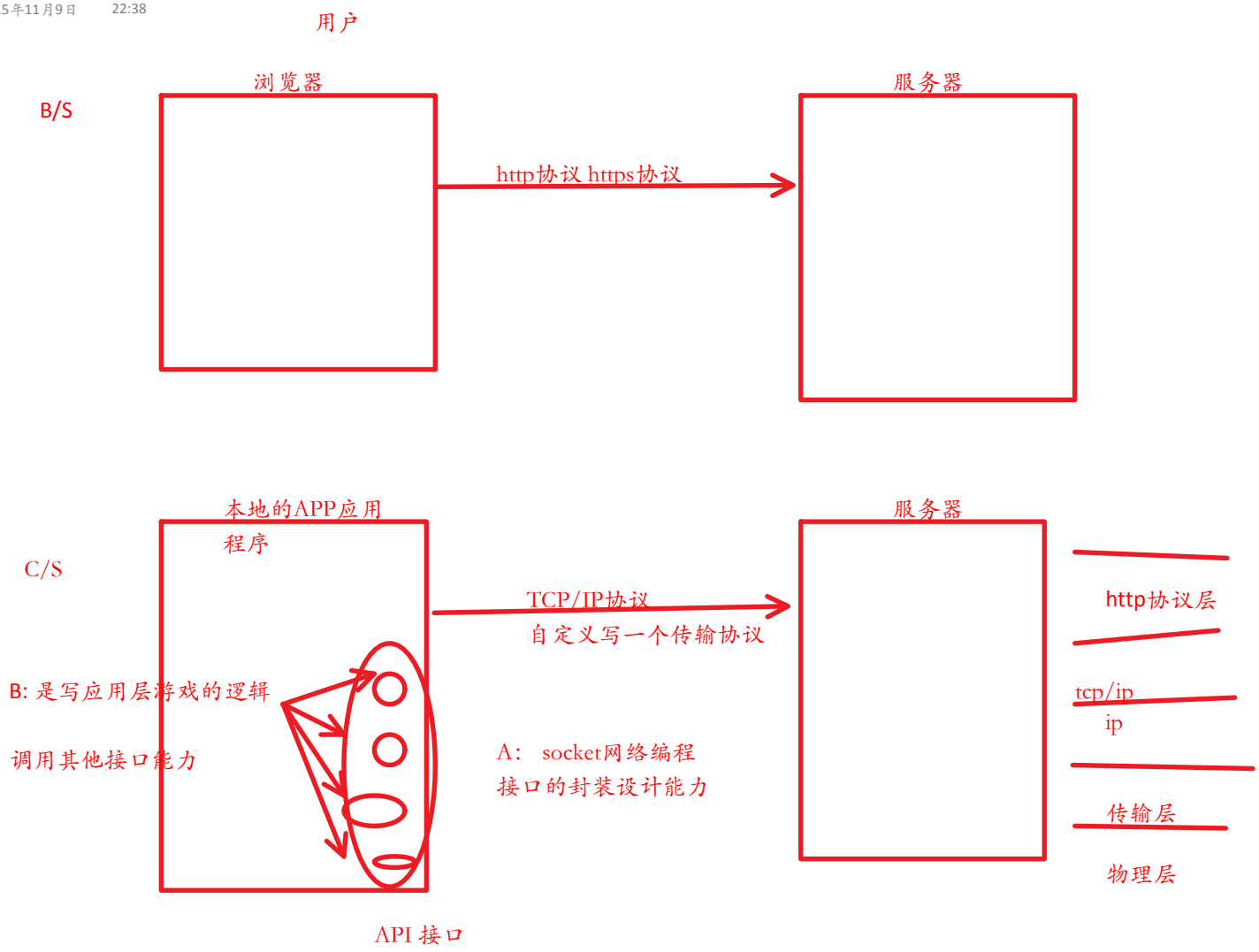
b/s c/s 结构

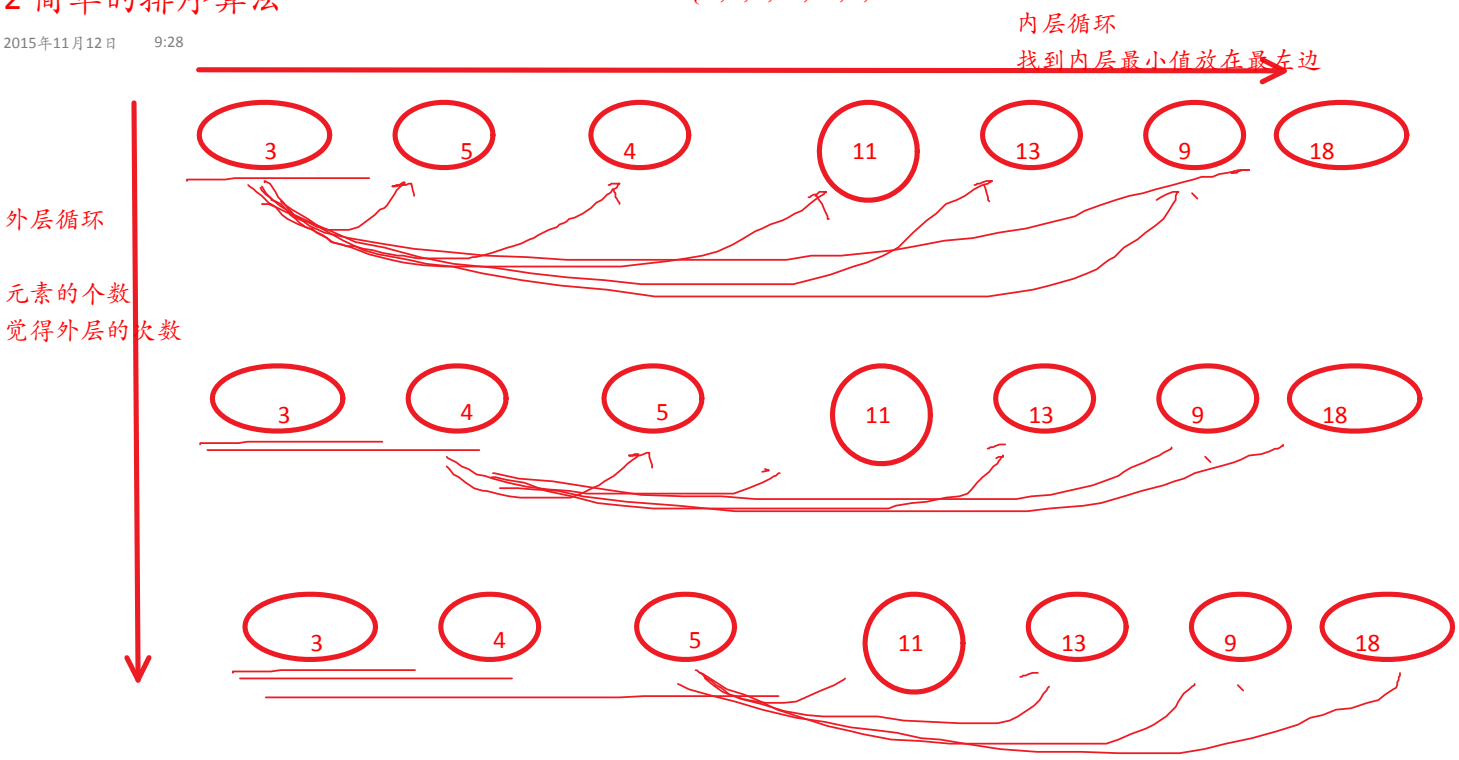
2015年11月9日 22:38



2 简单的排序算法

{ 3, 5, 4, 11, 13, 9, 18

2015年11月12日 9:28



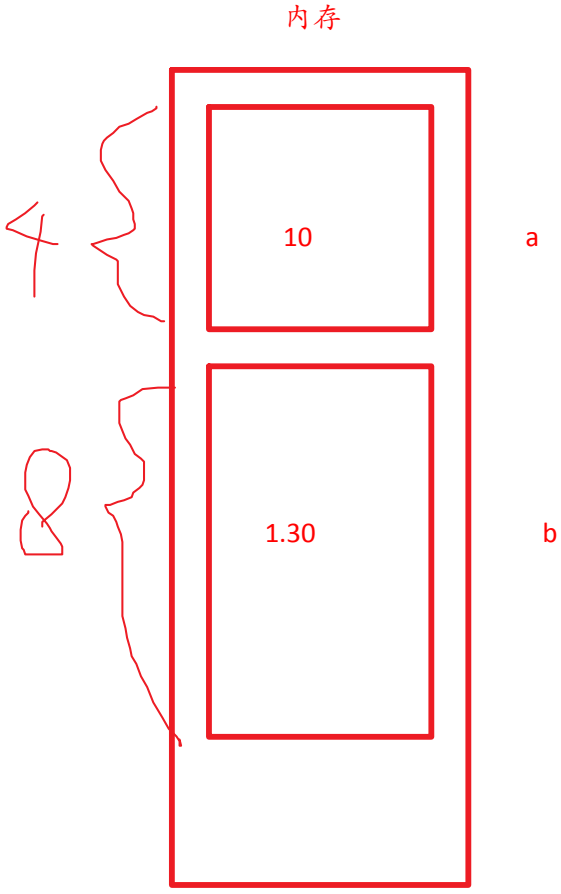
3 数据类型

2015年11月12日 10:14

int a =10;
//给我来一块4个字节大小的内存 a = 10;

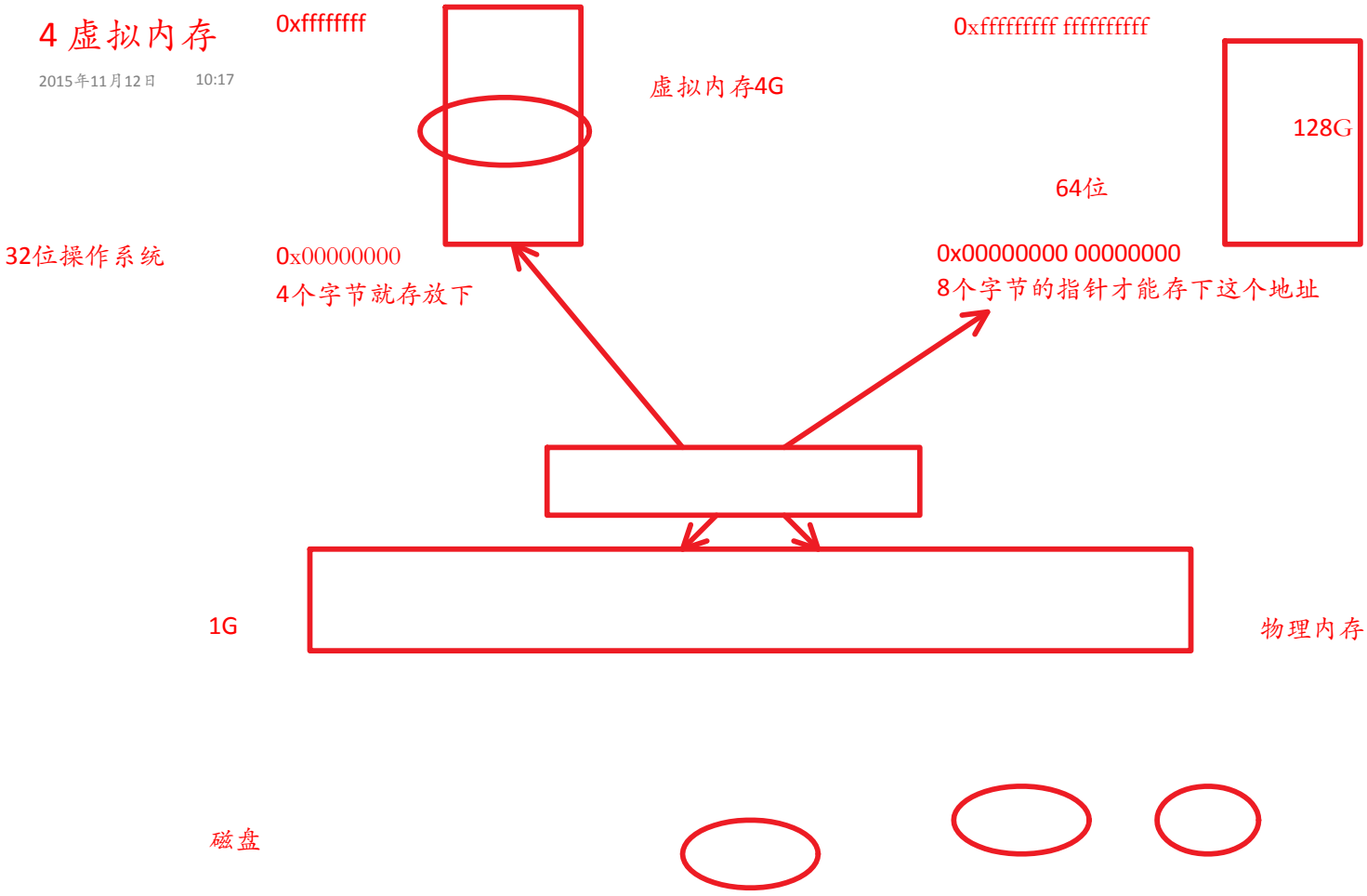
double b =1.30;
//8个大小的字节内存 b = 1.30;

数据类型的本质 内存块大小的 一个别名



4 虚拟内存

2015年11月12日 10:17



5 指针

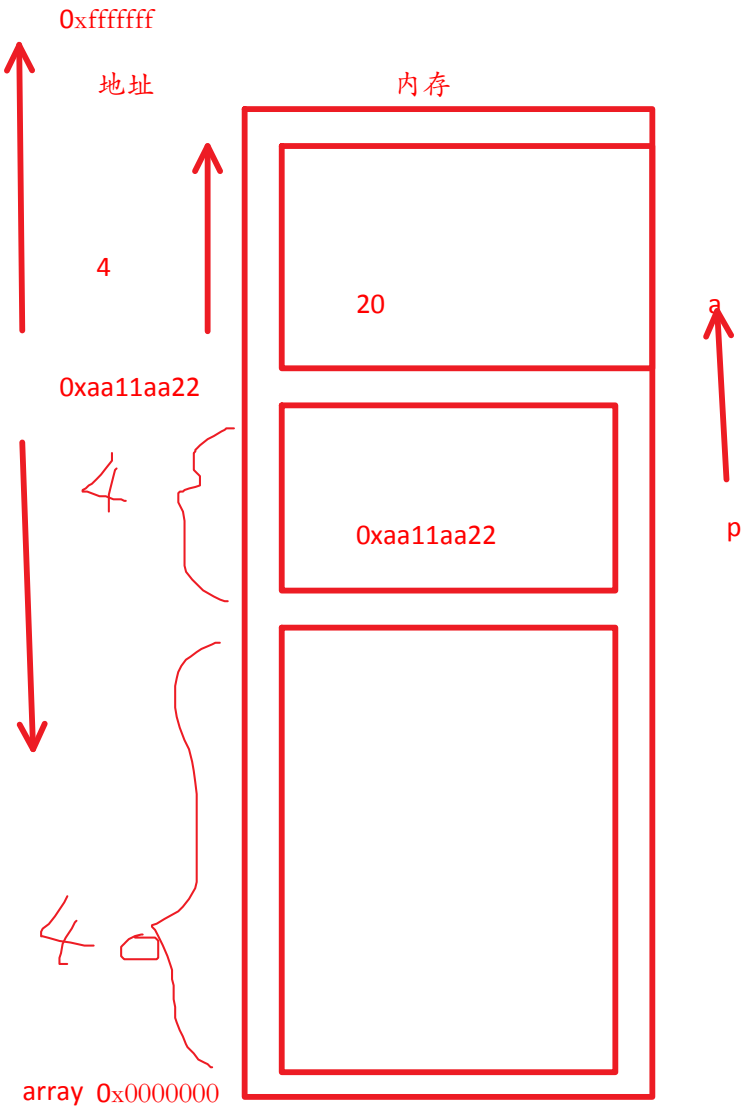
2015年11月12日 10:22

int a = 10;

int* p = &a;

p+1;

*p = 20;



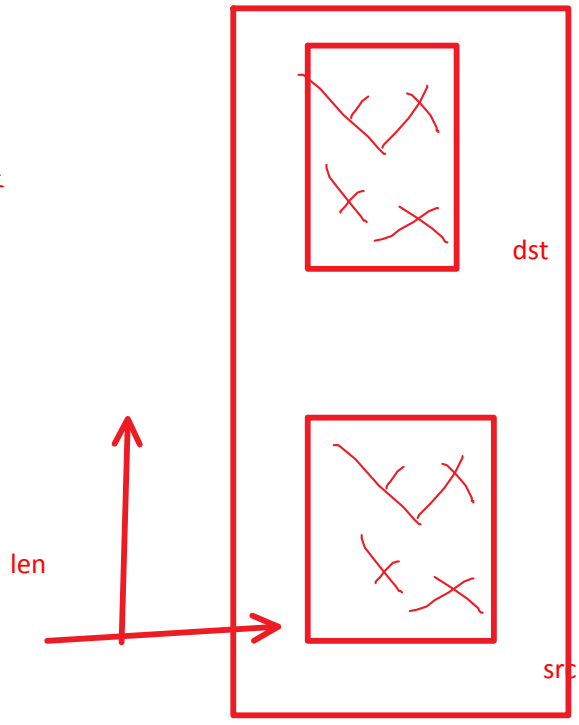
int array[10]; //编译器发下有一个数组类型 int[10]

6 数据类型的封装

2015年11月12日 11:00

void* p 是万能指针 p =a, p = b;
int* a a =p;
double* b

```
void * memcpy(void *dest, const void *src, size_t len);
```



void * 作为函数参数，起到了对数据类型 一个通用和 封装的特性

```
void main()
{
    void a;
}
```

7 变量的本质

2015年11月12日 11:20

int a = 10;

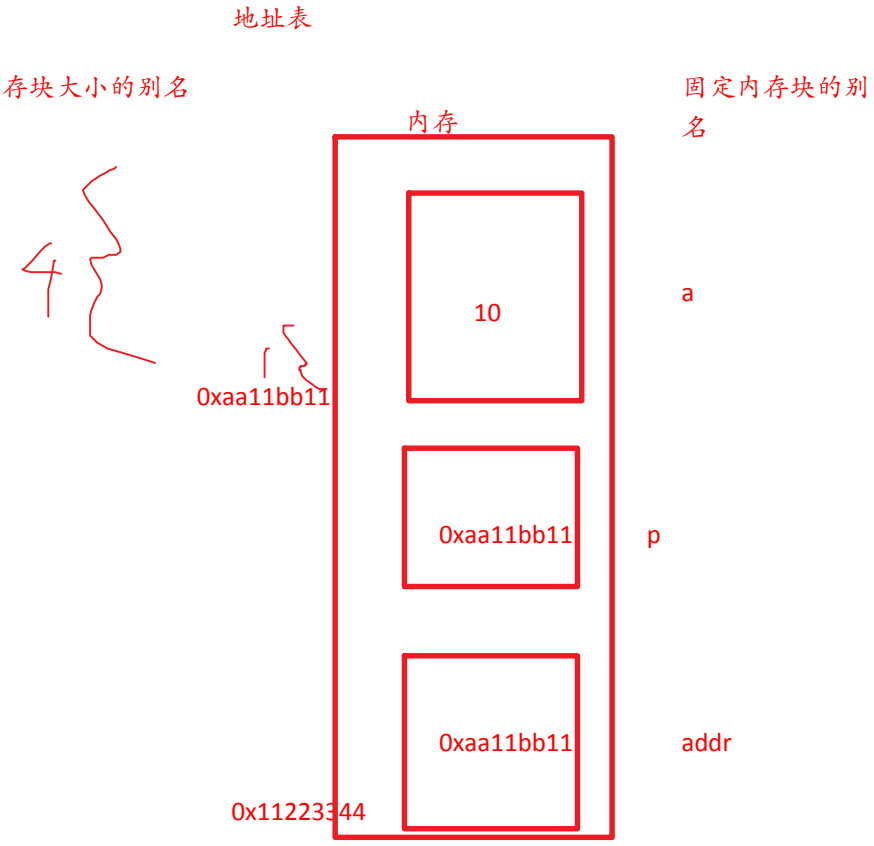
a = 10; //直接修改一个变量

间接修改一个变量

int *p = &a;

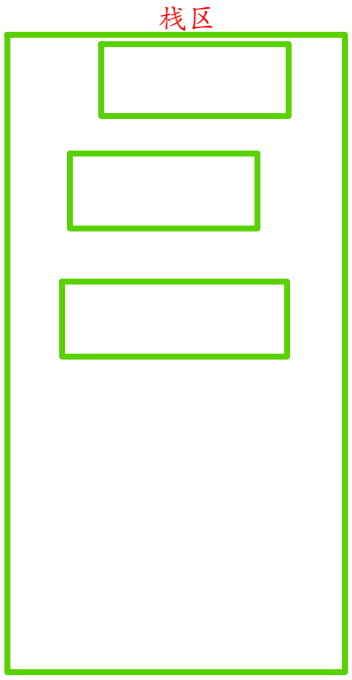
*p = 20;

int addr = (int) &a;

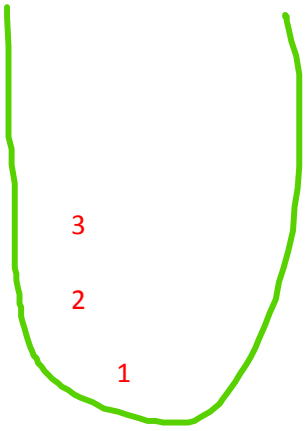


8 栈区

2015年11月12日 11:38



栈： 先进后出



栈区不是栈， 栈是一种数据结构

栈区是一种内存z

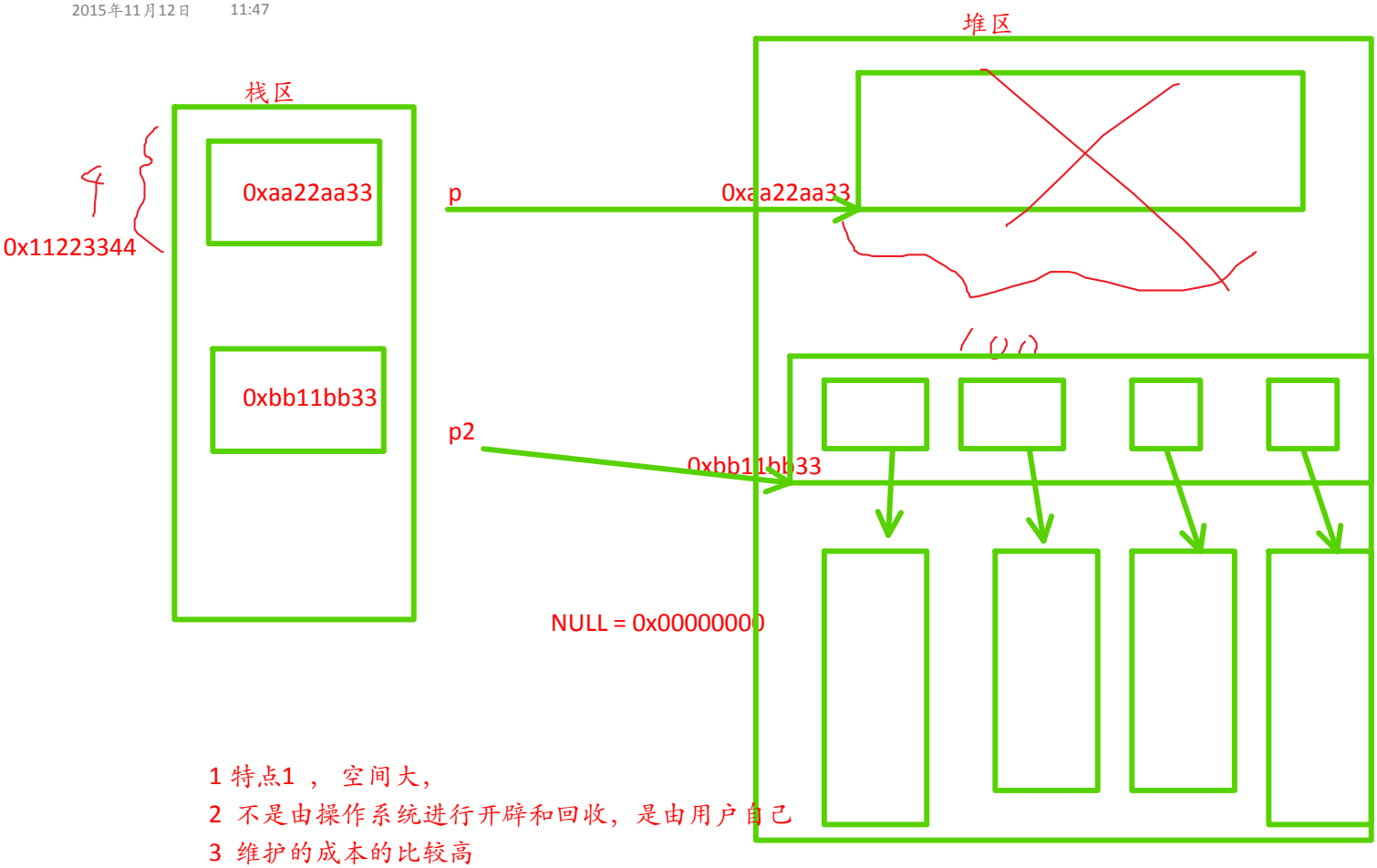
只不过 存放数据的方式，用栈的方式

是由操作系统进行分配和回收，

速度非常的快。 栈的资源少。

9 堆区

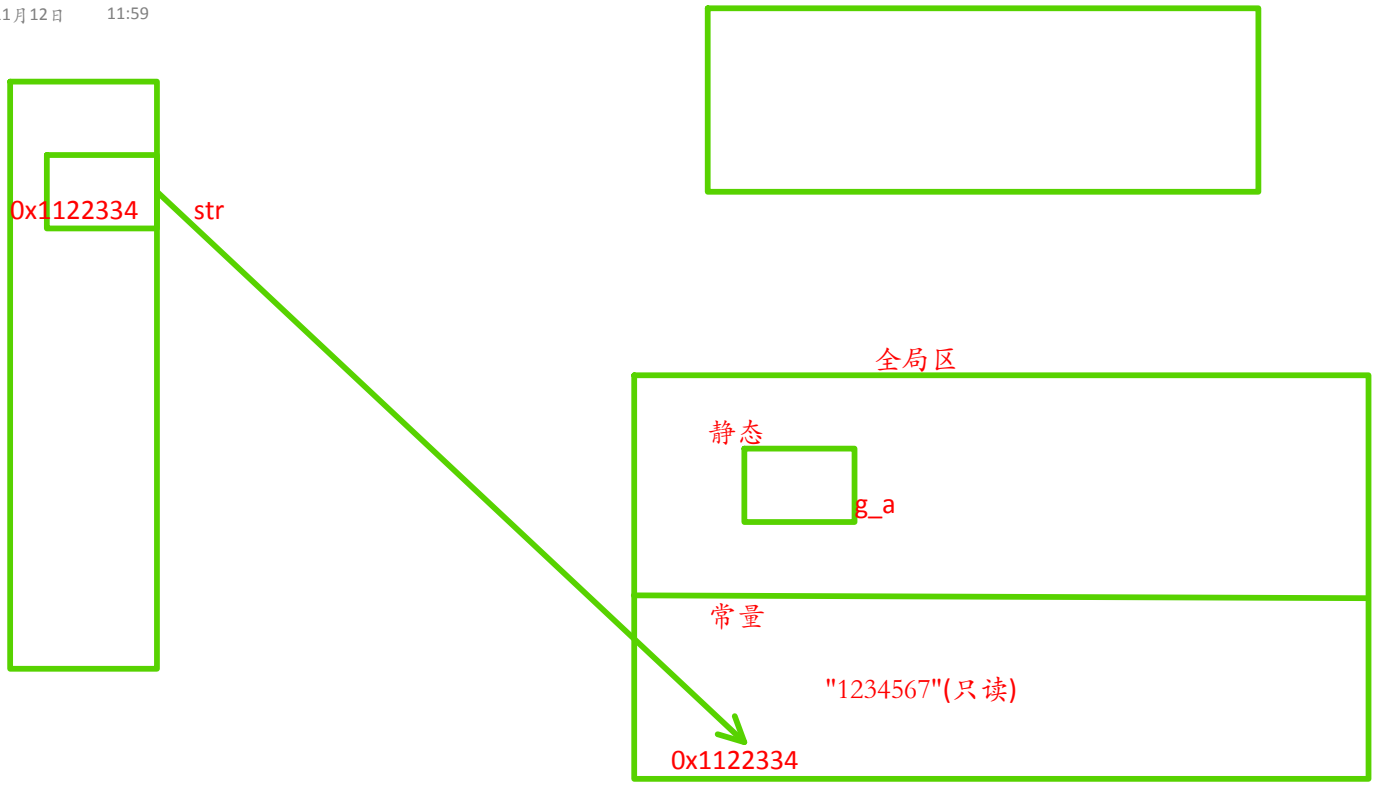
2015年11月12日 11:47



1 特点1 ， 空间大，
2 不是由操作系统进行开辟和回收，是由用户自己
3 维护的成本的比较高
4 使用与 容量较大， 管理方式比较单一的数据

10 全局区和常量区

2015年11月12日 11:59



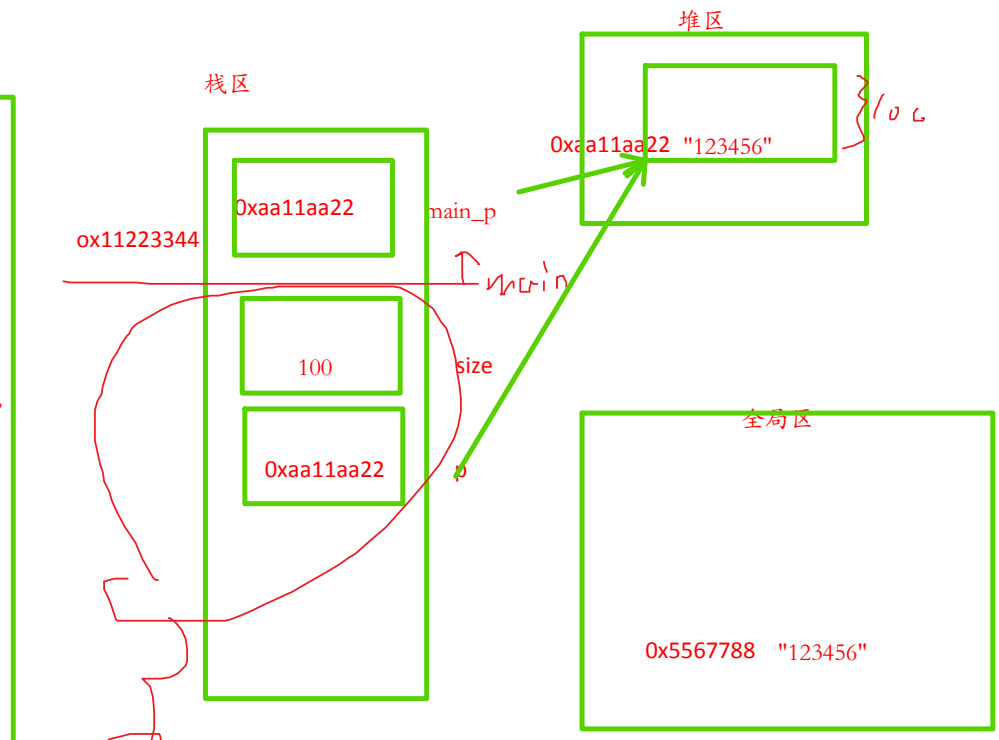

```
int g_a; //全局的
int main()
{
	char *str = "1234567";
}
```

11 内存四区

2015年11月12日 14:49

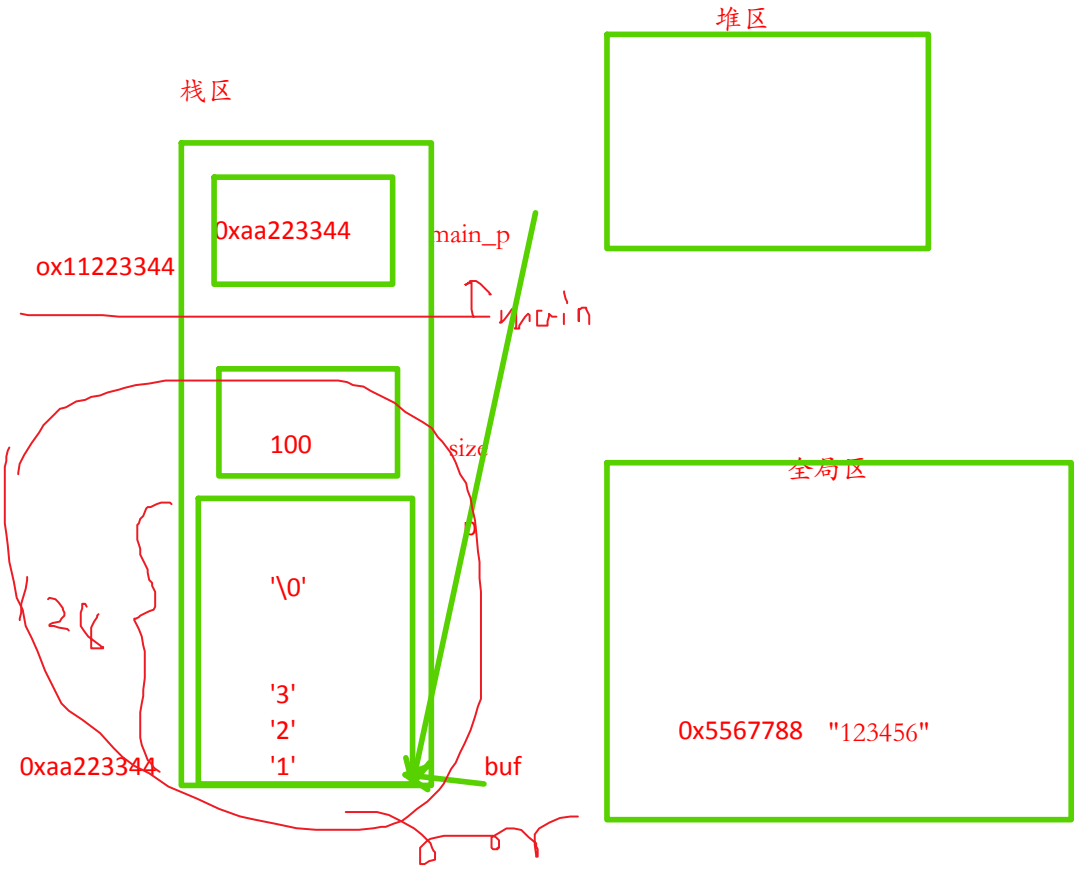
代码区

```
//在堆上开辟一些内存
char* get_mem(int size)
{
	char *p = NULL; //

	p = malloc(size);
	if (p == NULL) {
		fprintf(stderr, "malloc %d err\n",
		size);
		return NULL;
	}

	return p;
} //main_p = p;
 int a = b;


int main(void)
{
	char *main_p = NULL;
	main_p = get_mem(100);

	if (main_p == NULL) {
		fprintf(stderr, "get_mem err\n");
		return -1;
	}

	strcpy(main_p, "123456");


	return 0;
}
```



12

2015年11月12日 15:00

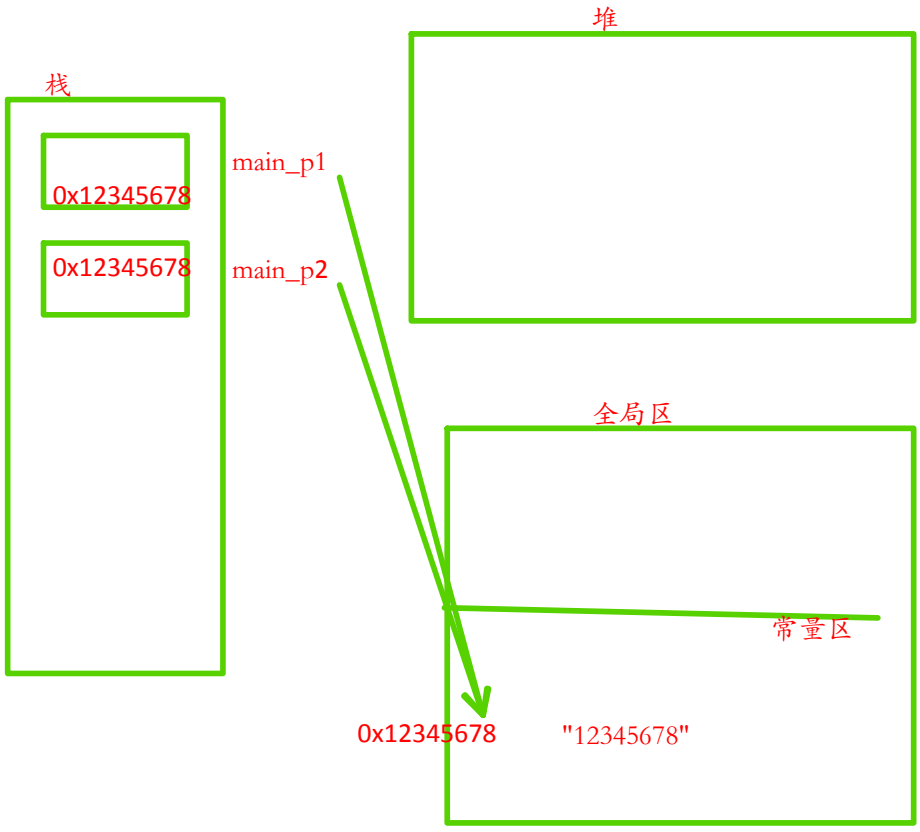



```
char* get_mem2(int
size)
{
    char buf[128]
    = { 0 };

    strcpy(buf,
    "123456");

    return buf;
}
//main_p = buf
```

2015年11月12日 15:10

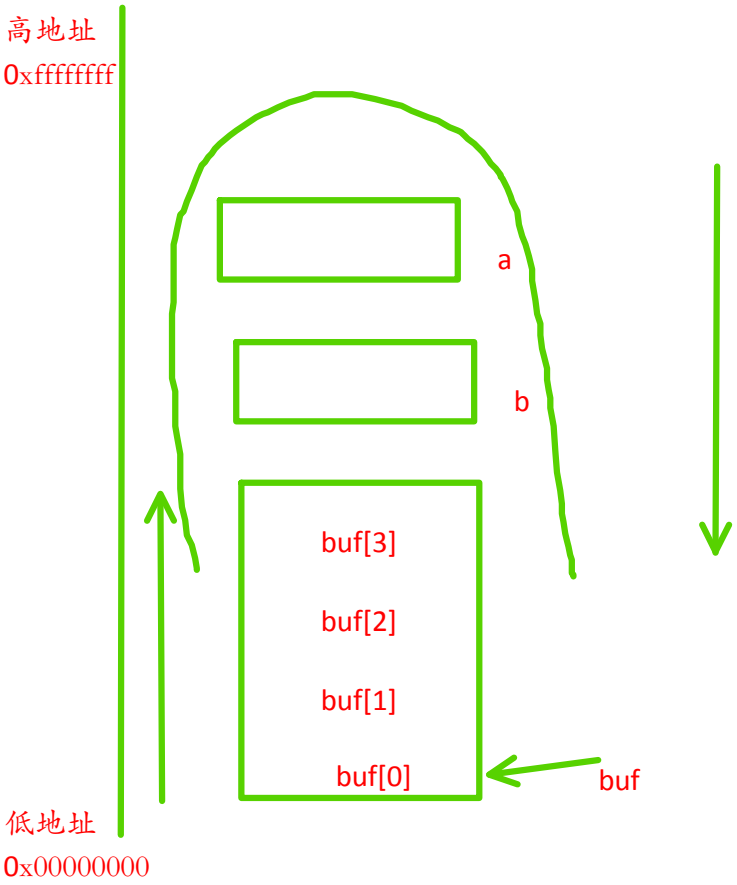
常量区

```
char *get_addr()
{
	char *p = "12345678";

	return p;
}

char *get_addr2()
{
	char *p = "12345679";

	return p;
}
main {

char *main_p1 = NULL;
char *main_p2 = NULL;

main_p1 = get_addr();
main_p2 = get_addr2();

if (main_p1 == main_p2) {
	printf("main_p1 和main_p2 指向同一个地址\n");
}
else {
	printf("main_p1 和main_p2不是 同一个地址\n");

}

}
```

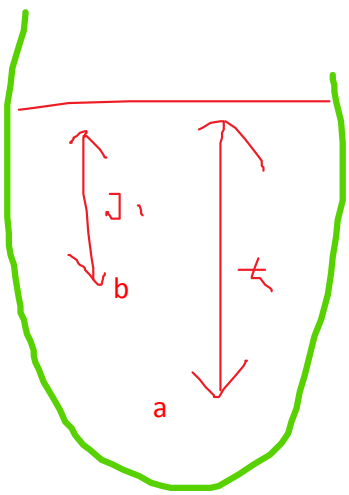


13 栈的开口方向

2015年11月12日 16:09

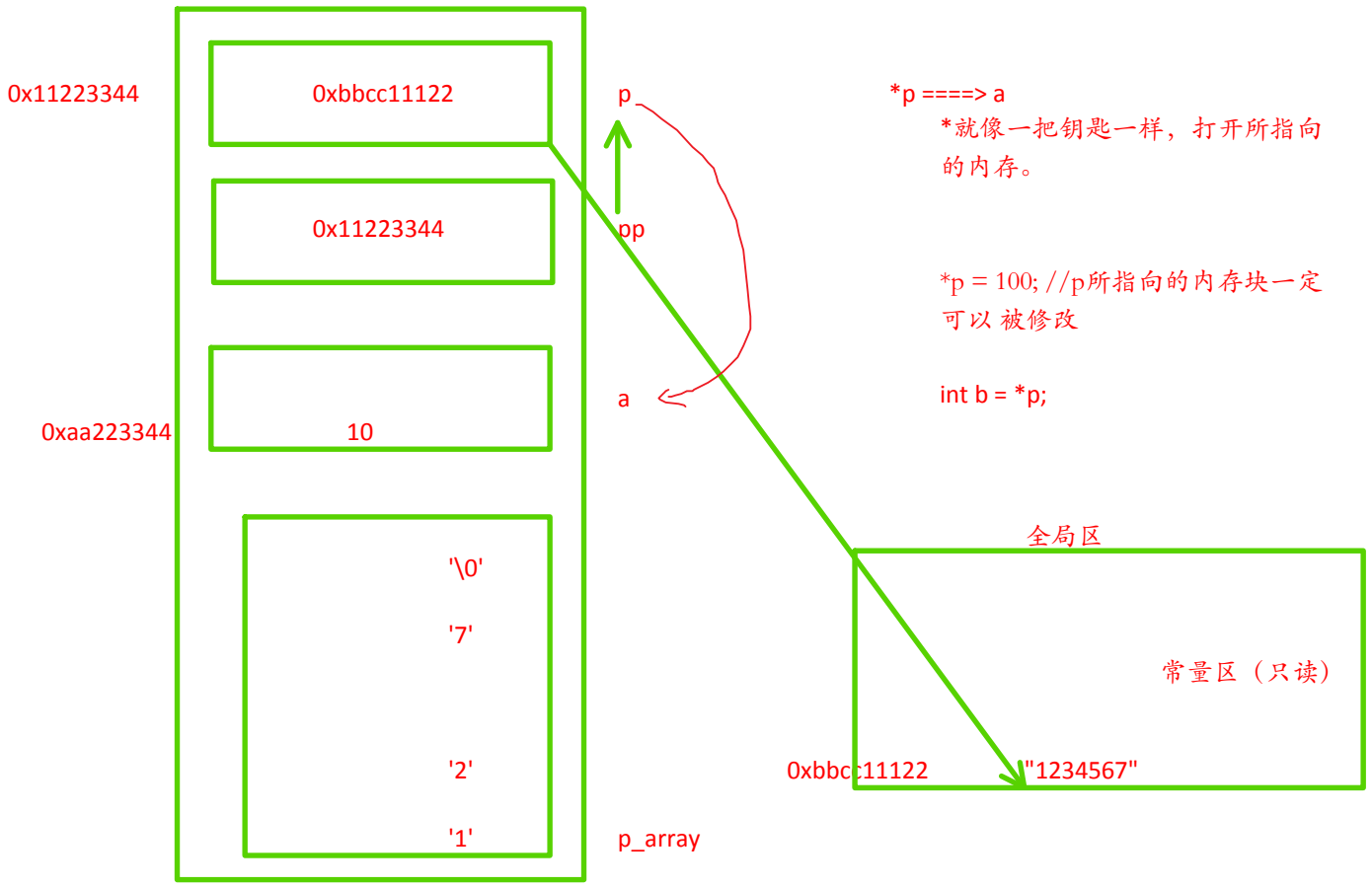


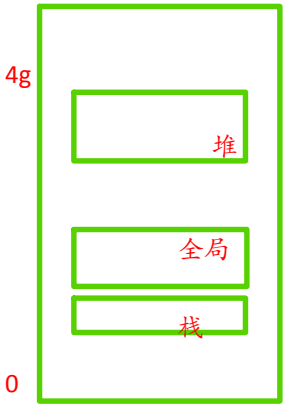
int a;
int b;



14 指针是一种数据类型

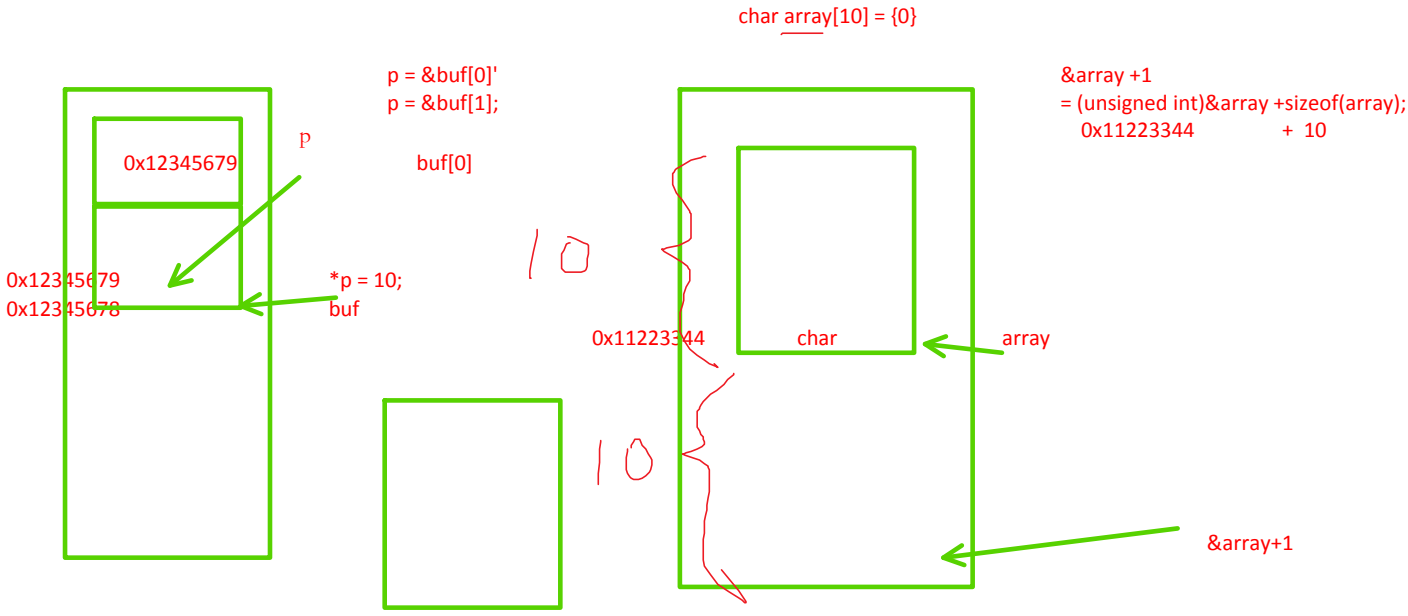
2015年11月12日 16:44





15 不断给指针赋值，是不断改变指针的方向

2015年11月12日 17:03



16 不断给指针赋值

2015年11月12日 17:13

```
p2 = (char *)malloc(100);
strcpy(p2, "abcdefg1212333333333311");
for (i=0; i<10; i++)
{
//不断的改变p1本身变量，跟p1指向的内存块无关

    p1 = p2+i;
    printf("%c ", *p1);
}
```