下面是如何在 csh 或 tcsh 中运行这个程序：

```
linux> (setenv LD_PRELOAD "./mymalloc.so"; ./intr; unsetenv LD_PRELOAD)
malloc(32) = 0x2157010
free(0x2157010)
```

请注意，你可以用 LD_PRELOAD 对任何可执行程序的库函数调用打桩！

```
linux> LD_PRELOAD="./mymalloc.so" /usr/bin/uptime
malloc(568) = 0x21bb010
free(0x21bb010)
malloc(15) = 0x21bb010
malloc(568) = 0x21bb030
malloc(2255) = 0x21bb270
free(0x21bb030)
malloc(20) = 0x21bb030
malloc(20) = 0x21bb050
malloc(20) = 0x21bb070
malloc(20) = 0x21bb090
malloc(20) = 0x21bb0b0
malloc(384) = 0x21bb0d0
 20:47:36 up 85 days,  6:04,  1 user,  load average: 0.10, 0.04, 0.05
```

7.14　处理目标文件的工具

在 Linux 系统中有大量可用的工具可以帮助你理解和处理目标文件。特别地，GNU binutils 包尤其有帮助，而且可以运行在每个 Linux 平台上。

- AR：创建静态库，插入、删除、列出和提取成员。
- STRINGS：列出一个目标文件中所有可打印的字符串。
- STRIP：从目标文件中删除符号表信息。
- NM：列出一个目标文件的符号表中定义的符号。
- SIZE：列出目标文件中节的名字和大小。
- READELF：显示一个目标文件的完整结构，包括 ELF 头中编码的所有信息。包含 SIZE 和 NM 的功能。
- OBJDUMP：所有二进制工具之母。能够显示一个目标文件中所有的信息。它最大的作用是反汇编 .text 节中的二进制指令。

Linux 系统为操作共享库还提供了 LDD 程序：

- LDD：列出一个可执行文件在运行时所需要的共享库。

7.15　小结

链接可以在编译时由静态编译器来完成，也可以在加载时和运行时由动态链接器来完成。链接器处理称为目标文件的二进制文件，它有 3 种不同的形式：可重定位的、可执行的和共享的。可重定位的目标文件由静态链接器合并成一个可执行的目标文件，它可以加载到内存中并执行。共享目标文件(共享库)是在运行时由动态链接器链接和加载的，或者隐含地在调用程序被加载和开始执行时，或者根据需要在程序调用 dlopen 库的函数时。

链接器的两个主要任务是符号解析和重定位，符号解析将目标文件中的每个全局符号都绑定到一个唯一的定义，而重定位确定每个符号的最终内存地址，并修改对那些目标的引用。