23 février 2015

Programmation structurée (420-PRS-DM)
Louis Marchand
Département d'informatique
Cégep de Drummondville

Devoir 3

Ce travail doit être fait individuellement. Il doit être remis avant le jeudi 9 avril 2015 à 23h59. La remise devra être faite par l'outil de soumission du logiciel Web Léa sur le portail. Vous devez placer votre code source dans une archive zip et vous devez soumettre que cette archive zip. Votre mandat consiste à programmer une commande shell et une commande dos en utilisant le langage de programmation Python3. Cette commande devra permettre la création d'un fichier d'archivage de fichiers.

Voici les spécifications que la commande devra respecter :

- Le code de votre application devra respecter les règles vues en classe. C'est à dire :
  - Le code devra être lisible et simple,
  - Le code devra être séparé en routine,
    - Il n'est pas nécessaire d'effectuer les jeux de test pour ce travail.
  - Les routines devront être documentées,
  - Les noms de variables et de routines devront être significatifs,
  - Dans votre code, vous devrez seulement inclure les fonctionnalités nécessaires des modules que vous utilisez.
  - Chaque fichier devra contenir une documentation d'en-tête.

- Votre application devra fonctionner autant sur le système d'exploitation Windows 7 (avec l'interpréteur MS-DOS) que sur le système d'exploitation Linux Ubuntu 14.04 (dans un terminal shell). Dans les deux cas, vous ne devrez pas avoir à utiliser l'extension « .py », ni le nom de l'interpréteur (python) pour démarrer a commande (ex : "archiveur arguments..." et non "python archiveur.py arguments…"). La commande devra fnctionner dans tout le système, peu importe l'emplacement de l'utilisateur.

- Vous devrez ajouter à votre projet un fichier LISEZMOI.txt contenant les informations suivantes :
  - Les créateurs de la commande
  - La date de création de la commande
  - Une description de la procédure d'installation et de la configuration de la commande (incluant la rendre accessible dans tout le système).
  - Une description de l'utilisation de la commande
  - Toutes autres notes que vous trouverez pertinentes.

- Votre commande devra effectuer deux tâches :
  - Archiver une liste de fichiers dans un unique fichier d'archivage
    - Seul le contenu des fichiers doit être récupérable
      - Ne dois pas stocker les noms de fichier, les attributs, etc.

    - Extraire les fichiers d'un fichier d'archivage
        - Nommer les fichiers extraient en fonction de leurs index dans le fichier archive.
            - Ex : « Fichier1 » pour le premier fichier de l'archive.a

- Votre commande devra être utilisée de la manière suivante :
    - L'archivage devra se faire de la façon suivante :

      ```
      archiveur archiver <fichier archive> <fichier1> <fichier2> …
      ```

    - L'extraction de tous les fichiers devra se faire de la façon suivante :

      ```
      archiveur extraire <fichier archive>
      ```

    - L'extraction de certains fichiers devra se faire de la façon suivante :

      ```
      archiveur extraire <fichier archive> <index1> <index2> …
      ```

L'implémentation de l'archiveur vous appartient, mais voici une liste de « trucs » qui pourrait vous être utile (en d'autres mots, ce qui suit est optionnel) :

- Le fichier d'archive contient deux champs (entête et corps) :
    - L'entête contient :
        - Le nombre de fichiers contenus dans l'archive
        - La taille de chaque fichier contenue dans l'archive
        - Index (position) du début de chaque fichier dans le corps
    - Le corps contient :
        - Le contenu binaire de chaque fichier

- À noter que lors des lectures de gros fichiers, nous avons la problématique suivante :
    - Si nous utilisons de trop grosses lectures (par exemple, lire le fichier au complet), l'application utilisera beaucoup trop de mémoire (RAM) pour le travail effectuer.
    - À l'inverse, si les lectures sont trop petites (par exemple, un octet par lecture), la performance de l'application sera grandement diminuée.

  Pour régler ce problème, il faut faire des lectures de petite taille côté mémoire et optimisée côté performance. Pour ce faire, il est une bonne pratique est d'utiliser la plus grande taille de cluster utilisé par les systèmes de fichiers courants (NTFS, FAT, ext4, etc.). Une taille de lecture souvent utilisée est 64k (ou 65536 octets).

- Utiliser la commande Unix suivante pour créer des fichiers au contenu aléatoire :

  ```
  dd if=/dev/urandom of=nom_fichier
  ```
  (CTRL-C pour arrêter)

- Utiliser la commande Unix suivante pour comparer le contenue de deux fichiers :

  ```
  diff nom_fichier_1 nom_fichier_2
  ```

- Utiliser la commande DOS suivante pour comparer le contenue de deux fichiers :

```
comp nom  _fichier_1 nom_fichier_2
```

- Un exemple de programme archiveur a été placé sur Léa.

- Voici un exemple de l'utilisation de l'archiveur :
  > dd if=/dev/urandom of=source1
  > dd if=/dev/urandom of=source2
  > dd if=/dev/urandom of=source3
  → Les trois fichiers sources sont créés.
  > archiveur archiver fichier_archive source1 source2 source3
  → Création du fichier_archive
  > archiveur extraire fichier_archive
  → Extraction des fichiers « Fichier1 », « Fichier2 » et « Fichier3 »
  > diff Fichier1 source1
  → Fichier identique
  > rm Fichier?
  → Supprimer les fichiers extraient.
  > archiveur extraire fichier_archive 3 1
  → Extrais les fichiers à l'index 3 et à l'index 1
  > diff source3 Fichier3
  → Fichier identique