16 Octobre 2016

ptar – un extracteur d'archives tar durable et parallèle

Antonin CELESTIN
Emanuel MARTINS

SOMMAIRE

Chapitre 1

Introduction

Ce projet avait pour principal objectif de réaliser un extracteur d'archives tar garantissant l'écriture sur le disque des fichiers extraits. Afin d'assurer des performances raisonnables, l'extracteur pourra utiliser plusieurs threads pour paralléliser les écritures sur le disque dur.

Ce travail nous permet d'atteindre plusieurs objectifs :

- Lecture de fichiers dans un format binaire assez complexe (tar)
- Création et manipulation de répertoires et fichiers avec l'API POSIX
- Manipulation de threads
- Utilisation de Git

Chapitre 2

Remerciements

Nous tenons à remercier Lucas NUSSBAUM qui a répondu aux questions posés sur le forum rapidement pour nous tenir informé des éventuelles modifications.

Nous le remerciions également pour les cours magistraux qu'il nous a dispensé, nous permettant d'avancer plus rapidement dans notre projet.

Nous avons réalisé ce projet avec l'aide de plusieurs sites, mais la plupart nous avons utilisé l'invite de commande pour connaître l'utilité de certaines commandes :

https://www.freebsd.org/cgi/man.cgi?query=tar&sektion=5
http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

https://www.garron.me/en/go2linux/ls-file-permissions.html
http://www.epochconverter.com/programming/c

Et plusieurs autre commande grâce à man :

http://www.epochconverter.com/programming/c
http://cyberzoide.developpez.com/unix/droits.php3

Chapitre 3

Choix de conception

Le premier choix de conception réalisé a été de séparer les différentes fonctions en plusieurs fichiers distincts. L'un contenant le "main", un autre contenant des fonctions dites "utilitaires" n'étant pas directement utilisées par le "main". Le dernier fichier "extracteur.c" contient les fonctions utilisant les fonctions utilitaires. Ces fonctions sont aussi celles utilisées par la fonction main.

Dans les fonctions "utilitaires" on retrouve notamment des fonctions de conversion

nécessaires au bon fonctionnement du programme mais ne composant pas le cœur de celui-ci. La signature des fonctions présentes dans les fichiers “extracteur.c” et “utilitaires.c” sont renseignées dans des fichiers ”.h” afin de pouvoir les utiliser dans les diverses fonctions du projet.

Ce choix de conception a conduit à un autre concernant le makefile. Pour ce dernier, en s’inspirant d’un makefile provenant d’un autre programme, nous avons défini les différentes librairies utilisées (ldl) ainsi que les fichiers nécessaires à la création du ptar de telle sorte qu’il soit très facile d’ajouter ou de modifier des dépendances. De plus nous avons fait en sorte que lors d’une nouvelle compilation, le précédent exécutable soit supprimé afin d’en créer un autre proprement.

En ce qui concerne les fonctions “core” du projet nous avons rapidement rencontré un problème avec l’archive que nous utilisions pour nos tests. Dans cette archive, certains dossiers pouvaient apparaître dans l’archive après leurs fichiers ou dossiers fils. Ceci nous a ainsi conduit à diviser la fonction extraction en deux. Au lieu de faire un choix suivant le type d’élément rencontré, nous avons ainsi été contraints de diviser le travail d’extraction en deux fonctions. La première créant d’abord les différents dossiers dans l’archive indépendamment de leur ordre grâce aux capacités de “mkdir”. La deuxième créant ensuite les fichiers restant dans l’archive.

Le problème de cette méthode vient des performances. En effet diviser l’extraction en deux étapes nécessite de parcourir par deux fois l’archive dans son entièreté. Ainsi une fois informé que ce cas n’était pas à gérer nous avons décidé de recourir à une méthode ne nécessitant pas de parcourir plusieurs fois l’archive.

Nous avons dû faire face à un problème similaire pour le changement de date des dossiers. Ces derniers voient en effet leur date modifiée chaque fois qu’un fichier y est créé. De fait deux options ont été envisagées. La première consistait à stocker les noms des différents dossiers avec leur date pour pouvoir les changer ultérieurement sans avoir à parcourir une fois de plus l’archive. La deuxième option consiste à l’inverse à parcourir de nouveau l’archive en changeant la date du dossier correspondant quand on le trouve dans l’archive.

La première méthode est donc coûteuse en mémoire et assez compliquée à mettre en place. La deuxième l’inverse. Pour cette raison ce fut la deuxième méthode qui fut choisie.

En ce qui concerne les différentes fonctions de parcours de l'archive (extract, listeur,listeur_detail,date_dossier), pour la condition d'arrêt de la boucle while nous avons décidé en accord avec la définition de POSIX ustar de nous servir des deux blocs de 512 0 présents à la fin de l'archive. Ainsi dès lors que la fonction read renvoie 0, c'est à dire qu'il ne reste lus rien à lire dans l'archive, la boucle while se termine.

Chapitre 4

Difficultés

La première et principale difficulté de ce projet fut l’utilisation de github. Chacun d’entre nous étant peu familier avec ce dernier les problèmes furent vite arrivés. De même au cours du projet plusieurs fautes d’étourderies furent à déplorer notamment au début avec l’utilisation d’une archive tar.gz en lieu d’archive test.

Une des grandes difficultés résidait également dans le fait de ne pas trop savoir les résultats qui devant être obtenus puis de devoir modifier son code afin de respecter un certain affichage.

En ce qui concerne l’application en elle-même, la difficulté est surtout venue de l’appréhension voire parfois de l’incompréhension des différentes fonctions devant être utilisées. La principale source pour ce projet fut d’ailleurs le manuel qui fut consulté à maintes reprises pour

s’assurer de la bonne compréhension des fonctions ou trouver l’origine de leur non-fonctionnement.

Parmi toutes on peut citer "lseek" dont l’utilisation à été au premier abord compliqué en raison d’une mauvaise compréhension du format POSIX_USTAR.

L’implémentation d’une fonction permettant d’afficher correctement la date prit également du temps. Si la structure "tm" et la fonction "strftime" furent vite identifiées comme nécessaires, la mise en place de leur utilisation mit un certain temps. La solution ne vint d’ailleurs pas directement de nous puisque c’est en s’inspirant d’une fonction trouvée sur le site "epochconverter.com" que la fonction d’affichage pu être terminée, grâce à la conversion du temps des fichiers.

La plus grosse perte de temps fut néanmoins probablement due à l’étape 6. Ici en raison d’une lecture trop rapide de la documentation "zlib" Aucune des fonctions d’accès "gzip" ne fut remarquée. Beaucoup de temps fut ainsi perdu à tenter de créer une fonction réalisant la décompression de ces fichiers uniquement à l’aide des fonctions externes "inflateInit", "inflate", "inflateEnd". Celle-ci fut d’ailleurs finalement réalisée mais au vu de la simplicité de l’utilisation des fonctions d’accès "gzip" ce sont ces dernières qui furent retenues.

Chapitre 5

Tableaux des temps

	Temps Emanuel MARTINS	Temps Antonin CELESTIN
Etape 1	5h	5h
Etape 2	2h	2h
Etape 3	8h	8h
Etape 4	10h	13h
Etape 5	4h	6h
Etape 6	0h	4h
Etape 7	0	1h
Rapport	1h	2h