宏与预处理器

March 21, 2017

在C程序中，以#开头的行是会被预处理器所处理，而预处理器是在真正的编译开始之前由编译器调用的独立程序。预处理器可以删除注释、包含其他文件以及执行宏替代。 下面介绍一些关于预处理器的有趣事实：

1. 使用include指令时，头文件中的内容将会复制到当前文件中。使用尖括号<和>，预处理器将会在预定义的缺省路径中寻找该文件；而使用双引号“和”，预处理器会先在当前目录中寻找该文件，若当前目录中不包含该文件，再到预定义的缺省路径下寻找文件。

2. 使用define定义一个常数时，预处理会在程序中将宏名替换成表达式。例如，下面的程序中MAX定义为100。

```c
#include<stdio.h>
#define MAX 100
int main(void)
{
  printf("max␣is␣%d", MAX);
  return 0;
}
```

```
Output: max is 100
```

3. 宏允许写成函数的形式，其中的变量不检查数据类型。如下面的程序中，宏INCREMENT(x)可用于任意类型的x。

```c
#include <stdio.h>
#define INCREMENT(x) ++x
int main(void)
{
  char *ptr = "GeeksQuiz";
  int x = 10;
  printf("%s␣␣", INCREMENT(ptr));
  printf("%d", INCREMENT(x));
  return 0;
}
```

```
eeksQuiz 11
```

4. 宏变量在宏展开之前不会被计算。考虑以下程序：

```c
#include <stdio.h>
#define MULTIPLY(a, b) a*b
int main(void)
{
  // The macro is expended as 2 + 3 * 3 + 5, not as 5*8
  printf("%d", MULTIPLY(2+3, 3+5));
  return 0;
}
```

```
16
```

5. 可使用##运算符将几个宏变量连接在一起。

```c
#include <stdio.h>
#define merge(a, b) a ## b
int main(void)
{
  printf("%d\n", merge(12, 34));
}
```

```
1234
```

6. 可使用#运算符将标识符转换为字符串字面值。

```c
#include <stdio.h>
#define get(a) #a
int main(void)
{
  // GeeksQuiz is changed to "GeeksQuiz"
  printf("%s\n", get(GeeksQuiz));
}
```

```
GeeksQuiz
```

7. 可用‘\’将宏写成多行，但最后一行不需要‘\’。

```c
#include <stdio.h>
#define PRINT(i, limit) while (i < limit) \
    {                       \
      printf("GeeksQuiz␣");       \
      i++;                  \
    }
int main(void)
{
  int i = 0;
  PRINT(i, 3);
  return 0;
}
```

```
GeeksQuiz  GeeksQuiz  GeeksQuiz
```

8. 使用带参量的宏需要谨慎。

```c
#include <stdio.h>
#define square(x) x*x
int main(void)
{
  int x = 36 / square(6); // Expended as 36/6*6
  printf("x␣=␣%d\n", x);
  return 0;
}
```

```
x = 36
```

9. 预处理器也支持if-else指令，通常用于条件编译。

```c
#include <stdio.h>
#define MSG 1
int main(void)
{
#if MSG == 1
  printf("Trace␣Message!\n");
#endif
}
```

```
Trace Message!
```

10. 可用标准宏__FILE__打印程序名、__DATE__打印编译日期、__TIME__打印编译时间、__LINE__打印C代码的行数。

```c
#include <stdio.h>
int main(void)
{
  printf("Current␣File:␣%s\n", __FILE__ );
  printf("Current␣Date:␣%s\n", __DATE__ );
  printf("Current␣Time:␣%s\n", __TIME__ );
  printf("Line␣␣Number:␣%d\n", __LINE__ );
  return 0;
}
```

```
Current File: macro10.c
Current Date: Feb 27 2017
Current Time: 21:02:36
Line  Number: 8
```