西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

编程任务的自动化

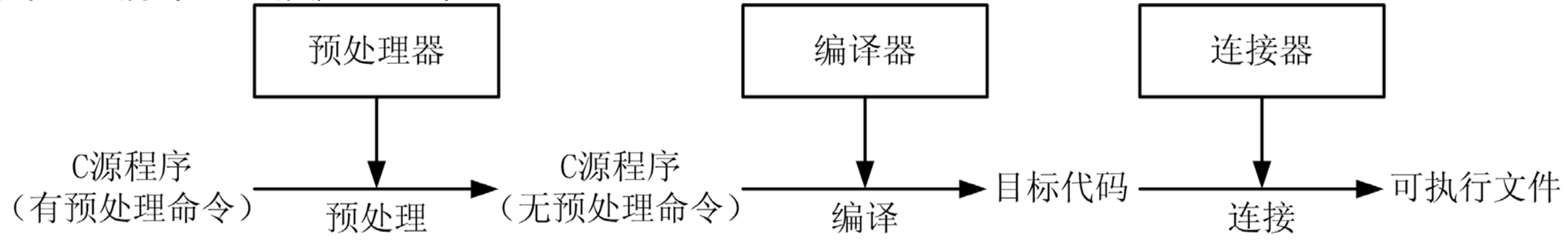
1、预处理命令

2、不带参数的宏定义

第5章 预处理命令

- C语言程序编译的处理过程一般为：预处理、编译（及汇编）和连接。

图5.1 编译、连接处理过程



第5章 预处理命令

- 预处理（preprocess）是在程序编译之前，由预处理器（preprocessor）对源程序中各种预处理命令进行先行处理，处理完毕自动进入对源程序的编译。

- 编译时先分析、后综合，分析是指编译器对源程序进行词法分析和语法分析；综合是指代码优化、存储分配和代码生成。为了完成这些分析综合任务，编译器采用对源程序进行多次扫描的办法，每次扫描集中完成一项或几项任务，也有一项任务分散到几次扫描去完成的。

- 大多数的编译器直接产生机器语言的目标代码，但有的编译器则先产生汇编语言的符号代码，然后调用汇编器（assembler）进行翻译加工处理，产生目标代码。
- 连接器将在不同文件中的目标代码和库函数代码经过重定位处理，连接成可执行文件。

- 预处理命令不是C语言本身的组成部分，更不是C语言语句，它是C语言标准规定的可以出现在C源程序文件中的命令。

- 这些命令必须以“#”开头，结尾不加分号，可以放置在源程序中的任何位置，其有效范围是从出现位置开始到源程序文件末尾。

- 预处理命令的操作对象是编译器和连接器，用来设置程序编译和连接时的各种参数，当编译工作完成后，预处理命令的作用即告完成，因而它不会出现在目标代码和可执行文件中。预处理命令是C语言的一个重要特点，合理地使用预处理功能，可以优化程序设计环境，提高程序的通用性、可读性和可移植性。

- C语言标准提供了多种预处理命令，如宏定义、文件包含、条件编译等。

5.1 宏定义

- 在C源程序中允许用一个标识符来代表一个字符文本，称为宏，标识符为宏名。宏是由宏定义命令事先定义的。预处理时，对程序中所有后续的宏名实例（称为宏引用），预处理器都用字符文本去替换，称为宏替换或宏展开。

- 宏定义通常用于定义程序中的符号常量、类型别名、运算式代换、语句代换等，其命令为#define，分为不带参数的宏定义和带参数的宏定义。

5.1.1 不带参数的宏定义

- 不带参数的宏定义的命令形式为：

```
#define 宏名 字符文本
```

- 其中：
- ①宏名按标识符语法取名，习惯上用大写字母，以便与变量等其他名称有所区别；
- ②字符文本可以为C语言允许的标识符、关键字、常量数值、表达式及各种符号等；
- ③宏名两侧至少用一个空白符间隔，且这个空白符不属于字符文本。

5.1.1 不带参数的宏定义

- 预处理时，程序中所有的宏名将被字符文本完全替换，然后再进行编译。例如有宏定义

```
#define PI  3.1415926
```

- 那么程序代码：

```
L=2*PI*r; //PI宏引用
```

- 在预处理时宏替换为：

```
L=2*3.1415926*r;
```

5.1.1 不带参数的宏定义

- 又如有宏定义

```
#define M y*y+5*y
```

- 那么程序代码：

```
S=3*M+4*M+5*M; //M宏引用
```

- 在预处理时宏替换为：

```
S=3*y*y+5*y+4*y*y+5*y+5*y*y+5*y;
```

5.1.1 不带参数的宏定义

- 使用宏定义，可以减少程序中重复书写某些字符文本的工作量，不容易出错；而且程序员习惯性将常量值定义为符号常量，无论是编写程序或是阅读程序都有记忆简单、见其名知其意的优点。

5.1.1 不带参数的宏定义

【例5.1】

计算半径为r的圆周长、圆面积、圆球表面积、圆球体积。

5.1.1 不带参数的宏定义

例5.1

```
#include <stdio.h>
#define PI  3.1415926 //不带参数的宏定义
int main()
{
   double r,L,S,SQ,V;
   scanf("%lf",&r); //输入半径
   L=2*PI*r; //计算圆周长
   S=PI*r*r; //计算圆面积
   SQ=4.0*PI*r*r; //计算圆球表面积
   V=4.0*PI*r*r*r/3.0; //计算圆球体积
   printf("L=%lf,S=%lf,SQ=%lf,V=%lf\n",L,S,SQ,V);
   return 0;
}
```

5.1.1 不带参数的宏定义

例5.1

```
1 #include <stdio.h>
```

程序运行屏幕

```
L=12.566370,S=12.566370,SQ=50.265482,V=3
3.510321
```

```
2.0↙
```

5.1.1 不带参数的宏定义

- 使用不带参数的宏定义需要注意：
- （1）宏定义用宏名来代表一个字符文本，在宏替换时又以该字符文本取代宏名，这只是一种简单的替换。
- 字符文本中可以包含任何字符，预处理器对它不作任何语法检查，即使有错误，也只有在编译已经宏替换后的源程序时才会发现。
- 因此不要在字符文本中放置任何多余的字符，比如在行末加分号，否则它们也将作为字符文本的组成部分。

5.1.1 不带参数的宏定义

- 例如：

```
#define PI  3.1415926;
S=2*PI*r; //错误 宏展开为 S=2*3.1415926;*r;
```

- 特别地，C语言标准允许在宏定义的字符文本后面出现C语言注释，例如：

```
#define E 2.718281828 //自然对数
#define G 9.8 //重力加速度
```

- 预处理时会忽略所有这样的注释。

5.1.1 不带参数的宏定义

- （2）宏替换时使用完整的字符文本替换宏名，既不会少也不会增加额外的字符。因此一些运算式代换中，字符文本中有无括号对宏替换的结果是有影响的。例如：

```
#define M1 a+b
#define M2 (a+b)
L=M1*M1; //宏展开为 L=a+b*a+b;
L=M2*M2; //宏展开为 L=(a+b)*(a+b);
```

5.1.1 不带参数的宏定义

- （3）源程序中的字符串常量、注释或标识符的一部分若有与宏名相同的字符，不会进行宏替换。假定已定义PI宏，则：

```
printf("PI=%lf\n",xPI*y); //xPI是变量名
```

- 不进行任何宏替换，因为代码中出现的PI均不是宏名。

5.1.1 不带参数的宏定义

- （4）一个宏名不要重复定义两次以上，否则引用的宏总是最后一次的宏定义；若要进行新的宏定义，应该先使用#undef命令。

5.1.1 不带参数的宏定义

- #undef命令的作用是取消已有的宏定义，形式为：

```
#undef  宏名
```

5.1.1 不带参数的宏定义

- 取消以后的宏名不再有定义，程序中若继续引用它将导致错误。例如：

```
1 #define MXY (x*x+y*y)
2 #define MXY (x*x+2*x*y+y*y) //重复宏定义
3 printf("%lf\n",MXY); //MXY宏展开为 (x*x+2*x*y+y*y)
4 #define MAB a+b
5 printf("%lf\n",MAB); //MAB宏展开为 a+b
6 #undef MAB  //取消MAB宏定义
7 printf("%lf\n",MAB-2); //错误，MAB未定义
8 #define MAB a*b //重新定义MAB
9 printf("%lf\n",MAB); //MAB宏展开为 a*b
```

5.1.1 不带参数的宏定义

- （5）一个宏的作用范围是从定义位置开始到#undef命令结束，如果没有对应的#undef命令，则到源程序文件末尾结束。通常，将宏定义写在源程序文件的开头或头文件中。

5.1.1 不带参数的宏定义

- （6）宏定义允许嵌套，即在宏定义的字符文本中可以引用已经定义的宏名，在宏替换时由预处理器层层替换。例如：

```
#define WIDTH 80
#define LENGTH (WIDTH+10)
L=WIDTH*a; //宏展开为 L=80*a;
S=LENGTH*b*WIDTH ; //宏展开为 S=(80+10)*b*80;
```


程序设计