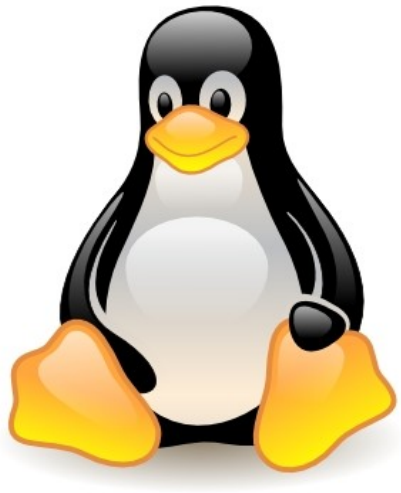
Système – Projet 2014

Commande mytar



Eric Uzenat, Mohamed Yennek

Table des matières

1. Mode d'emploi

1.1 Présentation

L'objectif de ce projet et d'implémenter une commande système d'archivage se rapprochant du fonctionnement de la commande tar mais toutefois beaucoup plus simplifié.

1.2 Compilation

Un outil de compilation makefile est fournie aveec le code source pour plus de simplicité. Ainsi, il suffira de tapper *`make mytar`* pour compiler l'ensemble du code source, et un binaire éxécutable du nom de mytar sera créer. Il est aussi possible de nettoyer tout les fichier générer lors de la compilation en tapant *`make clean`* ou *`make mytar clean.`*

1.3 Execution

Le programme s'éxécute en ligne de commande de la manière suivante :

`./mytar [options] -f archive.mtr[.gz] [path1] [path2] … [pathn] [-C rep]`

les options seront décrite à la section suivante. Il faut savoir que l'option *`-f`* suivit d'un fichier au format *`.mtr`* ou *`.mtr.gz`* est obligatoire.

1.4 Options

-f archive.mtr [path1] … [pathn]:

fournit le chemin vers l'archive et vers une liste de fichier `[pathi]` qui pourra resté vide.
Cette option est obligatoire.

-c:

option pour archiver les fichiers de la liste `[pathi]`, celle ci ne doit pas être vide.
Cette option ne peut pas être utilisé avec: `-x`, `-a`, `-d`, `-l`

-x:

Si `[pathi]` est vide, alors extrait le totalité de l'archive, sinon extrait uniquement les `[pathi]` et reconstruit les arborescence si besoin est
Cette option ne peut pas être utilisé avec: `-c`, `-a`, `-d`, `-l`

-a:

Ajoute les `[pathi]` à l'archive, celle ci ne doit pas être vide.
Cette option ne peut pas être utilisé avec: `-c, -x, -d, -l`

-d:

Supprime les [pathi] de l'archive, celle ci ne doit pas être vide.
Cette option ne peut pas être utilisé avec: `-c, -x, -a, -l`

-l:

Liste le contenue de l'archive, en affichant les droit, la date de derniere modification, et la reference de larchive. Si `[pathi]` est vide, liste l'integralité de l'archive, sinon liste uniquement les [pathi]
Cette option ne peut pas être utilisé avec: `-c, -x, -a, -d`

-k:

Lors de l'extraction des fichiers, indique que les fichier existant ne doivent pas etre remplacé. Un message d'avertissement doit etre affiché, et l'extraction continue.
Cette option doit etre utilisé uniquement avec `-x`

-C rep:

rep devient la racine de l'arborescence archivé. Utilsé en mode création (archivage: -c) cela signifie que les chemins path1 ... pathn, ainsi que les chemins écrits dans l'archive, sont relatifs à *rep ;* en mode extraction, cela signifie que l'archive doit être restaurée dans *rep.*
Cette option s'utilise uniquement avec `-c` et `-x`.

-z:

S'utilise avec l'archivage ou l'extraction. Elle permet de compressé l'ensemble des fichier

apres leur archivage.

2. Descrition du programme

2.1 Le module tools

Le module tools contient la structure d'entete d'un fichier archivé ainsi qu'un certain nombre de fonction annexe nécéssaire au bon fonctionnement du programme.

2.1.1 L'entete

```
struct header {
      size_t path_length;
      off_t file_length;
      mode_t mode;
      time_t m_time;
      time_t a_time;
      char [32] checksum;
};
```

La taille de file_length sera nul si il s'agit d'un repertoire.
Deux methode seront associé au header, celle pour l'initialisé qui prendra en argument l'adresse d'un header, et les donnée nécéssaire a son initilaisation. Elle aura pour prototype:

```
void set_header(header *hd,
                size_t pl,
                off_t fl,
                mode_t m,
                time_t mt,
                time_t at);
```

La seconde servira a recuperer les données correspondant à un header stocké dans un fichier, elle prendra en argument l'addresse d'un header, est un descripteur de fichier qui contient ces informations (on supposera que la tete de lecture sera exactement à l'endroit ou le header est ecrit):

```
void get_header(header *hg, int desc);
```

2.1.2 Les outils

- `void getChecksum(const char *filename, char chsm[32])`

Cette methode prend en argument un nom de fichier est un tableau de caractere, elle créer un processus fils qui, grace a la fonction exec execute la commande shell md5sum est insere l'emprunte md5 du fichier filename dans le tableau chsm.

- `int isMTR(const char *s)`

Cette methde prend en argument une chaine de caractere et test si la chaine termine par “.mtr”. Elle retourne 0 si c'est faux et 1 si c'est vrai.

- `int isMTRGZ(const char *s)`

Cette methde prend en argument une chaine de caractere et test si la chaine termine par “.mtr.gz”. Elle retourne 0 si c'est faux et 1 si c'est vrai.

- `int init_flag(char **argv, int argc, int *flagf, int *flagc, int *flagx, int *flaga, int *flagd, int *flagl, int *flagk, int *flagC, int *flagv, int *flagz, int *lenPath)`

Cette fonction prend en argument le tableau des argument passé en ligne de commande ainsi que la taille de ce tableau et toute les variable de flag. Elle initialise toutes ces variables en fonction du tableau d'argument. La fonction renvoie 0 si les argument de la ligne de commande son possible et -1 sinon.

- `char **get_path(char **arg, int n)`

Cette fonctin prend le tableau d'argument rentré en ligne de commande, et un entier n qui crrespond a la taille de ce tableau. Elle retourne le tableau des fichiers qui sont situé apres le fichier d'archive.

- `int get_pathLen(char **arg, int n)`

Cette fonctin prend le tableau d'argument rentré en ligne de commande, et un entier n qui crrespond a la taille de ce tableau. Elle retourne le nombre de fichiers qui sont situé apres le

fichier d'archive.

- `char *get_archive(char **arg, int n);`

Cette fonction prend le tableau d'argument rentré en ligne de commande, et un entier n qui correspond a la taille de ce tableau. Elle retourne le nom de l'archive situé apres l'option -f.

- `char *get_repC(char **arg, int n)`

Cette fonction prend le tableau d'argument rentré en ligne de commande, et un entier n qui correspond a la taille de ce tableau. Elle retourne le nom du repertoire situé apres l'option -C

- `void suppr_slash(char *s);`

Cette fonction prend en argument un nom et supprime si il existe le caractere '/' si c'est le dernier caractere du nom.

- `int cpyDirTmp(const char *filename)`

Cette fonction prend en argument une référence vers un fichier et le copie dans le reprtoire /tmp en lancant un processus fils qui execute la commande shell `cp`. Elle retourne 0 si elle s'est déroulé correctment et -1 sinon.

- `int compression(char *filename)`

Cette fonction prend en argument une référence vers un fichier et le compresse en lancant un processus fils qui execute la commande shell `gzip`. Elle retourne 0 si elle s'est déroulé correctment et -1 sinon.

- `int decompression(char *filename)`

Cette fonction prend en argument une référence vers un fichier et le compresse en lancant un processus fils qui execute la commande shell `gunzip`. Elle retourne 0 si elle s'est déroulé correctment et -1 sinon.

2.2 L'archivage et l'ajout

2.2.1 Archivage

Le procedé d'archivage est plutot simple, on creer l'entete du fichier [pathi], on l'ecrit dans le fichier de sortie d'extansion .mtr, puis on ecrit, le path, et le contenue du fichier (ci celui ci n'est pas un repertoire. On repete ce procédéer pour tout les path1 … pathn

2.2.2 Ajout

L'ajout utilise exactement le même procédé que l'archivage a la seul difference que le fichier d'extension .mtr est ouvert en mode ajout, on suppose donc qu'il éxiste deja.
Si le fichier en compressé aved l'extension .mtr.gz, alors le fichier est decompresser, puis on ajoute les fichier a archiver puis il est a nouveau compressé.

2.2.3 Les fonctions

- `int archive(char *archive, char **path, int lenPath, int flagv)`
Cette fonction prend en argument une reference, vers une archive à créer, un tableau de référence de fichier à archiver, la taille de ce tableau et le flag v. Elle retourne 0 si elle c'est déroulé correctement et -1 sinon.

- `int archiveC(char *archive, char **path, int lenPath, char *repC, int flagv)`
Cette fonction prend en argument une reference, vers une archive à créer, un tableau de référence de fichier à archiver, la taille de ce tableau, une réfrence vers un repertoire dans lequel on veut archiver tous les fichier et le flag v. Elle retourne 0 si elle c'est déroulé correctement et -1 sinon.

- `int addArchive(char *archive, char **path, int lenPath, int flagv, int flagz)`
Cette fonction prend en argument une reference, vers une archive à créer, un tableau de référence de fichier à ajouter à l'archive, la taille de ce tableau et le flag v et z. Elle retourne 0 si elle c'est déroulé correctement et -1 sinon.

- `int addArchiveC(char *archive, char **path,int lenPath, char *repC, int flagv, int flagz)`

Cette fonction prend en argument une reference, vers une archive à créer, un tableau de référence de fichier à ajouter à l'archive, la taille de ce tableau, une réfrence vers un repertoire dans lequel on veut archiver tous les fichier et le flag v. et z Elle retourne 0 si elle c'est déroulé correctement et -1 sinon.

- `int archiveREG(int archive, char *path)`

Fonction auxiliaire qui prend en argument un descripteur de fichier et une reference vers un fichier et archive le fichier. Si la fonction s'est déroulé correctement elle retourne 0 sinon -1.

- `int archiveDIR(int archive, char *path)`

Fonction qui prend en argument un descripteur de fichier, et une référence vers un reprtoire et archive toute l'arborescence du repertoire.Elle renvoie si elle s'est déroulé correctement et -1 sinon.

2.3 L'extraction

2.3.1 Fonctionnement

Pour extraire les fichiers, le programme lit les données d'entete créer le fichier ou le repertoire a partir des données d'entete et recopie leur contenu. Le fichier d'archive reste inchangé.
Si l'archive est compréssé, alors elle est copier dans le repertoire /tmp puis decompréssé et ensuite on extrait les données.

2.3.2 Les fonctions

- `int extract(char *archive, int flagv, int flagz, int flagk)`

Cette fonction prend en argument une référence vers une archive, ainsi que les trois flag v, z, et k. Puis retourne 0 si elle s'est déroulé correctement et -1 sinon.

- `int extractC(char *archive, char *repC, int flagv, int flagz, int flagk)`

Cette fonction en argument une référence vers une archive, ainsi que une référence vers un répertoire vers lequel on veut extraire tout les fichiers, et les trois flag v, z, k.

2.4 L'affichage

2.4.1 Fonctionnement

Si il s'agit s'implement d'une archive, alors le programme recupere les données d'entete, récupere ensuite le nom des fichier archiver grace au donnée puis saute le contenue des fichier grace a des lseek.

Si l'archive est compresser au format .gz, le prgramme copie le fichier dans /tmp, le decompresse et recupere les donnée de la même maniere de cette archive décompréssé.

2.4.2 Les fonctions

```
-int list(char *archive);
```

Une seule fonction est définie pour l'affichage des données. Le procédé est en effet court et très simple. La seul difficulté viens du tratement de l'extantion du fichier si il est compréssé ou non.

Cette fonction prend en argument la référence vers le fichier d'archive et retourne 0 si elle s'est déroulé correctement, et -1 sinon.

2.5 La suppression

2.5.1 Fonctionnement

La suppression copie les données d'archive dans un fichier temporaire, sans les références donées a la ligne de commande, puis le recopie dans le fichier d'origine, de sorte de garder éxactement le même numero d'inoeud.

Si l'archive est compresser, alors on la décompresse, puis une fois la suppression effectuer on la recompresse.

2.5.2 Les fonctions

- `int delete(char *archive, char **path, int lenPath, int flagz)`

La fonction prend en argument une référence vers une archive, un tableau de reference vers des fichier à archiver, la taille du tableau et le flagz. Elle retourne 0 si elle c'est déroulé correctement et -1 sinon.

3. Diagramme