# BIT-0-程序环境和预处理

### 版权©比特科技

本章重点：

程序的翻译环境程序的执行环境

详解：C语言程序的编译+链接预处理指令 #define

宏和函数的对比

预处理操作符#和##的介绍命令定义

预处理指令 #include

预处理指令 #undef

条件编译

# 正文开始

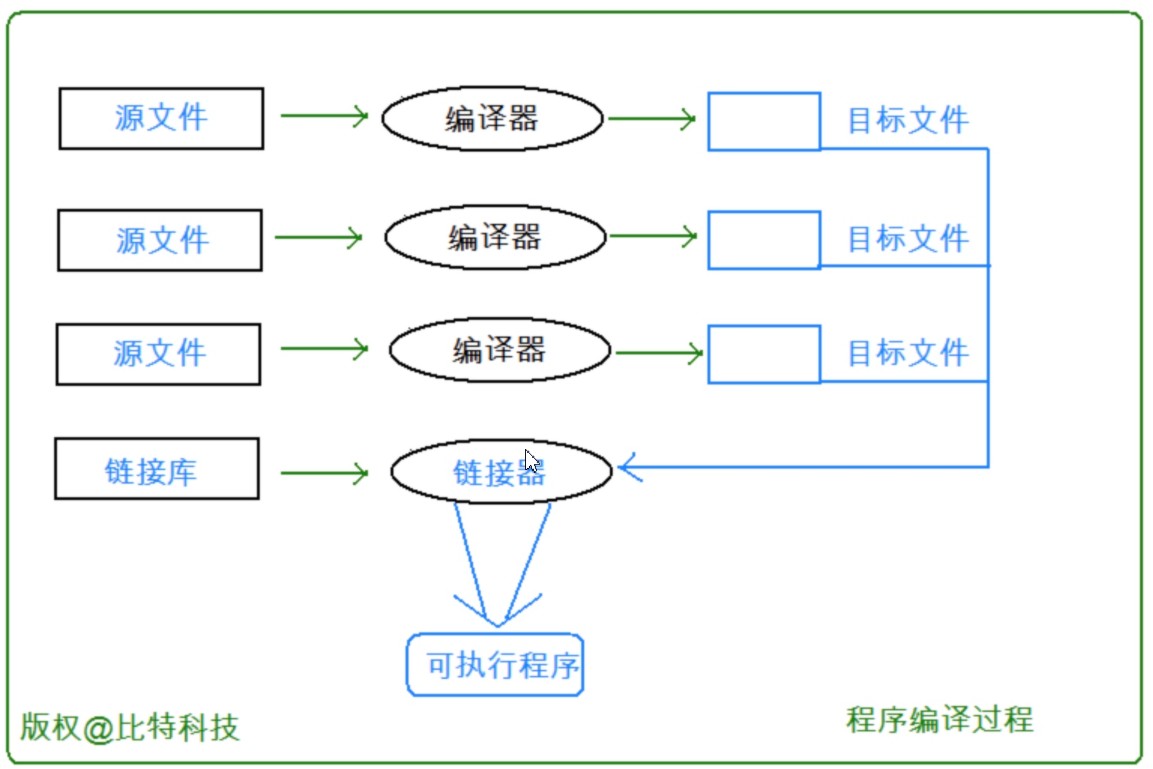
## 程序的翻译环境和执行环境

在ANSI C的任何一种实现中，存在两个不同的环境。

第1种是翻译环境，在这个环境中源代码被转换为可执行的机器指令。 第2种是执行环境，它用于实际执行代码。

## 详解编译+链接

### 翻译环境



组成一个程序的每个源文件通过编译过程分别转换成目标代码（object code）。每个目标文件由链接器（linker）捆绑在一起，形成一个单一而完整的可执行程序。

链接器同时也会引入标准C函数库中任何被该程序所用到的函数，而且它可以搜索程序员个人的程序 库，将其需要的函数也链接到程序中。

**编译本身也分为几个阶段组成**看代码： sum.c

int g\_val = 2016;

void print(const char \*str)

{

printf("%s\n", str);

}

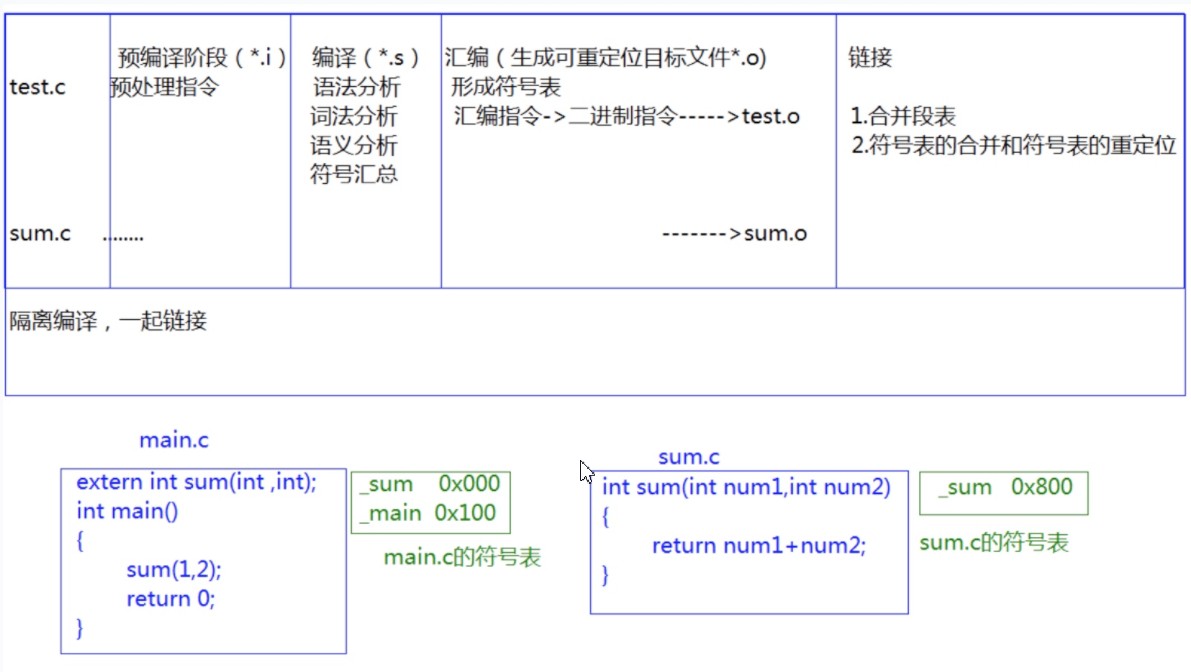
test.c

#include <stdio.h> int main()

{

extern void print(char \*str); extern int g\_val; printf("%d\n", g\_val); print("hello bit.\n"); return 0;

}



如何查看编译期间的每一步发生了什么呢？

test.c

#include <stdio.h> int main()

{

int i = 0; for(i=0; i<10; i++)

{

printf("%d ", i);

}

return 0;

}

1. 预处理 选项中。

gcc -E test.c -o test.i

1. 编译 选项

gcc -S test.c

1. 汇编

gcc -c test.c

VIM学习资料

简明VIM练级攻略：

预处理完成之后就停下来，预处理之后产生的结果都放在test.i文件

编译完成之后就停下来，结果保存在test.s中。汇编完成之后就停下来，结果保存在test.o中。

<https://coolshell.cn/articles/5426.html>给程序员的VIM速查卡<https://coolshell.cn/articles/5479.html>

### 运行环境

程序执行的过程：

1. 程序必须载入内存中。在有操作系统的环境中：一般这个由操作系统完成。在独立的环境中，程序的载入必 须由手工安排，也可能是通过可执行代码置入只读内存来完成。
2. 程序的执行便开始。接着便调用main函数。
3. 开始执行程序代码。这个时候程序将使用一个运行时堆栈（stack），存储函数的局部变量和返回地址。程序 同时也可以使用静态（static）内存，存储于静态内存中的变量在程序的整个执行过程一直保留他们的值。
4. 终止程序。正常终止main函数；也有可能是意外终止。

**注：** 介绍一本书《程序员的自我修养》

## 预处理详解

### 预定义符号

FILE

//进行编译的源文件

LINE //文件当前的行号

DATE //文件被编译的日期

TIME //文件被编译的时间

STDC //如果编译器遵循ANSI C，其值为1，否则未定义

这些预定义符号都是语言内置的。 举个栗子：

printf("file:%s line:%d\n", FILE , LINE );

## #deﬁne

### #deﬁne 定义标识符

语法：

#define name stuff

举个栗子：

#define MAX 1000 #define reg register

#define do\_forever for(;;)

#define CASE break;case

//为 register这个关键字，创建一个简短的名字

//用更形象的符号来替换一种实现

//在写case语句的时候自动把 break写上。

// 如果定义的 stuff过长，可以分成几行写，除了最后一行外，每行的后面都加一个反斜杠(续行符)。

#define DEBUG\_PRINT printf("file:%s\tline:%d\t \

date:%s\ttime:%s\n" ,\

FILE , LINE , \

DATE , TIME )

### 提问：

在deﬁne定义标识符的时候，要不要在最后加上 ; ?

比如：

#define MAX 1000;

#define MAX 1000

建议不要加上 ; ,这样容易导致问题。 比如下面的场景：

if(condition)

max = MAX; else

max = 0;

这里会出现语法错误。

### #deﬁne 定义宏

#deﬁne 机制包括了一个规定，允许把参数替换到文本中，这种实现通常称为宏（macro）或定义宏

（deﬁne macro）。

下面是宏的申明方式：

#define name( parament-list ) stuff

parament-list

在stuﬀ中。

其中的

是一个由逗号隔开的符号表，它们可能出现

**注意：** 参数列表的左括号必须与name紧邻。 如果两者之间有任何空白存在，参数列表就会被解释为stuﬀ的一部分。

如：

#define SQUARE( x ) x \* x

这个宏接收一个参数 . 如果在上述声明之后，你把



x

SQUARE( 5 );

置于程序中，预处理器就会用下面这个表达式替换上面的表达式：

5 \* 5

**警告：** 这个宏存在一个问题： 观察下面的代码段：

int a = 5;

printf("%d\n" ,SQUARE( a + 1) );

乍一看，你可能觉得这段代码将打印36这个值。 事实上，它将打印11. 为什么？

替换文本时，参数x被替换成a + 1,所以这条语句实际上变成了： printf ("%d\n",a + 1 \* a + 1 );

这样就比较清晰了，由替换产生的表达式并没有按照预想的次序进行求值。在宏定义上加上两个括号，这个问题便轻松的解决了：

#define SQUARE(x) (x) \* (x)

这样预处理之后就产生了预期的效果：