Projet RS

ptar – un extracteur d'archives tar durable et parallèle

Auteurs :
M. Théo Le Donné
M. Léo Jouët-Pastré

Encadrants :
Pr. Lucas Nussbaum

December 1, 2016

Contents

1 Introduction **3**
1.1 Purpose 3

2 Basic Listing **4**
2.1 Fonctionnement 4

3 Option -x **5**
3.1 Fonctionnement 5

4 Option -l **6**
4.1 Fonctionnement 6

5 Conclusion **7**
5.1 Nombre d'heures 7

1 Introduction

1.1 Purpose

This document describes the operations of the internship management program at Telecom Nancy. You can also find the main goals, the different users, requirements and some visual previews of the software.

2 Basic Listing

2.1 Fonctionnement

Le cas du listeur simple est executé lorsque qu'aucune des options n'est sélectionnée. Une fois l'archive ouverte, une boucle while permet de lire l'intégralité des headers de taille 512 octets. Etant donné que la fin de l'archive est indiquée lorsque le champs ustar vaut "magic", on affiche l'intégralité des fichiers présents dans l'archive. L'une des difficultés rencontrée fut liée au "saut" de la taille de l'élément dans l'archive. En effet nous avions mal compris la structure d'une archive tar. Nous pensions qu'il y avait seulement des headers, nos fichiers étant vides dans nos premières archives test,donc notre programme fonctionnait. Après le premier test blanc nous avons fais d'autres tests avec des archives contenant des fichiers tels que des images. Le programme ne fonctionnait alors plus. Après quelques recherches nous avons vu que le contenu du fichier était stocké après le header. On opère donc un saut avec lseek. (int) strtol permet de convertir ma_struct.size en long puis de le caster en int. Comme les archives tar sont des blocs de 512 octets on doit sauter avec lseek du bon nombre de bloc de 512. La fonction arrondi512(size) permet de calculer ce nombre de bloc. Par exemple si le fichier fait 1000octet il faudra sauter de 2 bloc de 512 soit de 1024 octets.

La gestion des options se fait grâce à un tableau de caractères dans lequel sont stockées les options souhaitées. Elle sont ensuite implémentées dans les différents cas d'un switch après l'appel à la fonction getopt().

3 Option -x

3.1 Fonctionnement

L'option -x permet d'extraire le contenu de l'archive TAR passée en paramètre. Nous commençons par ouvrir l'archive tar avec un descripteur de fichier fdx. Nous parcourons ensuite les headers des éléments de l'archive tar afin de créer ceux-ci. Les tests ma_struct.typeflag[0]=='i' permettent de savoir si l'élément est un fichier (i=0), un dossier(i=5), ou un symlink (i=2). Le parcours des headers se fait en remplissant la structure header_posi_ustar appelée ma_struct pour chaque header. On avance donc de 512 octets pour remplir la structure, on analyse ensuite le type de l'élément avant de le créer. Finalement on avance de la taille de l'élément dans l'archive afin d'accèder au header suivant. Les fichiers sont créés avec creat, les dossiers avec mkdir. On notera que l'on rentre le mode de l'élément m en paramètre lors de la création de fichier. Lors de la création de dossier on caste m en (mode_t). Les symlinks sont créés avec symlink(char[] linkname, char[] name).

Après ce premier parcours de l'archive tar on parcourt une seconde fois pour remplir les fichiers. De même on teste si l'élément est un fichier avec ma_struct.typeflag[0]=='0'. On crée un buffer de la taille de la taille du fichier (size). On remplit ensuite ce buffer du contenu du fichier de l'archive tar avec read avant d'écrire sur le fichier que l'on a créé auparavant. Pour ceci on utilise un descripteur fdfile: c'est le descripteur du fichier créé. On fait ensuite un saut avec lseek jusqu'au prochain header.7 Nous avons fait le choix de faire plusieurs boucles while pour plus de facilité. Nous sommes conscient que le temps d'execution est plus long qu'avec une seule boucle while.

On va ensuite modifier le temps de modifications des éléments créé. Là encore nous avons fait une nouvelle boucle while. En effet si l'on modifie le temps de modification d'un élément puis que l'on modifie l'élément, cela aurait servi à rien. On réalise donc cette opération en dernier. ma_struct.mtime correspond au temps de modification de l'élément contenu dans l'archive. On souhaite que l'élément créé ai ce même temps de modification. On va donc modifié le temps contenu dans le header de l'élément. On y parvient avec utime pour les fichier et les dossiers, lutimes pour les symlinks. On remplit donc une structure utimbuf avant d'appeler utime. On notera que pour remplir le champ actime de utimbuf, qui correspond au temps d'accès, on doit acceder utiliser le temps d'accès du fichier créé.

4 Option -l

4.1 Fonctionnement

L'option -l permet un listing détaillé d'une archive sous la forme : permission uid/gid taille date heure fichier . Dans le cas d'un symlink, on fait apparitre un lien vers le fichier pointé " -¿ dest" . Tout d'abord, dans le cas d'un listing détaillé, on ne souhaite pas faire apparaitre le listing simple. C'est pourquoi la variable lcase passe à 1, un test sur lcase en sortie du switch permet d'effectuer, ou non, le listing simple. Pour la permission, on récupère tout d'abord le typeflag pour indiquer de quel type de fichier/dossier il s'agit. Puis on récupère le mode. Une fois ce dernier converti en entier (ex : 777), on stocke chaque chiffre de cet entier dans une case d'un tableau. Chaque case du tableau correspond alors aux permissions pour chaque utilisateur. Il suffit ensuite de récupérer les informations manquantes : uid, gid, size. La valeur de mtime est ensuite formatée au format désiré grâce à l'appel de la fonction strftime().

5 Conclusion

5.1 Nombre d'heures

	Léo Jouët-Pastré	Théo Le Donné
Etape 1	X h	X h
Etape 2	X h	X h
Etape 3	X h	X h
Etape 4	X h	X h

Table 5.1: Nombre d'heures Léo Jouët-Pastré