1 System design

I originally was going to use getopt, but due to the non-POSIX nature of the command structure, I decided to manually interpret the user input. Obviously as there were user requirements for this assignment, the system design was outlined around that. The solution was generated by creating functions for each command line option, sketching out the likely algorithm to use, and divide functionality into reusable subfunctions as appropriate.

One of the critical design decisions was to ensure that instances of `struct ar_hdr` remain null terminated. Thus the static buffer-based `fix_*` functions for time, permissions, and generic strings were implemented. Headers are grabbed using an iterator (`get_next_header`), although I originally decided to construct a list of all headers this seemed to be more efficient overall.

The major subfunction (not related to archive integrity or conversion) that was able to be used multiple times was `delete_file`. If I were to rewrite this code, I would probably attempt to break down into functions further, however there were no major possible refactoring based on the current implementation.

2 Work log

```
Date:   Mon Feb 4 22:05:40 2013 -0800
Date:   Mon Feb 4 17:57:20 2013 -0800
    Fixed errors preventing writeup from compiling
Date:   Mon Feb 4 17:48:32 2013 -0800
    Switched to provided template, won't compile..
Date:   Mon Feb 4 17:23:21 2013 -0800
    Update to writeup
Date:   Mon Feb 4 17:05:21 2013 -0800
    Deletes
Date:   Mon Feb 4 17:02:40 2013 -0800
    Changed hierarchy, working on writeup
Date:   Mon Feb 4 11:03:13 2013 -0800
    Finished makefile
Date:   Mon Feb 4 10:41:22 2013 -0800
    Begin work on write up
Date:   Sun Feb 3 17:47:34 2013 -0800
    Everything happy apparently maybe hopefully
Date:   Sun Feb 3 16:26:18 2013 -0800
    Style, error checking fixes
Date:   Sat Feb 2 22:09:14 2013 -0800
    Finished extraction, finished main functions

    TODO:
    -expand and clean up error checking
    -extra credit function
    -extra credit duplicates
    -make file
    -header file?
    -LaTeX writeup
Date:   Sat Feb 2 21:05:28 2013 -0800
    Add error checking, rebased mistake
Date:   Sat Feb 2 20:12:43 2013 -0800
    Finished "append all" method
Date:   Sat Feb 2 15:36:05 2013 -0800
    Fix for delete func, added more test files
Date:   Sat Feb 2 13:56:24 2013 -0800
    Delete unnecessary function, fix for perms, adding
Date:   Fri Feb 1 22:56:25 2013 -0800
```

```
    Remove unnecessary define
Date:   Fri Feb 1 22:41:31 2013 -0800
    Fix for issue involving files starting at odd byte
Date:   Fri Feb 1 22:14:42 2013 -0800
    Added broken ar archive for testing
Date:   Fri Feb 1 22:10:30 2013 -0800
    Append option working (on some ar modes, need test)
Date:   Fri Feb 1 12:32:15 2013 -0800
    Delete function changed, still doesn't work...
Date:   Wed Jan 30 21:35:38 2013 -0800
    Sketched out file from archive delete logic
Date:   Wed Jan 30 17:19:11 2013 -0800
    Fix formatting issues associated with ar -tv
Date:   Wed Jan 30 08:11:33 2013 -0800
    Structure change (LaTeX template removed)

    Switching to McGrath's makefile based LaTeX approach.
Date:   Tue Jan 29 22:59:08 2013 -0800
    Got complex printing done (minus formatting)
Date:   Mon Jan 28 21:02:19 2013 -0800
    Got basic printing working, no lstat yet.
Date:   Sun Jan 27 19:47:48 2013 -0800
Date:   Sun Jan 27 19:44:29 2013 -0800
    ASSIGNMENT 2: begin work
```

3 Challenges

The biggest challenge was simply building an intuitive interface with rigorous error checking enabled. All system calls require error checking to be ran properly, so implementing this in a helpful, informative, and visually pleasing method was... difficult. I also had difficulty reporting errors in an informative fashion, instead of just puking and exiting. Before my next assignment begins, I intend to spend a lot of time brushing up on C error handling (especially in regards to a CLI program).

4 Questions

4.1 Main point of the assignment

The main point of the assignment was most likely to get everyone comfortable in C programming by providing a thorough refresher. Due to the complexity of this program, a lot of research and man page viewing was required to get up to speed, as well as a plethora of library functions and system calls.

4.2 Solution testing

Since we were essentially black boxing `ar`, my testing largely consisted of (1) does it meet assignment requirements for expected functionality and (2) do so in a way that is highly compatible with the existing `ar` utility. This meant ensuring that all archive were valid to `ar` after all operations performed by my implementation of it. Fortunately the archive is very human readable, so it was easy to tell when things weren't working correctly (and then fairly easy to fix using `gdb`).

4.3 I learned...

I have a reasonable amount of C and GNU/Linux experience, so what I mostly learned was how to access Linux system calls properly. I am familiar with the utilities that you interface with (such as `rm`) however I had not previously used their system call correspondents (e.g. `unlink`).