Programming Assignment #4 Documentation

Authors: Sneheel Sarangi , Ryan Elliot Hsu

Overview:

The purpose of this assignment is to create an archiving application which performs two services: (1) flattening a logical hierarchy(-ies) of a file system then storing this hierarchy(-ies) in a single file, and (2) recreating a file directory using the single file created. There are four main functions available for the user: (1) archiving a file, (2) extracting, or recreating files from the archived file, (3) displaying the metadata (owner, group, rights), and (4) displaying the logical hierarchy of the file system.

Archive:

This function creates a single archive file which stores all the content and metadata of files listed by the user, including all the files and directories under the given files. The structure of the archive file begins with a 20 byte header, followed by the content of all files, followed by all the metadata at the end. Archiving the file system is mainly done in two steps: archiving the file content, and archiving the metadata of the files. To archive the file content, the program reads in data from the files to be archived byte by byte, then writes to the archive file byte by byte. To archive the metadata, we create a metadata string and add all metadata to the string. At the beginning of the metadata string for each file, we append a ‘f’ or ‘d’ to represent whether the file is a standard file or a directory; we use the ‘?’ character to differentiate between different elements of the metadata; and the ‘:’ character to mark the end of the metadata for each file. Since archiving a file often requires going through several layers in a logical hierarchy, we recursively call our archive function each time we reach a directory, this is executed in a depth first fashion.

Extract:

To extract a file, we read the archive file, read the header to locate the metadata, go to the metadata, then begin recreating the logical hierarchy by using the metadata, which contains the address of the file content, then change the owner, groups, and rights of each file also provided by the metadata. Similar to how we archive a file, this is carried out recursively, as each time we

reach a directory, we call the extract function. When using the metadata, we use a metadata struct to store the inode information, this helps make the program more understandable and achieve OOP principles. For changing the metadata, functions such as fchmod() and fchown() are used here.

PRINT METADATA:

This function displays the metadata, which is the owner, groups, and rights of all files in the archive file. This is achieved with the help of a function getstruct() we defined in the program.

PRINT HIERARCHY:

This function displays the logical hierarchy of all files archived. This is also achieved with the help of a function getstruct() we defined in the program.

HOW TO RUN THE PROGRAM:

The program can be invoked in the command line with the following commands:

1. make (- only needed to create the executable)
2. ./adzip {-c|-a|-x|-m|-p} <archive-file> <file/directory list> (- to run the program)

References

Parts of this application utilized ideas and code found in the following sources:

1. Code given in the recitation by Dena Ahmed.