第二章习题

yinxuhao [xuhao_yin@163.com]

December 27, 2022

Contents

1 2.5

思考下面对show_bytes的三次调用：

```
int val = 0x87654321;
byte_pointer valp = (byte_pointer) &val;
show_bytes(valp, 1);  /* A. */
show_bytes(valp, 2);  /* B. */
show_bytes(valp, 3);  /* C. */
```

指出在小端法机器和大端法机器上，每次调用的输出值。

A. 小端法 21 大端法 87
B. 小端法 21 43 大端法 87 65
C. 小端法 21 43 65 大端法 87 65 43

2 2.6

使用show_int和show_float，我们确定整数 3510593 的十六进制表示为 0x00359141，而浮点数 3510593.0 的十六进制表示为 0x4A564504。

2.1 写出这两个十六进制的二进制表示

3510593 0000 0000 0011 0101 1001 0001 0100 0001
3510593.0 0100 1010 0101 0110 0100 0101 0000 0100

2.2 移动这两个二进制串的相对位置，使得它们相匹配的位数最多。有多少位相匹配呢？

3510593: 0000 0000 001 10101100100010100000 1
3510593.0 0100 1010 0 10101100100010100000 1 00

2.3 串中的什么部分不相匹配？

除了最高有效位 1，整数的其他所有位都嵌在浮点数中。
另外，浮点数中一些非零的高位不与整数中的高位相匹配。

3 2.25

考虑下列代码，这段代码试图计算数组 a 中所有元素的和，其中元素的数量由参数 length 给出。

```
/* WARNING: This is buggy code */
float sum_elements(float a[], unsigned length) {
    int i;
    float result = 0;

    for (i = 0; i <= length - 1; i++)
        result += a[i];
    return result;
}
```

因为 length 是无符号数，所以当它为 0 时，执行 $length - 1$, 会产生一个非常大的正数。所以程序永远都不会停，直至发生内存错误。

要修改代码，需要将 length 定义为 int 类型，或者用`i < length` 作为判断条件，防止无符号数隐式转换为 int。

4 3.26

现在给你一个任务，写一个函数用来判定一个字符串是否比另一个更长。前提是你要用字符串库函数 strlen，它的声明如下：

```
/* Prototype for library function strlen */
size_t strlen(const char *s);
```

最开始你写的函数是这样的：

```
/* Determine whether string s is longer than string t */
/* WARNING: This function is buggy */
int strlonger(char *s, char *t) {
    return strlen(s) - strlen(t) > 0;
}
```

当你在一些示例数据上测试这个函数时，一切似乎都是正常的。进一步研究发现在头文件`stdio.h`中数据类型`size_t` 是定义成 unsigned int 的。

4.1 在什么情况下，这个函数会产生不正确的结果？

s 的长度小于 t

4.2 解释为什么会出现这样不正确的结果。

当 s 的长度小于 t 时，理应会产生一个负数。但是由于 `size_t`是无符号数，所以会转换为一个超大的正数，导致检测值永远为 true。

4.3 说明如何修改这段代码好让它能更可靠地工作。

修改减法运算为直接比较，如下修改：

```
return strlen(s) > strlen(t);
```

即可。