

俞楚凡 ICS 4.7 Lab

`array_store.c`: 实现第一个课堂练习。

代码参见源代码。输出如下:

```
ainfinity@AInfinity:~/ics/lab6$ ./array_store

0
1
2
3
4
5
6
7
8
9
0
1
2
3
4
5
6
7
8
9
ainfinity@AInfinity:~/ics/lab6$ |
```

可以看见，一开始，由于栈上什么都没有，所以输出全是空。从第二次循环开始，由于每次调用函数，调用的都是栈上的同一块区域，所以事实上将会输出上一次存储的内容。

struct.c：实现第二个练习。

输出如下：

```
ainfinity@AInfinity:~/ics/lab6$ ./struct
8 12 8
Address of ss1.name: 0x7ffcf35c44dc
Address of ss1.code: 0x7ffcf35c44de
Address of ss1.value: 0x7ffcf35c44e0

Address of ss2.code: 0x7ffcf35c44ec
Address of ss2.value: 0x7ffcf35c44f0
Address of ss2.name: 0x7ffcf35c44f4
Address of ss3.value: 0x7ffcf35c44e4
Address of ss3.name: 0x7ffcf35c44e8
Address of ss3.code: 0x7ffcf35c44ea
ainfinity@AInfinity:~/ics/lab6$ |
```

可以看到，s1和s3占用的空间是一样的，而s2占用的空间更多。

8 12 8事实上是数据对齐导致的。第一个结构体是 $2 + 2 + 4$ ，第二是 $4 + 4 + 4$ ，第三个是 $4 + 2 + 2$ 。下面的地址之间的差也说明了这一点。

func.c：实现第三个实验。

```
ainfinity@AInfinity:~/ics/lab6$ ./func
385
385
385
385
ainfinity@AInfinity:~/ics/lab6$ |
```

可以看到，四个输出全部都是函数返回值。这是因为，对于函数指针而言，调用的时候会自动进行解引用，而函数本质上调用的是函数的起始地址的值，因此这四种写法全部都是等价的。