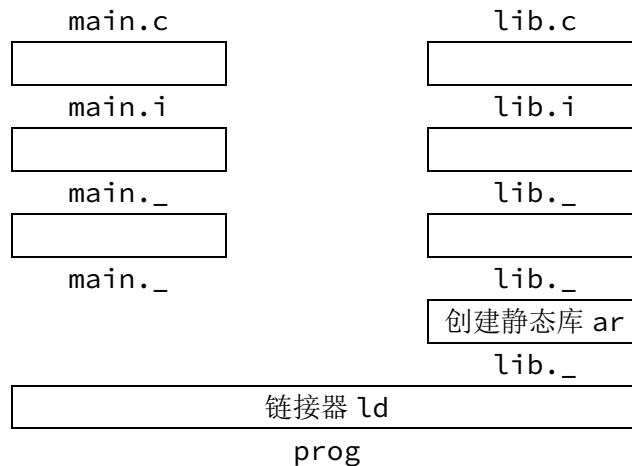


1. 对于可执行文件和加载过程，以下说法正确的是：

- () `_start` 函数是程序的入口点。
 () ASLR 不会影响代码段和数据段间的相对偏移，这样位置无关代码才能正确使用。

2. 以下展示了一个典型的链接过程，请将正确的过程补充完整。

- A. 汇编器 `as` B. 预处理器 `cpp` C. 编译器 `gcc`
 D. `*.a` E. `*.s` F. `*.o`



3. 圈出以下两个 C 程序中出现的符号。

main.c	count.c
<pre>#include <stdio.h> int buf[] = {1, 2}; void swap(); int main() { swap(); printf("buf:%d,%d\n", buf[0], buf[1]); return 0; }</pre>	<pre>extern int buf[]; int x; void swap() { int temp; temp = buf[0]; buf[0] = buf[1]; buf[1] = temp; return; }</pre>

4. 对于静态链接，判断以下说法是否正确

- () 链接时，链接器会拷贝静态库(.a)中的所有模块(.o)
 () 链接时，链接器只会从每个模块(.o)中拷贝出被用到的函数。
 () 链接时，如果所有的输入文件都是.o 或.c 文件，那么任意交换输入文件的顺序，都不会影响链接是否成功。
 () 链接时，通过合理地安排静态库和模块的顺序，每个静态库都可以在命令中出现至多一次。

5. 有下面两个程序。将他们先分别编译为.o 文件，再链接为可执行文件。

main.c	count.c
#include <stdio.h>	____B____

<pre> ____A____ int foo(int n) { static int ans = 0; ans = ans + x; return n + ans; } int bar(int n); void op(void) { x = x + 1; } int main() { for (int i = 0; i < 3; i++) { int a1 = foo(0); int a2 = bar(0); op(); printf("%d %d ", a1, a2); } return 0; } </pre>	<pre> int bar(int n) { static int ans = 0; ans = ans + x; return n + ans; } </pre>
---	--

(1) 当 A 处为 **int** x = 1; B 处为 **int** x; 时, 完成下表。如果某个变量不在符号表中, 那么在名字那一栏打×; 如果它在符号表中的名字含有随机数字, 那么请用不同的四位数字区分多个不同的符号。对于局部符号, 不需要填最后一栏。

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.c	x			
	bar			
	ans			
count.c	x			
	bar			
	ans			

程序能够链接成功吗? 如果可以, 程序的运行结果是什么? 如果不可以, 链接器报什么错?

(2) 当 A 处为 **static int** x = 1; , B 处为 **static int** x = 1; 时, 完成下表。

文件名	变量名	在符号表中的名字	是局部符号吗?	是强符号吗?
main.c	x			
	bar			
	ans			
count.c	x			
	bar			
	ans			

程序能够链接成功吗? 如果可以, 程序的运行结果是什么? 如果不可以, 链接器报什么错?

(3) 当 A 处为 **int** x = 1; , B 处为 **int** x = 1; 时。程序能够链接成功吗? 如果可以, 程序的运行结果是什么? 如果不可以, 链接器报什么错?

6. 在 gcc-7 编译系统下, 以下的两个文件能够顺利编译并被执行。在 x86-64 机器上, 若某次运行时得到输出 0xffffffff3, 请你判断这个值产生自?

f1.c	f2.c
<pre>void p2(void); int main() { p2(); return 0; }</pre>	<pre>#include <stdio.h> int main; void p2() { printf("0x%x\n", main); }</pre>

- A. 垃圾值
 B. main 函数汇编地址的最低字节按有符号补齐的结果
 C. main 函数汇编地址的最高字节按有符号补齐的结果
 D. main 函数汇编的第一个字节按有符号补齐的结果
7. 已知 x86-64 汇编指令 ret 的十六进制机器码为 0xc3。如果在一台现代 Intel x86 机器上使用 gcc 编译 foo.c 和 bar.c 得到可执行文件 a.out, 再执行它, 则会遇到如下哪种情况?

foo.c	bar.c
<pre>void foo(void); int main(){ foo(); return 0; }</pre>	<pre>int foo = 0xc3;</pre>

8. 有如下 C 代码

```
#define k 100
long foo(long n);
long bar(long n) {
    static long ans = 0;
    long acc = 0;
    for (int i = 0; i < n; i++) {
        ans += i;
        acc += ans * n;
    }
    return ans + acc;
}
long t;
static long y;
extern long z;
int main() {
    long x;
    myScanf("%ld%ld%ld", &x, &y, &z);
    myPrintf("%ld %ld\n", foo(x + y + t), bar(z + k));
    return 0;
}
```

采用命令 `gcc test.c -c -Og -no-pie -fno-pie` 与 `readelf -a test.o > t.txt` 后得到解析文件。

t.txt 中的部分节头部表信息如下：

节头:				
[号]	名称	类型	地址	偏移量
[1]	.text	PROGBITS	0000000000000000	00000040
[3]	.data	PROGBITS	0000000000000000	000000ff
[4]	.bss	NOBITS	0000000000000000	00000100
[5]	.rodata.str1.1	PROGBITS	0000000000000000	00000100
[10]	.symtab	SYMTAB	0000000000000000	00000190

t.txt 中的部分符号表如下：

Num	Size	Type	Bind	Vis	Ndx	Name
5	-----	OBJECT	-----	DEFAULT	-----	ans.1797
7	-----	OBJECT	-----	DEFAULT	-----	y
11	52	FUNC	GLOBAL	DEFAULT	-----	bar
12	139	FUNC	GLOBAL	DEFAULT	-----	main
13	0	NOTYPE	-----	DEFAULT	-----	z
15	-----	OBJECT	-----	DEFAULT	-----	t

(1) 除了上述已经列出的符号外，判断下列名字是否在符号表中。

名称	k	ans	acc	foo	y.????	x	n
在/不在							

(2) 补全上述符号表中漏掉的信息。其中 Bind 可以是 LOCAL 或者 GLOBAL，Ndx 可以是表示节头标号的数字，也可以是 UND (undefined) 或 COM (common)。

(3) 字符串 "%ld %ld\n" 位于哪个节中？

- A. .bss B. .data C. .rodata D. .text

(4) 假设在全局区定义 `long A[1000000]`。那么在 test.o 中，.bss 节占用的空间为 _____ 字节。