Contents

1 Overall System Design

1.1 Overall Layout / theory

The overall design of the myar aplication is fairly straightforward. When a user runs the program, the command line parser parses it, and determins what sub-sections of the program to run.

It then opens the archive, after verifying that it begins with the proper ARMAG, and then executes the portion of the program that should deal with whatever action you wanted it to run.

1.2 Command Line Parsing

The parser here is fairly simple. Initially, I was going to use getopt() but after re-reading the requirements, it appeared you wanted all the arguments first, with the files passed on the end, not right after the argument, as per standards.

- It starts with your first argument, (the flags, or options) and passes them into a variable to deal with later.
- Next it takes your second argument, (the path to the archive) and stores it into a variable to deal with later.
- Next, It will iterate through up to 7 more arguments, saving them into an array to be dealt with later.
- Then it double checks that you have provided enough / correct arguments. If not, it dies with an error.
- Finally, It itterates through a list of acceptable arguments, and executes the proper portion of the program, based on whitchever one it encounters first.
- it will check arguments in the following order:

  -t

  -v

  -q

  -x

  -d

  -A

  errors out if nothing was found.

1.2.1 Problems

There were major problems with passing invalid arguments (mainly not enough arguments) and the program crashing and burning. However, Recently, most of these cases have been trapped, and should no longer ocour.

1.3 Listing files

After the archive has been verified, and opened for editing / reading, all archives and read modes are initially parsed the same way. It starts by calling getList(), whether you wanted a quicklist, or the detailed list.

getList() fetches the address of the end of the archive, initializes variables, and then begins looping through the file. It starts imediently after ARMAG, and will read the first header, then convert the length of that header's file to an integer.

It will then lseek that far through the file, to the next file header, and read the next one. It continues through the file in this manner, until it finally seeks to or past the end of the file.

Each time, it writes the file header into the global array of structs that stores all this data.

Once that has finished, it will call fixList() which simple iterats through the list, removing trailing slashes, invalid files, and pretying up the list. It then falls back into whichever form of list you asked for. The simple list pretty much just outputs what is stored in the fixed list, while the detailed list is more complex.

1.4 Extracting Files

Extracting files with myar is simple. Simply use the command:

- myar -x ¡name of archive¿ ¡file to extract¿ [¡another file to extract¿]...

Internally, the commandline parser will parse through your arguments, and put all arguments after the name of the archive into an array to be used by this extraction process.

It then iterates through the list of files in the archive, and in a nested loop, iterates through the arguments you passed it. For all matches, it will call the function to actually extract said file to its file.

1.4.1 Caveats

- I have thoughly tested with ar, and it does not seem to preserve directories, at least not absolute directories. Hence, myar will also discard absolute directories.
- If you specify the same file name twice, myar will sometimes extract the file twice to the same file, duplicating it. Avoid this.

1.5 Adding Files

Not implemented in program yet. :(

2 Work Log

Coming soon. In the meantime, you can find a rough outline here: `https://github.com/1n5aN1aC/myar/commits/master`

3 Challenges

4 Questions And Answers

4.1 what do you think the main point of this assignment is?

- Make sure we know how to use all the tools of the course (C, tex, etc.)

- Get us used to basic system calls, and moving around inside of and reading/writing files.

4.2 how did you ensure your solution was correct? Testing details, for instance.

insert info here.

4.3 what did you learn?

- Although I previously knew some C++ this assignment has been a rude awakening on the front of c-strings, character arrays and the like
- Overall, I would say I strongly dislike C from this couse so far. I like my objects in Java / Python too much.
- Although, I have mostly learned how to work with structs and character arrays in C, even though I greatly dislike it.