【c陷阱与缺陷 笔记】

单目运算符 包括：！ ~ ++   --  -  (type)   \*  &  sizeof

它们的结合性从右向左

特别的，＊结合性从右向左 ，因此＊p＋＋  被解释成＊（p＋＋）

三目运算符？:  的结合性也是右向左，因此应该从右向左读。

else始终与最近的if结合，因此需要小心多个没有用括号嵌套的else if语句。如：

if()

     if()

          ……

else

          ....

如果没有括号，else是与最近的if结合，用括号可改变，达到想要的效果。

强制p指向一个内存地址位置，如0x500    p = （int \*）（0x500）；

无符号整数相加不会溢出，但是会被截断

首先在无符号表示中 a + 2^n = a;

如果a，b两个无符号整数，都未溢出：a < 2^n, b < 2^n, 且a+b > 2^n，

那么sum = a + b - 2^n ， 即sum - a = b - 2^n < 0==> sum < a，同理可得 sum < b。

而如果有符号整数相加会溢出，对于2个非负的有符号整数相加，可能会出现负数，判断溢出的方法可以采用：

1 转换为无符号整数

    if( (unsigned)a +(unsigned)b > INT\_MAX )

               ……

2 使用if(a > INT\_MAX - b)进行判断

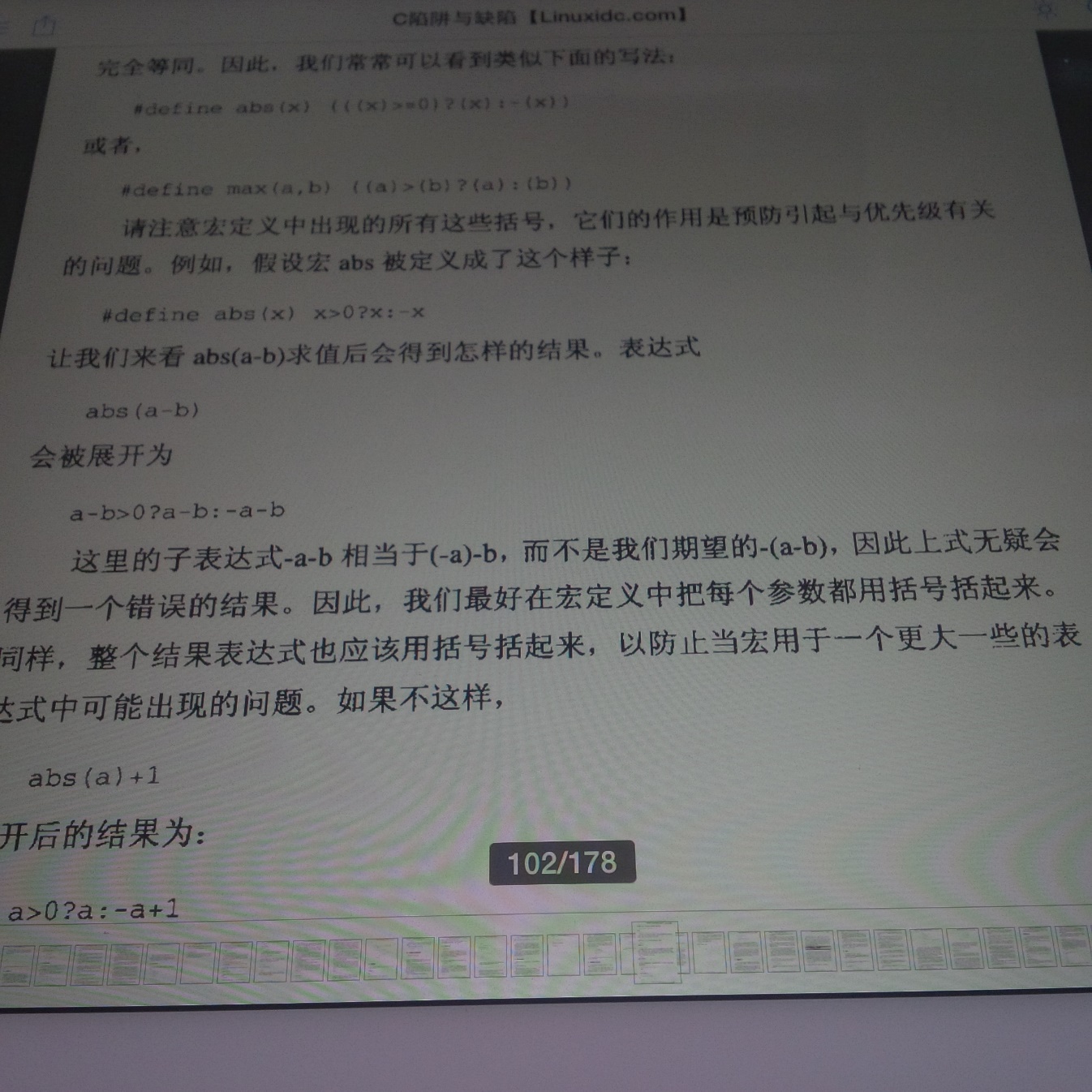
static作用：

1. 用static声明外部变量 static变量的作用域是所在的文件，不同于全局变量。因此可以隐藏变量，防止命名冲突。

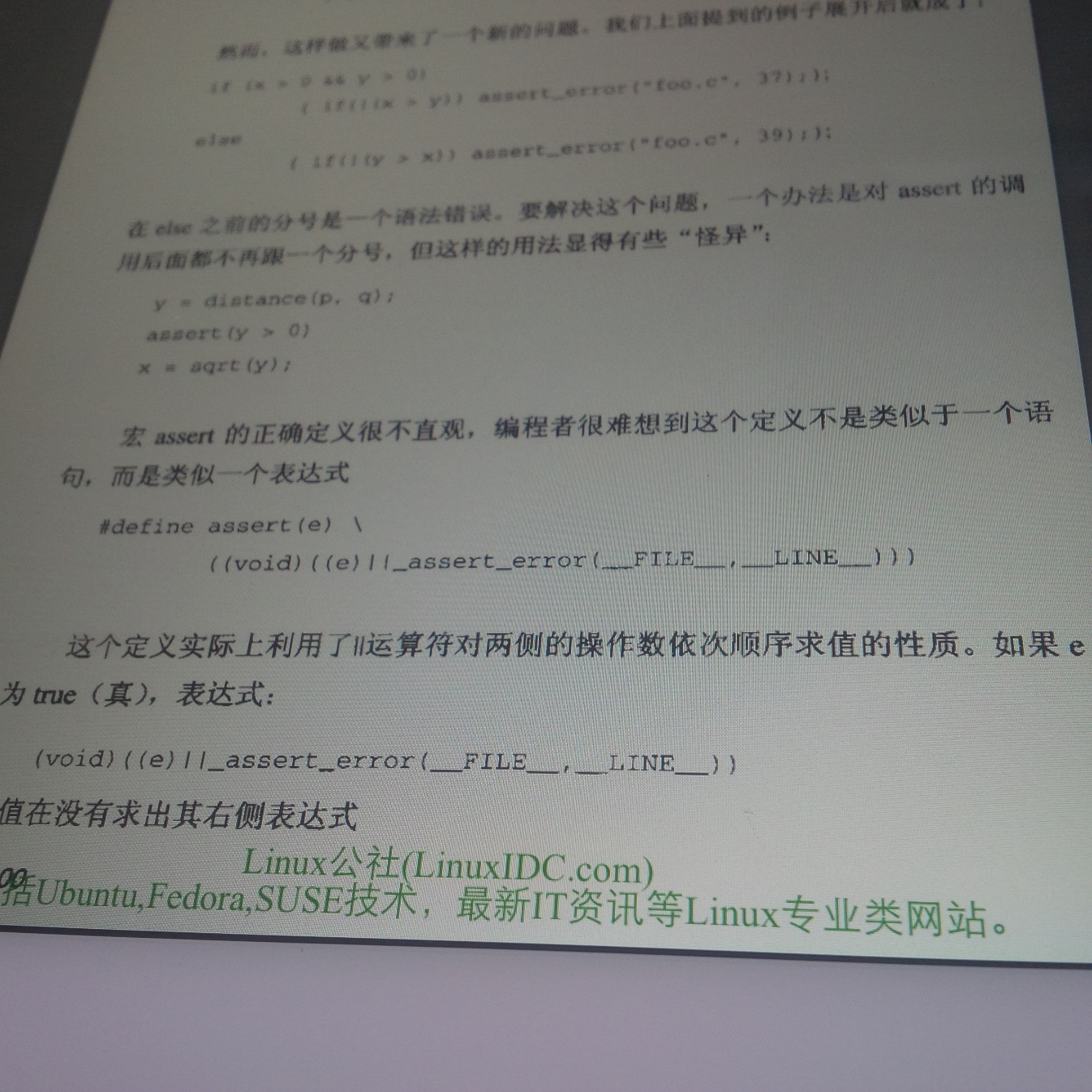
2. 和全局变量相同，变量放在静态存储区，并且默认初始化为0

getchar()函数的返回值是int 因为有些字符不是ANSIC的，故不要用char型变量存储返回值，防止截断

宏定义时候注意括号 不要混合三目运算防止a＋＋,具体如图所示



assert宏定义为如下图，是防止if展开错误



Typedef struct Foo \*T

而不是#define T struct Foo \*

以防止 T a b；展开为struct foo ＊a，b；

一般，typedef int \*p即可。 可以写成typedef int （＊p）但是不能typedef (int\*) p

类型名上不能加括号 如不能(int) a 或许说明int\*是一种数据类型？？？

char (\*pFun)(int) 定义函数指针

typedef char (\*PTRFUN)(int)

用括号将\*扩起来因为括号比 \*优先级高，否则就变成返回值为指针的函数了

外部函数大小写是否区分

对于\*指针靠左还是靠右的问题

一般情况下靠右边，int \*a, \*b可以防止定义b时候错误

对于函数返回值是指针时候，char\* fun(int) 星号靠右方便理解