TELECOM Nancy
193 Avenue Paul Muller
54602, Villers-lès-Nancy
Université de Lorraine

Projet de Réseaux et systèmes

Extracteur d'archives tar durable et parallèle

Lucas MARTINEZ
Yann PRONO

2016 - 2017

REMERCIEMENTS

Le déroulement de se projet nous a amenés à nous poser de nombreuses questions. Le bien connu site web StackOverFlow nous a régulièrement aidés sur des difficultés d'aspect techniques liés à l'utilisation du langage C, la communauté y est très active et des explications sont fournies. Le site unixtimestamp.com nous a également aidés. Concernant les différentes API utilisés dans ce projet, nous nous sommes documentés sur les manuels linux. Enfin, nous tenons à remercier Lucas Nussbaum pour avoir répondu à nos questions de manière efficace et rapide sur des connaissances nous manquant pour réaliser ce projet à bien.

INTRODUCTION

Le projet de Réseaux et Systèmes a pour objectif la création d'un programme d'extraction d'archive tar. Plusieurs options sont implémentées. L'écriture des données sur le disque doit être garantie. En contrepartie, cette garantie entraîne un ralentissement du système. Pour pallier ce problème de performance, nous avons utilisé le principe de multithreading pour améliorer la vitesse d'exécution du programme.

CONCEPTION

Le programme repose sur deux structures de base : *header_posix_ustar* et *w_info*. *header_posix_ustar* est une structure C représentant l'en-tête d'une entrée dans une archive tar (nom de l'entrée, typeflag, permissions ...). Cette structure directement expliquée dans le manuel de tar a une taille totale de 512 octets.
La deuxième structure est *w_info*. Cette dernière regroupe un pointeur vers *header_posix_ustar* ainsi qu'un pointeur vers les données liées à cette en-tête. C'est cette structure qui sera passée en paramètre des threads afin de permettre l'écriture des données en simultanée.

Dans l'objectif de réaliser un programme maintenable au fur et à mesure des tests, nous avons décidé d'éclater notre programme en plusieurs modules :

- *header_posix_ustar* : représente la structure accompagnée de getters adéquates.
- *extract* : contient toutes les fonctions propres à l'extraction des données.
- *option* : permet de gérer les différentes options du programme.
- *print* : permet d'afficher dans la sortie standard les différentes messages d'erreur.
- *utils* : contient un ensemble de fonctions non utiles à l'extraction proprement dites des données mais utiles pour le programme (conversion d'octal en décimal, récupération du nom de fichier sans son extension...).
- *w_info* : représente la structure décrite précédemment.
- *ptar* : contient l'ensemble des fonctions réalisant la lecture de l'archive tar suivant les options de l'utilisateur.
- *main* : contient la fonction principale du programme.

OPTIONS IMPLÉMENTÉES

Toutes les options du sujet ont été implémentées dans notre programme :

- l : Permet d'afficher une liste détaillé des entrées de l'archive tar
- x : Permet l'extraction de l'archive tar sans optimisation des performances
- p : Permet l'extraction optimisée de l'archive avec N threads. Il est nécessaire d'avoir l'option -x d'actif pour extraire.
- z : Permet d'extraire une archive tar compressée au format *gzip*.
- Un Checksum des fichiers extraits est calculé pour vérifier que l'archive n'est pas corrompue.

ORGANISATION

L'ensemble du projet a été versionné à l'aide de l'outil git. L'utilisation de cet outil s'est révélé agréable En effet, nous nous étions répartis le travail de manière équitable. En général, un membre était responsable d'une étape du projet, les étapes étant liées les une autres, il nous ai également arrivé de travailler sur les parties de l'autre membre. Un fichier README a été mis en place au tout début du projet afin d'en comprendre sa structure. Nous avons lu de la documentation sur le format tar pour comprendre la structure d'une archive et les informations contenues dans le header.

Nous avons également utilisé make de manière efficace. Le make file nous permettait de créer, les fichiers, l'archive, nettoyer le répertoire courant et de lancer la compilation en mode debug.

Voici le nombres d'heures de chaque membre pour les différentes phases du projet :

Travail	**Lucas Martinez**	**Yann Prono**
conception	5	6
Codage	18	16
Tests	5	4
Bugs/Fuites mémoires	2	3
Rédaction du rapport	2	2
Total heures	32 heures	31 heures

UTILISATION DES THREADS

La partie des threads fonctionne de la façon suivante : Le thread principal passe le header ansi que les données du fichier associé sous forme d'un char[] au thread qui se charge d'écrire le fichier avec la fonction fsync().

Le nombre de threads disponibles est géré par une sémaphore qui empêche le thread principal de dépasser le nombre de threads autorisés. Les identificateurs des thread se trouvent dans un tableau dont chaque case représente un thread. Un tableau d'entier, dont chaque case est gérée par au maximum unique thread permet au thread principal de connaître quels sont les threads disponibles pour une nouvelle extraction.

Lorsque tous les fichiers ont été extraits, le thread principal exécute une boucle sur le tableau de thread avec pthread join sur chaque case pour s'assurer que les exécutions des threads sont finies.

FORMAT GZIP

Pour gérer les archives gzip, nous chargeons dynamiquement la libraire zlib avec dlopen.

DIFFICULTÉS RENCONTRÉES

Nous nous sommes demandés si les threads devaient juste gérer l'écriture des fichiers ou la lecture et l'écriture, auquel cas le temps d'exécution du programme pouvait être amélioré. Une version thread en lecture et écriture à été faite, mais seule la version thread en écriture à été rendue, car plus conforme au sujet.

Deux semaines avant le rendu, nous avons commencé à trouver les bugs et fuites mémoires de notre programme avec l'outil valgrind. Une grande partie de ces fuites ont été corrigés mais certaines subsistent, notamment avec l'utilisation de la fonction chargeant dynamiquement les librairies (dlopen). Quelques difficultés ont étés rencontrées autour de la gestion des dates. La documentation sur les fonctions des dates sont nombreuses.