后，动态链接器通过执行下面的重定位完成链接任务：

- 重定位 libc.so 的文本和数据到某个内存段。
- 重定位 libvector.so 的文本和数据到另一个内存段。
- 重定位 prog2l 中所有对由 libc.so 和 libvector.so 定义的符号的引用。

最后，动态链接器将控制传递给应用程序。从这个时刻开始，共享库的位置就固定了，并且在程序执行的过程中都不会改变。

7.11 从应用程序中加载和链接共享库

到目前为止，我们已经讨论了在应用程序被加载后执行前时，动态链接器加载和链接共享库的情景。然而，应用程序还可能在它运行时要求动态链接器加载和链接某个共享库，而无需在编译时将那些库链接到应用中。

动态链接是一项强大有用的技术。下面是一些现实世界中的例子：

- 分发软件。微软 Windows 应用的开发者常常利用共享库来分发软件更新。他们生成一个共享库的新版本，然后用户可以下载，并用它替代当前的版本。下一次他们运行应用程序时，应用将自动链接和加载新的共享库。
- 构建高性能 Web 服务器。许多 Web 服务器生成动态内容，比如个性化的 Web 页面、账户余额和广告标语。早期的 Web 服务器通过使用 fork 和 execve 创建一个子进程，并在该子进程的上下文中运行 CGI 程序来生成动态内容。然而，现代高性能的 Web 服务器可以使用基于动态链接的更有效和完善的方法来生成动态内容。

其思路是将每个生成动态内容的函数打包在共享库中。当一个来自 Web 浏览器的请求到达时，服务器动态地加载和链接适当的函数，然后直接调用它，而不是使用 fork 和 execve 在子进程的上下文中运行函数。函数会一直缓存在服务器的地址空间中，所以只要一个简单的函数调用的开销就可以处理随后的请求了。这对一个繁忙的网站来说是有很大影响的。更进一步地说，在运行时无需停止服务器，就可以更新已存在的函数，以及添加新的函数。

Linux 系统为动态链接器提供了一个简单的接口，允许应用程序在运行时加载和链接共享库。

```
#include <dlfcn.h>

void *dlopen(const char *filename, int flag);
```

返回：若成功则为指向句柄的指针，若出错则为 NULL。

dlopen 函数加载和链接共享库 filename。用已用带 RTLD_GLOBAL 选项打开了的库解析 filename 中的外部符号。如果当前可执行文件是带-rdynamic 选项编译的，那么对符号解析而言，它的全局符号也是可用的。flag 参数必须要么包括 RTLD_NOW，该标志告诉链接器立即解析对外部符号的引用，要么包括 RTLD_LAZY 标志，该标志指示链接器推迟符号解析直到执行来自库中的代码。这两个值中的任意一个都可以和 RTLD_GLOBAL 标志取或。

```
#include <dlfcn.h>

void *dlsym(void *handle, char *symbol);
```

返回：若成功则为指向符号的指针，若出错则为 NULL。