

## 十六进制

---

0~F

egRGB的表示——0x作为十六进制的标识



#000000, 黑色, 没有任何光

#FFFFFF, 白色, 全部RGB

## 内存, 地址

---

也是用十六进制表示

**& 并不是逻辑与, 而是地址**

**\*不是乘法, 是去内存的地址**

%p, 地址

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int n = 50;
6     printf("%p\n", &n);
7 }
```

```
~/ $ ./address
0x7ffd80792f7c
```

\*&n=n

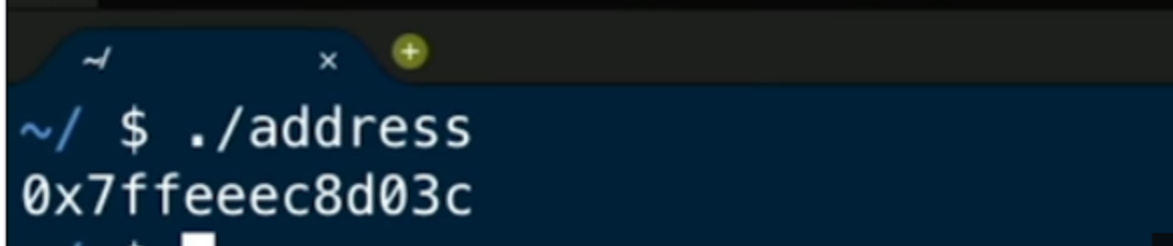
然后同理，&n=\*p

## 指针pointer

---

一种不同类型的变量，储存地址

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int n = 50;
6     int *p = &n;
7     printf("%p\n", p);
8 }
```



比喻，指针就像邮箱，里边可以放各种东西（数据）

string技术上只是第一个项的地址

string其实是char\*



同理，get\_string返回的就是第一个项的地址

strcmp的实现，实际上是比较两个地址储存的值以及随后的值

```
5
6 int main(void)
7 {
8     char *s = get_string("s:
9
10    char *t = s;
11
12    t[0] = toupper(t[0]);
13
14    printf("s: %s\n", s);
15    printf("t: %s\n", t);
16 }
```

```
~/ $ ./copy
s: hi!
s: Hi!
t: Hi!
```

s和t指向一个东西

正确的复制一个string的方式：遍历全部

## malloc

请求内存分配

```

7 int main(void)
8 {
9     char *s = get_string("s: ");
10
11     char *t = malloc(strlen(s) + 1);
12
13     for (int i = 0, n = strlen(s); i <= n; i++)
14     {
15         *(t+i) = *(s+i);
16     }
17
18     t[0] = toupper(t[0]);
19

```

所以如果你想去 t 的地址加上任何 i 来抵消你自

```

~/ $ ./copy
s: hi!
s: hi!
t: Hi!

```

用处：复制s到t，而不是仅仅复制一个指针

NULL空指针

\0或者NUL是空内容

直接用strcpy (a,b) 也可

## free()

归还之前的内存

strcpy () 也要free

get\_string 就是用了malloc和free

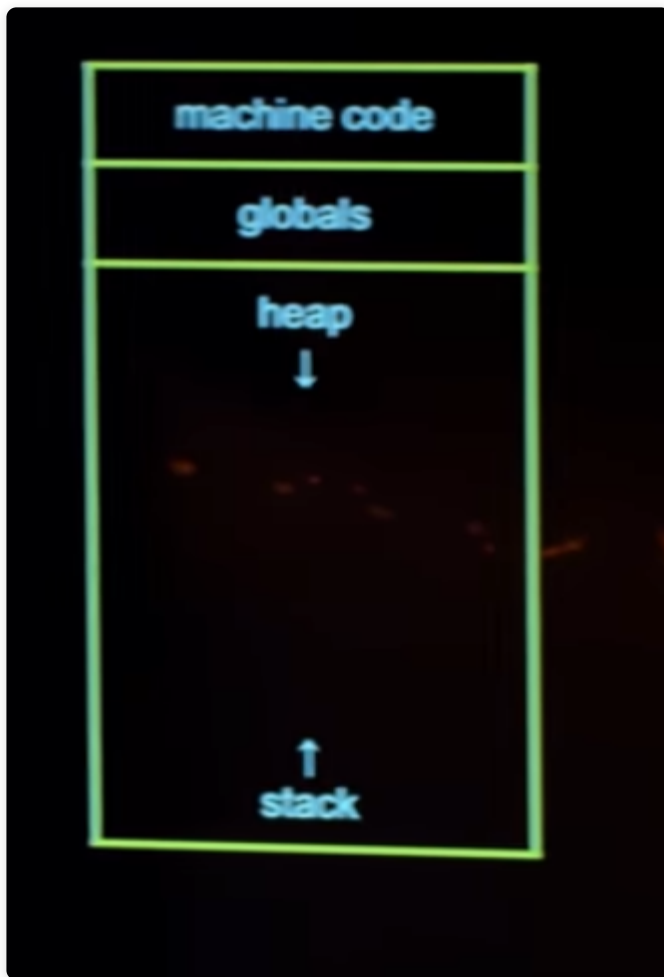
## 一些修复和内存相关的bug的工具：valgrind

valgrind ./文件名

## 垃圾值

如果在内存里没有指定值的话，之前会存在一些垃圾值，是之前过的

## 内存的布局



最上层是你代码的机器编码，然后后面是全局变量，在下面是堆-栈。堆是malloc调用的内存，栈是函数储存的地方，但是当内存过小，会互相覆盖。

## 实现交换

```
void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

但是这不行，因为没有return，其原始值并没有被改变

真正的方法是做两个指针，然后在函数里改变指针所指的内存格子的值

(但，不可以return两个值吗...?)

## 堆溢出or栈溢出

导致程序崩溃x

比如递归没有结束条件

## 解决方案：缓冲区buffer overflow

就像视频的缓冲

## scanf

---

即get系列函数的实现

get int

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int x;
6     printf("x: ");
7     scanf("%i", &x);
8     printf("x: %i\n", x);
9 }
```

get string更复杂，要用malloc

## file I/O

---

文件永久保存

FILE\* 变量类型，指向文件地址的指针

fopen () 打开文件的参数，名称+打开模式

```
FILE *file = fopen("phonebook.csv", "a");
if (file == NULL)
{
    return 1;
}
```

```
char *name = get_string("Name: ");
char *number = get_string("Number: ");
```

```
fprintf(file, "%s,%s\n", name, number)
fclose(file);
```

不同文件格式的头都有标志自身类型的数据

```
// Read first three bytes
BYTE bytes[3];
fread(bytes, sizeof(BYTE), 3, file);

// Check first three bytes
if (bytes[0] == 0xff && bytes[1] == 0xd8 && bytes[2] == 0xff)
{
    printf("Maybe\n");
}
```

对jpeg，读取其前三个字节

fread的使用orz