

第 1 章 预处理

1.1 c 语言编译过程

```
gcc -E hello.c -o hello.i    1、预处理
gcc -S hello.i -o hello.s    2、编译
gcc -c hello.s -o hello.o    3、汇编
gcc hello.o -o hello_elf     4、链接
```

1: 预编译

将.c 中的头文件展开、宏展开
生成的文件是.i 文件

2: 编译

将预处理之后的.i 文件生成 .s 汇编文件

3、汇编

将.s 汇编文件生成.o 目标文件

4、链接

将.o 文件链接成目标文件

预处理有几种啊？

1.2 include

#include <> //用尖括号包含头文件，在系统指定的路径下找头文件

#include "" //用双引号包含头文件，先在当前目录下找头文件，找不到，再到系统指定的路径下找。

注意：include 经常用来包含头文件，可以包含 .c 文件，但是大家不要包含.c

因为 include 包含的文件会在预编译被展开，如果一个.c 被包含多次，展开多次，会导致函数重复定义。
所以不要包含.c 文件。

注意：预处理只是对 include 等预处理操作进行处理并不会进行语法检查
这个阶段有语法错误也不会报错，第二个阶段即编译阶段才进行语法检查。

1.3 define

定义宏用 define 去定义

宏是在预编译的时候进行替换。

1、不带参宏

```
#define PI 3.14
```

在预编译的时候如果代码中出现了 PI 就用 3.14 去替换。

宏的好处：只要修改宏定义，其他地方在预编译的时候就会重新替换。

做真实的自己，用良心做教育

注意：宏定义后边不要加分号。

例 2：

```
#define PI 3.1415926

int main()
{
    double f;
    printf("%lf\n",PI);

    f=PI;
    return 0;
}
```

宏定义的作用范围，从定义的地方到本文件末尾。

如果想在中间终止宏的定义范围

#undef PI //终止 PI 的作用

例 3：

```
#define PI 3.1415926

int main()
{
    double f;
    printf("%lf\n",PI);

#undef PI

#define PI 3.14

    f=PI;
    return 0;
}
```

2、带参宏

#define S(a,b) a*b

注意带参宏的形参 a 和 b 没有类型名，

S(2,4) 将来在预处理的时候替换成 实参替代字符串的形参，其他字符保留， $2 * 4$

例 4：

```
#define S(a,b) a*b

int main(int argc, char *argv[])
{
```

做真实的自己，用良心做教育

```
int num;  
num=S(2,4);  
  
return 0;  
}
```

S(2+4,3)被替换成 2+4 * 3

注意：带参宏，是在预处理的时候进行替换
解决歧义方法

例 5：

```
#define S(a,b) (a)*(b)
```

```
int main(int argc, char *argv[])  
{  
    int num;  
    num=S(2+3,5); //(2+3 ) *(5)  
  
    return 0;  
}
```

3、带参宏和带参函数的区别

带参宏被调用多少次就会展开多少次，执行代码的时候没有函数调用的过程，不需要压栈弹栈。所以带参宏，是浪费了空间，因为被展开多次，节省时间。

带参函数，代码只有一份，存在代码段，调用的时候去代码段取指令，调用的时候要压栈弹栈。有个调用的过程。所以说，带参函数是浪费了时间，节省了空间。

带参函数的形参是有类型的，带参宏的形参没有类型名。

1.4 选择性编译

1、

```
#ifdef AAA  
    代码段一  
#else  
    代码段二  
#endif
```

如果在当前.c ifdef 上边定义过 AAA，就编译代码段一，否则编译代码段二

注意和 if else 语句的区别，if else 语句都会被编译，通过条件选择性执行代码
而 选择性编译，只有一块代码被编译

做真实的自己，用良心做教育

例:6 :

```
#define AAA

int main(int argc, char *argv[])
{
    #ifdef AAA
        printf("hello kitty!!\n");
    #else
        printf("hello 千锋 edu\n");
    #endif

    return 0;
}
```

2、

```
#ifndef AAA
    代码段一
#else
    代码段二
#endif
```

和第一种互补。

这种方法，经常用在防止头文件重复包含。

防止头文件重复包含：

3、

```
#if 表达式
    程序段一
#else
    程序段二
#endif
```

如果表达式为真，编译第一段代码，否则编译第二段代码

选择性编译都是在预编译阶段干的事情。