## 2.

1. 通过链接将多个 C 语言源程序模块组合起来生成一个可执行目标文件。  
   转换过程为：源程序—预处理—编译—汇编—链接—完全可执行的目标文件
2. 链接带来的好处有：  
   ① 模块化：一个程序可以分成很多源程序文件，可构建公共函数库  
   ② 效率高：时间上,可分开编译；空间上,无需包含共享库所有代码，用到只需要调用即可
3. 可重定位目标文件和可执行目标文件的差别如下：  
   ① ELF头中的字段e\_entry的值不一样  
   ② 可执行目标文件后多出.init节和.fini节  
   ③ 可执行目标文件中少了.rel.txt节和.rel.data节等重定位信息节  
   ④ 可执行目标文件少了程序头表
4. 符号解析和重定位
5. .txt：目标代码部分  
   .rodata：只读数据  
   .data：已初始化的全局变量  
   .bss：未初始化的全局变量
6. （与可重定位目标文件大致相同）  
   .txt：目标代码部分  
   .rodata：只读数据  
   .data：已初始化的全局变量  
   .bss：未初始化的全局变量
7. ① 只读代码段和可读写数据段  
   ② 只读代码段：ELF头、程序头表、.init节、.fini节、.txt节、.rodata节  
   ③ 可读写数据段：.data节、.bss节
8. 程序头表
9. 简化了链接器的设计和实现
10. 静态链接将库函数合并包含在可执行文件中；动态链接库只有当程序被执行的时候才加载库函数

## 3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 符号 | 是否在test.o表中 | 定义模块 | 符号类型 | 节 |
| a | 是 | main.o | 外部 | .data |
| val | 是 | test.o | 全局 | .data |
| sum | 是 | test.o | 全局 | .text |
| i | 否 |  |  |  |

## 4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 符号 | 是否在test.o表中 | 定义模块 | 符号类型 | 节 |
| buf | 是 | main.o | 外部 | .data |
| bufp0 | 是 | swap.o | 全局 | .data |
| bufp1 | 是 | swap.o | 本地 | .bss |
| incr | 是 | swap.o | 本地 | .text |
| count | 是 | swap.o | 本地 | .data |
| swap | 是 | swap.o | 全局 | .text |
| temp | 否 |  |  |  |

## 5.

1. main.c：强符号：x,y,main，弱符号：y,proc1;  
   proc1.c：强符号：proc1，弱符号：x
2. 打印前：  
   &z：02 00  
   &x：01 01 00 00  
   打印后：  
   &z：00 00 F8 BF  
   &x：00 00 00 00  
   更改后打印结果应为：x=0,z=0
3. 将proc1.c的第一行改为static double x;