outlineL28-w15T-student

Wednesday, December 14, 2022 4:01 PM

outlineL28-w15T-stu...

CS 354 - Machine Organization & Programming
Tuesday December 13th, 2022

Course Evals

https://aefis.wisc.edu Course: CS354 Instructor: DEPPELER

Final Exam -Wednesday Dec 21st, 7:25 PM - 9:25 PM

Your final exam room will be randomly assigned and sent to you via email.
You must attend the exam room as assigned in the email you receive.
Arrive early if possible with UW ID and #2 pencils. See additional exam info on course web site.

All office hours, TA consulting, and Peer Mentoring end on Wed December 14th

Homework hw8: DUE on or before Monday Dec 12

Homework hw9: DUE on or before Wednesday Dec 14 (NO LATE DAY)

Project p6: Due on last day of classes (NO LATE DAY or OOPS PERIOD). If you plan on getting help in labs, be sure to bring your own laptop in case there is no workstation available.

Last Week

Issues with Multiple Signals Forward Declaration Multifile Coding Multifile Compilation Makefiles	Relocatable Object Files Static Linking Linker Symbols Linker Symbol Table Symbol Resolution

This Week

Resolving Globals Symbol Relocation Executable Object File Loader What's next? take OS cs537 as soon as possible and Compilers cs536, too!	

Next Week: FINAL EXAM

Watch your email for your exam room assignment. All students must take the final exam in their assigned final exam room.

Students with accomodations should receive email with exam date/time/venue by 12/13.

Resolving Globals

Confusing Globals

main.c

```
int m;
int n = 11;
short o;

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

//code continues...
```

fun1.c

```
int m = 22;
int n;
int o;

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

//code continues...
```

fun2.c

```
extern int m;
extern int n;
char o;

static int x = 33;
static int y;
int z;

//code continues...
```

extern

extern

Linker in

DECL

DECL ONLY

Ok, linker picks

OK

OK

OK

Strong and Weak Symbols

strong: All function definitions & global variables

weak: Function declarations & uninitialized global variables

→ Which code statements above correspond to strong symbols?

Rules for Resolving Globals

→ Which code statements above correspond to definitions?

Recall: extern is only declaration. , The rest are definition

1. Multiple ~~strong~~ symbols In public global scope are not allowed unless you use linker option -z muldefs
2. Given one strong symbol and one or more weak symbols, Declare weak symbols as 'extern'
3. Given only weak symbols, Linker chooses first definition found

❋ *Use `extern` to ~~clearly~~ indicate when* A global var is decl. only

❋ *Use `static` to clearly ~~indicate~~ when* A global var is meant to be private

Symbol Relocation

What? *Symbol relocation* Combines ROFs/SOFs so that addresses can be determined for "remaining" linker symbols in code

How?

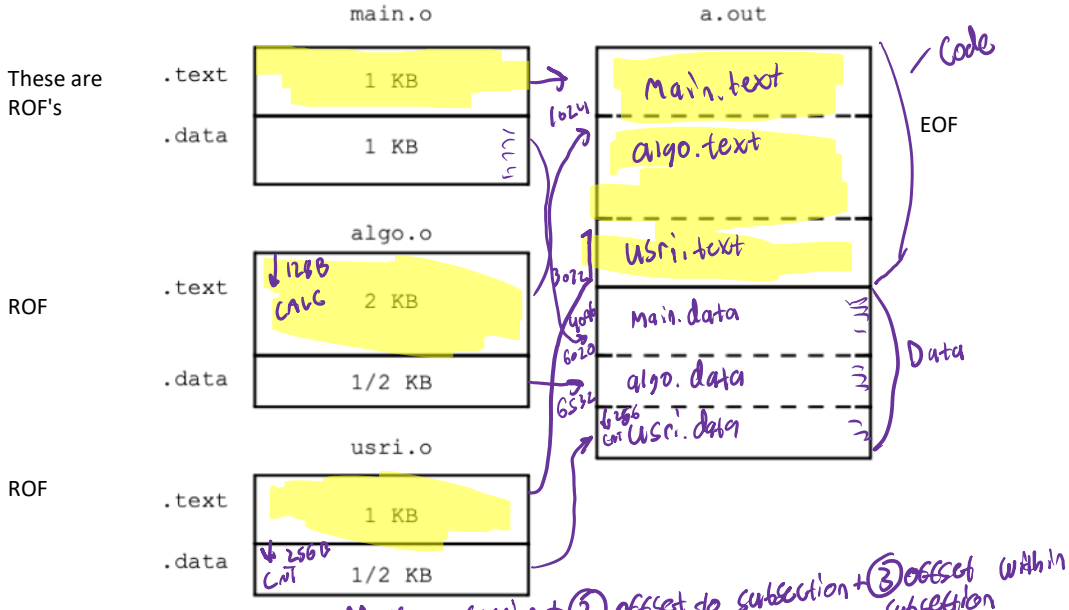
1. Merges the same sections Of ROFs into one aggregate for each type section

2. Assigns virtual addresses To each aggregate section and each symbol definition

3. Updates symbol references Listed in our ROFs reloc sections
   With virtual addresses

Example

Consider the .text and .data sections of 3 object files below combined into an executable:



readelf

Excutable Object File (EOF)

What? An EOF, like an ROF, is a file containing obj. code

It is a produced by the linker and can be loaded into the memory and ran.

Executable and Linkable Format

Same as ROF with extra sections and information.

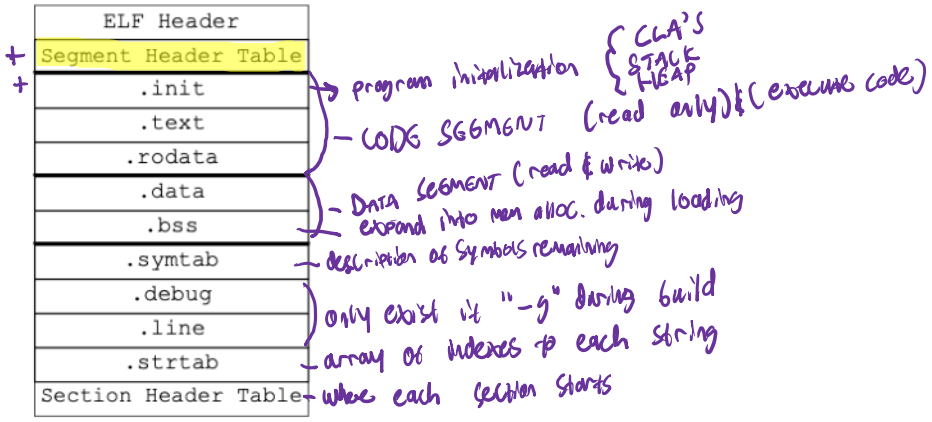
ELF Header

+Entry point - address of first instruction

+ Segment Header Table

Has information for each segment to be loaded into memory

- Has the offset for the file
- What the alignment is (what alignment rules it's following)
- Page size
- Size in the file and the size in the memory
- Runtime permissions (Read, Write , eXecute)



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

Everything has been relocated already - all symbols have been replaced with their address since we Assume static linking.

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

Because the .bss section does not have space allocated in EOF, but does take space in memory

Loader

What? The *loader*

- Kernel code that starts a program executing
- Can be invoked by any linux program using the 'execve' syscall.

Loading

1. Copies CODE and DATA segments from EOF into memory.
2. Starts the execution of the program by jumping to the ENTRY POINT

Execution - the final story

1. shell Creates a child process with a command called 'fork()'

2. child process invokes the Loader with execve (virtual environment)

3. loader creates a new runtime image

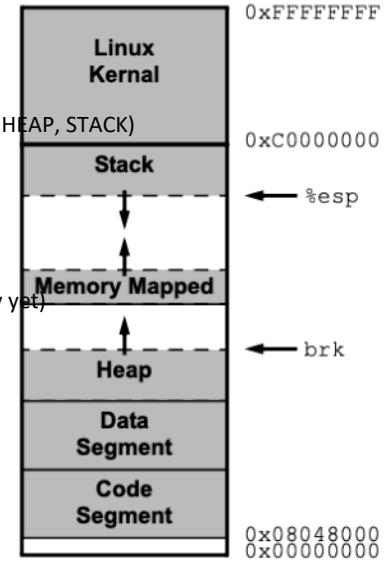
   a. Deletes the current segments
   (they were copied from the shell) (CODE, DATA, HEAP, STACK)

   b. Creates new segments (CODE, DATA)

   c. Heap and Stack created with an init size of 0

   d. EOF's CODE & DATA are mapped into page table
   In page-size chunks (but not copied into memory yet)

4. loader jumps to start address, causes a page fault




```
call __libc_init_first
call _init
call atexit
call main
call _exit
```

At least 2 questions on caching, at least 2 on heap
Always something on Stack vs. Heap 2D Arrays
Address arithmetic
Hits and misses on a stack

CS 354 - Machine Organization & Programming
Tuesday December 13th, 2022

Course Evals

https://aefis.wisc.edu Course: CS354 Instructor: DEPPELER

Final Exam -Wednesday Dec 21st, 7:25 PM - 9:25 PM

Your final exam room will be randomly assigned and sent to you via email.
You must attend the exam room as assigned in the email you receive.
Arrive early if possible with UW ID and #2 pencils. See additional exam info on course web site.

All office hours, TA consulting, and Peer Mentoring end on Wed December 14th

Homework hw8: DUE on or before Monday Dec 12

Homework hw9: DUE on or before Wednesday Dec 14 (NO LATE DAY)

Project p6: Due on last day of classes (NO LATE DAY or OOPS PERIOD). If you plan on getting help in labs, be sure to bring your own laptop in case there is no workstation available.

Last Week

Issues with Multiple Signals Forward Declaration Multifile Coding Multifile Compilation Makefiles	Relocatable Object Files Static Linking Linker Symbols Linker Symbol Table Symbol Resolution

This Week

Resolving Globals Symbol Relocation Executable Object File Loader What's next? take OS cs537 as soon as possible and Compilers cs536, too!	

Next Week: FINAL EXAM

Watch your email for your exam room assignment. All students must take the final exam in their assigned final exam room.

Students with accomodations should receive email with exam date/time/venue by 12/13.

Resolving Globals

Confusing Globals

main.c

```
int m;
int n = 11;
short o;

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

//code continues...
```

fun1.c

```
int m = 22;
int n;
int o;

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

//code continues...
```

fun2.c

```
int m;
extern int n;
char o;

static int x = 33;
static int y;
int z;

//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?

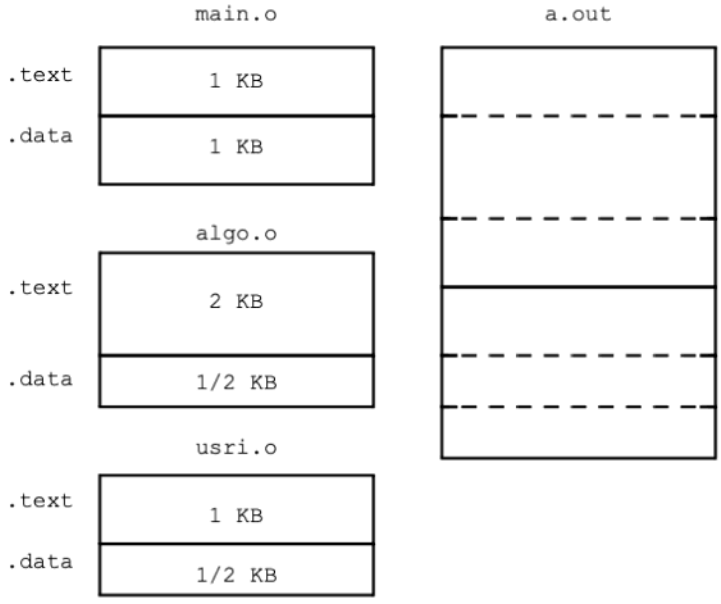
1. Merges the same sections

2. Assigns virtual addresses

3. Updates symbol references

Example

Consider the .text and .data sections of 3 object files below combined into an executable:



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*

-
-

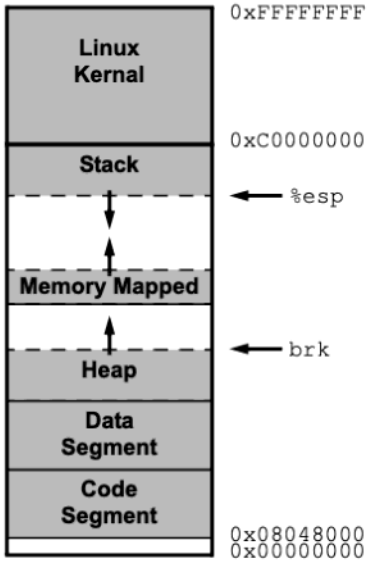
Loading

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
```



CS 354 - Machine Organization & Programming
Tuesday December 13th, 2022

Course Evals

https://aefis.wisc.edu Course: CS354 Instructor: DEPPELER

Final Exam -Wednesday Dec 21st, 7:25 PM - 9:25 PM

Your final exam room will be randomly assigned and sent to you via email.
You must attend the exam room as assigned in the email you receive.
Arrive early if possible with UW ID and #2 pencils. See additional exam info on course web site.

All office hours, TA consulting, and Peer Mentoring end on Wed December 14th

Homework hw8: DUE on or before Monday Dec 12

Homework hw9: DUE on or before Wednesday Dec 14 (NO LATE DAY)

Project p6: Due on last day of classes (NO LATE DAY or OOPS PERIOD). If you plan on getting help in labs, be sure to bring your own laptop in case there is no workstation available.

Last Week

Issues with Multiple Signals Forward Declaration Multifile Coding Multifile Compilation Makefiles	Relocatable Object Files Static Linking Linker Symbols Linker Symbol Table Symbol Resolution

This Week

Resolving Globals Symbol Relocation Executable Object File Loader What's next? take OS cs537 as soon as possible and Compilers cs536, too!	

Next Week: FINAL EXAM

Watch your email for your exam room assignment. All students must take the final exam in their assigned final exam room.

Students with accomodations should receive email with exam date/time/venue by 12/13.

Resolving Globals

Confusing Globals

main.c

```
int m;
int n = 11;
short o;

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

//code continues...
```

fun1.c

```
int m = 22;
int n;
int o;

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

//code continues...
```

fun2.c

```
int m;
extern int n;
char o;

static int x = 33;
static int y;
int z;

//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?

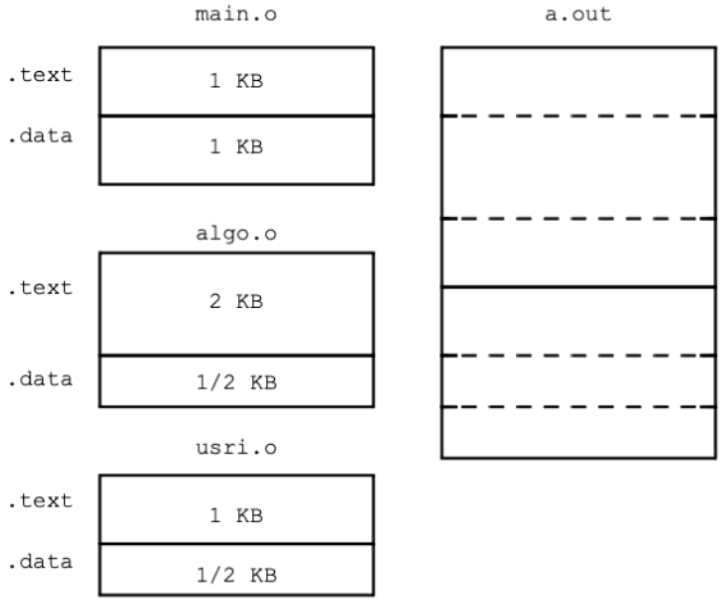
1. Merges the same sections

2. Assigns virtual addresses

3. Updates symbol references

Example

Consider the .text and .data sections of 3 object files below combined into an executable:



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
```