Linker

CS 217

Linker

- Combines multiple object files into a single executable file (**a.out**)
    - **a.out** is also an ELF file
    - also called a "link editor" or "linker/loader"
- Two main functions
    - relocation: assembler assumes each **.o** file begins at address 0, but when multiple object files are linked together some addresses must be relocated
    - resolution: all external references must be resolved; this also involves searching libraries to find any undefined symbols

---

Unix **ld** Command

- Implicit (called by compiler)
    - **gcc foo.c bar.c**
- Explicit (supports separate compilation)
    - **gcc –c foo.c**
    - **gcc –c bar.c**
    - **gcc foo.o bar.o**

  alternatively
    - **ld /usr/lib/crt0.o foo.o bar.o –lc –lm**
    - (do **"gcc –v foo.o bar.o"** to see full ugliness)

Example

main.c   f.c
```
static int a;       int b;
extern int b;       f(int a, int b) {
int c[10];              return a + b;
main() {            }
   f(a, b);         g(int a, int b) {
   g(a, b);             return a – b;
}                   }
```

Example (cont)

Then run…
```
% gcc –c main.c    % gcc –c f.c
% nm main.o        % nm f.o

00000050 b _a      00000004 C _b
         U _b      00000000 T _f
00000028 C _c      0000000c T _g
         U _f
         U _g
00000000 T _main
```

Relocation

- Concatenate `.o` files to output `a.out`
```
% gcc main.o f.o
% nm a.out
00004000 d __DYNAMIC
000040c8 b _a
00004098 B _b
000040a0 B _c
00004098 D _edata
000040d0 B _end
00004090 D _environ
000024c4 T _etext
000022e0 T _f
000022ec T _g
00002290 T _main
```

Symbol Resolution

- Linking must resolve <u>all</u> symbols
  ```
  % gcc main.o
  ld: Undefined symbol
    _b
    _f
    _g
  ```
- Linking must define each symbol <u>once</u>
  ```
  % cp f.o g.o
  % gcc main.o f.o g.o
  ld: g.o: _f multiply defined
  g.o: _g multiply defined
  ```

Resolution (cont)

- Linkers <u>do not type check</u> across modules
- Linkers combine all modules, even if some are not referenced
- Undefined symbols drive library searching
  - only those modules that define one of the unresolved symbols are linked
  - search for a particular symbol stops as soon as a match is found
- Linkers can build an object file incrementally
  ```
  % ld –o ab.o a.o b.o
  % ld –o a.out ab.o c.o
  ```

Possible Edits

- Resolution only
  ```
  R_SPARC_WDISP30     call
  R_SPARC_WDISP22     ba, be, bne,…
  ```
- Resolution and/or Relocation
  ```
  R_SPARC_22          sethi
  R_SPARC_13          add, ld, and,…
  R_SPARC_5           sll, srl, sra
  R_SPARC_HI22        sethi
  R_SPARC_LO10        or
  ```

Symbol Table

- Executable files contain a symbol table
  - useful for debugging
- Takes up space; remove by "stripping"

```
% gcc main.o f.o
% ls -l a.out
-rwxrwxr-x  llp  13872  a.out
% strip a.out
% ls -l a.out
-rwxrwxr-x  llp   3256  a.out
% nm a.out
nm: a.out: no symbols
```

Fall 2001   10

Advanced Topics

- Dynamic Linking
  - complete linking step at execution time
  - supports dynamic libraries
- Shared Objects
  - text segment shared by multiple processes
  - requires less physical memory

Fall 2001   11