# Rapport de Projet - CTAR

## I. Introduction

Dans le cadre de ce projet, nous avons réalisé un générateur/extracteur d'archive .tar conforme à la norme GNU POSIX 1003.1-1988 "ustar". Ce rapport détaillera nos choix de conception, les difficultés rencontrées, leurs solutions, et fournira un guide d'utilisation du programme.

## II. Choix de Conception

### 1. Structure du Projet

Le projet est structuré de manière modulaire avec des fichiers .h et .c . Cette approche facilite la maintenance et la lisibilité du code.

### 2. Utilisation de zlib

Nous avons intégré la bibliothèque zlib comme recommandé pour ajouter la fonctionnalité de compression à notre archive.

### 3. Approche Incrémentale

L'approche incrémentale a été suivie avec succès, en commençant par la récupération des paramètres, puis la lecture, l'extraction, et enfin la création d'archive. Cela a facilité le suivi du développement et la détection précoce des erreurs via des tests réguliers.

### 4. Respect des Contraintes

Nous avons basé notre conception sur les contraintes imposées. Voici un récapitulatif des contraintes imposés et leur respect :

✔️ **CT01** – La compilation du projet doit se faire via un ***Makefile***

✔️ **CT02** – La définition des structures doit se faire dans un fichier ***typedef.h***.

✔️ **CT03** – La définition des méthodes protoype (*.h*) & implémentation (*.c*) doit se faire de manière séparée autant que faire se peut.

✔️ **CT04** – Le code produit doit être documenté.

✔️ **CT05** – La récupération des paramètres doit se faire via les fonctions usuelles ***getopt()*** / ***getoptlong()*** .

✔️ **CT06** – La gestion des erreurs doit se faire via « les mécanismes proposés par errno ».

Contraintes facultatives :

✔️ **CTO01** – La documentation du code générée via l’utilitaire ***doxygen****. DoxyFile dans le dossier doc*

✔️ **CTO02** – Le code est soumis à un contrôle de couverture via l’utilitaire ***gcov****. Execution via le makefile*

❌ **CTO03** – Une page de manuel Linux est rédigée pour détailler l’exécution du binaire.

## III. Difficultés Rencontrées et Solutions

### 1. Compréhension du Système Tar et Gestion des Entêtes

#### 1.1. Structure Tar et Norme "ustar"

La première difficulté a résidé dans la compréhension détaillée de la structure tar, qui repose sur une organisation en blocs de 512 octets. Cela a nécessité une analyse approfondie de la norme GNU POSIX 1003.1-1988 "ustar". La création de la structure struct header\_tar pour représenter les entêtes des fichiers a été cruciale, demandant une gestion précise de champs tels que le nom du fichier, les permissions et la taille, encodés en octal, ...

#### 1.2. Lecture Séquentielle et Analyse des Entêtes

La seconde difficulté a été la mise en œuvre d'une lecture séquentielle précise des blocs pour extraire les entêtes. Chaque fichier dans l'archive est précédé de son entête, et comprendre comment extraire correctement ces informations a été un défi. La manipulation d'entêtes a demandé une attention particulière à la conversion des valeurs octales en types de données appropriés.

### 2. Lecture de Fichiers et Gestion des Buffers

#### 2.1. Blocs de 512 Octets et Itérations

La première partie de la gestion des fichiers a nécessité une lecture précise des blocs de 512 octets. La création de buffers pour stocker temporairement ces blocs a été cruciale, assurant une lecture itérative jusqu'à la fin de l'archive. La gestion correcte des tailles de fichiers a été un point clé pour éviter des erreurs de lecture ou d'extraction.

#### 2.2 Taille des Fichiers et Buffering

La gestion des tailles de fichiers a été cruciale pour garantir une lecture précise et complète des données. La création et la gestion de buffers ont été essentielles pour stocker les données lues temporairement. S'assurer que la taille des fichiers est correctement gérée et que la lecture des blocs se fait de manière ordonnée ont été des aspects techniques à résoudre.

### 3 Compression et Décompression avec Gzip

L'intégration de la compression Gzip dans le projet a constitué un défi supplémentaire. La manipulation des flux compressés a nécessité une compréhension approfondie des mécanismes de Gzip, ajoutant une complexité supplémentaire à la lecture et à la création d'archives.

## IV. Temps Passé sur le Projet (38h)

Ayant opté pour le pair programming nous ne pouvons pas réellement représenter le temps/personne

### Conception (4h)

- Conception Globale du Projet : 3 heures

- Définition des Structures et Interfaces : 1 heures

### 2. Codage (25h)

- Implémentation de la Lecture d'Archive : 5 heures

- Implémentation de l'Extracteur d'Archive : 10 heures

- Implémentation du Générateur d'Archive : 10 heures

### 3. Tests (4h)

- Tests Unitaires : 2 heures

- Correction des Bugs et Optimisation : 2 heures

### 4. Rédaction du Rapport (5h)

- Rédaction de la Documentation : 3 heures

- Rédaction du Rapport de Projet : 2 heures

## V. Guide d'Utilisation

Le programme propose plusieurs fonctionnalités accessibles via des options en ligne de commande :

**FM01** – Le binaire est capable de lister les éléments d’une archive :



**FM02** – Le binaire est capable d’extraire l’intégralité d’une archive passée en paramètre

**ET**

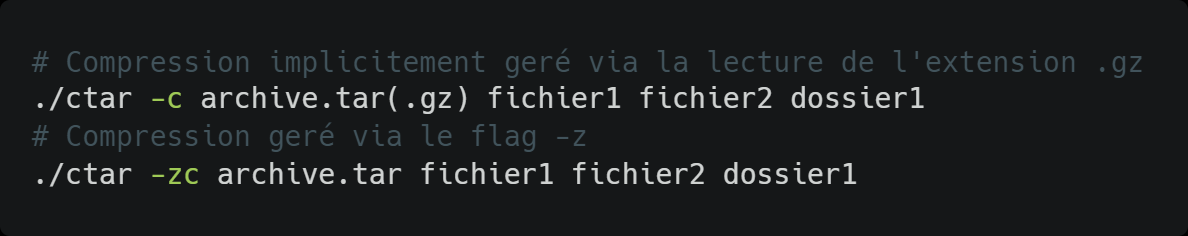
**FMO02** – La prise en charge de la décompression d’une archive tar.gz (gzip) :



**FM03** – L’application est capable de générer une archive.

**ET**

**FMO01** – La prise en charge de la compression d’une archive tar (via la libraire zlib) :



## VI. Conclusion

En conclusion, ce projet nous a permis de mettre en pratique les concepts de programmation en langage C, la gestion des fichiers, et la conception de programmes modulaires. Les choix de conception ont été guidés par la volonté de produire un code clair, modulaire, et conforme aux spécifications du cahier des charges. Ce projet a également renforcé notre compréhension de la gestion des erreurs et de la compression de fichiers.