CS 354 - Machine Organization & Programming
Thursday, December 5, 2019

Final Exam - Monday December 16th, 5:05 - 7:05 PM

- **A - H Last/Family Name:** room **2**650 Humanities
- **I - R Last/Family Name:** room 105 Psychology
- **S - Z Last/Family Name:** room B130 Van Vleck
- UW ID required
- #2 pencils required
- closed book, no notes, no electronic devices (e.g., calculators, phones, watches)
- see "Final Exam" on course site Assignments for topics

Project p6 (4.5%): DUE at 10 pm on Saturday, December 14th

Homework hw8 (1.5%): Due TOMORROW at 10 pm on Friday, December 6th

Homework hw9 (1.5%): Assigned Monday

Last Time

Issues with Multiple Signals
Forward Declaration
Multifile Coding
Multifile Compilation
Makefiles

Today

Makefiles (from last time)
Relocatable Object Files
Static Linking
Linker Symbols
Course Feedback - **BRING YOUR DEVICES TO DO ONLINE COURSE EVALUATION**
Linker Symbol Table
Symbol Resolution

Next Time

Symbol Relocation, Loading
Read: B&O 7.7 - 7.9

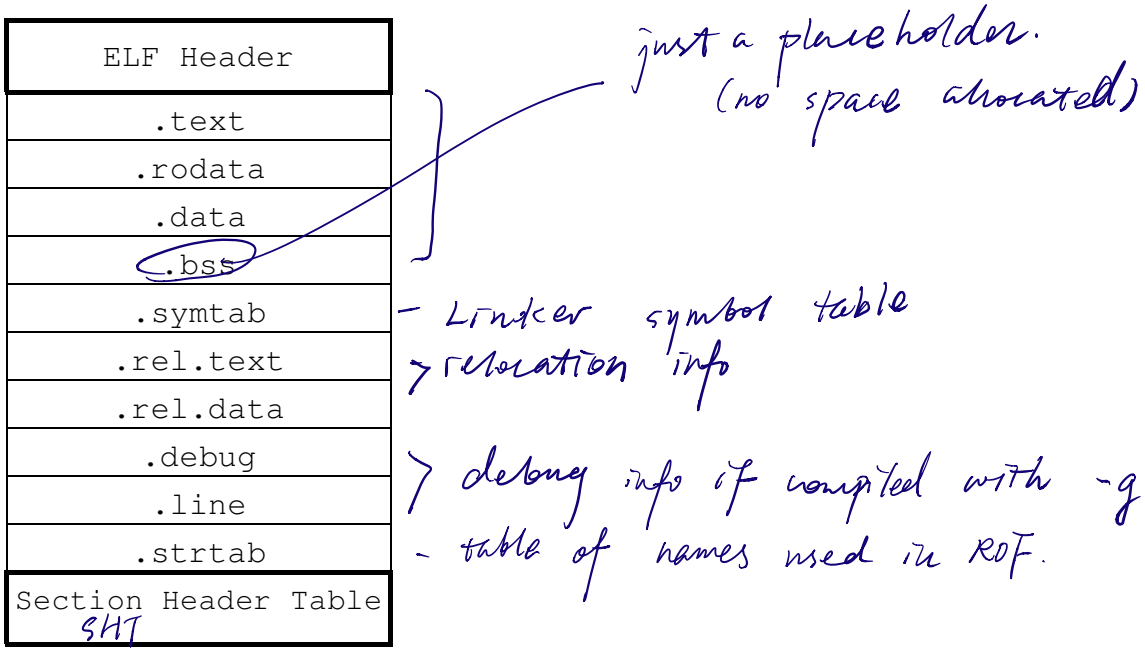
Relocatable Object Files (ROFs)

AKA object file

What? A *relocatable object* file is

- a file containing object code
- in format the linker can easily combine with other objectfile

Executable and Linkable Format (ELF) (diff sys → diff obj file)
what linux uses.



ELF Header contains gen info:
ELF HDR size, object file (ROF, SOF, EOF)
offset to SHT, size of SHT, num of events in SHT
also arch info, word size, byte ordering, machine type.

Section Header Table (SHT)
contains the location and size of each section.

Static Linking

What? *Static linking* generated a complete EOF with no identifiers remaining in code.

	static	vs.	dynamic
executable size:	larger		smaller
library code:	included		no, linked in during load or run times

How?

note: all (one translation are done (ee, as) needed only to combine ROF/SOF into EOF.

→ What issues arise from combining ROFs?

1. symbol need to be checked that they have exactly one definition.

2. symbol addrs need to be determined for the resulting EOF. Relocation

Making Things Private

→ Are functions and global variables only in a source file actually private if they're not in the correspondng header file? no!

They can still be accessed by adding declaration for them in your code.

→ How do you make them truly private?

declare global function & variable as "static".

Linker Symbols

What? *Linker symbols* are identifiers of variables & functions in a source file that are managed by Linker.

→ Which kinds of variables need linker symbols?

those that are allocated in the data seg

1. local variables
2. `static` local variables ✓
3. parameter variables
4. global variables ✓
5. `static` global variables ✓
6. `extern` global variables ✓

→ Which kinds of functions need linker symbols?

those that are used in one ROF but defined in another.

1. extern funcs
   linker connects extern func called in one ROF to its definition in another ROF.
2. Non-static func.
   Linker *might* need to connect func definition in this ROF to a func call in another ROF.

Linker Symbol Table

What? The linker *symbol table* is

- built by the assembler using symbols exported by the compiler.
- represented as array of ELF_symbol structures.

`ELF_Symbol` Data Members and their Use

`int name` byte offset in .strtab where the symbol name is formed.

`int value` symbol's addr

if ROF offset from beginning symbol's section.

if EOF virtual addr

`int size` number of bytes for mem alloc

`char type:4` data (object), funcs (FUNC), extern (notype)

`binding:4` mostly global, but sometimes local

`char section` index of the symbol's section in ROF.

Ndx

pseudo sections:

Assume section 1. TEXT 2. Ro data 3 data 4. BSS

ABS:olute symbols, should not be relocated

UND:efined external symbols reference in this ROS but defined in another

COM:mon symbols in .BSS (uninitialized data)

`value` now means alignment requirement.

`size` - - - minimum size. (need how many bytes).

Example

$readelf -s <object file name>

```
Num:  Value Size  Type    Bind    Ot Ndx  Name
1 - 7 not shown
8:    0     4     OBJECT  GLOBAL  0  3    bufp0
9:    0     0     NOTYPE  GLOBAL  0  UND  buf
10:   0     39    FUNC    GLOBAL  0  1    swap
11:   4     4     OBJECT  GLOBAL  0  COM  bufp1
```

Start then look here.

→ Is `bufp0` initialized? yes 3 is .data.

→ Was `buf` defined in the source file or declared `extern`?

→ What is the function's name? swap

→ What is the alignment and size of `bufp1`?

4 4

Symbol Resolution

What? *Symbol resolution*

- ensures that each symbol has exactly one defn.
- work is divided between compiler & linker.

Compiler's Resolution Work

resolves symbols in one SRC file at a time.

- locals checks that there is one defn within scope

  static locals also resolved by compiler.
  but also ensures that each static locals has a unique name for the linker.
- globals leaves for linker if non-static

❋ *If a symbol is not defined* (global), the compiler assumes it is defined in another SRC file.

Linker's Resolution Work

resolves symbols across multiple ROFs

- locals compiler handled these

  static locals also checked by compiler.
  but linker must relocate.
- globals linker check there's only on defn in all ROFs (non-static)
  and also relocate

❋ *If a symbol is not defined or is multiply defined* (global) then that is a linker error.