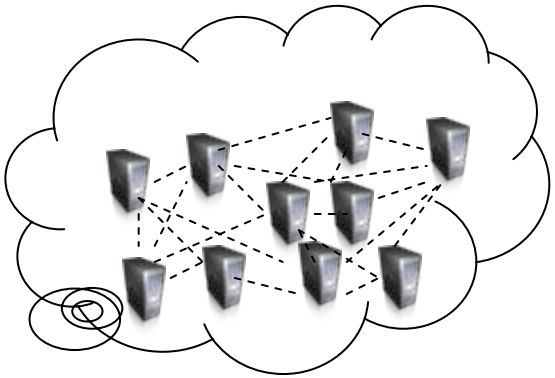
厦门大学非计算机专业本科生公共课

（2011-2012第2学期）



C语言程序设计

林子雨

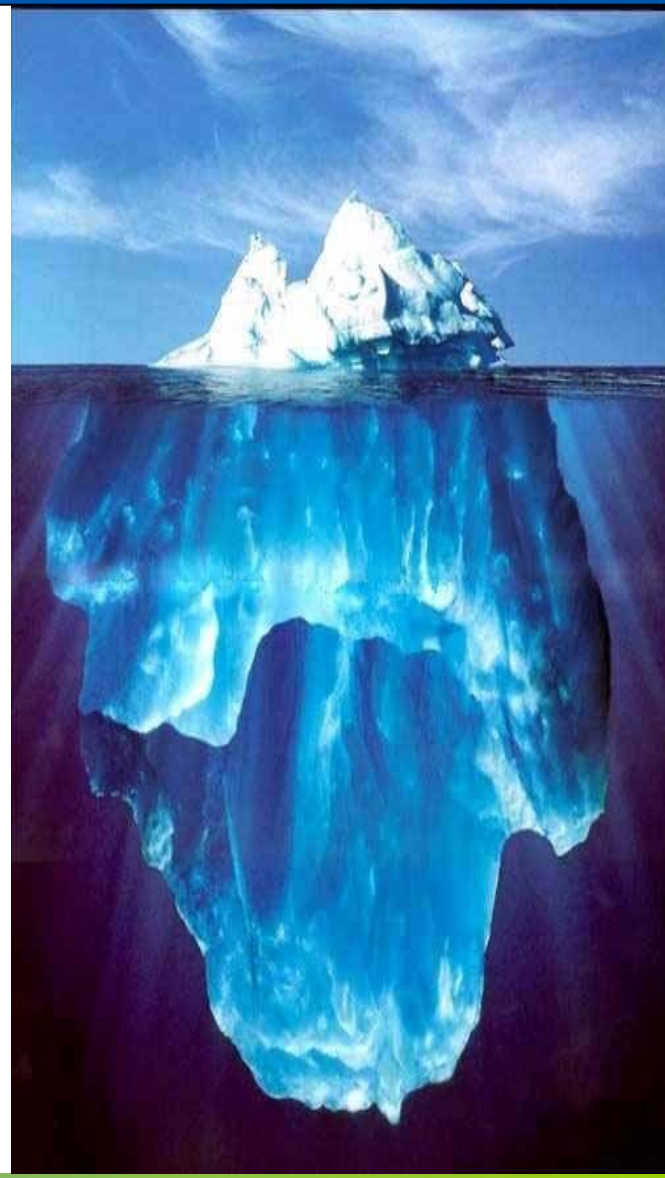
厦门大学计算机科学系

E-mail: ziyulin@xmu.edu.cn

个人主页：**http://www.cs.xmu.edu.cn/linziyu**

课程提要

- 第一章 绪论
- 第二章 C语言基础
- 第三章 结构化程序设计
- 第四章 选择结构
- 第五章 循环结构程序设计
- 第六章 函数
- 第七章 编译预处理
- 第八章 数组
- 第九章 结构体、共用体和枚举类型
- 第十章 指针



第7章 编译预处理

7.0 预处理命令
7.1 宏定义
7.2 条件编译(*)
7.3 文件包含
7.4 多文件组织

7.0 预处理命令

- 预处理命令：以"#"开头的命令；
- 预处理命令不是C语言的组成部分，但也是由ANSI C统一规定的，为了区别于一般的C语句，规定预处理命令必须从新的一行开始，以"#"开头，以回车符结束。
- 预处理命令不能被编译程序识别，必须在编译之前由专门的预处理程序进行转换。
- 现在的C编译系统一般都包括预处理、编译和链接等部分，C编译系统对C源程序的一般处理过程是：
  - 首先，运行预处理程序扫描源代码，对源程序中的预处理命令进行转换和处理；
  - 然后，运行编译程序，把源程序编译成目标代码；
  - 最后，运行链接程序，把目标代码链接成可执行文件。
- C语言提供了宏定义、文件包含、条件编译等多种预处理功能，有效扩展了C语言程序设计的环境，减少程序设计和维护工作量，增强程序可读性。

7.1 宏定义

宏提供了一种文本替换机制，C语言中用预处理命令#define定义宏。宏定义有不带参数和带参数两类，不带参数的宏定义实现简单文本替换，带参数的宏定义具有类似函数的功能。

- 7.1.1 不带参数的宏
- 7.1.2 带参数的宏
- 7.1.3 取消宏定义

7.1.1 不带参数的宏

- 定义形式：

  #define 标识符 [字符序列]

  例如：#define PI 3.14159

- 标识符就是宏的名字，字符序列可以为空，也可以是一串字符，用于在预处理时替代宏名，称作替换文本；标识符与字符序列之间应当用1个以上的空格或制表符隔开。
- 预处理时，预处理程序把源程序中出现在宏定义之后的所有宏名逐一替换成相应的替换文本，这样的替换过程成为“宏扩展”或“宏替换”。

  例如：s=PI*r*r;

  经过预处理被替换成：s=3.14159*r*r;

使用宏的好处：

- 提高程序的可读性，描述性的宏名有助于更好地理解对应的替换文本的含义和用途；
- 可以减少程序中同一个常量的重复书写，并方便对该变量的修改。

7.1.1 不带参数的宏

- 注意：
- （1）宏定义只能以“回车”结束，预处理程序将宏定义中从宏名之后的第一个非空白字符开始到换行符之前的所有字符作为替换文本。
- （2）如果宏定义超过一行，可以在该行行末加一个反斜杠”\”来续行。例如：

  #define LONG_STRING this is\

  not a very long string
- （3）如果在字符常量、字符串和注释中出现宏名，则不做扩展。例如：

  #define HI hello

  语句printf(“HI”);输出的仍然是HI，而不是hello。
- （4）允许嵌套使用宏，即一个宏名可以出现在另一个宏的替换文本中。例如：

  #define X 5

  #define Y X+1

  #define Z Y*X

  例子：a=Z;逐层替换：a=Y*X; →a=X+1*5; → a=5+1*5;
- （5）尽管宏名也是一个标识符，但它不是变量，不分配内存空间，因此，不能作为变量使用。
- （6）宏定义中可以没有替换文本，例如：#define EMPTY

  这种宏定义通常作为条件编译检测的一个标志。

7.1.2 带参数的宏

- 定义形式：
  #define 标识符(参数表) [字符序列]
- 参数表是一系列由逗号分隔的标识符，这些标识符的作用与C语言函数中形参类似；标识符与括号”(”之间不能有空格。
- 定义带参数的宏后，在后继的源程序中可以采用如下的类似函数调用带参数的宏：
  宏名(实参表)
  例如：#define MULT(a,b) a*b
  在后继程序中可以用下面语句调用该宏：
  printf(“%d\n”,MULT(1+2,3+4));
- 宏替换分两步：
  - 首先，用宏定义中的替换文本替换整个宏调用；
  - 然后，将替换文本中出现的各个形参分别用宏调用中对应的实参替换。

  则上面的语句被替换为：
  printf(“%d\n”,1+2*3+4);

7.1.2 带参数的宏

带参数宏和函数的区别

- （1）函数调用时，先对实参表达式求值，然后把实参的值赋给形参；而调用带参数宏时，只是进行单纯的文本替换。

  例如： printf(“%d\n”,MULT(1+2,3+4));

  为了保证宏扩展后表达式运算的正确性，有时需要对形式参数加括号或对整个替换文本加括号。

  例如：#define MULT(a,b) ((a)*(b))

- （2）函数调用时，要求实参和形参的数据类型一致，如不一致，系统自动把实参转换为形参的类型；而宏扩展不存在类型问题，宏名和它的参数都是一种符号表示，没有类型，宏扩展时只要代入相应的字符序列即可。宏扩展也没有返回值。

7.1.2 带参数的宏

带参数宏和函数的区别

例7.1.1：比较下面的MAX宏和max( )函数。

```
# include  <stdio.h>
# define  MAX(x,y)  ((x)>(y)?(x):(y))
int max(int x, int y) { return x>y?x:y;}
void  main()
{
    int     a=10, b=20;
    float  x=3.14, y=31.5;
    printf("%d\n",MAX(a,b));
    printf("%f\n",MAX(x,y));
    printf("%d\n",max(a,b));
    printf("%d\n",max(x,y));
    printf("%f\n",max(x,y));
 }
```

运行结果：

```
20
31.500000
20
31
0.000000
```

7.1.2 带参数的宏

带参数宏和函数的区别

- （3）函数调用是在程序运行时处理的，调用函数时需要为函数中定义的局部变量（包括形参）分配存储单元，把实参的值传递给形参，函数调用结束时要释放这些存储单元，返回函数值。而带宏参数的扩展在编译前进行，只做单纯的文本替换，不需要分配内存单元，不进行数据传递，也没有返回值的概念。

例7.1.2：比较下面的宏扩展和函数调用。

```
# include  <stdio.h>
# define CUBE(x) ((x)*(x)*(x))
int cube(int x) { return x*x*x ;}
void  main()
        {
                int         a=2,b;
                b=CUBE(a++);
                printf("%d\n",b);
                b=CUBE(a);
                printf("%d\n",b);
                a=2;
                b=cube(a++);
                printf("%d\n",b);
                b=cube(a);
                printf("%d\n",b);
        }
```

运行结果：
8
125
8
27

7.1.3 取消宏定义

- 如果在文件中不再使用某个在前面的文件首部已经定义的宏，可以取消该宏定义，语法形式为：

  #undef 宏名

- 使用#undef的主要目的是把宏名局部化，即把宏名的作用域局限于需要它的代码中；
- 宏可以重新定义，但在重新定义之前，应该用#undef命令把原来的定义取消，否则重新定义宏时系统会提出警告。

```
#define BLOCK_SIZE 512    //定义宏
…
#undef BLOCK_SIZE         //取消宏
#define BLOCK_SIZE 128    //重新定义宏
…
```

7.3 文件包含

- 文件包含是指，一个源文件可以将另一个源文件的内容全部包含进来，即把另一个文件插入到本文件中。
- C程序通常分成多个模块由多个程序员合作编写。对于程序中公用的符号常量定义、类型定义、外部变量声明、函数原型声明和宏定义等，通常单独组成文件，供各模块共享。这种供各模块共用的源文件，称为“标题文件”或“头文件”，扩展名为.h。
- 一般头文件不包含变量定义和函数定义。
- 采用头文件的方法，可避免在每个文件开头书写这些公用量，减少了程序设计人员的重复劳动。而且，当需要修改这些公用量时，也不必逐一修改各个文件，而只需要修改这个公用文件即可。

7.3 文件包含

- 文件包含操作由预处理命令#include实现，它指示预处理程序将指定文件的全部内容插到#include命令所在的位置。
- 命令格式为：

  #include<文件名>

  或

  #include”文件名”

  其中，文件名必须是一个完整的文件名（扩展名不可省略），可以包含路径，例如：

  #include “c:\mydir\prog.cpp”
- #include<文件名>命令指示预处理程序到Include文件夹中去查找所指定的文件，如果找不到指定文件，则显示出错信息；
- #include”文件名”命令指示预处理程序先到当前用户文件夹中查找，如果没有找到，再到Include文件夹中去查找，如果还找不到，则显示出错信息。

7.3 文件包含

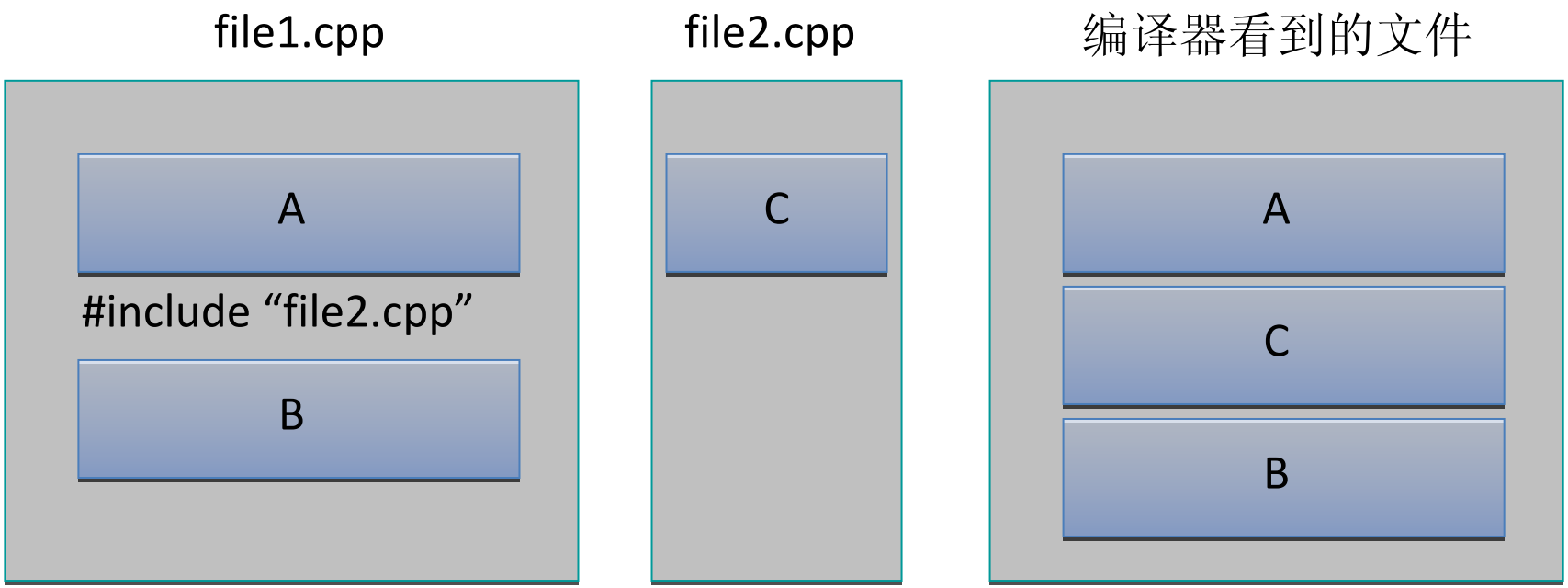


图7.3.1 #include 命令处理示意图

7.4 多文件组织

- 在结构化程序设计中，程序由若干模块组成，而C语言的基本模块就是函数。通常为了程序设计和调试，将一个大的程序分成多个源文件存储，每个源文件包含一个或多个函数（一个函数不能存储在两个文件中）。
- 当一个程序由多个源文件组成时，某源文件中定义的标识符（如全局变量名、函数名等）能否被其他源文件中的函数访问，取决于该标识符的连接属性。连接属性有两种：内部连接和外部连接。

- 7.4.1 内部连接
- 7.4.2 外部连接

7.4.1 内部连接

- 内部连接也称为静态连接，如果一个标识符（全局变量或函数）具有内部连接属性，那么它只能被本文件中的函数调用。这样的标识符称为文件级标识符。
- 定义一个标识符的内部连接属性的方法是：在标识符前面用关键字static修饰，即：static 标识符

例7.4.1：内部函数的非法调用示例。

```
//File1.cpp
static int iMax(int x, int y)
{
        return x>y?x:y ;
}
```

```
//File2.cpp
void main()
{
        int M;
        int iMax(int,int);
        M=iMax(6,2);
}
```

说明：由于static修饰，File2.cpp中的不能调用File1.cpp的iMax()函数。

7.4.2 外部连接

- 具有外部连接属性的标识符可以在其他文件中经过声明后被访问。
- 定义一个标识符的外部连接属性的方法是：在名字前面用关键字extern修饰，即：extern 标识符
- 外部连接属性是C语言默认的属性，在定义全局变量或函数时，如果省略连接属性定义，则隐含具有外部连接属性。具有外部连接属性的标识符是程序级的，即在程序的任何文件中都可调用它。

例7.4.2：外部函数调用示例。

```
//File3.cpp
extern int eMax(int x, int y)
// extern可以省略
 {
            return x>y?x:y ;
 }
```

```
//File4.cpp
void main()
{           int M;
            extern int eMax(int,int);
            // extern可以省略
            M=eMax(6,2);
}
```

说明：在File3.cpp中将eMax()定义为具有外部连接属性，在File4.cpp中eMax()函数经声明后可以被调用。可以将声明放在头文件中。

附件：课程教师和助教（2011-2012第2学期）

主讲教师：林子雨

单位：厦门大学信息科学与技术学院计算机科学系
办公地点：福建省厦门市思明区厦门大学海韵园
E-mail:ziyulin@xmu.edu.cn
个人主页：http://www.cs.xmu.edu.cn/linziyu

助教：林尚青

单位：厦门大学计算机科学系2010级硕士研究生
E-mail:lsq1015@qq.com
手机：15959206201

助教：赖明星

单位：厦门大学计算机科学系2011级硕士研究生
E-mail:joy_lmx@163.com
手机：18050056577

附件：课程FTP （2011-2012第2学期）

- FTP地址：ftp://218.193.53.74
- 用户名： stu_linziyu
- 密码：123456
- 目录：“下载教学内容”→”C语言”

附件：课程教材 （2011-2012第2学期）

- 《C语言程序设计（第2版）》
- 清华大学出版社，黄保和，江弋 编著
- 版次：2011年10月第2版
- ISBN:978-7-302-26972-4
- 定价：35元

Department of Computer Science, Xiamen University, April 15, 2012