Myar Archiving Program (due 9/27/2020)

Learning Objectives

Upon completion of this assignment, you should be able:

1. Be able to efficiently use low level fileio system calls
2. Understand and use the metadata that describes a file
3. Be able to interact with directory structures
4. Interact with system data structures compatibly (ar_hdr, archive format)

New mechanisms you will see and use include:

- system calls: read, write, lseek, stat, utime, unlink
- library routines: Directory Routines, String Routines (malloc, free only for buffer size from stat)
- umask

NAME

`myar` – Archiving Utility

SYNOPSIS

`myar` [qxotvd:A:] archive-file [file1 .....]

DESCRIPTION

myar operates on the archive-file listed adding, deleting, listing contents according to the operations selected, using the files specified as appropriate. Myar maintains compatibility with “ar” untility on Linux, with the exception of the “A” option which is an added option to myar, and any item mentioned in the notes below.

-q Quickly append the files specified to the archive-file
-x Extract the specified files from the archive-file
-o Used in combination with “x” restore the orginal permission and mtime
-t List the file names in the archive-file
-v Used in combination with “t” print the verbose info as “ar” does.
-d Delete the files specified from the archive-file
-A Quickly append all regular files in the directory modified more than N days ago

Exit Status

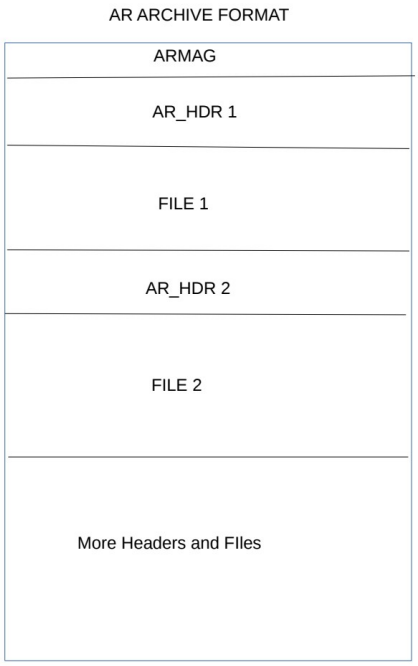
myar returns 0 on success and 1 on failure

ERROR MESSAGES

Same as in ar.

DISCUSSION

This is what an AR Archive Files looks like



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 and ar must be tested with archives created by myar. 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;
       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 "-t -v" option.

NOTES

- 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.

- You may assume all files are in the current directory, no filename is longer than 15 bytes, and no archive index will be attempted. Unlike “ar” you will not follow symbolic links. In the “A” option you skip symbolic links.
-
- The ar_hdr and the ARMAG have even size. ar_hdr’s must always be on an even boundary. This is maintained if the file being archived is even. However when archiving a file who size is odd, a padding byte is required after it to preserve ar_hdr allignmnent.

- You MUST NOT consult source code for other implementations of "ar

- Create archives with permission 0666 subject to umask. Extraction permission in the -x -o option is also subject to the umask. (Good news, this means there is nothing to do).

- When running the “ar” command on Ubuntu use “qU” instead of “q”. Ubuntu does not include the metadata without the “U”. This is a recent optimization justified by the typical use of “ar” to maintain binary libraries. Myar should always include the metadata.

- 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 newarchive 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 as many files as appear on the command line. 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.

HAND in myar.c, Makefile, any other files (not ar.h), README in a lab2 directory created in <yourid>/cs551