3.10　在机器级程序中将控制与数据结合起来

到目前为止，我们已经分别讨论机器级代码如何实现程序的控制部分和如何实现不同的数据结构。在本节中，我们会看看数据和控制如何交互。首先，深入审视一下指针，它是C编程语言中最重要的概念之一，但是许多程序员对它的理解都非常浅显。我们复习符号调试器GDB的使用，用它仔细检查机器级程序的详细运行。接下来，看看理解机器级程序如何帮助我们研究缓冲区溢出，这是现实世界许多系统中一种很重要的安全漏洞。最后，查看机器级程序如何实现函数要求的栈空间大小在每次执行时都可能不同的情况。

3.10.1　理解指针

指针是C语言的一个核心特色。它们以一种统一方式，对不同数据结构中的元素产生引用。对于编程新手来说，指针总是会带来很多的困惑，但是基本概念其实非常简单。在此，我们重点介绍一些指针和它们映射到机器代码的关键原则。

- 每个指针都对应一个类型。这个类型表明该指针指向的是哪一类对象。以下面的指针声明为例：

```
int *ip;
char **cpp;
```

  变量ip是一个指向int类型对象的指针，而cpp指针指向的对象自身就是一个指向char类型对象的指针。通常，如果对象类型为T，那么指针的类型为$T*$。特殊的void * 类型代表通用指针。比如说，malloc函数返回一个通用指针，然后通过显式强制类型转换或者赋值操作那样的隐式强制类型转换，将它转换成一个有类型的指针。指针类型不是机器代码中的一部分；它们是C语言提供的一种抽象，帮助程序员避免寻址错误。

- 每个指针都有一个值。这个值是某个指定类型的对象的地址。特殊的NULL(0)值表示该指针没有指向任何地方。
- 指针用'&'运算符创建。这个运算符可以应用到任何lvalue类的C表达式上，lvalue意指可以出现在赋值语句左边的表达式。这样的例子包括变量以及结构、联合和数组的元素。我们已经看到，因为leaq指令是设计用来计算内存引用的地址的，&运算符的机器代码实现常常用这条指令来计算表达式的值。
- * 操作符用于间接引用指针。其结果是一个值，它的类型与该指针的类型一致。间接引用是用内存引用来实现的，要么是存储到一个指定的地址，要么是从指定的地址读取。
- 数组与指针紧密联系。一个数组的名字可以像一个指针变量一样引用(但是不能修改)。数组引用(例如a[3])与指针运算和间接引用(例如* (a+ 3))有一样的效果。数组引用和指针运算都需要用对象大小对偏移量进行伸缩。当我们写表达式p+ i，这里指针p的值为p，得到的地址计算为$p+L\cdot i$，这里L是与p相关联的数据类型的大小。
- 将指针从一种类型强制转换成另一种类型，只改变它的类型，而不改变它的值。强制类型转换的一个效果是改变指针运算的伸缩。例如，如果p是一个char * 类型的指针，它的值为p，那么表达式(int *)p+ 7计算为$p+28$，而(int *) (p+ 7)计算为$p+7$。(回想一下，强制类型转换的优先级高于加法。)