char”的对象的指针。这样一个字节指针引用一个字节序列，其中每个字节都被认为是一个非负整数。第一个例程 show_bytes 的输入是一个字节序列的地址，它用一个字节指针以及一个字节数来指示。该字节数指定为数据类型 size_t，表示数据结构大小的首选数据类型。show_bytes 打印出每个以十六进制表示的字节。C 格式化指令“%.2x”表明整数必须用至少两个数字的十六进制格式输出。

```
1   #include <stdio.h>
2
3   typedef unsigned char *byte_pointer;
4
5   void show_bytes(byte_pointer start, size_t len) {
6       size_t i;
7       for (i = 0; i < len; i++)
8           printf(" %.2x", start[i]);
9       printf("\n");
10  }
11
12  void show_int(int x) {
13      show_bytes((byte_pointer) &x, sizeof(int));
14  }
15
16  void show_float(float x) {
17      show_bytes((byte_pointer) &x, sizeof(float));
18  }
19
20  void show_pointer(void *x) {
21      show_bytes((byte_pointer) &x, sizeof(void *));
22  }
```

图 2-4　打印程序对象的字节表示。这段代码使用强制类型转换来规避类型系统。很容易定义针对其他数据类型的类似函数

过程 show_int、show_float 和 show_pointer 展示了如何使用程序 show_bytes 来分别输出类型为 int、float 和 void * 的 C 程序对象的字节表示。可以观察到它们仅仅传递给 show_bytes 一个指向它们参数 x 的指针 &x，且这个指针被强制类型转换为“unsigned char * ”。这种强制类型转换告诉编译器，程序应该把这个指针看成指向一个字节序列，而不是指向一个原始数据类型的对象。然后，这个指针会被看成是对象使用的最低字节地址。

这些过程使用 C 语言的运算符 sizeof 来确定对象使用的字节数。一般来说，表达式 sizeof(*T*)返回存储一个类型为 *T* 的对象所需要的字节数。使用 sizeof 而不是一个固定的值，是向编写在不同机器类型上可移植的代码迈进了一步。

在几种不同的机器上运行如图 2-5 所示的代码，得到如图 2-6 所示的结果。我们使用了以下几种机器：

Linux 32：运行 Linux 的 Intel IA32 处理器。

Windows：运行 Windows 的 Intel IA32 处理器。

Sun：运行 Solaris 的 Sun Microsystems SPARC 处理器。（这些机器现在由 Oracle 生产。）

Linux 64：运行 Linux 的 Intel x86-64 处理器。