# CS311 - FA13: Archive

Trevor Bramwell

October 22, 2013

## 1 Design

This assignment is to create a simple implementation of the UNIX utility *ar*. The utility will support the operations on archive files of appending, creating, extracting, listing and deleting. It will also provide a flag for verbose output and if time permits, a way to add modified files, after a delay, from a directory.

Since this *ar* revolves around the previously mentioned operations, I will be structuring the code in the same manner. Depending on the flag passed in, the program will call the corresponding function to perform the operation. A set of utility functions will be created as well to reduce code duplication, and improve code in a more self-documenting way.

When I start writing my program, I will begin with the user interface, that is the command line. Having the interface created first will make it easier for me to focus on the functionality of the program.

I will then walk though each operation that can happen document the steps it needs to take, i.e. open archive file, write file, etc. to flesh out what the utility functions might be. I will then focus on the single operation of *listing*, and use a pre-created archive file so as to reduce the number of concerns. From there I will itertively enable each flag until they all work and the program works as intended.

### 1.1 Versions

Up to 0.x GNU ar will be used for testing.

- 0.0 Tests (all flags, all flags w/verbose)
- 0.1 List (-t)
- 0.x Extract (-x)
- 0.x Append (-q)
- 0.x Quick Append (-A)
- 0.x Create Archive when using (-q, -A)
- 0.x Delete (-d)
- 1.0 Verbose (-v) - Change all printf statements to only execute when flag passed.
- 1.1 Timeout (-w)

### 1.2 Notes

Remove is an Extract w/Delete, Extract should take a flag to also remove the file.

Append takes the directory, or a list of files. Version that takes a directory will pass in all regular files in the directory to the version that takes a list.

### 1.3 Deviation

places your implementation deviated from this design

## 2 Challenge

any challenges you overcame in completing this assignment

## 3 Questions

What do you think the main point of this assignment is?

> The main point of this assignment is to get us reaquanted with the C programming language, and to learn UNIX system calls.

How did you ensure your solution was correct? Testing details, for instance. This section should be very thorough.

> I ensured my program was correct by writing a suite of interface tests. For example: I test for the listing functionality by having a pre-determined list of files in an archive, running my *ar*, and ensuring the returned list matches my pre-determined one.

What did you learn?

## 4 Work Log

a work log, detailing what you did when – this can fairly easily be created if you are using some form of revision control

# Instructions

Write a C program called myar.c. This program will illustrate the use of file I/O on UNIX by maintaining a UNIX archive library, in the standard archive format.

For this assignment, the following is the syntax your program must support:

myar [options] archive [member...]

where archive is the name of the archive file to be used, and options is one of the following options (all are silent unless verbose is specified):

-v iff specified with other options, print a verbose version of the output -q quickly append named files to archive -x extract named files -t print a concise table of contents of the archive -d delete named files from archive -A quickly append all "regular" files in the current directory (except the archive itself)

-w Extra credit: for a given timeout, add all modified files to the archive. (except the archive itself)

You must use getopt to parse the command line options (and only getopt – do not use getopt_long).

The archive file maintained must use exactly the standard format defined in /usr/inc1ude/ar.h, and in fact may be tested with archives created with the ar command. The options listed above are compatible with the options having the same name in the ar command. -A is a new option not in the usual ar command.

Notes:

For the -q command myar should create an archive file if it doesn't exist, using permissions "666". For the other commands myar reports an error if the archive does not exist, or is in the wrong format.

You will have to use the system calls stat and utime to properly deal with extracting and restoring the proper timestamps. Since the archive format only allows one timestamp, store the mtime and use it to restore both the atime and mtime. Permissions should also be restored to the original value, subject to umask limitation.

The -q and -A commands do not check to see if a file by the chosen name already exists. It simply appends the files to the end of the archive.

The -x and -d commands operate on the first file matched in the archive, without checking for further matches.

In the case of the -d option, you will have to build a new archive file to recover the space. Do this by unlinking the original file after it is opened, and creating a new archive with the original name.

You are required to handle multiple file names as arguments.

Since file I/O is expensive, do not make more than one pass through the archive file, an issue especially relevant to the multiple delete case.

Compilation will be done with icc and use (at minimum) the following flags: '-std=c99 -Wall'.

For the -w flag, the command will take as long as specified by the timeout argument. You should print out a status message upon adding a new file only if -v is specified. This may result in many different copies of the same file in the archive.

For extra credit, any time a file is added that already exists, remove the old copy from the archive, but ONLY if it is not the same. If identical, do not add the new file. Size is NOT a valid indication of isomorphism. You will need to use a more advanced technique.

## Writeup Instructions

Your write-up should include (at minimum) the following:

a design for your system, as well as places your implementation deviated from this design

a work log, detailing what you did when – this can fairly easily be created if you are using some form of revision control

any challenges you overcame in completing this assignment

answers to the following questions: What do you think the main point of this assignment is?

How did you ensure your solution was correct? Testing details, for instance. This section should be very thorough.

What did you learn?

## Deliverable

Things to turn in:

C source code write-up, as a tex document any support files necessary to compile your tex document or source a makefile to build both source code and your tex file

You are not allowed to use Lyx to create the document. You must create the document "by hand", instead.