dlsym 函数的输入是一个指向前面已经打开了的共享库的句柄和一个 symbol 名字，如果该符号存在，就返回符号的地址，否则返回 NULL。

```
#include <dlfcn.h>

int dlclose (void *handle);
```

返回：若成功则为 0，若出错则为 −1。

如果没有其他共享库还在使用这个共享库，dlclose 函数就卸载该共享库。

```
#include <dlfcn.h>

const char *dlerror(void);
```

返回：如果前面对 dlopen、dlsym 或 dlclose 的调用失败，则为错误消息，如果前面的调用成功，则为 NULL。

dlerror 函数返回一个字符串，它描述的是调用 dlopen、dlsym 或者 dlclose 函数时发生的最近的错误，如果没有错误发生，就返回 NULL。

图 7-17 展示了如何利用这个接口动态链接我们的 libvector.so 共享库，然后调用它的 addvec 例程。要编译这个程序，我们将以下面的方式调用 GCC：

```
linux> gcc -rdynamic -o prog2r dll.c -ldl
```

code/link/dll.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <dlfcn.h>
4
5   int x[2] = {1, 2};
6   int y[2] = {3, 4};
7   int z[2];
8
9   int main()
10  {
11      void *handle;
12      void (*addvec)(int *, int *, int *, int);
13      char *error;
14
15      /* Dynamically load the shared library containing addvec() */
16      handle = dlopen("./libvector.so", RTLD_LAZY);
17      if (!handle) {
18          fprintf(stderr, "%s\n", dlerror());
19          exit(1);
20      }
21
22      /* Get a pointer to the addvec() function we just loaded */
23      addvec = dlsym(handle, "addvec");
24      if ((error = dlerror()) != NULL) {
25          fprintf(stderr, "%s\n", error);
```

图 7-17　示例程序 3。在运行时动态加载和链接共享库 libvector.so