动程序总是传送 libc.a 给链接器，所以前面提到的对 libc.a 的引用是不必要的）。

在 Linux 系统中，静态库以一种称为*存档*(archive)的特殊文件格式存放在磁盘中。存档文件是一组连接起来的可重定位目标文件的集合，有一个头部用来描述每个成员目标文件的大小和位置。存档文件名由后缀 .a 标识。

为了使我们对库的讨论更加形象具体，考虑图 7-6 中的两个向量例程。每个例程，定义在它自己的目标模块中，对两个输入向量进行一个向量操作，并把结果存放在一个输出向量中。每个例程有一个副作用，会记录它自己被调用的次数，每次被调用会把一个全局变量加 1。（当我们在 7.12 节中解释位置无关代码的思想时会起作用。）

*code/link/addvec.c*

```
1  int addcnt = 0;
2
3  void addvec(int *x, int *y,
4              int *z, int n)
5  {
6      int i;
7
8      addcnt++;
9
10     for (i = 0; i < n; i++)
11         z[i] = x[i] + y[i];
12 }
```

*code/link/addvec.c*

a) addvec.o

*code/link/multvec.c*

```
1  int multcnt = 0;
2
3  void multvec(int *x, int *y,
4               int *z, int n)
5  {
6      int i;
7
8      multcnt++;
9
10     for (i = 0; i < n; i++)
11         z[i] = x[i] * y[i];
12 }
```

*code/link/multvec.c*

b) multvec.o

图 7-6　libvector 库中的成员目标文件

要创建这些函数的一个静态库，我们将使用 AR 工具，如下：

```
linux> gcc -c addvec.c multvec.c
linux> ar rcs libvector.a addvec.o multvec.o
```

为了使用这个库，我们可以编写一个应用，比如图 7-7 中的 main2.c，它调用 addvec 库例程。包含(或头)文件 vector.h 定义了 libvector.a 中例程的函数原型。

*code/link/main2.c*

```
1  #include <stdio.h>
2  #include "vector.h"
3
4  int x[2] = {1, 2};
5  int y[2] = {3, 4};
6  int z[2];
7
8  int main()
9  {
10     addvec(x, y, z, 2);
11     printf("z = [%d %d]\n", z[0], z[1]);
12     return 0;
13 }
```

*code/link/main2.c*

图 7-7　示例程序 2。这个程序调用 libvector 库中的函数

为了创建这个可执行文件，我们要编译和链接输入文件 main.o 和 libvector.a：