Week 8
More Linking

21 November 2018

CS 35L Lab 4

Jeremy Rotman

Announcements

- Assignment #7 is due Monday by 11:55pm
- Assignment #10 Presentations
  - **Email me to tell me what story you are choosing**
  - Here is the link to see what stories people have signed up for already
- Happy slightly early Thanksgiving!
  - To the few that did show up
  - As well as those that are already home, or away, for the holiday
    - Or was suddenly struck by an inexplicable and completely legitimate situation, which meant they could not make it to class today

Outline

- ➔ Creating static libraries
- ➔ ELF (sadly not the movie)
- ➔ Super secret fun things that everyone who isn’t here misses out on

Questions?

Creating Static Libraries

To create a static library, or to add additional object files to an existing static library, you can use a command like this

```
ar rcs my_library.a file1.o file2.o
```

This adds the object files, file1.o and file2.o to a library my_library.a, which is created if it does not already exist

Creating Shared Libraries

Simple example of compiling a shared library

```
gcc -fPIC -g -c -Wall a.c
gcc -fPIC -g -c -Wall b.c
gcc -shared -Wl,-rpath=$PWD -o libmystuff.so a.o b.o -ldl
```

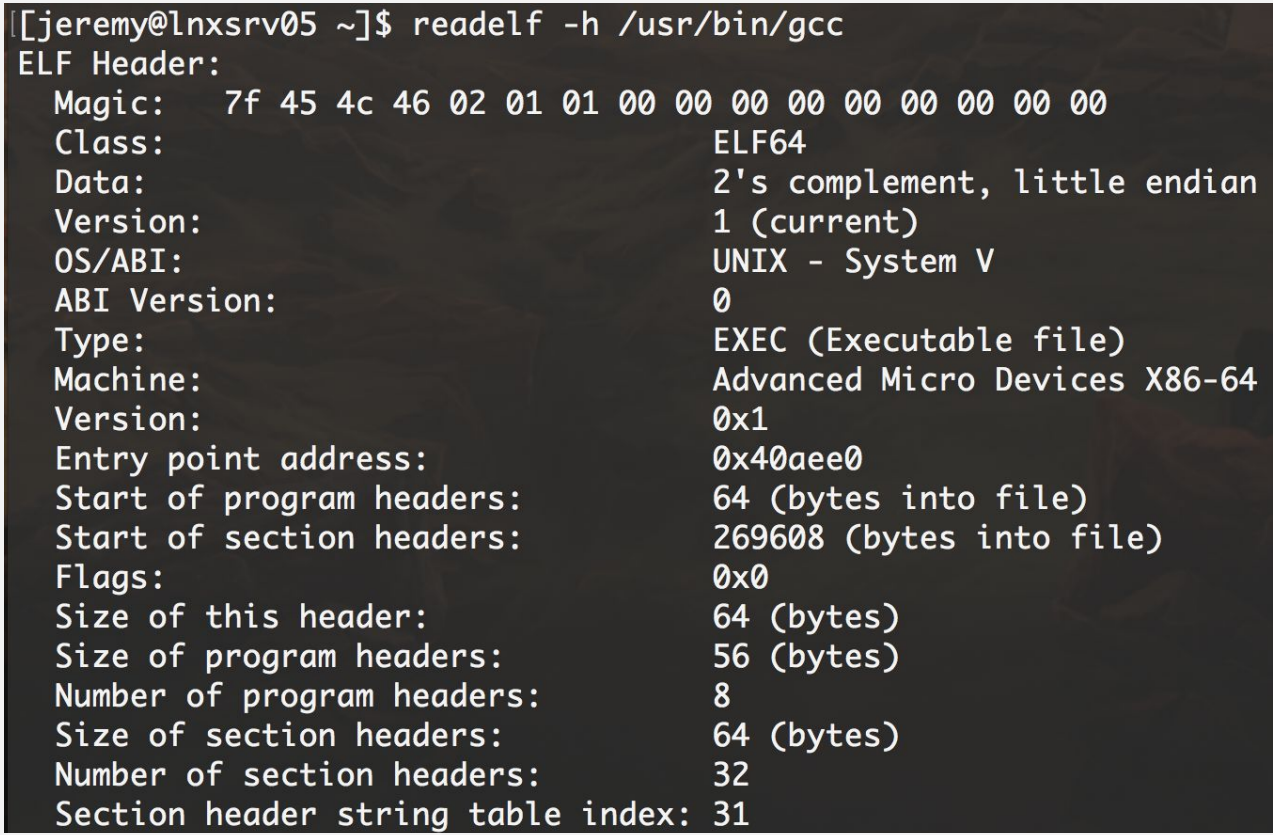
This adds the object files a.o and b.o to the shared library libmystuff.so

ELF files

- Executable and Linking Format file
  - This file type is used for binaries, libraries and core files
  - Extensions include:
    - none
    - .bin
    - .o
    - .so
    - And others

ELF headers

Every ELF file begins with an ELF header which lists information about the file



ELF Headers

- Data
  - Can define whether the data is stored as big endian or little endian
- Type
  - Defines what type of ELF file it is (4 possible values)
    - 1: REL (relocatable file)
    - 2: EXEC (executable file)
    - 3: DYN (shared object file)
    - 4: CORE (core file)
- Program headers
  - Describe how to create a process/memory image for runtime execution

ELF Program Headers

- INTERP defines what dynamic linker to use
  - Similar to the interpreter command in a bash script
    - #!/bin/bash
- This also defines how to map between section and segment

```
[jeremy@lnxsrv05 ~]$ readelf -l /usr/bin/gcc

Elf file type is EXEC (Executable file)
Entry point 0x40aee0
There are 8 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                 0x00000000000001c0 0x00000000000001c0  R E    8
  INTERP         0x0000000000000200 0x0000000000400200 0x0000000000400200
                 0x000000000000001c 0x000000000000001c  R      1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x000000000003f27c 0x000000000003f27c  R E    200000
  LOAD           0x000000000003f280 0x000000000063f280 0x000000000063f280
                 0x00000000000020a8 0x00000000000020a8  RW     200000
  DYNAMIC        0x000000000003f2a8 0x000000000063f2a8 0x000000000063f2a8
                 0x0000000000000190 0x0000000000000190  RW     8
  NOTE           0x000000000000021c 0x000000000040021c 0x000000000040021c
                 0x0000000000000044 0x0000000000000044  R      4
  GNU_EH_FRAME   0x000000000003d260 0x000000000043d260 0x000000000043d260
                 0x00000000000004e4 0x00000000000004e4  R      4
  GNU_STACK      0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     8

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .note.gnu.build-id .gnu.hash .dynsym .gnu.liblis
t .gnu.version .gnu.version_r .rela.dyn .rela.plt .init .plt .text .fini .rodata
 .eh_frame_hdr .eh_frame
   03     .ctors .dtors .jcr .dynamic .got .got.plt .data .dynbss .bss .dynstr .
gnu.conflict
   04     .dynamic
   05     .note.ABI-tag .note.gnu.build-id
   06     .eh_frame_hdr
   07
```

ELF Program Headers

- Sections
  - Exist before linking
  - Are in the object files
  - .data holds the initialized data
  - .text holds the executable code
- Segments
  - Exist after linking
  - Are in the executable files
  - One segment can hold multiple sections
  - Essentially instructions on how to map sections to virtual space
    - Using the mmap system call

Questions?