计算机系统原理

课后作业

要求:

* 请在本周**日**23:00之前完成。
* 如果遇到在不适合在word中输入完成的题目，可以拍照后粘贴到word文档中。
* 完成后将word文档压缩后上传BB平台，上传的文档格式为：学号\_姓名.rar 。

**1. 简答题**

（1）引入链接的好处是什么？

**链接带来的好处1：模块化**

**1.一个程序可以分成很多源程序文件**

**2.可构建公共函数库，如数学库，标准C库等**

**链接带来的好处2：效率高**

**1.时间上，可分开编译只需重新编译修改的源程序文件，然后重新链接**

**2.空间上，无需包含共享库所有代码源文件中无需包含共享库函数的源码，只**

**要直接调用即可可执行文件和运行时的内存中只需包含所调用函数的代码而不**

**需要包含整个共享库**

（2）可重定位目标文件和可执行目标文件的主要差别是什么？

**1.ELF头中字段e\_entr y给出系统将控制权转移到的起始的虚拟地址（入口**

**点），即执行程序时第一条指令的地址。而在可重定位文件中，此字段为0。**

**2.多了一个.init节，其中定义了一个init函数，用于可执行目标文件开始执**

**行时的初始化工作。**

**3.少了两个.rel节，因为可执行目标文件中的指令和数据己被重定位，故可去**

**掉用于重定位的节。**

**4.多了一个程序头表，也称段头表(cpp)s)eg ment header table)，它是一个**

**结构数组。**

（3）可执行目标文件中有哪两种可装入段？哪些节组合成只读代码段？哪些节组合成可读写数据段？

第一可装入段对应第**0x00000~0x004d3**字节（包括ELF头、程序头

表、.init、.text和.rodata节），映射到虚拟地址0x8048000开始长度

为**0x4d4**

字节的区域，按**0x1000=212=4K字节对齐，**具有只读/执行权限（Flg= RE），

是**只读代码段**（read-only code）。

第二可装入段对应第0x000f0c开始长度为

0x108字节的.data节，映射到虚拟地址**0x8049f0c**开始的长度为**0x110**字节

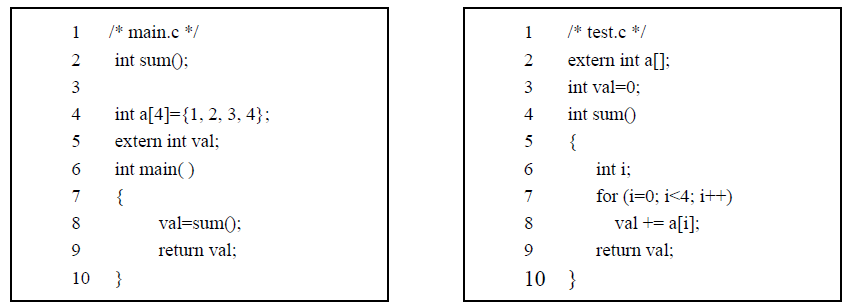
的存储区域，在0x110=272字节的存储区中，前0x108=264字节用于.data节内

容初始化，而后面272-2 64=8个字节对应.bss节，初始化为0，该段按0x1000=4KB

对齐，具有可读可写权限（Flg= RW），因此，它是一个**可读写数据段**（read/write

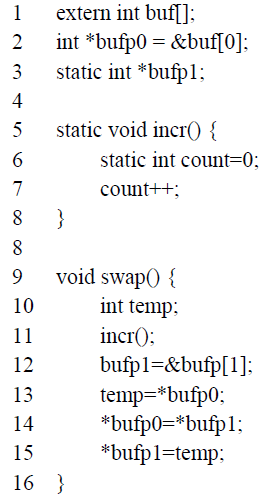
data segment）

2. 假设一个C语言程序有两个源文件：main.c 和test.c，它们的内容如图4.23 所示。对于编译生成的可重定位目标文件 test.o，填写下表中各符号的情况，说明每个符号是否出现在test.o的符号表（.symtab 节）中，如果是的话，定义该符号的模块是main.o 还是test.o、该符号的类型是全局、外部还是本地符号、该符号出现在test.o 中的哪个节（.text、.data 或.bss）



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 符号 | 是否在 test.o 的符号表中 | 定义模块 | 符号类型 | 节 |
| a | **在** | **main.o** | **global** | **.data** |
| val | **在** | **test.o** | **global** | **.data** |
| sum | **在** | **test.o** | **global** | **.text** |
| i | **不在** |  |  |  |

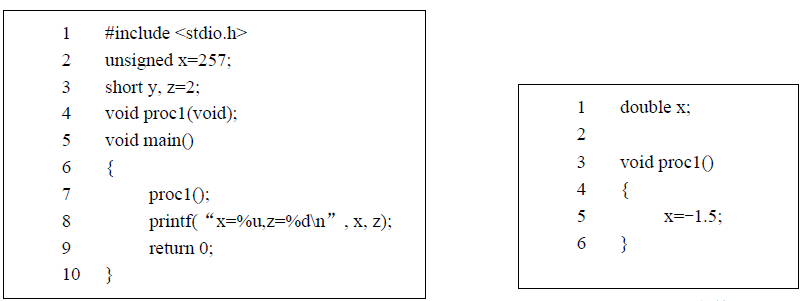
3. 假设一个C 语言程序有两个源文件：main.c 和swap.c，其中，main.c 的内容如图4-7a 所示，而swap.c的内容如下：



对于编译生成的可重定位目标文件swap.o，填写下表中各符号的情况，说明每个符号是否出现在swap.o的符号表（.symtab 节）中，如果是的话，定义该符号的模块是main.o 还是swap.o、该符号的类型是全局、外部还是本地符号、该符号出现在swap.o 中的哪个节（.text、.data 或.bss）。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 符号 | 是否在 swap.o 的符号表中 | 定义模块 | 符号类型 | 节 |
| buf | **在** | **main.o** | **global** | **.data** |
| bufp0 | **在** | **swap.o** | **global** | **.data** |
| bufp1 | **在** | **swap.o** | **local** | **.bss** |
| incr | **在** | **swap.o** | **local** | **.text** |
| count | **在** | **swap.o** | **local** | **.data** |
| swap | **在** | **swap.o** | **global** | **.text** |
| temp | **不在** |  |  |  |

4. 假设一个C 语言程序有两个源文件：main.c 和proc1.c，它们的内容如图所示。



回答下列问题。

1. 在上述两个文件中出现的符号哪些是强符号？哪些是弱符号？

**强符号**

**main.c中的proc1**

**proc1.c中的proc1**

**弱符合**

**main.c中的x,y,z,main**

**proc1.c中的x**

1. 程序执行后打印的结果是什么？请分别画出执行第7 行的proc1()函数调用前、后，在地址&x 和&z 中存放的内容。

**打印结果：x= 0,z=0**

**调用前:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **&z** | **02** | **00** | **--** | **--** |
| **&x** | **01** | **01** | **00** | **00** |

**调用后:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **&z** | **00** | **00** | **F8** | **BF** |
| **&x** | **00** | **00** | **00** | **00** |

（3）修改文件proc1，使得main.c 能输出正确的结果（即x=257, z=2）。要求修改时不能改变任何变量的数据类型和名字。

**将**

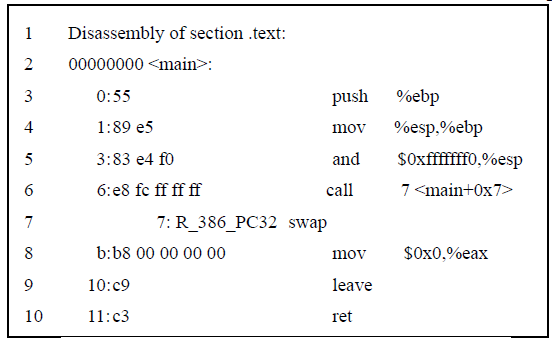
**double x**

**改为**

**static double x**

5. 给出了图4.7a 所示的main 源代码对应的main.o 中.text 节和.rel.text 节的内容，图中显示其.text节中有一处需重定位。假定链接后main 函数代码起始地址是0x8048386，紧跟在main 后的是swap函数的代码，且首地址按4 字节边界对齐。

要求根据对图4.15 的分析，指出main.o的.text 节中需重定位的符号名、相对于.text 节起始位置的位移、所在指令行号、重定位类型、重定位前的内容、重定位后的内容，并给出重定位值的计算过程。



**main.o的.text 节中需重定位的符号名:swap**

**由第6行可知，相对于.text节的位移为7**

**所在指令行号为6**

**重定位类型为PC相对地址方式**

**重定位前内容为0xFFFFFFFC 为-4**

**main函数占18字节，起始地址为0x8048386，因为swap紧贴main后面，所以swap起始地址为0x8048398**

**重定位值计算公式**

**ADDR(r-sym)-(ADDR(.text)+r\_offest-init)**

**=0x8048386-((0x8048386+7)-(-4))=7**

**所以重定位后内容为0x07000000 为7**

没搞懂，真tm难