在当前进程中的程序执行了如下的 execve 调用：

```
execve("a.out", NULL, NULL);
```

正如在第 8 章中学到的，execve 函数在当前进程中加载并运行包含在可执行目标文件 a.out 中的程序，用 a.out 程序有效地替代了当前程序。加载并运行 a.out 需要以下几个步骤：

- 删除已存在的用户区域。删除当前进程虚拟地址的用户部分中的已存在的区域结构。
- 映射私有区域。为新程序的代码、数据、bss 和栈区域创建新的区域结构。所有这些新的区域都是私有的、写时复制的。代码和数据区域被映射为 a.out 文件中的 .text 和 .data 区。bss 区域是请求二进制零的，映射到匿名文件，其大小包含在 a.out 中。栈和堆区域也是请求二进制零的，初始长度为零。图 9-31 概括了私有区域的不同映射。
- 映射共享区域。如果 a.out 程序与共享对象（或目标）链接，比如标准 C 库 libc.so，那么这些对象都是动态链接到这个程序的，然后再映射到用户虚拟地址空间中的共享区域内。
- 设置程序计数器（PC）。execve 做的最后一件事情就是设置当前进程上下文中的程序计数器，使之指向代码区域的入口点。

下一次调度这个进程时，它将从这个入口点开始执行。Linux 将根据需要换入代码和数据页面。

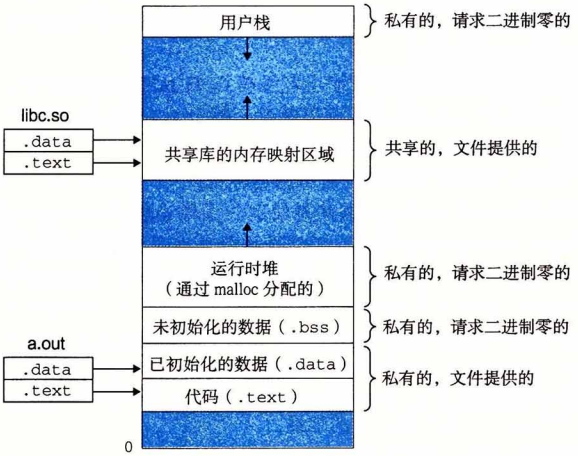

图 9-31　加载器是如何映射用户地址空间的区域的

9.8.4　使用 mmap 函数的用户级内存映射

Linux 进程可以使用 mmap 函数来创建新的虚拟内存区域，并将对象映射到这些区域中。

```
#include <unistd.h>
#include <sys/mman.h>

void  *mmap(void *start, size_t length, int prot, int flags,
            int fd, off_t offset);
                        返回：若成功时则为指向映射区域的指针，若出错则为 MAP_FAILED(-1)。
```