ESGI
Année 2012-2013

Langage de scripting/Python

Projet
Manipulation de fichiers `tar`

Important :
Ce projet est à réaliser par groupes de 3 étudiants, et devra être présenté en soutenance le **17-04-2013**.
Lors de cette soutenance, vous présenterez votre projet, et le responsable du cours testera les différentes fonctionnalités. Vous devrez expliquer ce que vous avez réussi à implémenter, et ce qui manque ou ne fonctionne pas correctement. À la fin de cette soutenance, vous devrez remettre le code source de votre programme, accompagné d'un rapport de 5 pages maximum, et dans lequel vous décrirez les choix effectués.

Objectif -

L'objectif de ce projet est de réaliser un outil de manipulation de fichiers `tar`.
Pour plus de détails sur ce format, vous pouvez vous référer à la page suivante : `http://en.wikipedia.org/wiki/Tar_(computing)`.

Le format `tar` -

Un fichier `tar` est un fichier qui regroupe à l'intérieur d'un même fichier (le fichier tar) toute une arborescence de fichiers, de façon structurée. Le format permet de récupérer cette arborescence, avec les noms des fichiers et répertoires, ainsi que divers autres attributs.

Le format de ce fichier est décrit en détail sous l'URL suivante : `http://en.wikipedia.org/wiki/Tar_(file_format)#Format_details`.
On utilisera dans un premier temps le format classique ; votre programme devra par la suite accepter également le format `USTAR`.
De même, on vous demandera dans un premier temps de ne traiter que les fichiers normaux et les répertoires avec des noms courts.

Il existe une commande `Unix`, la commande `tar`, qui permet de construire un archive tar.
Par exemple, pour construire le fichier tar contenant toute une arborescence dont la racine est le répertoire `dir`, il suffit d'utiliser la commande suivante : `tar cvf dir.tar dir`.

Classes et méthodes -

Votre code source devra au moins contenir les éléments suivants :

- une classe Tar, qui représentera une archive `tar`, avec :
  - un constructeur, qui prend en paramètre le nom du fichier tar à ouvrir ;

- *pwd* qui affiche le nom du répertoire courant (le chemin complet) ;
- *ls* qui liste le contenu du répertoire courant ; avec l'option *-l*, elle retourne en plus les attributs des fichiers trouvés dans l'archive ;
- *cat* `<file>` qui affiche le contenu du fichier `file`, se trouvant dans le répertoire courant ;
- *part* `<start> <end> <file>` qui affiche le contenu du fichier `<file>` entre les octets `start` et `end` ;

Optionnel : Gestion des droits -
Vous devez gérer les droits des fichiers et répertoires à l'intérieur de l'archive `tar` de la même manière que sous un environnement Unix. C'est-à-dire que tout fichier (ou répertoire) de l'archive possède des droits, et avant l'ouverture, la lecture, et l'accès à un répertoire, il faut s'assurer que l'utilisateur de votre programme possède les droits pour ces opérations.
Rappelons que :

- pour lister le contenu d'un répertoire, il faut avoir le droit de lecture du répertoire ;
- pour traverser un répertoire, il faut posséder le droit d'exécution sur ce répertoire ;
- pour lire le contenu d'un fichier, il faut posséder le droit d'exécution sur les répertoires permettant d'y accéder, ainsi que le droit de lecture du fichier en question ;

Améliorations possibles -
Vous pouvez apporter différentes améliorations à votre programme, parmi lesquelles :

- gestion des droits ;
- accepter le format `tar` classique ainsi que le format `USTAR` ;
- accepter la commande *ls -R* pour lister récursivement le contenu du répertoire courant dans votre interpréteur ;
- ...

Vous pouvez apporter toutes les améliorations que vous jugerez utiles ; pensez à les expliquer dans votre rapport.

Evaluation -
Le respect des noms des classes et des méthodes précisés dans ce sujet sont un aspect important de l'évaluation finale de votre projet. Si vous le jugez nécessaire, vous pourrez rajouter des paramètres à certaines méthodes ; mais tâchez de l'expliquer dans votre rapport.

Une attention particulière devra être portée à la détection et à la gestion des erreurs qui pourront se produire lors de l'utilisation de votre programme.

Si certaines fonctionnalités manquent, ou si elles ne fonctionnent pas correctement, veuillez le préciser également dans votre rapport. Attachez un soin particulier à la rédaction de votre rapport, ainsi qu'aux commentaires dans le code source.

- une méthode *open_tar(self)* qui se charge de l'ouverture de l'archive (dont le nom a été spécifié dans le constructeur) ;
- une méthode *close_tar(self)* afin de fermer l'archive (si elle est ouverte) ;
- une méthode *chdir_tar(self, path)* qui permet de changer le répertoire courant ; lors de l'ouverture de l'archive tar, le répertoire courant est la racine de l'archive (représenté par */*) ;
- une méthode *getpwd_tar(self)* (comparable à la commande Unix *pwd*) qui retourne le nom répertoire courant ;
- une méthode *opendir_tar(self)* qui retourne un objet de la classe TarDir ;

- une classe TarDir, qui représente un répertoire à l'intérieur de l'archive tar, avec les méthodes suivantes :
  - un constructeur, qui prend en paramètre le nom d'un répertoire, et se charge de sa gestion ;
  - une méthode *readdir(self)* (comparable à la commande Unix *ls*) qui liste le contenu du répertoire courant (dans un premier temps sans mode récursif) ;
  - une méthode *fopen(self, filename)* qui ouvre le fichier dont le nom est donné en paramètre, s'il existe, et retourne un objet de la classe TarFile ci-dessous ;
  - une méthode *close(self)* qui ferme le répertoire, s'il est ouvert ;
- une classe TarFile, qui représente un fichier à l'intérieur de l'archive tar, avec les méthodes suivantes :
  - un constructeur, qui prend en paramètre le nom d'un fichier de l'archive, et se charge de sa gestion ;
  - une méthode *read(self, size)* qui lit *size* octets dans le fichier (depuis sa position courante, initialement à la position *0*), et les retourne sous forme de chaîne de caractères ;
  - une méthode *seek(self, offset, whence=0)* qui place la position courante dans le fichier (voir la méthode *seek* de l'objet file en Python pour plus de détails) ;
  - une méthode *close(self)* qui ferme le fichier, s'il est ouvert ;

Remarque : Les noms de fichiers et de répertoires utilisés seront relatifs au répertoire courant dans l'archive, ou à la racine de l'archive s'ils commencent par */*.

Interpréteur de commandes -

Il vous est également demandé d'écrire un interpréteur de commandes qui utilise les classes et méthodes ci-dessus. Cet interpréteur contiendra au moins les commandes suivantes :

- *opentar* `<fichier>` qui ouvre un fichier `tar` ; une seule archive tar pourra être ouverte à la fois ;
- *closetar* qui ferme l'archive tar courante ;
- *cd* `<rep>` qui permet de se déplacer dans l'arborescence de l'archive tar ; après ouverture avec *opentar*, le répertoire courant est la racine de l'archive ;