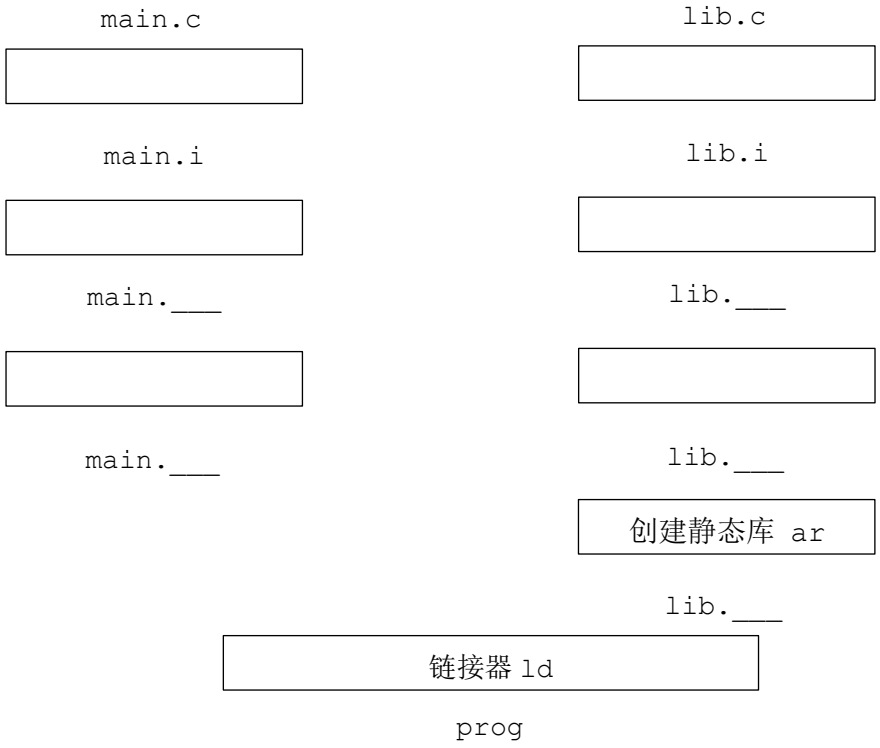
ICS 第七章

1．下图为一个典型的编译过程。将正确的过程填上，并补充缺失的拓展名。

A．汇编器 as　　B．预处理器 cpp　　C．编译器 ccl

D．*.a　　E．*.s　　F．*.o



【答】BCA、EFD

2．判断下面关于静态链接的说法是否正确。

(1) (　　)链接时，链接器会拷贝静态库(.a)中的所有模块(.o)

(2) (　　)链接时，链接器只会从每个模块(.o)中拷贝出被用到的函数。

(3) (　　)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。

(4) (　　)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。

【答】(1)错，只会拷贝被用到的模块。(2)错，会把所有函数拷贝出来。(3)对。(4)错。考虑下面这个极端的例子。foo.o 中定义了 p5()并引用了 p0()；对于所有的 i=1,2,3,4,5，bar.o 中定义了 *pi-1*()并引用了 *pi*()。Q135.a 包含了 bar1,bar3,bar5,R24.a 包含了 bar2,bar4。输入文件为 Q135.a,R24.a 与 foo.o。可以证明，gcc foo.o Q135.a R24.a Q135.a R24.a Q135.a 是最短的命令，并且 Q135.a 必须要出现至少 3 次、R24.a 必须要出现至少 2 次。

3. 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

```
#include <stdio.h>
        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;
}
```

count.c

```
        B
int bar(int n) {
    static int ans = 0;
    ans = ans + x;
    return n + ans;
}
```

(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.o	x	x	×	√
	bar	bar	×	×
	ans	ans.1597	√	
count.o	x	x	×	×
	bar	bar	×	√
	ans	ans.0344	√	

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？

1 1 3 3 6 6

(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.o	x	x	√	
	bar	bar	×	×
	ans	ans.1597	√	
count.o	x	x	√	
	bar	bar	×	√
	ans	ans.0344	√	

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？
1 1 3 2 6 3。两个 x 在各自的.o 文件中的名字都为 x，因为它们不是过程中的静态变量。思考：对于非过程间的静态变量，为什么**编译器**不需要作这样的区分？

(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？

链接错误，x 被定义多次。

4. 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：

```
Symbol table '.symtab' contains ?? entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     5: 0000000000000000     8 OBJECT  LOCAL  DEFAULT    4 ans.1797
     7: 0000000000000008     8 OBJECT  LOCAL  DEFAULT    4 y
    11: 0000000000000000    52 FUNC    GLOBAL DEFAULT    1 bar
    12: 0000000000000034   139 FUNC    GLOBAL DEFAULT    1 main
    13: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT  UND z
    15: 0000000000000008     8 OBJECT  GLOBAL DEFAULT  COM t
```

PART A. 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
√/×	×	×	×	√	×	×	×

PART B. 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）。

PART C. 问答题

(1) 字符串"%ld %ld\n"位于哪个节中？_____。

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

(2) 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。

PART D. 使用 objdump -dx test.o 查看发现有如下的汇编代码：

```
0000000000000000 <bar>:
   0:   b9 00 00 00 00       mov    $0x0,%ecx
...
0000000000000034 <main>:
  34:   53                   push   %rbx
...
  6b:   e8 00 00 00 00       callq  70 <main+0x3c>
          6c: R_X86_64_PC32    bar-0x4
...
  90:   bf 00 00 00 00       mov    $0x0,%edi
          91: R_X86_64_32  .rodata.str1.1+0xa
```

现在将若干个.o 文件链接成可执行文件 done。假设链接器已经确定：test.o 的.text 节在 done 中的起始地址为 ADDR(.text)=0x400517。

链接后，test.o 中的 6b 处的指令变为 done 中如下的指令：

```
_________:    e8 ___ ___ ___ ___          callq  400517 <bar>
```

请补充以上五个空格（其中第一个空格是指令地址，之后五个空格是机器码）

test.o 中 90 处的指令变为 done 中如下的指令：

```
  4005a7:  bf  9e  06  40  00       mov    $0x40069e,%edi
```

则可执行文件 done 中，.rodata.str1.1 的起始地址为 0x_____。

PART E. 对 done 使用 objdump，发现有如下的函数：

```
0000000000400430 <_start>
000000000040054b <main>
0000000000600ff0 <__libc_start_main@GLIBC_2.2.5>
```

则 done 的入口点地址是 0x_____。

【答】

C. (1) C (2) 0

D. 由题意，
ADDR(s) = ADDR(.text) = 0x400517
ADDR(r.symbol) = ADDR(bar) = 0x400517
r.offset = 0x6c
r.addend = -4
因此
refaddr = ADDR(.text) + r.offset = 0x400583
*refptr = ADDR(r.symbol) + r.addend - refaddr = 0xFFFFFF90
所以代码地址为 400582，机器码为 e8 90 ff ff ff
0x40069e-0xa=0x400694

E. 0x400430