Linking

Annalise Tarhan

September 25, 2020

7 Homework Problems

7.6 Fill in the table with information about each symbol that is defined and referenced in swap.o.

Symbol	.symtab entry?	Symbol type	Module where defined	Section
buf	yes	extern	m.o	.data
bufp0	yes	global	swap.o	.data
bufp1	yes	local	swap.o	.bss
swap	yes	global	swap.o	.text
temp	no			
incr	yes	local	swap.o	.text
count	yes	local	swap.o	.bss

7.7 Without changing any variable names, modify bar5.c so that foo5.c prints the correct values of x and y.

```
static double x;
void f()
{
        x = -0.0;
}
```

7.8 For each example, indicate how the linker would resolve references to the multiply-defined symbol.

7.8.1

REF(main.1)→DEF(main.1)

REF(main.2)→DEF(main.2)

7.8.2

REF(x.1)→DEF(UNKNOWN)

REF(x.2)→DEF(UNKNOWN)

7.8.3

REF(x.1)→DEF(ERROR)

REF(x.2)→DEF(ERROR)

7.9 Explain why foo6 prints the string 0x48 and terminates normally even though the function never initializes the variable main.

The variable main is declared twice, first as a function and second as an uninitialized char. The function is considered strong and the char weak, so when the linker needs to choose one to print, it chooses the function, whose value, or location, is apparently 0x48.

7.10 For each of the following dependency chains, show the minimal command line that will allow the static linker to resolve all symbol references.

7.10.1 p.o → libx.a → p.o

gcc p.o libx.a

7.10.2 p.o → libx.a → liby.a AND liby.a → libx.a

gcc p.o libx.a liby.a libx.a

7.10.3 p.o → libx.a → liby.a → libz.a AND liby.a → libx.a → libz.a

gcc p.o libx.a liby.a libx.z libz.a

7.11 The program header indicates that the data segment occupies 0x230 bytes in memory, but only the first 0x228 bytes come from the sections of the executable file. What causes this discrepancy?

The final eight bytes are reserved for .bss data that will be initialized to zero at run time.

7.12 Consider the call to swap in object file m.o with the given relocation entry.

7.12.1 Suppose the linker relocates .text in m.o to address 0x4004e0 and swap to address 0x4004f8. Then what is the value of the relocated reference to swap in the callq instruction?

reference pointer = symbol address + entry addend - reference address
reference address = section address + entry offset

ref_address = 0x4004e0 + 0xa = 0x4004ea
ref_pointer = 0x4004f8 - 0x4 - 0x4004ea = 0xa

The value of the relocated reference is 0xA.

7.12.2 Suppose the linker relocates .text in m.o to address 0x4004d0 and swap to address 0x400500. Then what is the value of the relocated reference to swap in the callq instruction?

ref_address = 0x4004d0 + 0xa = 0x4004da
ref_pointer = 0x400500 - 0x4 - 0x4004da = 0x22

The value of the relocated reference is 0x22.

7.13 For the following questions, use the described tools to manipulate object files.

7.13.1 How many object files are contained in the versions of libc.a and libm.a on your system?

The standard C library on OSX is part of libSystem.dylib, which is itself an object file.

7.13.2 Does gcc -0g produce different executable code than gcc -0g -g?

Yes. Using diff on the output of each command on a test file confirms that the binary files differ. I couldn't spot any difference between the objdumps, though, so I have no idea what those differences might be.

7.13.3 What shared libraries does the GCC driver on your system use?

Using otool -L on libSystem.B.dylib shows 36 shared libraries.