MOULKAF Anouar
CHAPOULIE Alexandre

IN501A3

RAPPORT

PROGRAMMATION SYSTEME

1 - Objectifs

Le but de ce projet de programmation est de recréer un systeme d'archivage de fichiers par le biais de la commande "tar".
L'archive contiendra donc un ensemble de fichiers en un seul.
Les possibilités fournies par cette commande seront l'archivage de fichiers et/ou répertoires, l'extraction de composants internes d'une archive dans le dossier courant, la suppression d'un fichier constituant une archive, ainsi que la mise à jour d'un fichier dans l'archive depuis un fichier modifié a l'exterieur de celle-ci.
Des options associées à cette commande seront aussi gérées, telles que le manuel par exemple.

2 - Travail Effectué

Nous avons commencé notre projet par l'implémentation d'un module "outils.c", dans lequel sont rassemblées un ensemble de fonctions qui seront utilisées par tous les autres modules, cette implémentation a été choisie pour éviter la duplication de code.
Nous avons egalement crée un struct dateModif permettant de gerer l'affichage et la manipulation d'informations concernant les dates de dernières modifications associées aux fichiers (nous y reviendrons plus loin).

Les différents modules disposent entre autres de fonctions qui renvoient des informations sur des fichiers (nom, taille, date de modification), de fonctions pour concatener des fichiers et de fonctions pour gérer les répértoires.

outils.c

Ce fichier correspond à une bibliotheque de toutes les fonctions utiles dans les fichiers archivage.c et desarchivage.c , en quelque sorte une boite à outils pour le codage de notre commande.

Tout d'abord on y retrouve des fonctions de recuperations d'informations indispensables à la creation d'archive telles que :

size : fonction qui prend en parametre un nom de fichier (ou dossier) et retourne sa taille.

permissionOfFile : fonction qui prend en parametre un nom et qui renvoie les droits d'accès associés à celui ci.

modifiedDate : fonction qui enregistre la date de derniere modification(ou de creation) d'un fichier passé en parametre

addHeader : Qui ajoute dans l'archive une en-tete correspondant au fichier (avec nom, type du fichier, date de modification , délimités entre eux par divers symboles)

addTo : Ajoute a la suite de l'entete, le contenu du fichier à ajouter, dans l'archive.

Autre groupement de fonctions tres important concerne les operations sur les fichiers et les operations associées :

getNextFileName : Cette fonction prend un descripteur de fichier et recupère le nom du prochain fichier à creer. Le curseur est egalement deplacé apres le délimiteur de nom.

getNextFileSize : Fonction semblable à la precedente , qui recupere la taille du fichier suivant.

readNextFile : Recupère le contenu du fichier dans une variable de type char *.

Ces fonctions sont également très utiles pour l'extraction!

archivage.c

Ce module contient un ensemble de fonctions en rapport avec l'archivage, le module génère une archive d'une liste de fichiers et répertoires en effectuant quelques appels à des fonctions du module outils.h

Archivage:

La fonction d'archivage prend en paramètre une liste de fichiers, le dernier nom de fichier est considéré comme étant le nom de l'archive. Pour chaque fichier de la liste, on crée un entête qui contient les informations associées (nom, taille, date de modification), puis on insère cet entête dans l'archive, suivi du contenu du fichier.

Pour avoir un programme d'archivage plus dynamique, nous avons conçu notre fonction d'archivage de façon à ce qu'elle puisse distinguer les fichiers des répertoires et les traiter adéquatement.

La fonction parcourt la liste des fichiers (ou répertoires), et en identifie le type, s'il s'agit d'un fichier, le contenu est rajouté à l'archive, sinon, dans le cas où il s'agit d'un répertoire, une fonction se charge de l'ouvrir et d'y appliquer le même test précédent. Cette fonction va etre appelée récursivement à chaque fois qu'un répertoire est trouvé. La fonction produit donc une archive tout en respectant la hiérarchie des sous dossiers.
L'utilisateur pourrait donc utiliser une liste de fichiers et dossiers sans se soucié de leurs types.
En ce qui concerne les en-têtes de fichiers conservé dans l'archive , nous avons sauvegardé la taille, qui va nous indiquer le nombre d'octets à lire pendant le désarchivage , la date de modification qui va nous sérvir pour savoir si le fichier a été modifié recement, et faire la mise à jour dans l'archive.

Voici la structure de l'en-tête suivant le type de fichier :

Fichier : +++<nom_fichier>$$$<taille_fichier>;;;<date_modif>***<contenu_fichier>

Repertoire : +++<nom_repertoire / nom_fichier>$$$<taille_fichier>;;;<IDEM>

Dans un premier temps, nous nous sommes contenté de la taille et la date, une extension a été prevue pour plus tard, nous avons envisagé de rajouter les droits du fichier, mais une contrainte de temps a été posée.

Désarchivage:

L'extraction de fichier d'une archive se fait de maniere inverse à celle de l'archivage.
On rentre en ligne de commande un nom d'archive et les fichiers (et/ou dossiers) sont extraits de cette dernière dans le repertoire courant.
La procédure va prendre soin de relever séparement les noms de fichiers, les tailles,les dates, ainsi que les contenus des fichiers à extraire (cette partie est gerée par les fonctions getNextFileName , getNextFileSize , readNextFile) , débarassés de leurs délimiteurs..

On va ensuite créer les fichiers associés a l'aide de descripteurs de fichiers en utilisant les noms,tailles,dates,contenus (extraits de l'archive.

Suppression de fichier :

Une fonction deleteFileArchive supprime un fichier passé en pramètre d'une archive. la fonction cherche le fichier dans l'archive et sauvegarde sa position, un fichier temporaire est crée ensuite où on sauvegarde tous ce qui suit le fichier que nous voulons supprimer, puis on enregistre à partir de ce fichier, dans le fichier archive à l'endroit du fichier que nous voulons supprimer afin de l'écraser, puis on tronque le reste. Une autre methode de suppression était de masquer le fichier dans l'archive, avec un simple boolean dans l'entete qui précise si le fichier est masqué ou non. Toutefois cette méthode nous a paru insatisfaisante puisque ne réduisant pas la taille de l'archive.

Mise à jour du fichier:

La fonction de mise à jour, récupere la date de derniere modification de l'archive grace à la fonction lastModifedArchive qui renvoie un pointeur vers une structure Date. on compare cette date avec celle du fichiers.Dans le cas où il faut faire la mise à jour, on supprime le fichier de l'archive avec la fonction de suppression et puis on le réecrit dans l'archive avec les fonctions simples d'archivage.

Gestion de versions

Lors de la réalisation de ce projet de programmation système nous nous sommes appuyés sur le programme de gestion de versions 'GIT'.
L'appel à cet utilitaire nous a permis une meilleure approche du travail en groupe ainsi qu'une mise en commun des idées, efficace.