Project 3: myar

Robert L. Phillips III

October 30, 2013

Contents

1 Design

I assumed for all these functions that the provided files were in the current working directory, so no files with absolute paths were given or relative paths such as ../file.txt.

1.1 Files

1. write.c/write.h
2. read.c/read.h
3. extract.c/extract.h
4. remove.c/remove.h
5. proj3.c

1.2 write.c/write.h

The write.c file was to implement all the functions needed to write to an archive. The main functionality that this file was to provide was the ability to append files to the end of an archive. The only deviation from this design that is seen in the final product is that this file was not originally built to append multiple files at once. It was built to append one file, and the main file, proj3.c, was going to implement the function that called the append function for multiple files. The final revision implemented the function to append multiple files in write.c to keep all of the functions that involve purely writing to an archive together.

1.3 read.c/read.h

The read.c file was to implement all the functions needed to read and move around in an archive. The main functionality that this was file was to provide was the ability to read the archive headers and either seek to the next header or print the header information to stdout. There was no deviation from the original design in the read.c file.

1.4 extract.c/extract.h

The extract.c file was to implement all the functions needed to extract files from an archive. As the program progressed, I realized that this file would be an abstraction layer on top of read.c and write.c because much of the functionality needed to perform the functions needed reused much of the code implemented in the other two files. This file was not intended to implement any remove functionality, but the final design implemented that functionality (for an explanation of why see the remove.c/remove.h section).

1.5 remove.c/remove.h

These files were originally intended to provide functions to remove files from an archive, but the final design did not include these files. The functionality that this was supposed to provide was instead included in the extract.c file. The reason for the design change is that as I began to understand how to remove a file from the archive further, I came to the conclusion that removing a file was simply extracting all files except for the ones to be removed into a new archive, so the functionality was moved to the extract.c file.

1.6 proj3.c

The proj3.c file was to put all the functionality of the other files together into one to make one cohesive program. This file became myar.c in the final design. The design changes to this file were minimal. The only deviation was that the append multiple files function was implemented in the write.c file instead of in this one.

2 Challenges

The biggest challenge I faced completing this assignment was identifying the source of my bugs. I typically code C using an IDE because I find it much easier to use a GUI debugger and am not familiar with GDB. Because GDB was somewhate foreign to me, I had to take some time to figure out how to use GDB and how to get the same functionality out of it that I would have with an IDE. I also think I underestimated the assignment, so my design was not nearly as simple or robust as it could have been if I would have spent more time in the design consideration phase.

3 Work Log

1. **Revision 1** (Wed 10 Oct 2012 8:08 PM) 1.2 hours: Attempted to implement the functionality to append to an archive. Adding to an archive was semi-successful (the resulting archive was not compatible with ar).

2. **Revision 2** (Thu 11 Oct 2012 1:30 PM) 4.4 hours: Finished the append functionality and added in the read functionality. The contents of ar created archives were successfully printed out (both verbose and regular).

3. **Revision 3** (Fri 12 Oct 2012 7:22 PM) 2.4 hours: Built off the finished read and write functions to add extract functionality.

4. **Revision 4** (Sat 13 Oct 2012 4:11 PM) 3.8 hours: Built off the finished read and write functions to add delete functionality, which revealed bugs a write function that was also fixed.

5. **Revision 5** (Sun 14 Oct 2012 8:57 PM) 2.1 hours: Combined all the functions to create the finished program. Worked out several bugs in the delete function.

6. **Revision 6** (Mon 22 Oct 2012 2:00 PM) 0.5 hours: Updated the makefile to use more variables and clean all the output files of the latex pdf generation.

4 Additional Questions

1. I believe the main point of this assignment was to get comfortable using system calls in a Unix environment. It also directly related to everything that has been talked about in class such as file IO.

2. To ensure that my solution was correct, I split each type of operation into its own file so that I could test each one individually. Before the project was put together into one, I had three seperate entities that had their own main function. This allowed me to test each function in that file by passing in some input and comparing the actual output with the expected output. Once I was satisfied that all three entities were working correctly, I was then able to combine them into one and use the ar commmand with my program to verify that my solution was completely compatible with ar.

3. I learned that everything I needed to know about file IO in a Unix environment can be found in the man pages. I also learned that read/write could read less than the amount of bytes requested. In the past, I have worked with reading and writing from sockets, and I always assumed that if it succeeded it read/wrote the number of bytes requested. This realization taught me the importance of checking return values so that errors could be caught before they caused my program to crash.