Problem 1

Read-only code segment

1 LOAD off 0x00000000 vaddr 0x08048000 paddr 0x08048000 align 2**12

2 filesz 0x00000448 memsz 0x00000448 flags r-x

Read/write data segment

3 LOAD off 0x00000448 vaddr 0x08049448 paddr 0x08049448 align 2**12

4 filesz 0x000000e8 memsz 0x00000104 flags rw-

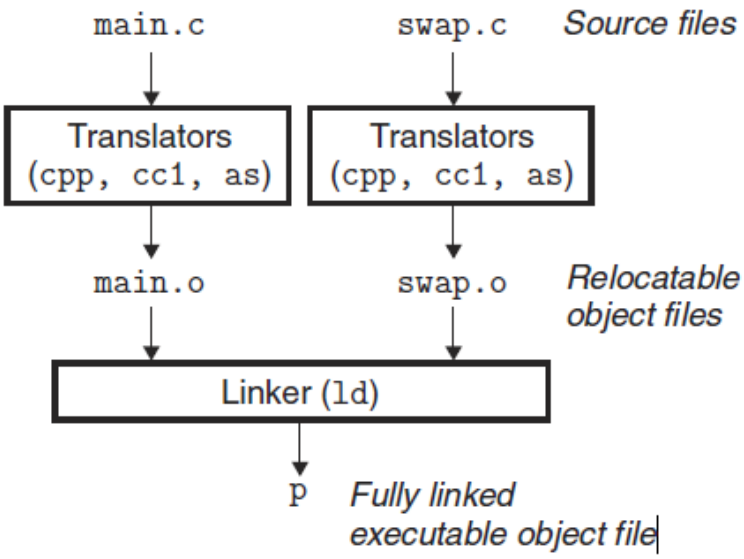
The segment header indicates that the data segment occupies 0x104 bytes in memory. However, only the first 0xe8 bytes of these come from the sections of the executable file. What causes this discrepancy?

因为该段中，0xe8 个字节来自于文件中.data 节，剩余下来的字节对应于运行时将被初始化为 0 的.bss 数据。

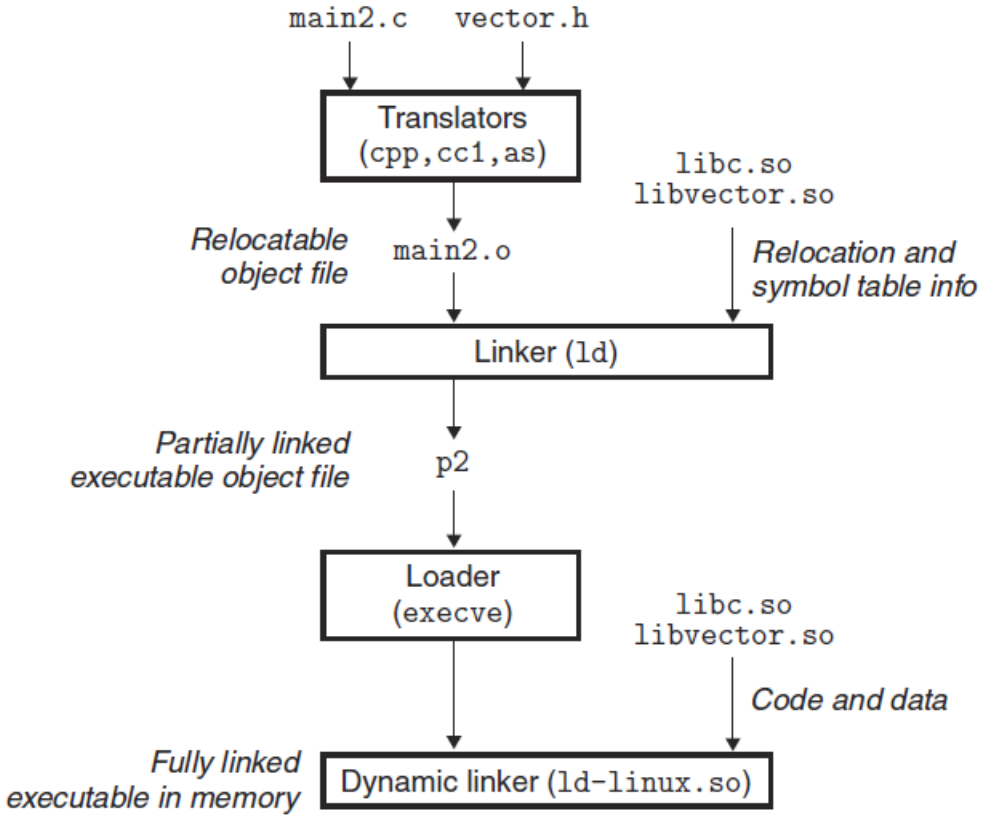
Problem 2

Express the difference between the procedure of static link and the procedure of dynamic link (with shared library). You can draw a picture to show that.

Static link:



Dynamic link:



Problem 3

1) Draw a Linux run-time memory image to show the location of GOT, PLT and shared library.
2) Express the function of PLT and GOT. And how to use them?
3) When the GOT is generated, and when the items of it are relocated?(Two situations: PIC Data References and PIC Function Calls)

1)

内核虚拟存储器
用户栈
共享库(shared library)
堆
读/写数据段 GOT（.data 中）
只读的代码和数据 PLT（位于.text 中）

2）GOT 作用是把位置无关的地址重定位到绝对地址；PLT 的作用是把位置独立的函数调用重定向到绝对位置。
PIC 数据引用时，通过位置无关代码，使程序跳转到 GOT 表中的相应条目，通过 GOT 间接引用每个全局变量。
PIC 函数调用时，将被调用函数绑定到 PLT 表中的相应条目中，并从条目的第一条指令开始执行，从而得到绝对地址。

3）GOT 表由静态链接器生成，在动态链接时由动态链接器重定位。

Problem 4

Suppose we have **main.c** and a shared library **foo.so**. In main.c, we called a function"**void flag()** "which is in the **foo.so**.

Complete the following Codes in **main.c** to call the **flag()**.

```
#include <stdio.h>
#inlcude <stdlib.h>
#inlcude <dlfcn.h>

Int main()
{
    // Your codes here.
    void *handle;
    void (*flag)();
    char *error;

    handle = dlopen("./foo.so", RTLD_LAZY);
    if(!handle){
        fprintf(stderr,"%s\n",dlerror());
        exit(1);
    }

    flag = dlsym(handle,"flag");
    If((error=dlerror()) != NULL){
        fprintf(stderr,"%s\n",error);
        exit(1);
    }


    flag();

    return 0;
}
```