_T和__T宏

```
ifdef _UNICODE
#define __TEXT(quote)   L##quote
#define __T(x)          L##x
...
#else
#define __TEXT(quote)   quote
#define __T(x)          x
...
#endif

#define _T(x)       __T(x)
```

需要由`_T到__T`中转一次的原因：

```
#define MY_MSG(x) #x
#define MSG(x) x

// 只用__T
__T(MSG(MY_MSG(hello))) -> LMSG(MY_MSG(hello))  // error

// 用_T
_T(MSG(MY_MSG(hello))) -> __T(MY_MSG(hello)) -> __T("hello") -> L"hello"
```

指针

直接访问与间接访问

- 直接访问
  - 找到变量地址，按照变量地址所存储的变量的值的类型做出解释
- 间接访问
  - 找到指针所指向的地址，对指针所存放的数据视为另一个地址，对另一个地址按数据变量所属类型做出解释

禁忌

- 指针定义要初始化，没有合适的值给**NULL**

指针的运算

```
type *ptr = ....;
int n = ...;

ptr[n] = *(type *)((int)ptr + sizeof(type) * n);
ptr + n = ptr + sizeof(type) * n;

type *ptr_a = ...;
type *ptr_b = ...;
ptr_a - ptr_b = ((int)ptr_a - (int)ptr_b) / sizeof(type);
```