Lecture 10: Unix Libraries

Kenneth M. Anderson

Software Methods and Tools

CSCI 3308 - Fall Semester, 2003

Reuse in a Unix Environment

- Two commonly reused software objects in Unix environments
  - source code
  - object code
- Source code Reuse
  - Pro: Can modify to suit new context
  - Con: MUST modify to suit new context
- Object code Reuse
  - Pro: No compilation required; just header file and lib
  - Con: No ability to change functionality; Arch-specific

Libraries

- Unix Library
  - a collection of object files, used for some purpose
    - e.g. math libraries, graphics libraries, etc.
- Can be reused in other programs
  - The rules of marshalling (covered in last lecture) ensure that the compiler knows how to call the object code contained in the library
    - Remember that object code is architecture-specific

Creating a Library

- Compile .c files to create .o files
- Use the ar command to create a library from the .o files
  - The .o files are stored in the archive such that they can be extracted at a later time
  - This allows a linker to be smart about using the object code in libraries
    - e.g. only those functions used are placed in the linked executable

Example

- main.c

```
#include "main.h"
main(){
    subject();
    verb();
}
```

- s1.c

```
int subject() {
    printf("Jane ");
}
```

- s2.c

```
int subject(){
    printf("Ken ");
}
```

- v1.c

```
int verb() {
    printf("codes.");
}
```

- v2.c

```
int verb(){
    printf("debugs.");
}
```

- main.h

```
int subject();
int verb();
```


Example, continued

- First, compile the support files
  - g++ -c s1.c
  - g++ -c s2.c
  - g++ -c v1.c
  - g++ -c v2.c
- Second, create two different libraries
  - ar -r libWords1.a s1.o v1.o
  - ar -r libWords2.a s2.o v2.o
- This creates two separate libraries
  - libWords1.a and libWords2.a

Checking library contents

- ar -t libWords1.a
  - s1.o
  - v1.o
- ar -t libWords2.a
  - s2.o
  - v2.o
- strings libWords1.a
  - …
  - Jane
  - codes.
  - …
- strings libWords2.a
  - …
  - Ken
  - debugs.
  - ...

Example, continued

- Third, compile main
  - g++ -c main.c
- Fourth, link executable
  - g++ main.o -o main1 -lWords1
  - g++ main.o -o main2 -lWords2
- Fifth, run programs
  - main1 -> Jane codes.
  - main2 -> Ken debugs.

More info on ar command

- ar is the ARchive command
- It is similar to tar: Tape Archive
  - Both store multiple files as a single collection
  - ar focuses on storing .o files to create libraries
- The similarity ends there
  - the command flags and behavior of these commands are sometimes quite different

ar command syntax

ar (d|q|r|t) archive [files…]

- r - Replace
  - replace .o files in archive with specified files
- q - Quick append
  - append specified files to archive
- d - Delete
  - delete specified files from archive
- t - Table of Contents
  - print table of contents of archive
- Note: This is just a sample of ar's functionality; see the ar man page for more details

Using Unix Libraries

- In order to use a Unix library, a compiler needs to know the location of the library, the location of its include file, and its name
- Unix compilers (g++, gcc, and cc) have command flags that let you specify this information
  - `-I` Directory for include files (uppercase i)
  - `-L` Directory for Libraries
  - `-l` Name of library (lowercase L)

More on include directories

- Any source file that wants to make use of a library, must include its header file
- The `-I` flag specifies a directory name for this purpose
- When a compiler encounters a "#include" statement, it looks in the current directory and the directory specified by the `-I` flag for the file

More on Library directories

- The `-L` option specifies a directory where Unix libraries are stored
- When a linker needs to locate a library (in order to link it into an executable), the linker will look in the directory specified by the `-L` flag
- Note: you can have more than one `-L` and `-I` flags in a single command

More on Library names

- The `-l` flag (lowercase L) specifies the name of a Unix library
- The compiler assumes that all libraries begin with "lib" and end in ".a"
- As such, you write "`-lmath`" rather than "`-llibmath.a`"
  - The latter would cause the compiler to look for a file called `liblibmath.a.a`!

Note: Order is significant

- The order of `-l` flags is significant

```
g++ main.c -o main -lWords1 -lWords2
```

- produces
  - "Jane codes."
- The object code in Words2 is ignored because the linker found matches for subject() and verb() in Words1
- Swapping the libraries in the above command produces
  - "Ken debugs."

Brooks' Corner: Why Did The Tower of Babel Fail?

- Communication, (the lack of it)
  - This made it impossible to coordinate
- How do you communicate in large project teams?
  - Informally (telephone, e-mail), meetings, workbook
- Workbook
  - It is a structure placed on a project's documents
  - Why is it important? Technical prose lives a long time; best to get it structured formally from the beginning; it also helps with the distribution of information

More on the Workbook

- OS/360
  - *Each* programmer should see *all* the material
  - Each book was updated quickly (one-day)
  - Problem
    - The workbook grew to 5 feet thick!
      - They switched to microfiche
    - We need to take advantage of on-line artifacts, information management techniques like open hypermedia, information retrieval, and the WWW

Reducing communication paths

- Communication needs are reduced by
  - division of labor
  - specialization of function
- A tree structure often results from applying this principle
  - However this serves power structures better than communication (since communication between siblings is often needed)
  - So communication structure is often a network

Organizational Structure

- Brooks outlines
  - mission, producer, director, schedule, division of labor, and interfaces between the parts
- The new items are the producer and the director
  - producer: manages project and obtains resources
  - director: manages technical details
- Microsoft's program and product manager
  - former is director, latter does more marketing than Brooks specifies for producer but has some overlap