code/data/show-bytes.c

```
1  void test_show_bytes(int val) {
2      int ival = val;
3      float fval = (float) ival;
4      int *pval = &ival;
5      show_int(ival);
6      show_float(fval);
7      show_pointer(pval);
8  }
```

code/data/show-bytes.c

图 2-5　字节表示的示例。这段代码打印示例数据对象的字节表示

机器	值	类型	字节（十六进制）
Linux 32	12 345	int	39 30 00 00
Windows	12 345	int	39 30 00 00
Sun	12 345	int	00 00 30 39
Linux 64	12 345	int	39 30 00 00
Linux 32	12 345.0	float	00 e4 40 46
Windows	12 345.0	float	00 e4 40 46
Sun	12 345.0	float	46 40 e4 00
Linux 64	12 345.0	float	00 e4 40 46
Linux 32	&ival	int *	e4 f9 ff bf
Windows	&ival	int *	b4 cc 22 00
Sun	&ival	int *	ef ff fa 0c
Linux 64	&ival	int *	b8 11 e5 ff ff 7f 00 00

图 2-6　不同数据值的字节表示。除了字节顺序以外，int 和 float 的结果是一样的。指针值与机器相关

参数 12 345 的十六进制表示为 0x00003039。对于 int 类型的数据，除了字节顺序以外，我们在所有机器上都得到相同的结果。特别地，我们可以看到在 Linux 32、Windows 和 Linux 64 上，最低有效字节值 0x39 最先输出，这说明它们是小端法机器；而在 Sun 上最后输出，这说明 Sun 是大端法机器。同样地，float 数据的字节，除了字节顺序以外，也都是相同的。另一方面，指针值却是完全不同的。不同的机器/操作系统配置使用不同的存储分配规则。一个值得注意的特性是 Linux 32、Windows 和 Sun 的机器使用 4 字节地址，而 Linux 64 使用 8 字节地址。

给 C 语言初学者　使用 typedef 来命名数据类型

C 语言中的 typedef 声明提供了一种给数据类型命名的方式。这能够极大地改善代码的可读性，因为深度嵌套的类型声明很难读懂。

typedef 的语法与声明变量的语法十分相像，除了它使用的是类型名，而不是变量名。因此，图 2-4 中 byte_pointer 的声明和将一个变量声明为类型“unsigned char *”有相同的形式。

例如，声明：

```
typedef int *int_pointer;
int_pointer ip;
```

将类型“int_pointer”定义为一个指向 int 的指针，并且声明了一个这种类型的变量 ip。我们还可以将这个变量直接声明为：

```
int *ip;
```