一、bzero

对单个字节的操作

```
void bzero(void *s, size_t n);
*s:指向字符串的指针，现在理解为字符数组
size_t n：置0多少位，以字节为单位。
```

二、memset

对单个字节的操作

```
void *memset(void *s, int c, size_t n);
void *s:现阶段理解为数组
int c：c是我们要置为什么数
size_t n：置多少个字节
```

```c
#include <stdio.h>
#include <strings.h>
#include <string.h>
int main(int argc, const char *argv[])
{
    char s1[] = "hello";
    //puts(s1);
    bzero(s1,3);
    memset(s1,'A',3);
    int arr[8] = {2,3,4,5,6,7,8,9};//定义了整型数组
    memset(arr,1,8);//对单个字节置1，0000 0001，一个int4字节，所以置了两个int
    printf("%s\n",s1+3);
    puts(s1);
    for (int i=0;i<8;i++)
    {
        printf("%d\n",arr[i]);
    }
    return 0;
}
```

输出：

```
lo
AAAlo
16843009
16843009
4
5
6
7
8
9
```

0000 0001 0000 0001 0000 0001 0000 0001

三、指针

指针变量保存的是，他指向的数据的首地址。

3.1定义

```
存储类型 数据类型 *指针变量名;
int *p; //定义了一个指针变量p
```

两个运算符：

1）&取地址符，去变量的地址。

2）*：

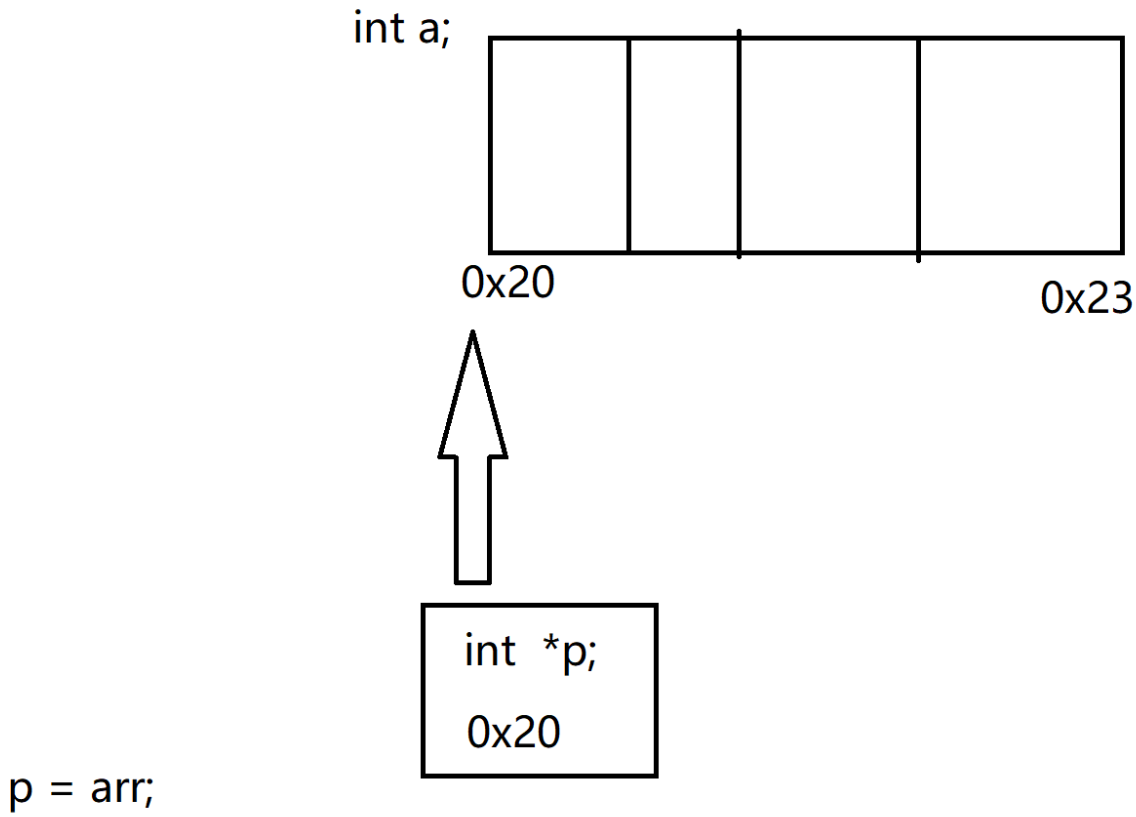
乘法运算符，

标识定义指针变量

解引用符，取地址中的元素。

3.2指针变量的初始化

```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int a=0;
    //int *p; //野指针
    int *p = &a; //用变量的地址，初始化指针
    //int *p = NULL; //定义一个指针指向空地址
    *p = 1;
    printf("%d\n",a);
    return 0;
}
```



3.3通过指针间接访问变量

```
int a = 20;
int *p = &a;
*p = 30;
再输出a，就变成30了，因为通过对指针的解引用操作，修改了指针指向的那块内存空间的值
```

3.4指针的大小

与数据类型无关，至于操作系统有关。

32位 4Byte

64位 8Byte

3.5指针的数据类型

指针的数据类型只是为了标识，+，-操作时偏移多大的空间，所以尽量与指针指向的变量的数据类型一致

3.6指针的运算

```
++  -- +  -  =
```

指针变量的数据类型，决定了他运算的时候能偏移几个字节的空间。

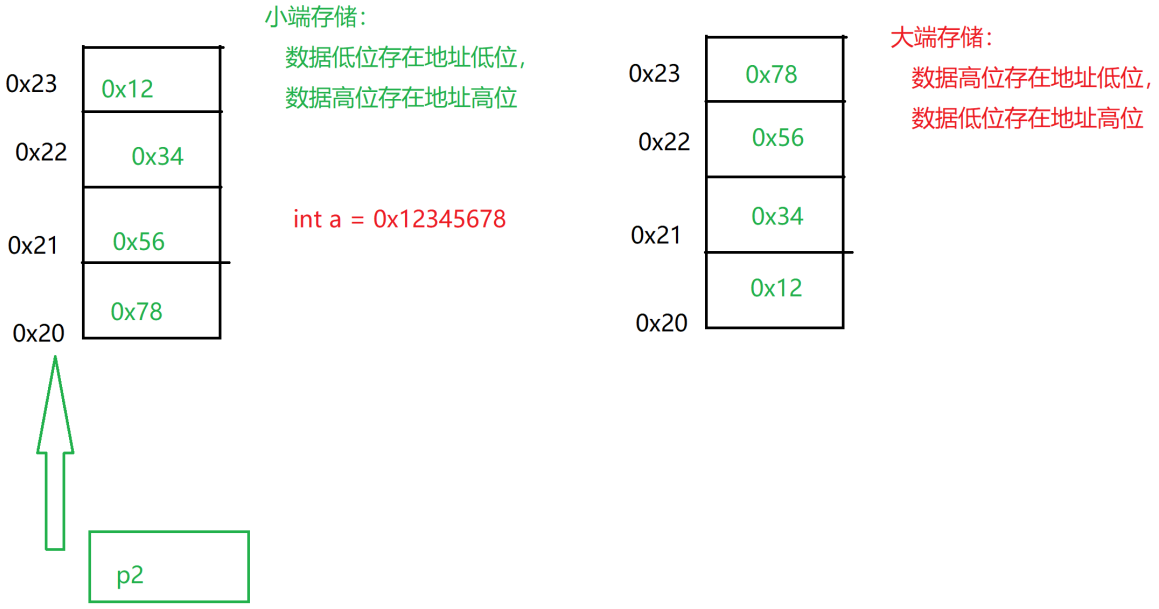
```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    //int *p; //野指针，间接访问发生段错误
```

```
    //int *p = NULL; //空指针，间接访问发生段错误
    int a =10;
    int *p = &a;//定义了一个指针变量p，指向a
    printf("%p\n",p);
    printf("%p\n",p+1);
    int *p1 = p;//定义了一个指针变量p1，指向a
    printf("%ld\n",sizeof(p));
    *p = 20; //把p这块地址里面的内容赋值为20
    int arr[5] = {1,2,3,4,5};
    int *p3 = arr;
    int i=0;
    for (i=0;i<5;i++)
    {
        //printf("%d\n",*(p3+i));
        //printf("%d\n",p3[i]);
        //printf("%d\n",*(arr+i));
        printf("%d\n",arr[i]);
    }
    (*arr)++;
    //p3++;  //p3是指针变量可以自增
    //arr++; //arr是一个常量不能自增，是数组首元素的地址。
    return 0;
}
```

大小端存储问题

```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    int a = 9999;
    int *p1 = &a;
    *p1 = 0x12345678;
    char *p2 = &a;
    printf("%#x  %#x  %#x  %#x\n",*p2,*(p2+1),*(p2+2),*(p2+3));
    return 0;
}
```

输出结果：0x78 0x56 0x34 0x12



3.7指针与一维数组

```
int arr[5] = {1,2,3,4,5};//定义数组
int *p3 = arr; //p3指向数组arr
int i=0;
    for (i=0;i<5;i++)  //访问数组的四种方式
    {
        //printf("%d\n",*(p3+i));
        //printf("%d\n",p3[i]);
        //printf("%d\n",*(arr+i));
        printf("%d\n",arr[i]);
    }
```

3.8指针与字符串

```
#include <stdio.h>
int main(int argc, const char *argv[])
{
    // != ==
    /*int a =0;
    int *p =&a;
    int *p1 = p;
    if (p!=p1)
    {
        printf("1");
    }*/
    //指针和字符串
    /*char s1[] = "hello";
    char *s = s1;
    //puts(s);
    printf("%s\n",s);
    int i;
    for (i=0;i<6;i++)
    {
        printf("%c",s[i]);
    }*/
    char *s = "hello world";
    *s = "hi";
    puts(s);

    return 0;
}
```

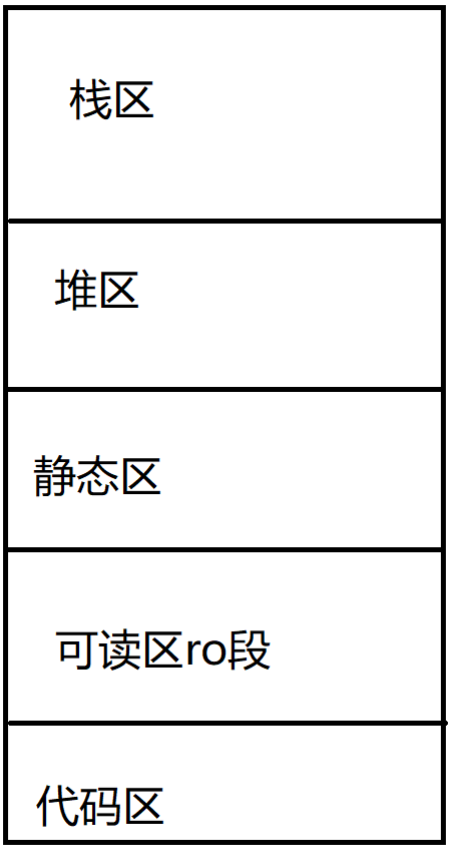
段错误：

1）数组越界

2）野指针的间接访问

3）空指针的间接访问

4）指针指向字符串常量，间接访问（修改）


栈区
这一片空间，由操作系统进行分配和回收。
堆区
程序员手动获取和释放
静态区
存放的是静态变量，全局变量
可读区ro段
常量
代码区
txt文本