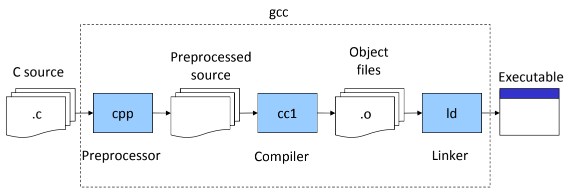
CS 449 – Executables and Linking

Jonathan Misurda
jmisurda@cs.pitt.edu

Compiler



Executables

- What do we need to store?
  - Code
  - Data
  - More?
- Agree on a common format (much like with ID3 tags)

Older Executable Formats

- a.out (Assembler OUTput)
  - Oldest UNIX format
  - No longer commonly used
- COFF (Common Object File Format)
  - Older UNIX Format
  - No longer commonly used

Modern Executable Formats

- PE (Portable Executable)
  - Based on COFF
  - Used in 32- and 64-bit Windows
- ELF (Executable and Linkable Format)
  - Linux/UNIX
- Mach-O file
  - Mac

a.out

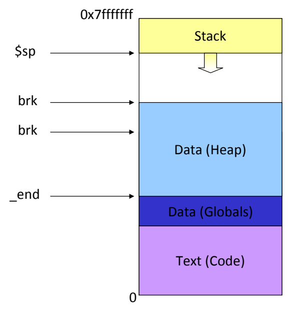
- exec header
- text segment
- data segment
- text relocations
- data relocations
- symbol table
- string table

Header

- Every a.out formatted binary file begins with an exec structure:

```
struct exec {
   unsigned long   a_midmag; //magic number
   unsigned long   a_text;
   unsigned long   a_data;
   unsigned long   a_bss;
   unsigned long   a_syms;
   unsigned long   a_entry;
   unsigned long   a_trsize;
   unsigned long   a_drsize;
};
```

Process's Address Space



Libraries

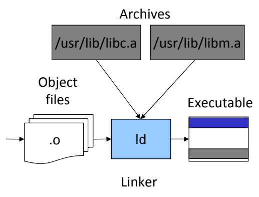
- Not all code in a program is what you wrote
- Use code that others have written in your own program
- How to include this code in your address space?

Linking

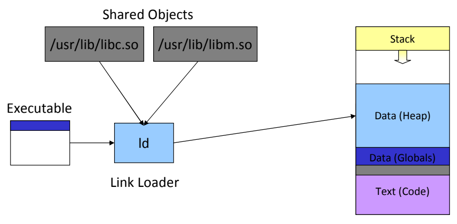
- Static Linking
  - Copy code into executable at compile time
  - Done by linker
- Dynamic Linking
  - Copy code into Address Space at load time or later
  - Done by link loader

Static Linking

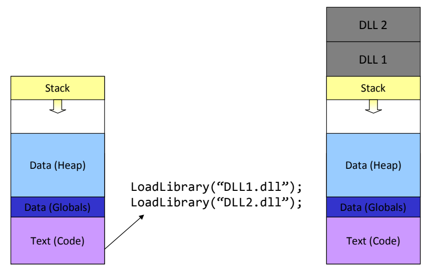
```
#include <stdio.h>
#include <math.h>

int main() {
   printf("The sqrt of 9 is %f\n", sqrt(9));
   return 0;
}
```



Dynamic Linking



Dynamic Loading



Function Pointers

- How do we call a function when we can't be sure what address it's loaded at?
- Need a level of indirection
- Use a function pointer

Function Pointers in C

```
#include <stdio.h>

int f(int x) {
   return x;
}

int main() {
   int (*g)(int x);

   g = f;
   printf("%d\n",g(3));
   return 0;
}
```