

Projet du module de Réseaux et Systèmes :
ptar, un extracteur d'archives *tar* durable et parallèle

GARCIA Guillaume et ZAMBAUX Gauthier

December 15, 2016



Remerciements

Introduction

L'application que nous avons développée ici a pour objectif de lire des archives *tar* puis d'en extraire le contenu dans des fichiers.

Le projet s'inscrivant dans le module de Réseaux et Systèmes,

1 Conception

1.1 Page de manuel

Pour l'écriture de la page de manuel *ptar(1)*, il a fallu assimiler le langage utilisé pour écrire ce type de documents. Nous avons ainsi été capable de reproduire la page donnée dans le sujet du projet. Pour l'édition et l'affichage de la page, nous avons utilisé l'utilitaire *groff-utf8* qui permet l'utilisation de caractères spéciaux comme les accents français.

Pour visualiser la page du manuel, il est possible d'utiliser la commande :

```
man ./manpage/ptar.1.gz
```

depuis la racine du dépôt. À noter que le format *.1.gz* est un format standard pour les pages (1) de manuels. Si toute fois on souhaite la consulter avec la commande :

```
man ptar
```

il est nécessaire de d'abord suivre la procédure d'installation décrite dans le *README* (nécessite les droits administrateur).

1.2 Étape 1 : Listeur basique

Tout au long de la conception du programme, la page de manuel *tar(5)* nous a servi de référence. Nous y avons d'abord trouvée la structure d'une entête d'archive *POSIX USTAR* (nous avons donc choisi de gérer ce type d'archives).

Pour parcourir l'archive, nous avons utilisé les fonctions de *fcntl.h* (*open()*, *read()*, *lseek()*, *close()*) et une boucle *do...while* dont la sortie est assurée par la détection d'une taille nulle du champ *name* du *header*. En effet, la fin de l'archive (*End Of File*) est indiquée par deux blocs successifs d'octets à 0 (voir *tar(5)*).

Nous avons également dû omettre les données suivant le *header* dans le cas d'un fichier non vide. Pour cela, nous récupérons le champ *size* du *header* afin d'appeler la fonction *lseek()* pour déplacer la tête de lecture courante du *offset* égal à *size*. Deux problèmes se sont alors présentés :

- Nous devons convertir *size*, une chaîne de caractère représentant la taille en octal, en un entier (*int*) en base décimale. Pour cela, nous avons utilisé la fonction :

```
long int strtol(char *string, char **endptr, int base)
```

Le deuxième argument ne nous intéresse pas ici et est mis à *NULL*.

- En appliquant simplement cela, nous nous sommes aperçu que le programme n'affichait pas ce qui était attendu. Cela était dû à la segmentation des archives *tar* en blocs de 512 octets et était particulièrement visible sur les données des fichiers (qui suivent le *header* concerné). Nous avons trouvé ce phénomène en utilisant la commande :

hexdump -C nomArchive.tar

Nous avons donc dû déterminer le multiple de 512 supérieur à *size* après conversion qui lui était le plus proche, tout en excluant le cas où *size* en était déjà un multiple.

1.4 Étapes 4 : Listing détaillé et application des attributs à l'extraction

Pour gérer les options en lignes de commande, nous avons utilisé la page de manuel *getopt(3)*. Cette partie n'a pas posé de problème particulier.

À cette étape, nous avons voulu vérifier l'existence et la validité (extension correcte) de l'archive passée en paramètre (voir *checkfile.h* dans les sources du dépôt). Nous avons aussi, dans la boucle principale, vérifié le champ *magic* du *header* pour nous assurer qu'il s'agissait bien d'une archive *USTAR*. Le cas échéant, *ptar* retourne immédiatement 1.

Pour ce qui est de l'extraction, nous avons créé une fonction

*int extraction(headerTar * header, char * data)*

où *data* sont les données éventuelles suivant le *header* récupérées à l'aide d'un appel à *read()* à la place de *lseek()*. La différenciation d'éléments se faisait sur le champ *typeflag* du *header* à l'aide d'un *switch...case* sur le premier caractère de ce champ. Pour extraire, nous avons utilisé diverses fonctions telles que *open()*, *write()*, *mkdir()* et *simlink()*. Il fallait faire attention à passer *size* converti non ramené au multiple de 512 calculé dans la boucle principale en argument au *write()* éventuel.

1.3 Étapes 2 et 3 : Options de lignes de commande et extraction