CS 354 - Machine Organization & Programming
Tuesday, December 10, 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 hw9 (1.5%): Due at 10 pm on Friday, December 13th

Homework hw10 (1.5%): DUE at 10 pm on Sunday, December 15th

sixteen 1/2pt questions, 2 attempts, 20 mins each

Last Time

Makefiles
Relocatable Object Files
Static Linking
Linker Symbols
Linker Symbol Table
Symbol Resolution

Today the last time!

Final Exam Information
Resolving Globals
Symbol Relocation
Executable Object File
Loading

Final Exam

The final is cumulative. Review your midterm exams.

Part I: 52 1-pt questions
Part II: 16 3-pt questions

Reference Page include in exam:

Powers of 2

$2^5 = 32$, $2^6 = 64$, $2^7 = 128$, $2^8 = 256$, $2^9 = 512$, $2^{10} = 1024$
$2^{10} = K$, $2^{20} = M$, $2^{30} = G$
$2^A \times 2^B = 2^{A+B}$
$2^A / 2^B = 2^{A-B}$

Hexadecimal Digits

$9_{16} = 9_{10} = 1001_2$
$A_{16} = 10_{10} = 1010_2$
$B_{16} = 11_{10} = 1011_2$
$C_{16} = 12_{10} = 1100_2$
$D_{16} = 13_{10} = 1101_2$
$E_{16} = 14_{10} = 1110_2$
$F_{16} = 15_{10} = 1111_2$

Registers

32 bit	16 bit	8 bit
`%eax`	`%ax`	`%ah, %al`
`%ecx`	`%cx`	`%ch, %cl`
`%edx`	`%dx`	`%bh, %bl`
`%ebx`	`%bx`	`%dh, %dl`
`%edi`	`%di`	
`%esi`	`%si`	
`%ebp`	`%bp`	
`%esp`	`%sp`	

32 bit registers: underlined are caller saved, others are callee saved

Assembly

Most instructions with two operands have the order: Source, Destination
e.g., `subl s,d` means d = d - s; `imull s,d` means d = d * s
Comparison (cmp) and test instructions have operand order: Source2, Source1
e.g., `cmpl s2,s1` means s1 - s2; `test s2,s1` means s1 & s2
Suffixes for set, jump, and conditional move instruction are:
e.g., `setns` means set not signed, `jl` means jump if less
general - `e`: equal, `ne`: not equal, `s`: signed, `ns`: not signed
unsigned - `b`: below, `be`: below or equal, `a`: above, `ae`: above or equal
signed - `l`: less, `le`: less or equal, `g`: greater, `ge`: greater or equal

Resolving Globals

Confusing Globals

```
main.c                  fun1.c                  fun2.c

int m;                  int m = 22;             int m;
int n = 11;             int n;                  extern int n;
short o;                int o;                  char o;

extern int x;           int x;                  static int x = 33;
int y;                  static int y = 33;      static int y;
static int z = 66;      static int z = 77;      int z;

//code continues...     //code continues...     //code continues...
```

Strong and Weak Symbols

strong:

weak:

→ Which code statements above correspond to strong symbols?

Rules for Resolving Globals

→ Which code statements above correspond to definitions?

1. Multiple strong symbols

2. Given one strong symbol and one or more weak symbols,

3.Given only weak symbols,

❉ *Use `extern` to clearly indicate when*

❉ *Use `static` to clearly indicate when*

Symbol Relocation

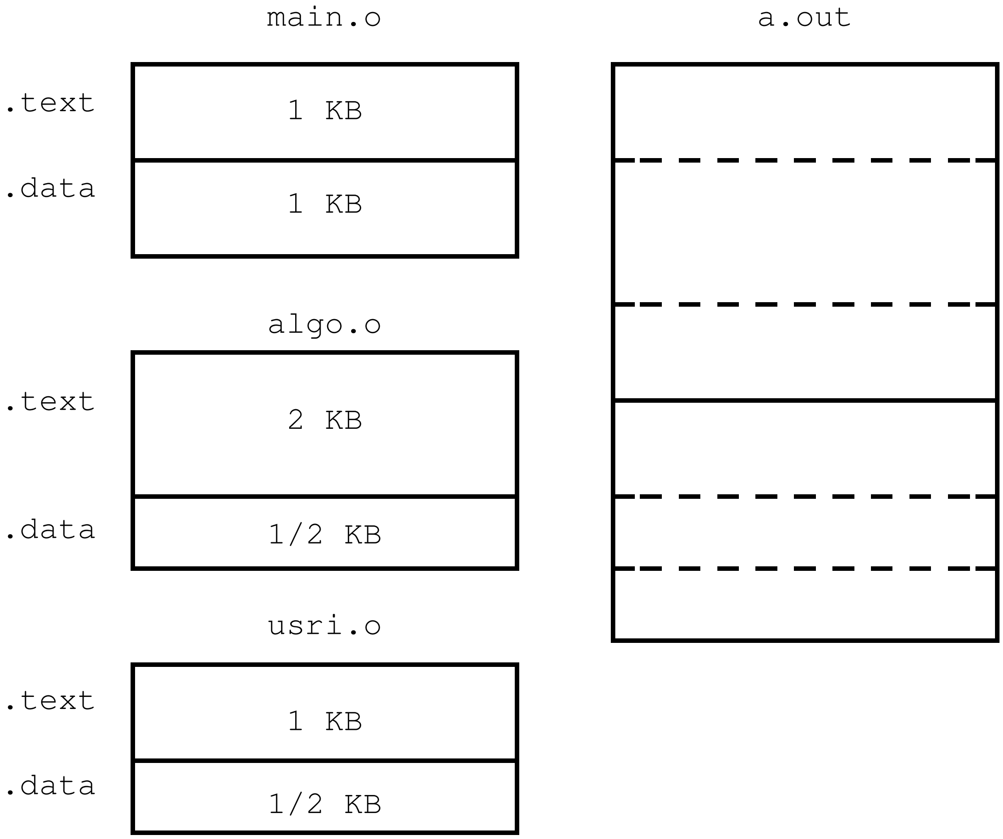
What? *Symbol relocation*

How?

1. Merges the same sections

2. Assigns virtual addresses

3. Updates symbol references



address =

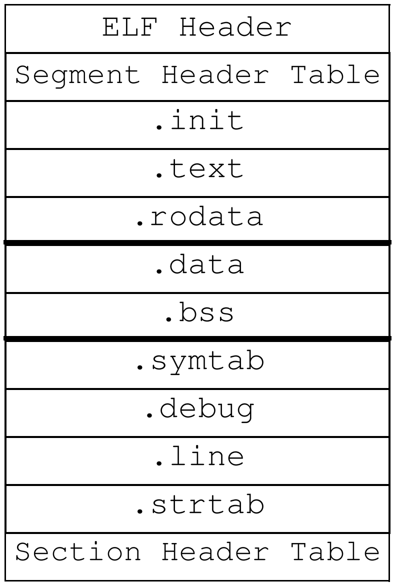
Excutable Object File (EOF)

What? An EOF, like an ROF, is

Executable and Linkable Format

ELF Header

+ Segment Header Table



ELF Header

Segment Header Table

.init

.text

.rodata

.data

.bss

.symtab

.debug

.line

.strtab

Section Header Table

→ Why aren't there relocation sections (.rel.text or .rel.data) in EOF?

➢ Why is the data segment's size in memory larger than its size in the EOF?

Loader

What? The *loader*

◆

◆

Loading

1.

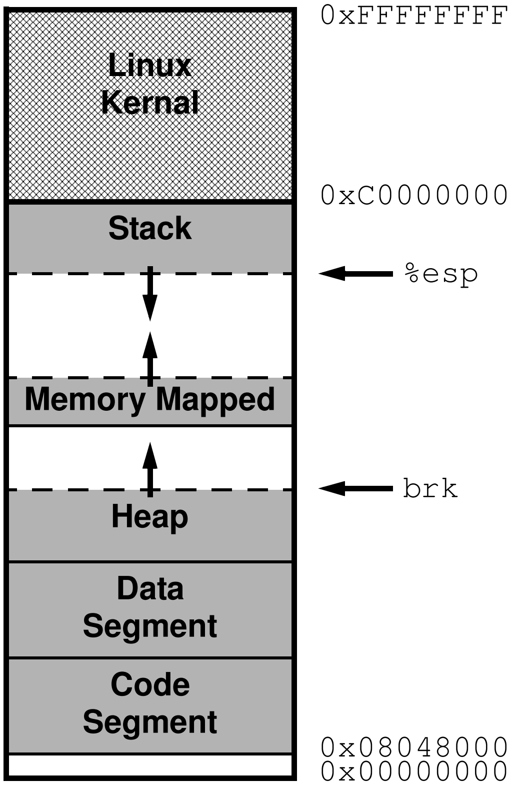
2.

Execution - the final story

1. shell

2. child process

3. loader



a.

b.

c.

d.

4. loader

```
call __libc_init_first

call _init

call atexit

call main

call _exit
```