# 目录

[目录 1](#_Toc520877880)

[第7章 C语言-预处理 2](#_Toc520877881)

[7.1 预处理 2](#_Toc520877882)

[7.1.1 宏定义 2](#_Toc520877883)

[7.1.2 条件编译 2](#_Toc520877884)

[7.2 关键字extern 2](#_Toc520877885)

[7.3 关键字static 2](#_Toc520877886)

[7.4 关键字typedef 2](#_Toc520877887)

# C语言-预处理

## 内存分区

在C语言中,有四大分区

1. 常量区
2. 主要用来存放常量数据、程序代码、字符串 char \*p = “dfsfdf”; char a[20] = “sdfdsfsd”;
3. 只能进行读操作，不能进行写操作
4. 静态区
5. 主要存放全局变量、静态局部变量(static)
6. 可以进行读写操作
7. 栈区
8. 主要存放局部变量、函数形参
9. 可以进行读写操作
10. 堆区
11. 主要存放程序员申请的空间 char a;//系统申请 malloc()函数
12. 空间的类型和空间的大小由程序员决定，申请成功后，返回该空间的首地址
13. 可以进行读写操作

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int c;//静态区  int main(void)  {  char \*p = "dfsgfg";//常量区  int a; //栈区  static int d;  int \*f = (int \*)malloc(4);//堆区  printf("常量区 = %#p\r\n", p);  printf(" 栈区 = %#p\r\n", &a);  printf("静态区 = %#p\r\n", &c);  printf(" 堆区 = %#p\r\n", f);  return 0;  } |

## 预处理

编译过程：预处理、编译、汇编、链接 C语言 🡪 机器语言

预处理：由#开头，都是预处命令 之前学过预处理：头文件包含 #include <stdio.h>

### 宏定义

#define 🡪 宏定义命令

由#开头，后面紧跟着define，这个形式就是宏定义， 宏定义又分为(宏名)带参宏和无参宏

1. 无参宏
2. #define 宏名 头文件模板：#define \_LED\_H\_

就是单纯定义一个名字，通常跟条件编译配合使用

1. #define 宏名 表达式

宏名：满足标识符的命名规则，具有含义，一般大写

表达式：常量、变量、表达式、简单的程序代码

宏定义的本质：文本替换，在预处理后，表达式就替换宏名

注意：

1. 单文件：一般放在头文件包含和主函数之间；在多文件中，放在.h文件中
2. 宏定义后面不要加分号
3. 方便书写
4. 方便查看代码
5. 方便修改

|  |
| --- |
| #define PI 3.14159264651564161  int main(void)  {  double s = 0, r = 2.5;  s = PI \* r \* r;  printf("%lf\r\n", s);  return 0;  } |

1. 带参宏

用法：#define 宏名(参数) 表达式

注意：

1. 不能加分号
2. 宏名和()之间不能有空格

|  |
| --- |
| #include <stdio.h>  #define MUL(a) ((a) \* (a))  int main(void)  {  int sum = 0;  sum = MUL(5+1) \* 10;//110 360  printf("%d\r\n", sum);  return 0;  } |

### 条件编译

实例：在一个产品，通常都有高配和低配之分

专门写一份代码给低配？

专门写一份代码给高配？

它们是共用同一套代码，用条件编译来选择代码进行编译

写法有三种

1. 第一种

|  |
| --- |
| #if 表达式 不能为变量，可以常量、宏名、常量表达式  代码段1；  #else  代码段2；  #endif  原理：如果表达式为真，那么选择代码段1进行编译，否则选择代码段2进行编译  如果代码段2不存在。可省略#else    if语句：选择语句 --> 代码执行过程中，根据条件选择相应代码执行  #if条件编译 --> 在预处理阶段，选择相应的代码进行编译 |

1. 第二种写法

|  |
| --- |
| #ifdef 宏名  代码段1；  #else  代码段2；  #endif  原理：如果定义了宏名，那么选择代码段1进行编译，否则选择代码段2进行编译  如果代码段2不存在。可以是#else |

1. 第三种写法

|  |
| --- |
| #ifndef 宏名  代码段1；  #else  代码段2；  #endif  原理：如果没有定义宏名，那么选择代码段1进行编译，否则选择代码段2进行编译  如果代码段2不存在。可以是#else |

## 关键字extern

声明/引用外部变量关键字

作用：主要扩展全局变量的作用域

全局变量作用域：从定义开始到以下所有地方 🡪 文本作用域

别的文件想用全局变量，用extern进行声明即可

用法：extern 数据类型 变量名；

## 关键字static

静态变量关键字

1. 修饰局部变量 static：静态区

作用：改变局部变量的生命周期

局部变量：从定义开始到函数结束

静态局部变量：从定义开始到程序结束(作用域不变)

注意：

用static修饰的局部变量语句，只会执行一次，所以修饰的这个局部变量进行初始化

|  |
| --- |
| #include <stdio.h>  void Printf(void);  int main(void)  {  int i;  for(i = 0; i < 10; i++)  {  Printf( );  }  printf("\r\n");  return 0;  }  void Printf(void)  {  static int i = 0;//只执行一次  i++;//1 2 10  printf("%d ", i);  } |

1. 修饰全局变量

作用：使全局变量只能在本文件中使用

## 关键字typedef

作用：给数据类型取别名 int char float

用法：typedef 数据类型 别名

#define u8 unsigned char 🡪 文本替换(预处理后)

typedef unsigned char u8;

注意：

1. 要加分号
2. 数据类型用typedef来进行取别名