Homework #3 **M1522.000800 System Programming**

Sample Solution

Question 1

Symbol Resolution

Consider the following program, which consists of two object modules:

```
/* foo.c */
void p2(void);

int main()
{
    p2();
    return 0;
}
```

```
/* bar.c */
#include <stdio.h>

char main;

void p2()
{
    printf("0x%x\n", main);
}
```

When this program is compiled and executed on a Linux system, it prints the string "0x55\n" and terminates normally, even though p2 never initializes variable main. Can you explain this?

=> There is a multiply-defined global symbol **main**. In **foo.c**, **main** is allocated **.text** section, and In **bar.c**, **main** would be allocated **.data** section. However, compiler does only care strong symbol with initialization. In this case, **foo.c**'s main symbol is strong than **bar.c**'s one. And, **p2()** print **main**'s address which is allocated in **.text** section. That's why **main** has a value without initialization.

Question 2

Linker & Reference Resolution

Let **a** and **b** denote object modules or static libraries in the current directory, and let **a** → **b** denote that **a** depends on **b,** in the sense that **b** defines a symbol that is referenced by **a**. For each of the following scenarios, show the minimal command line (i.e., one with the least number of file object file and library arguments) that will allow the static linker to resolve all symbol references.

A. p.o → libx.a → p.o.
B. p.o → libx.a → liby.a **&** liby.a → libx.a
C. p.o → libx.a → liby.a → libz.a **&** liby.a → libx.a → libz.a

=> **A.** *gcc -static p.c ./libx.a*
=> **B.** *gcc -static p.c ./libx.a ./liby.a ./libx.a*
=> **C.** *gcc -static p.c ./libx.a ./liby.a ./libx.a ./libz.a*

Question 3

Relocating Symbol References

The example program consists of two source files, *main.c* and *swap.c*.

```
/* main.c */
void swap();

int buf[2] = {1, 2};

int main()
{
    swap();
    return 0;
}
```

```
/* swap.c */
extern int buf[];

int *bufp0 = &buf[0];
int *bufp1;

void swap()
{
    int temp;

    bufp1 = &buf[1];
    temp = *bufp0;
    *bufp0 = *bufp1;
    *bufp1 = temp;
}
```

The *swap* routine in the following plots (Relocated **.text** section & **.data** section) contains five relocated references.

```
    .text
 1 080483b4  <main>:
 2 80483b4:      55                          push   %ebp
 3 80483b5:      89 e5                       mov    %esp, %ebp
 4 80483b7:      83 ec 08                    sub    $0x8, %esp
 5 80483ba:      e8 09 00 00 00              call   80483c8 <swap>              swap();
 6 80483bf:      31 c0                       xor    %eax, %eax
 7 80483c1:      89 ec                       mov    %ebp, %esp
 8 80483c4:      5d                          pop    %ebp
 9 80483c5:      c3                          ret
10 80483c5:      90                          nop
11 80483c5:      90                          nop
12 80483c5:      90                          nop

13 080483c8  <swap>:
14 80483c8:      55                          push   %ebp
15 80483c9:      8b 15 5c 94 04 08           mov    0x804945c, %edx             Get *bufp0
16 80483cf:      a1 58 94 04 08              mov    0x8049458, %eax             Get buf[1]
17 80483d4:      89 e5                       mov    %esp, %ebp
18 80483d6:      c7 05 48 95 04 08 58        movl   $0x8049458,0x8049548        bufp1 = &buf[1]
19 80483dd:      94 04 08
20 80483e0:      89 ec                       mov    %ebp, %esp
21 80483e2:      8b 0a                       mov    (%edx), %ecx
22 80483e4:      89 02                       mov    %eax, (%edx)
23 80483e6:      a1 48 95 04 08              mov    0x8049548, %eax             Get *bufp1
24 80483eb:      89 08                       mov    %ecx, (%eax)
25 80483ed:      5d                          pop    %ebp
26 80483ee:      c3                          ret
```

```
   .data
 1 08049454  <buf>:
 2 8049454:      01 00 00 00 02 00 00 00

 3 0804945c  <bufp0>:
 4 804945c:      54 94 04 08                                 Relocated!
```

For each relocated reference, indicate its line number in the upper relocated section, its run-time memory address, and its value in the table below.

Line # in the relocated section	Address	Value
15 <bufp0>	**0x80483cb**	**0x804945c**
16 <buf[1]>	**0x80483d0**	**0x8049458**
18 <bufp1>	**0x80483d8**	**0x8049548**
18 <buf[1]>	**0x80483dc**	**0x8049458**
23 <bufp1>	**0x80483e7**	**0x8049548**

To do you will need the original code and relocation entries in the *swap.o* module:

```
 1 00000000  <swap>:
 2       0:      55                          push   %ebp
 3       1:      8b 15 00 00 00 00           mov    0x0, %edx            get *bufp0=&buf[0]
 4                                           3: R_386_32    bufp0        relocation entry
 5       7:      a1 04 00 00 00              mov    0x4, %eax            get buf[1]
 6                                           8: R_386_32    buf          relocation entry
 7       c:      89 e5                       mov    %esp, %ebp
 8       e:      c7 05 00 00 00 00 04        movl   $0x4, 0x0            bufp1 = &buf[1]
 9      15:      00 00 00
10                                          10: R_386_32    bufp1        relocation entry
11                                          14: R_386_32    buf          relocation entry
12      18:      89 ec                       mov    %ebp, %esp
13      1a:      8b 0a                       mov    (%edx), %ecx         temp = buf[0];
14      1c:      89 02                       mov    %eax, (%edx)         buf[0]=buf[1];
15      1e:      a1 00 00 00 00              mov    0x0, %eax            get *bufp1=&buf[1]
16                                          1f: R_386_32    bufp1        relocation entry
17      23:      89 08                       mov    %ecx, (%eax)         buf[1]=temp;
18      25:      5d                          pop    %ebp
19      26:      c3                          ret
```