CS 350: Assignment 2
Doing Linux File I/O
Due Monday October 28, 2019 at 9PM

Write a C program on Linux 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.

Once compiled your program should run in a manner similar to the standard Linux command "ar". You may wish to look at the Linux man page for "ar" for insight, however, for this assignment the following is the syntax your program must support:

myar key afile name ...

where afile is the name of the archive file to be used, and key is one of the following options:

q	quickly append named files to archive
xo	extract named files with meta data
t	print a concise table of contents of the archive
tv	print a verbose 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)

The archive file maintained must use exactly the standard format defined in "/usr/include/ar.h", and in fact must be tested with archives created with the "ar" command. You must

```
#include <ar.h>
```

for the definition of struct ar_hdr. In addition, define the following

```
struct meta {
        char name[16]; //room for null
        int  mode;
        int size;
        int uid;
        int gid;
        time_t  mtime; // a time_t is a long
}
```

and write functions

```
    int fill_ar_hdr(char *filename, struct ar_hdr *hdr);

    int fill_meta( struct ar_hdr hdr, struct meta *meta);
```

The first of these "stat"s the filename, and fills in the ar_hdr. stat structures have binary fields, ar_hdr fields are all printable ascii, so conversions are necessary. This is used in adding members to the archive.

The second goes in the opposite direction, converting fields in the ar_hdr into binary for the meta structure. The meta structure is used when extracting members from the archive, and for the "tv" option.

The options listed above are compatible with the options having the same name in the "ar" command The "A" command is a new option not in the usual "ar" command.

Notes:

(1)

(2) This is an exercise in efficiently using read/write/lseek directly, and you cannot use the buffered I/O library (fopen, fread, putc, etc). You should be reading and writing ar_hdrs in a single system call. File contents is written in buffers sized by stat.

(3) You may assume all files are in the current directory, no filename is longer than 15 bytes, and no archive index will be attempted. You MUST NOT consult source code for other implementations of "ar".

(4) For the "q" command "myar" should create an archive file if it doesn't exist, using permissions "0666". For the other commands "myar" reports an error if the archive doesn't exist, or has the wrong format. (When testing the system ar on Linux use "qU" so that it includes the meta information).

(5) 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" if the "o" option is specified. Permissions should also be restored to the original value, subject to "umask" limitation.

(6) 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. They save only ordinary files, and skips symbolic links. This is different than what the system "q" does. Your programs will not be tested with "*.o" files, so no index should be created.

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

(8) 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.

(9) In case you do the multiple file option, note that 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.