**概述**

编译器的error系统也是一个重要的组成部分，虽然绝大多数情况下，写程序只要printf一下，然后报一下error就完事了。但是需要注意的一件事是，这和<error.h>是不一致的，这里是编译期中，对于C源文件中不符合标准的部分进行的处理。

* **内部error**

这主要是用于我写代码以及测试时使用，同时包括输出error和test的信息。

正常情况下，这部分内容对于用户肯定是不可见的，所以自己怎么方便怎么来就可以了。

在MAKEFILE中，我使用了-D定义了\_TEST\_宏，从而控制是否进行测试，如果有需要不加测试，请在MAKEFILE中CFLAGS处取消这部分，正式发布的程序也应当取消这部分。

在代码中可以调用m\_error开头的函数来使用，表示mcc编译器的内部error。

为了不使得各种报错信息格式乱来，做以下约定：

* + **Error**

报错应当涉及到四个部分：开头的

|  |
| --- |
| ”mcc:ERROR\n” |

作为错误的标识开始。

随后是需要标识出具体的出错文件，出错函数，便于查找，格式为换行后：

|  |
| --- |
| “\t Infile :%s , in function :%s”,（文件名），（函数名） |

这里不加行号，因为写代码很容易增删，行号需要实时更新，太麻烦

随后换行，表示出错原因种类：

|  |
| --- |
| “\t ERRORreason： %s” |

最后一行，由于很多参数需要在运行中才能产生，所以这里一个argument表示具体出错的问题。

|  |
| --- |
| “\t Argument: %s或者其他%d之类的” |

但是，有些地方不需要参数，有些地方的参数类型不确定，数量也不知道，因此不定参数本来是个好选择，可是，他的类型也不确定，这就很捉急了，如果再复杂一些，怕不是要实现个printf么。因此，这里面单独设置一个函数，表示argument的输出就行了

* + **Test**

对于测试，由于各个测试项目的不同，缺少统一的标准，这里对通用的test输出只做以下约定，而特定的可以确定的测试，则在后续列出：

测试项目名称：

|  |
| --- |
| “mcc:TEST %s”,（项目名称） |

对于测试中每一项item条目的输出，包括item名称和item的值：

|  |
| --- |
| “%s:”,（item名称）  “\t%s”，（item值）  或者  “\t%d”，（item值） |

* + - **Token**

对于token类型的值，他内容一定是确定的，包括type和value两个部分，因此输出如下：

|  |
| --- |
| <type\t:\tvalue>\n |

* **C标准error**

C标准的error是面向用户的，如果用户用于编译的代码或者编译的命令等等出现错误，那么就会报这些错误，也就是常见的‘fatalerrorCxxxx‘。

（老实说，我确实没找到那一大堆标准编号的列表和原因说明，哪怕是英文的，有人能赞助一下下么？）

在考虑那一大堆编号的问题之前，更重要的两个事情是：

1. 如何定位
2. 该以怎样的数据结构来组织所有的错误

第一个问题产生的原因在于，当预处理器，将一大堆include的文件附加到文件中，以及进行预处理器的宏定义进行替换之后，相对于原始的文件，某个任意出错字符的位置，可能早就已经不是原来的那个位置了。那么，如何定位呢？

对于这个问题，我在lex中，将每个token都给出对应的line，输出的时候，输出文件名，行号，token——当然如果需要更精细的操作，可以给到具体的行和第几个字符。

但是我认为这没有必要，因为，通常来说，C程序员不会在一行中，给出一个非常长且复杂的语句，这不太符合编程和项目的规范——非常长的语句可能发生，但这往往是有迹可循的，比如多个并列的条件判断啦，初始化一个很长的数组啦等等。

而且，在某一行中，某个token出现的次数大多数情况下是不多的，尽管出错的地方未必是该token。

因此当给定某一行，以及相关的错误和token的时候，我认为已经足够程序员判断到底出错在哪里了。

没有必要指定该token所在符号位置，另外一个让我决定不这么做的原因在于编译预处理之后的替换问题，由于目前没有涉及到预处理器，因此，这让我很难去解决因为宏替换导致的token在行中的位置的偏移。这使得行号可以准确，但offset变得困难。

第二个问题是，报错是可能发生在各个地方的，除了每个阶段之后可以进行一轮输出之外，其余的，需要将所有的报错都存放起来，在每个阶段结束之后，进行输出。

这我设计了一个向量，用于存放error，对于error的类型，我设计了一个

* **编译器warning**

这里确实有一堆warning，通常是在C11标准中没有提及，但是在一些情况下，通常没有一个正常人会这么去做的情况。或者未定义的implement等等，这容易导致错误。而且貌似不同编译器还不一样。

我并不想复制一遍error的相关数据结构，两者通常差不多，所以我把两者都放在一起去了，但做了一些区分，以用于实现error和warning的不同的打印输出。