d'abord surtout par les Apple Macintosh (utilitaire binhex), c'est maintenant le plus utilisé (en particulier par le format MIME pour les courriers électroniques).

Pour transmettre un fichier binaire, on le code en ASCII, le transmet, puis le récepteur le décode afin de retrouver les données originales.

2 Compression des données

Les techniques de *compression de données* visent à réduire le volume des données pour économiser de l'espace mémoire (disque) ou accélérer les transmissions réseau. Ces techniques sont basées sur l'exploitation des *redondances* : en général, le contenu d'un fichier n'est pas complètement aléatoire. Exemple extrême : un fichier constitué de 1024 octets à zéro occupe 1Ko, mais peut être codé par la phrase "1024 octets à 0", qui n'occupe elle que 15 octets !

On distingue deux types de techniques :
- *Méthodes à reconstruction exacte :* les données originales peuvent être reconstituées à l'identique à partir du code compressé. Ces méthodes sont utilisées pour les fichiers informatiques (textes, programmes) pour lesquels une erreur d'un seul bit serait intolérable. Elles ne permettent pas une très forte compression (en gros un facteur 3 à 5 sur du texte).
- *Méthodes à reconstruction inexacte :* utilisées lorsque l'on peut tolérer une légère déformation des données : image, son. Les taux de compression peuvent être très élevés, au prix d'une dégradation de l'information ; il faut trouver un compromis. Par exemple, la méthode JPEG de compression d'image ; plus le taux de compression utilisé est fort, plus l'image reconstruite apparait floue. Les standards de compression de fichier les plus connus sont :
  - `TIFF G4` pour les images noir et blancs (bitmaps), par exemple les télécopies (fax) (compression exacte).
  - `JPEG` pour les images couleur fixes (par exemple pour la photographie numérique) ;
  - `MPEG` pour les séquences d'images (flux vidéos) ;
  - `MP3` pour les sons (musique).

Notons enfin que les modems font de la compression/décompression (exacte) de données au vol.

Outils de compression (exacte) répandus :

1. Sous UNIX :
   - `compress`, `uncompress`, `zcat` : commandes standard UNIX ;
   - `gzip`, `gunzip` : GNU zip, versions gratuites s'utilisant exactement comme `compress` et `uncompress`, mais plus performantes (rapidité et taux de compression).
   - `bzip2`, `bunzip2` (Burrows-Wheeler + Huffman), encore plus performant (quoique souvant plus lent).
2. Sous DOS/Windows : plusieurs utilitaires, gartuits ou commerciaux, le plus utilisé étant `zip`. (Une version gratuite de `zip` existe sous UNIX).

3 Archivage

En informatique, une *archive* est un ensemble de fichiers réunis dans un seul fichier (l'archive), pour pouvoir plus facilement les sauvegarder (sur une bande magnétique par exemple) ou les transmettre (par e-mail par exemple). Le fichier archive contient en général une "table des matières" permettant de retrouver rapidement les fichiers inclus dans l'archive.

Les "archiveurs" sont des programmes permettant de créer et manipuler des archives. Les

plus fréquemment utilisés sont :

1. Sous UNIX : `tar` ;
2. Sous DOS : `zip`.

Lorsque l'on envoie un courrier électronique, on a souvent besoin de joindre des documents (textes, images, programmes...), réunis dans le même message. On doit donc créer une sorte d'archive, dont le format est défini par la norme MIME (que nous n'étudions pas en détail ici).

Chaque partie du message est appelée un "attachement". Les attachements sont mis les uns à la suite des autres dans le message, séparés par des lignes spéciales ("séparateurs").

Chaque attachement doit aussi être converti en ASCII 7 bits.

EXERCICE 3 - Codage 7 bits sous UNIX

On a écrit une commande `7bits` qui force le 7ième bit de chaque octet à zéro. La commande s'utilise comme un filtre de texte :

```
$ cat toto | 7bits > toto.7
```

`toto.7` est la version 7 bits de `toto`.
La documentation des commandes `uuencode` et `uudecode` est fournie en annexe.

1- Créer un fichier contenant un court texte avec des accents. Utiliser la commande `7bits` pour vérifier que les accents ne sont pas correctement transmis.

2- Utiliser `uuencode` pour coder un petit fichier binaire. Quelle est la ligne d'entête du fichier codée ? Comparer la taille du fichier binaire et du fichier codé.

3- Décoder le fichier codé, et vérifier que l'on retrouve les informations de départ.

4- La commande `7bits` simule la transmission sur un réseau.
Simuler en une seule ligne shell le codage, la transmission et le décodage d'un fichier, à l'aide de 2 tubes.
(Aide : si l'on ne précise pas de nom de fichier, les commandes `uuencode`/`uudecode` utilisent la sortie et l'entrée standard.)

EXERCICE 4 - Fichiers e-mail MIME

1- A l'aide de Netscape, composer un message électronique, lui attacher quelques petits fichiers binaires.
Indiquez a Netscape d'envoyer le message *plus tard*.
Note : Vous n'êtes pas connecté à Internet, il est impossible d'envoyer réellement le message. L'option d'envoi différé permet de composer ses messages lorsque l'on n'est pas connecté, pour qu'ils partent lorsque la connexion est étable (par exemple via un modem). Dans ce cas, netscape sauve le message dans un fichier (dossier `Unsent Messages` sous UNIX).

2- Retrouver le fichier généré par Netscape pour votre message (faire une recherche en utilisant la date de création du fichier).

3- Lire ce fichier dans un éditeur. Examiner l'en-tête.

1. Comment sont séparés les fichiers attachés ?
2. Comment sont codés les données binaires attachés ?
3. S'il y a plusieurs messages, comment peut-on les individualiser ?

EXERCICE 5 - Comparaisons des commandes de compression

1- Écrire en Python un script créant un fichier texte contenant N lignes, avec sur chaque ligne un nombre aléatoire entre 0 et 1 (fonction `random()` du module `whrandom`. N est un paramètre spécifié sur la ligne de commande. Le fichier s'appelera `hasard-N.txt`, où `N` est la valeur de N.

2- A l'aide de votre script, créer plusieurs fichiers de tailles croissantes et compresser les à l'aide de `compress`, `gzip`, `zip` et si possible `bzip2`.
Le taux de compression est le rapport entre la taille du fichier compressé et la taille originale. Remplir en tableau de la forme

Taille originale	taux compress	taux gzip	taux zip	taux bzip2
1Ko 10Ko 100Ko ... 1Mo				

Conclure.

EXERCICE 6 - Archivage de votre compte UNIX

1- Avec les commandes `tar` et `compress`, créer une archive compressée contenant tout votre compte UNIX.
(voir les exemples d'utilisation de `tar` dans le support de cours UNIX).
Quelle est la taille de l'archive ?

2- Supprimer le fichier archive. A l'aide de la commande `du -s`, comparer la taille de votre compte à celle de l'archive compressée.

3- Fin du TP : faire le ménage sur votre compte UNIX. Transférer vos exercices sur disquette (sauvegarde en cas de problème), puis, à l'aide de la commande `find`, supprimer les fichiers de sauvegarde automatique d'emacs (noms terminant par ~) de tous vos répertoires.

Annexe 1 : source de "7bits.c"

```
/* Filtre de texte :
 *   force a zero le bit de poids fort de chaque octet
 *
 * E. Viennet, Janvier 1999
 */
#include <stdio.h>

void main(void) {
  char c;
  do {
    c = getc(stdin);     /* lit un caractere sur l'entree */
    if (c != EOF) {
      c = c & 0x7F;      /* force bit 7 a zero */
      putc( c, stdout ); /* ecrit le caractere */
    }
  } while (c != EOF);
}
```

Annexe 2 : Commandes uuencode et uudecode

uuencode [fichier] nom

Utilisé pour coder un fichier en n'utilisant que des caractères ascii 7 bits (codes entre 0 et 127), par exemple pour le transmettre sur un réseau ne transmettant que les 7 bits de poids faible.

`uudencode` lit le fichier (ou l'entrée standard si l'on ne précise pas de nom) et écrit la version codée sur la sortie standard. Le paramètre `nom` précise le nom du fichier pour le décodage par `uudecode`.

uudecode [fichier]

Décode un fichier codé par `uuencode`. Si l'argument `fichier` n'est pas spécifié, lit l'entrée standard. Le nom du fichier résultat est précisé dans le codage (paramètre `nom` de `uuencode`).