重定向

符号

```
>       输出重定向到一个文件或设备，覆盖原来的文件
>!      输出重定向到一个文件或设备，强制覆盖原来的文件
>>      输出重定向到一个文件或设备，追加原来的文件
<       输入重定向到一个程序
```

标准错误重定向符号

```
2>      将一个标准错误输出重定向到一个文件或设备，覆盖原来的文件
2>>     将一个标准错误输出重定向到一个文件或设备，追加到原来的文件
2>&1    将一个标准错误输出重定向到标准输出
>&      将一个标准错误输出重定向到一个文件或设备，覆盖原来的文件
|&      将一个标准错误、管道、输送到另一个命令作为输入
```

作用域与生命期

	作用域	生命期	存放区域
局部	从定义处开始到函数结束	函数开始到函数结束	栈空间
块	从定义处开始到块结束	函数开始到函数结束	栈空间
全局	工程的任何地方	程序模块载入到卸载	数据区
静态全局	从定义处开始到文件结束	同全局	同全局
静态局部	从定义处开始到函数结束	同全局	同全局

注解

- 未赋初值的全局变量，不占磁盘空间
- 静态局部变量初始化为常量，在汇编层面上没有产生代码

```
void foo()
{
    static int var = 1234;  /* 汇编上不产生代码，赋值是直接从存储的地方拿
                               所以行为就跟“初始化一次，之后不在赋值”一样*/
    ....
}
```

- 如果静态局部变量初始化为变量的值

```
void foo(int n)
{
    static int var = n;
    ....
}

// 在C中无法通过编译，但是在C++中可以。保持行为一致，等效于以下代码
bit flag = 0;            // 低版本的在var附近，高版本考虑线程安全转移到其他地方了
void foo(int n)
{
    if(flag == 0) {      // 以标志位进行判断是否应该去初始化
        var = n;
        flag = 1;
    }
    ....
}
```

- 寄存器变量
  - 只有小于寄存器位数大小的**整型、字符型、指针**可以，浮点、静态、全局不行
  - 在VS中的行为：Debug，寄存器永远不够。Release，会优先使用寄存器

宏

作用

1. 符号化常量
   `#define PI 3.14`
2. 提高兼容性

```
// 在VC6与VS中写法会有兼容问题
for(int i = 0; i < 5; i++) {
    ...
}
for(int i = 0; i < 5; i++) {    // 如果在VC6中会报错，因为VC6这里的i是函数作用域
    ...
}

// 更改
#define for if(1)for            // 可以解决兼容问题，将i变为块作用域
for(int i = 0; i < 5; i++) {
    ...
}
for(int i = 0; i < 5; i++) {
    ...
}
```

3. 说明性

```
#define IN
#define OUT
// 标明传入与传出参数
int foo(IN int x, OUT int *y) {
    ...
}
```

4. 表达式
   `#define GET_AREA(r) ((r) * (r) * PI)`
   注意加**括号**
5. 功能性
   `#define SHOW_MAX(x, y) { printf("%d", (((x) > (y)) ? (x) : (y)); }`

预处理相关

include 包含

`#incldue`预处理不认文件后缀名，只做文本粘贴
头文件中不要有任何的**动作行为**（赋值、定义等）
头文件注意加防卫式声明

```
/***** file: foo.h *****/
#ifndef __FOO_H__
#define __FOO_H__
...
#endif  // !__FOO_H__
```

条件编译

```
#ifdef _DEBUG
    ...         // Debug版参与编译
#else
    ...         // 非Debug版参与编译
#end
```