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 15, 2016

Contents

1 Basic Listing **3**
1.1 Fonctionnement 3
1.2 Gestion des options 3

2 Option -x **4**
2.1 Fonctionnement 4

3 Option -l **6**
3.1 Fonctionnement 6

4 Conclusion **7**
4.1 Nombre d'heures 7
4.2 Bilan personnel 7
4.3 Bibliographie et remerciements 7

1 Basic Listing

1.1 Fonctionnement

Le cas du listeur simple est exécuté à chaque utilisation de la fonction sauf lors de l'utilisation de l'option -l. Une fois l'archive ouverte, une boucle while permet de lire l'intégralité des headers de taille 512 octets.Le parcours des headers se fait en remplissant la structure header_posix_ustar appelée ma_struct pour chaque header. 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. 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.

1.2 Gestion des options

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().

2 Option -x

2.1 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). 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() et les symlinks sont créés avec symlink().. 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).

Dans une première version du programme, nous avions mal compris le sujet et nous utilisions des appels système pour la création de fichiers et dossiers. Il nous a fallu corriger cette conception. L'implémentation finale est plus simple.

Après ce premier parcours de l'archive tar on parcourt une seconde fois pour remplir les fichiers. Ce fonctionnement n'est pas optimal mais nous a permis d'éliminer des erreurs. L'écriture se fait à l'aide de deux descripteurs de fichier et des fonctions read() et write(). On fait ensuite un saut avec lseek() jusqu'au prochain header. Nous avons fait le choix de faire plusieurs boucles while pour plus de facilité. Nous sommes conscient que le temps d'exécution est plus long qu'avec une seule boucle while.

On va ensuite modifier le temps de modifications des éléments créés. 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 modifier 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 accéder utiliser le temps d'accès de l'élément créé. C'est avec la fonction stat() (lstat() pour les symlinks) que l'on y parvient. La fonction stat() remplit une structure stat, cette dernière contient les informations à propos de l'élément (taille totale en octet, heure du dernier accès, heure de la dernière modification...). Avec cette structure nous pouvons donc accéder

à ces informations, sans les modifier. Dans un premier temps nous pensions que nous pouvions les modifier en modifiant la structure, après plusieurs test nous avons réalisé que la structure n'était en rien lié avec le fichier. C'est à ce moment là que nous avons cherché une fonction pour modifier le temps et nous avons trouvé utime. Ce fut l'une des nombreuses difficultés rencontrées lors du développement de l'option -x.
La fonction lutimes() a un fonctionnement différent, en effet on doit lui donner le temps d'accès (times[0]) et le temps de modification (times[1]). Ces temps sont en secondes et micro-secondes. On doit caster en long les données que l'on a récupéré.

3 Option -l

3.1 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 apparaître un lien vers le fichier pointé. Tout d'abord, dans le cas d'un listing détaillé, on ne souhaite pas faire apparaître 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 et on convertit ce dernier converti en entier (ex : 777) avec la fonction atoi(), on stocke chaque chiffre de cet entier dans une case d'un tableau. Pour cela, on utilise la division par 100 pour avoir le premier chiffre, puis la division par 10 des deux derniers chiffres pour le deuxième et enfin on obtient le dernier par soustraction des deux précédents. Chaque case du tableau correspond alors aux permissions de chaque utilisateur au format "rwx". On utilise la fonction strcat() pour concaténer deux chaînes de caractères entre elles. Par exemple si on a l'entier 6 on concatène "rw-".
Il suffit ensuite de récupérer les informations manquantes : uid, gid, size, mtime. Ces informations sont dans la structure ma_struct en ASCII en système octal. On utilise la fonction strtol() pour convertir le champ en long int puis on caste en int. mtime est stockée dans l'entier mt, ce dernier est casté dans la variable tt de type time_t. On utilise ensuite la fonction localtime() et strftime() pour avoir le format désiré.

4 Conclusion

4.1 Nombre d'heures

	Léo Jouët-Pastré	Théo Le Donné
Etape 1	7h	7 h
Etape 2	2 h	2 h
Etape 3	19 h	20 h
Etape 4	7 h	8 h
Rapport	2 h	1 h

Table 4.1: Nombre d'heures Léo Jouët-Pastré

4.2 Bilan personnel

Léo Jouët-Pastré : Au cours de notre travail et des nombreuses difficultés rencontrées j'ai beaucoup appris. Ce projet intéressant malgré les difficultés qu'il nous a causé, m'a apporté une réelle expérience. Ce fut l'occasion de découvrir de nouvelles fonctions en C et l'interface de programmation POSIX.

Théo Le Donné : Ce projet en binome m'a permis d'approfondir mes connaissances en C et en système. Par manque de temps, nous n'avons pas approfondi l'implémentation des threads à l'étape 5. Néanmoins Ce projet fut enrichissant et il est intéressant de découvrir ce qui se passe derrière un outil que l'on utilise quotidiennement.

4.3 Bibliographie et remerciements

Pour ce projet nous nous sommes aidés des sites web suivant :
https://doc.ubuntu-fr.org/permissions
https://lipn.univ-paris13.fr/ poinsot/save/L2%20Archi/Cours/Cours%209%20-%20Print.pdf
http://www.epochconverter.com/programming/c
https://openclassrooms.com
http://manpagesfr.free.fr
Nous avons également discuté avec le groupe Escamez-Tardivon sur la conception et les différences entre nos projets.