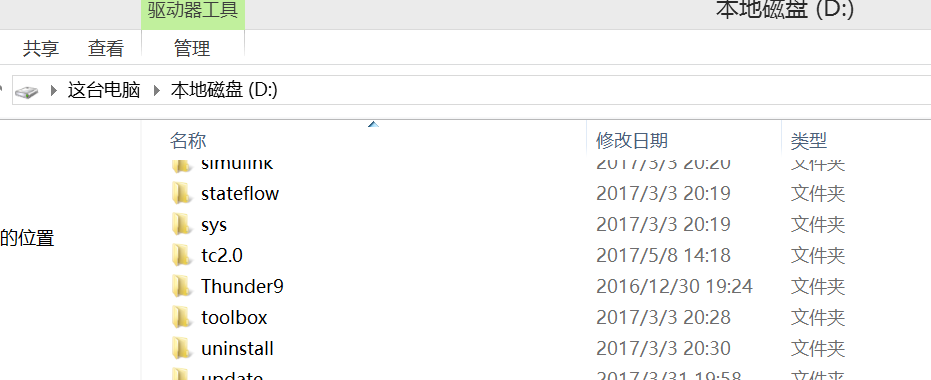
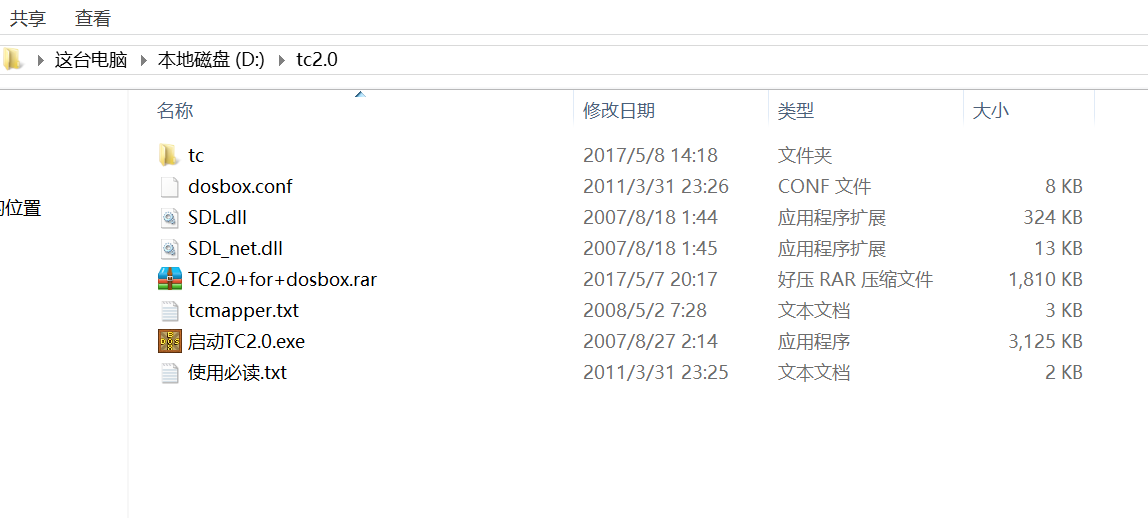
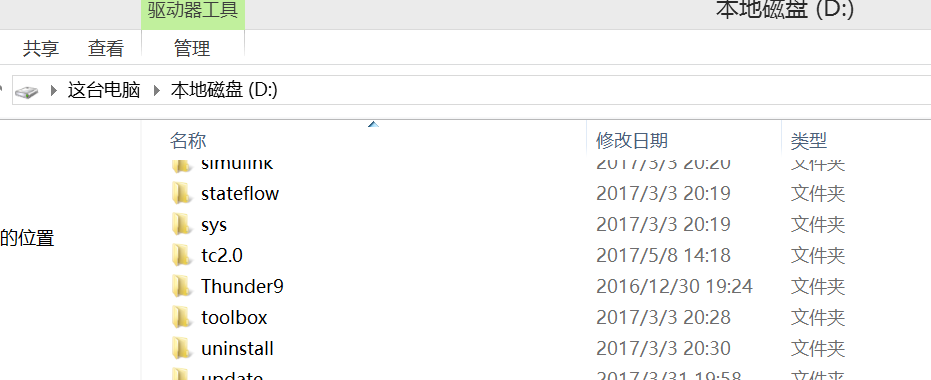
**综合研究一研究报告**

1. **研究过程展示**

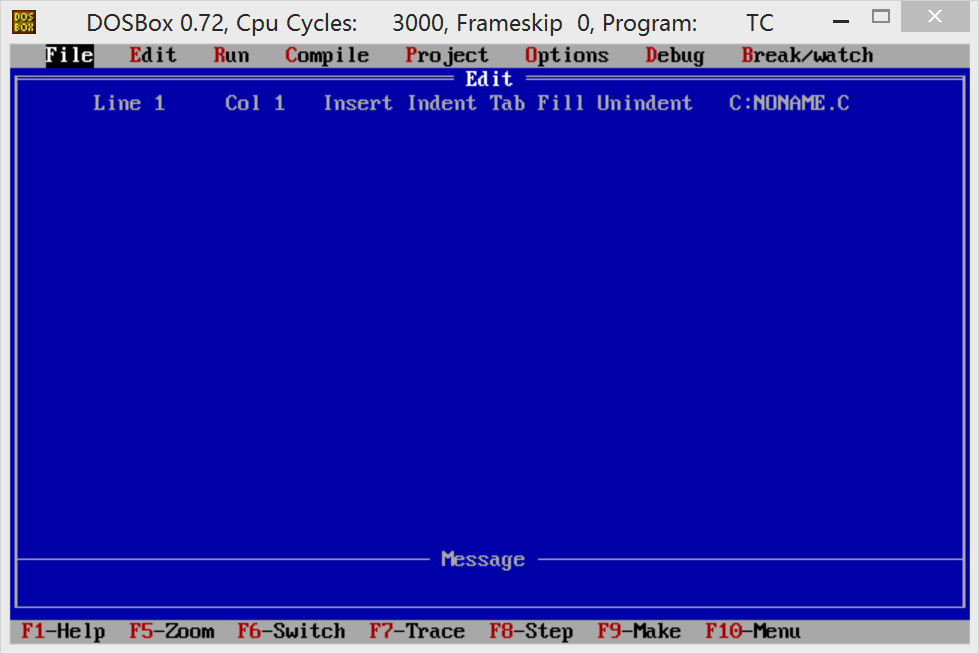
****

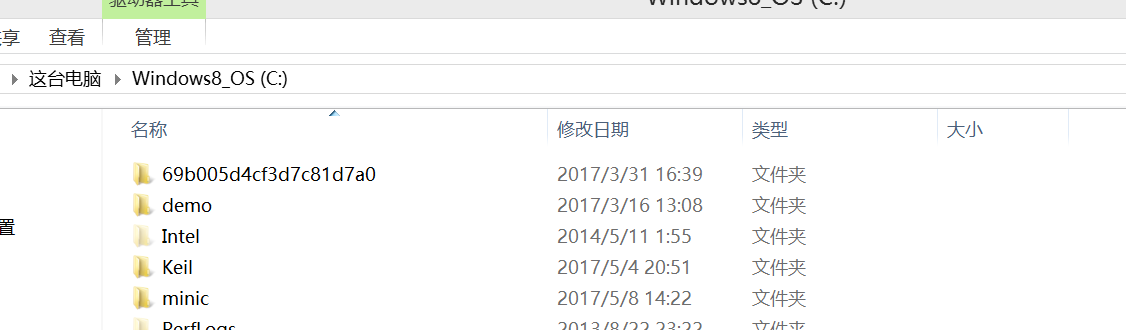
**如图在D盘新建：D:\tc2.0**

双击打开如下图：

****

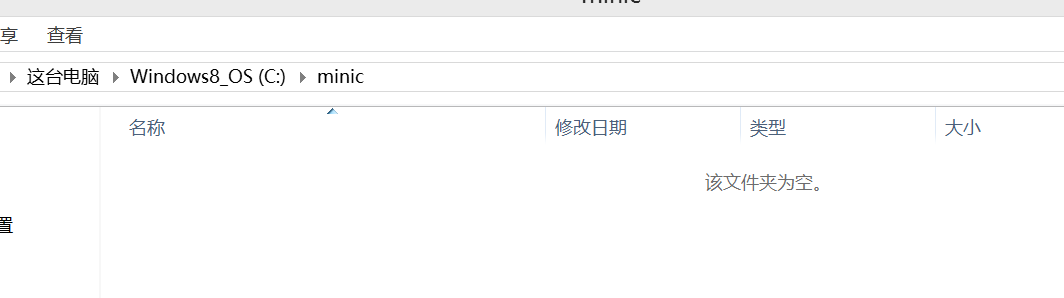
双击打开“启动TC2.0.exe”效果如下图：

****

****

**如图在C盘新建：C:\minic**

双击打开如下图：

****

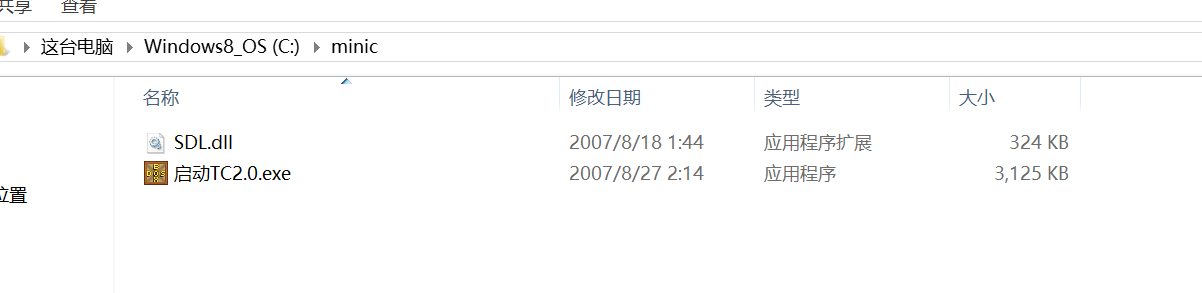
依次拷贝文件效果如下图：

****

双击之后：

****

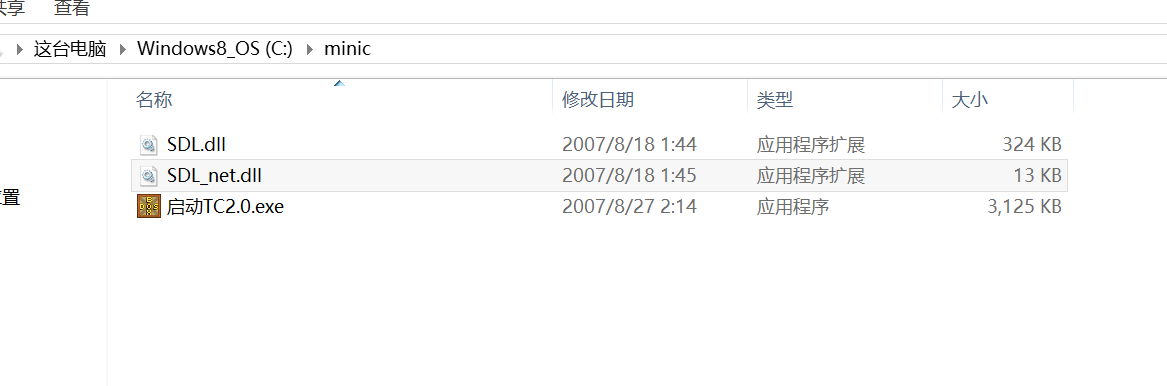
按要求拷贝**D:\tc2.0\SDL.dll**文件：



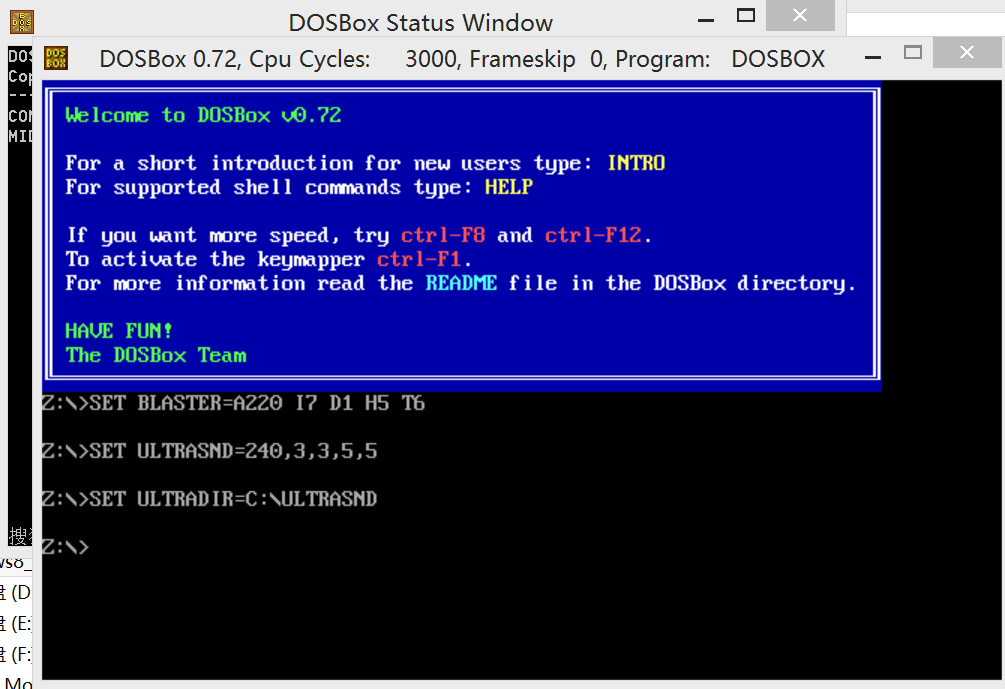
再次双击之后：

****

按要求拷贝**D:\tc2.0\SDL\_net.dll**文件：

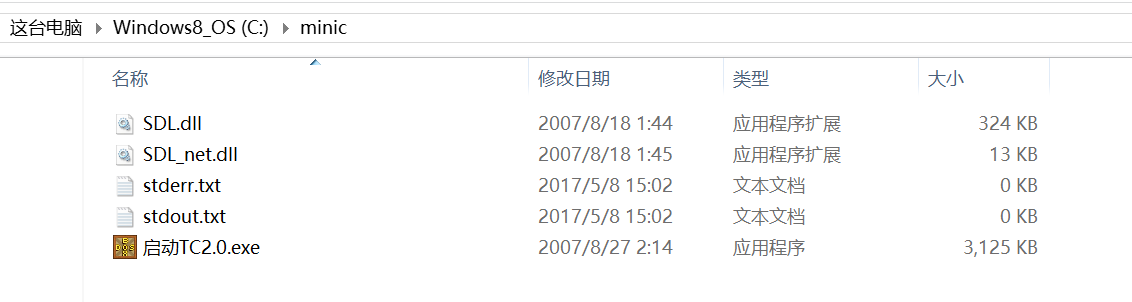
****

第三次双击之后：

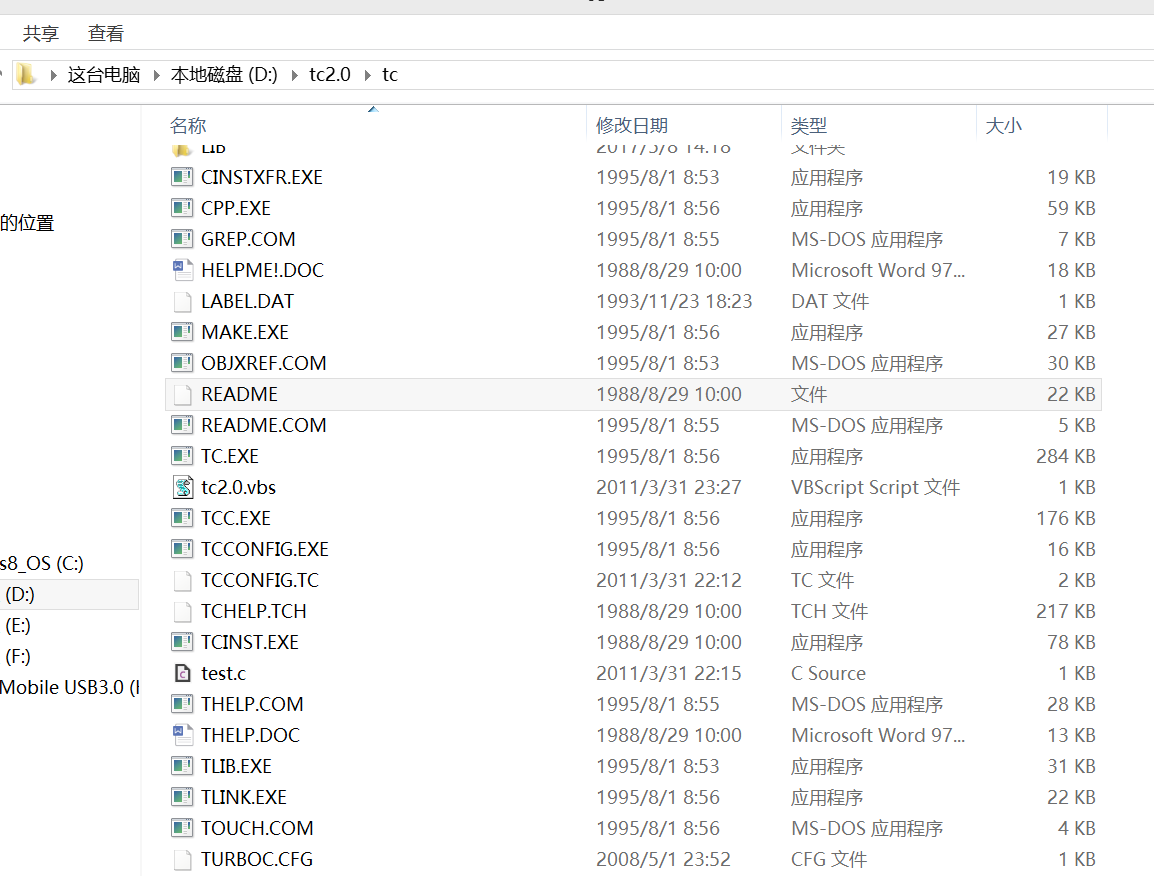


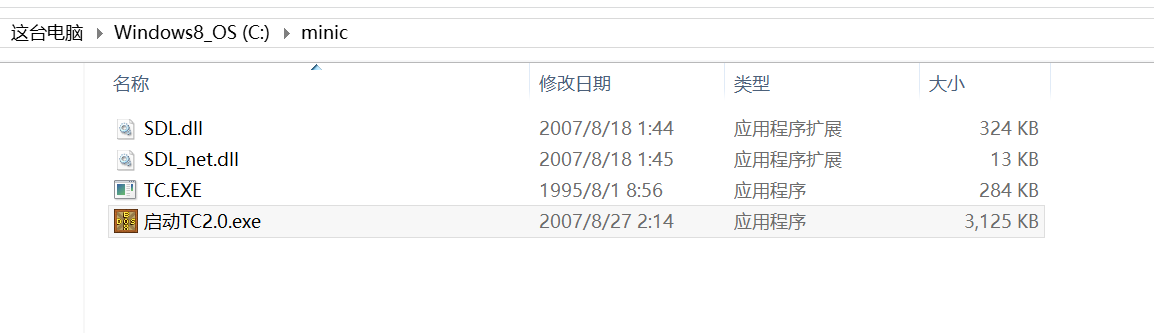
出现dosbox启动界面，不过有一些配置参数如图中横线。

并且在“未关闭状态”，此时的目录下文件情况如下图，可以看到，明显多出两个文本文档“stderr.txt、stdout.txt”；而当我关闭时，这两个文本文档又不在了。

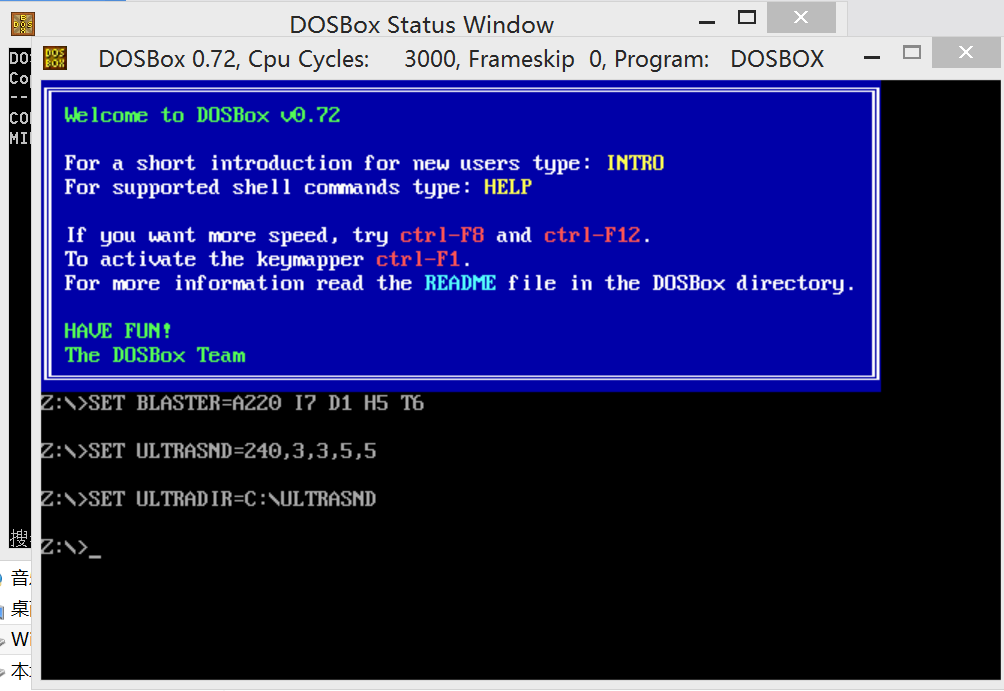


拷贝**D:\tc2.0\tc\TC.EXE**文件：





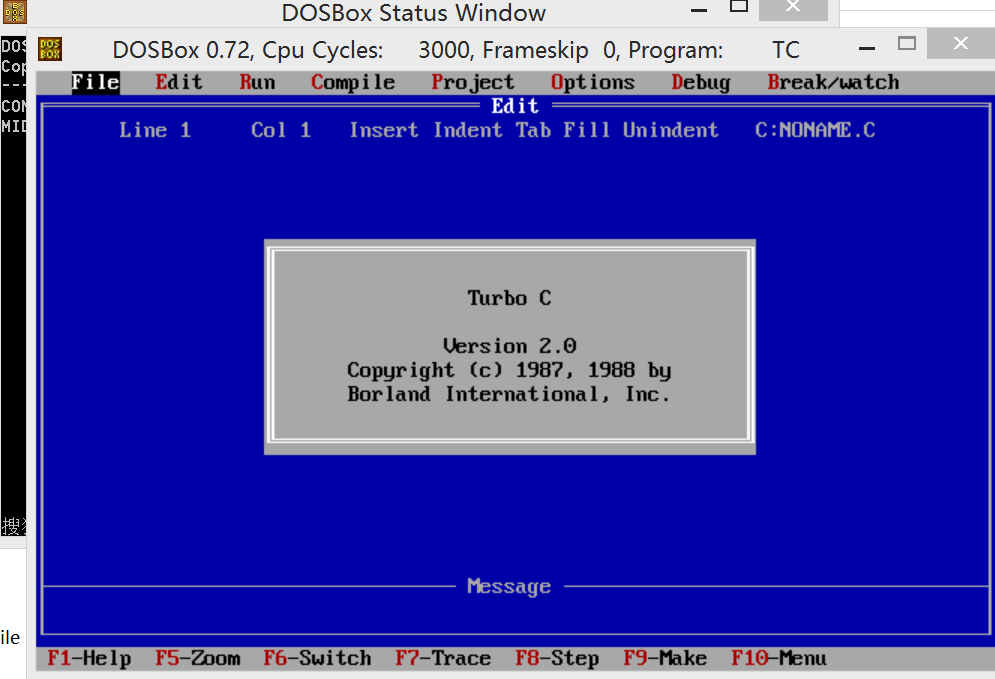
第四次双击之后：



拷贝**D:\tc2.0\dosbox.conf**文件：

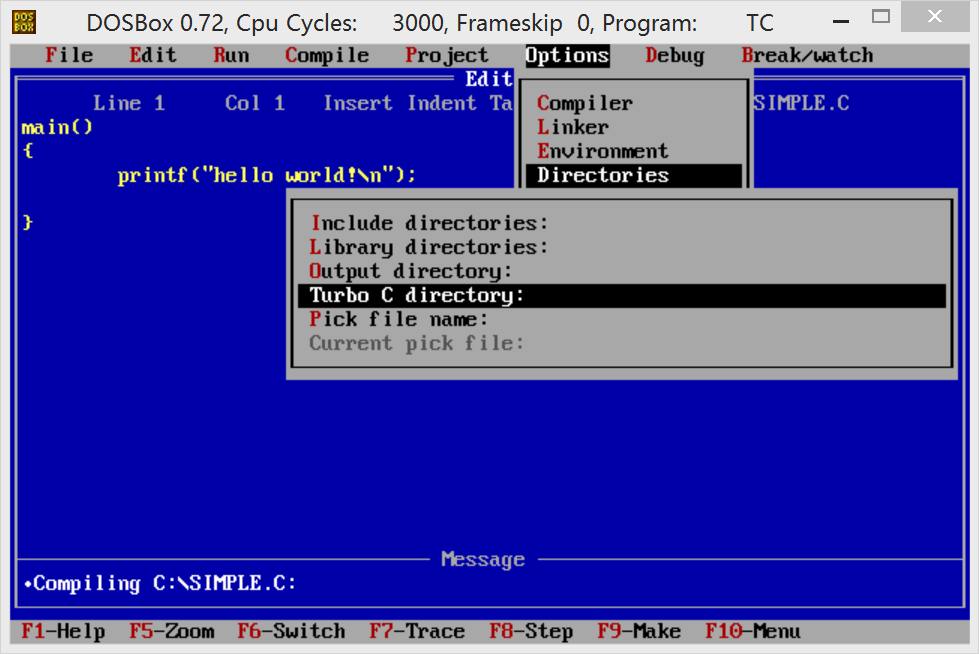
****

第五次双击之后：

****

效果至此，看起来好像可以用了。

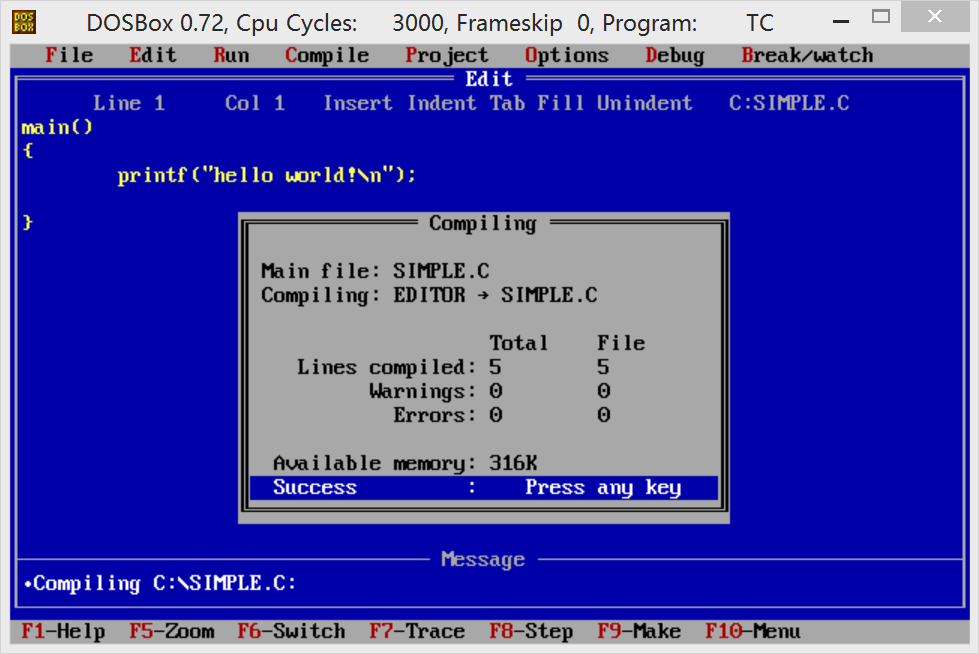
按照书上步骤做到第4步：

****

全部清除

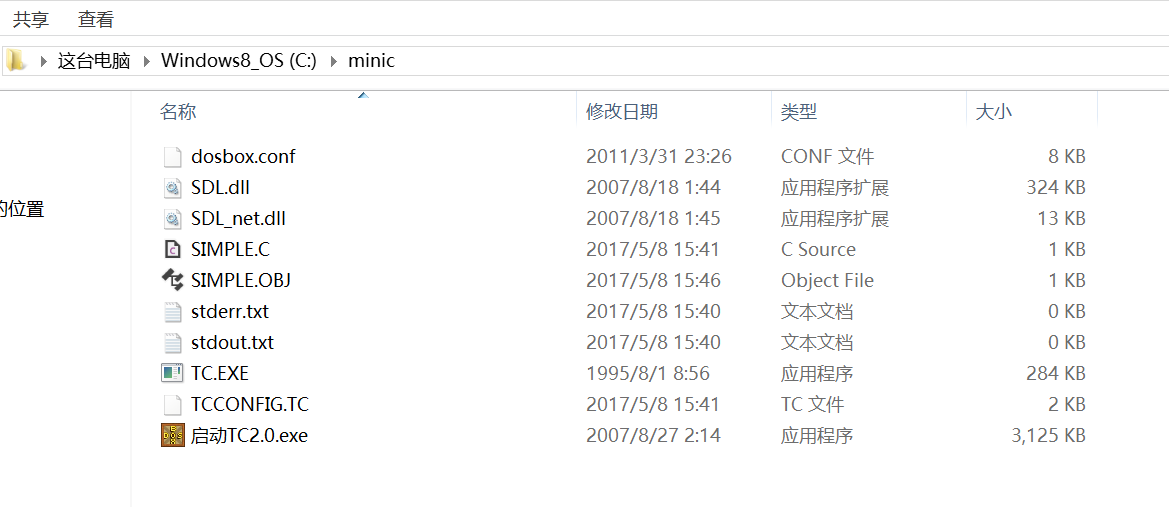
按照书上步骤做到第6步：

****

****

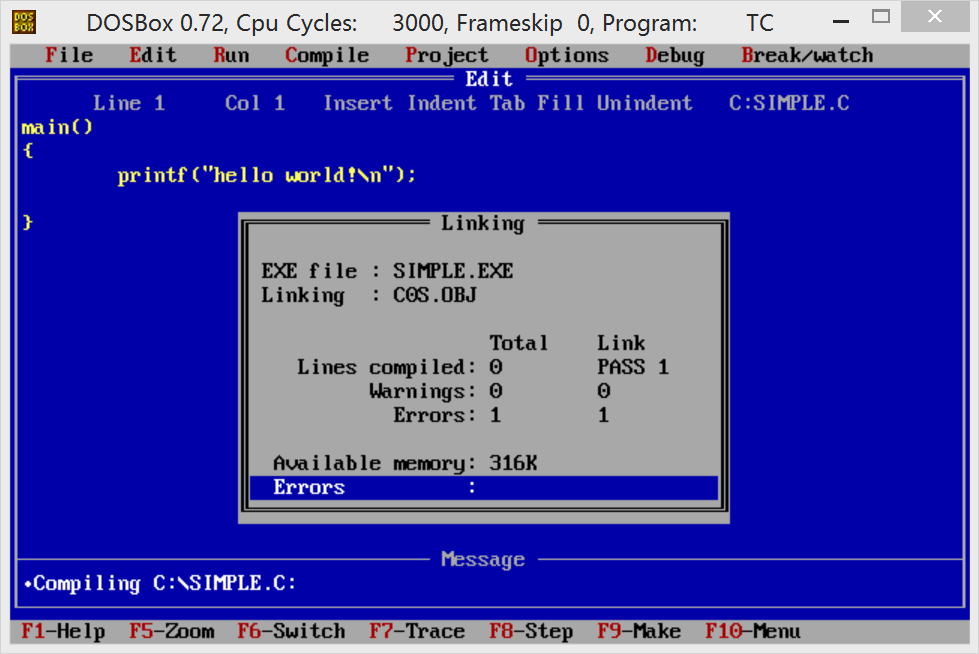
出现“Success”，应该是“编译成功OBJ文件”

再次检查**C:\minic** :

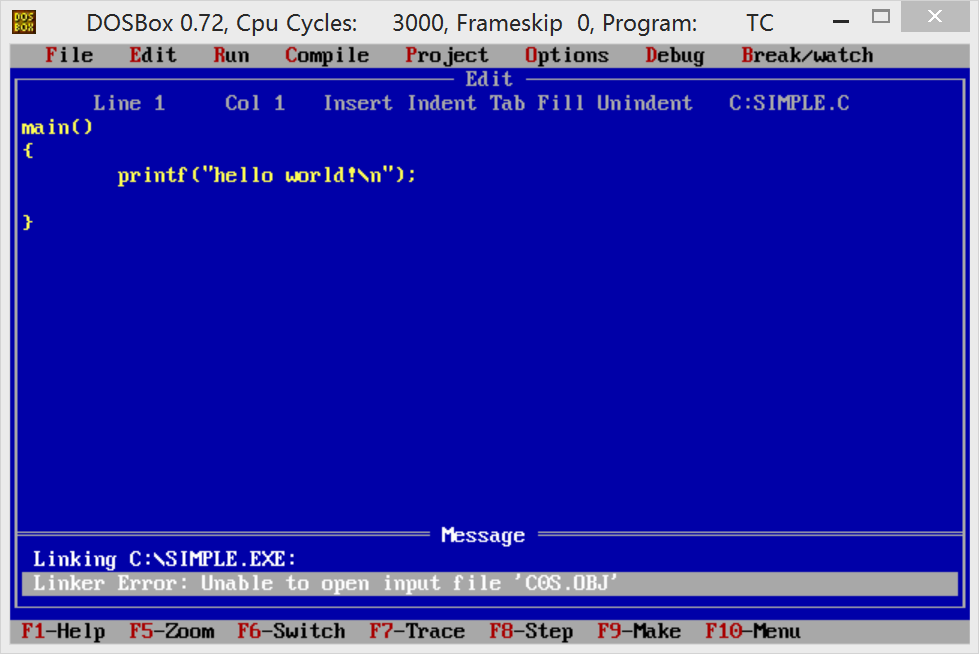
****

由此，确认“编译成功simple.OBJ文件”

按照书上步骤做到第7步：

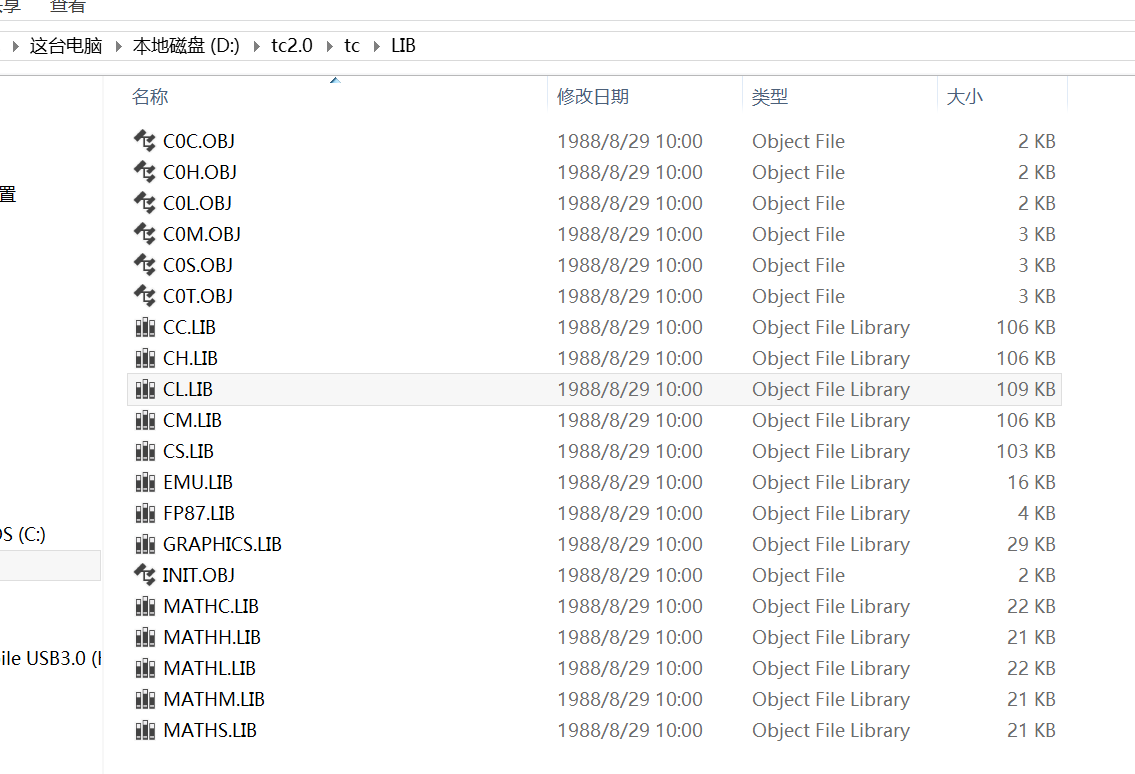
****

出现“Errors”，应该是“连接错误”

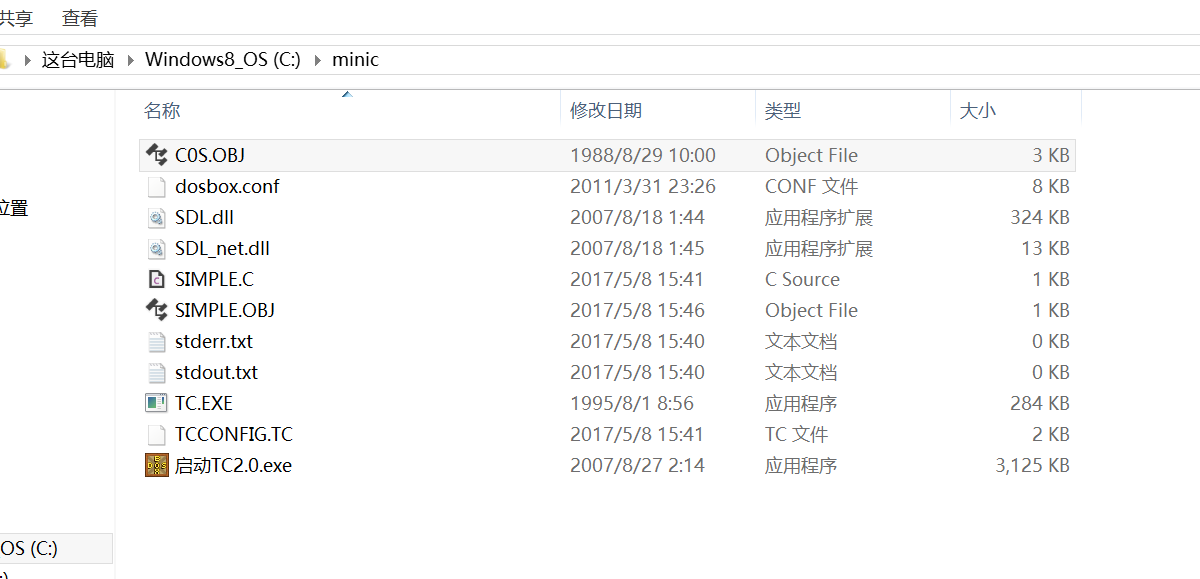
****

出现和书上预想的一样的错误提示信息。

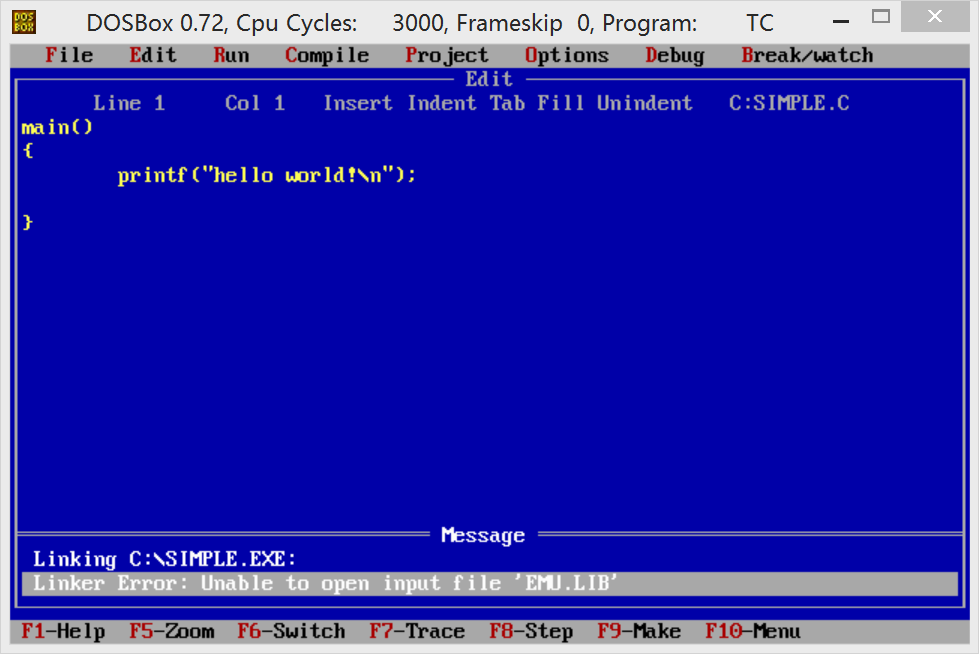
找到C0S.OBJ文件，路径如图：

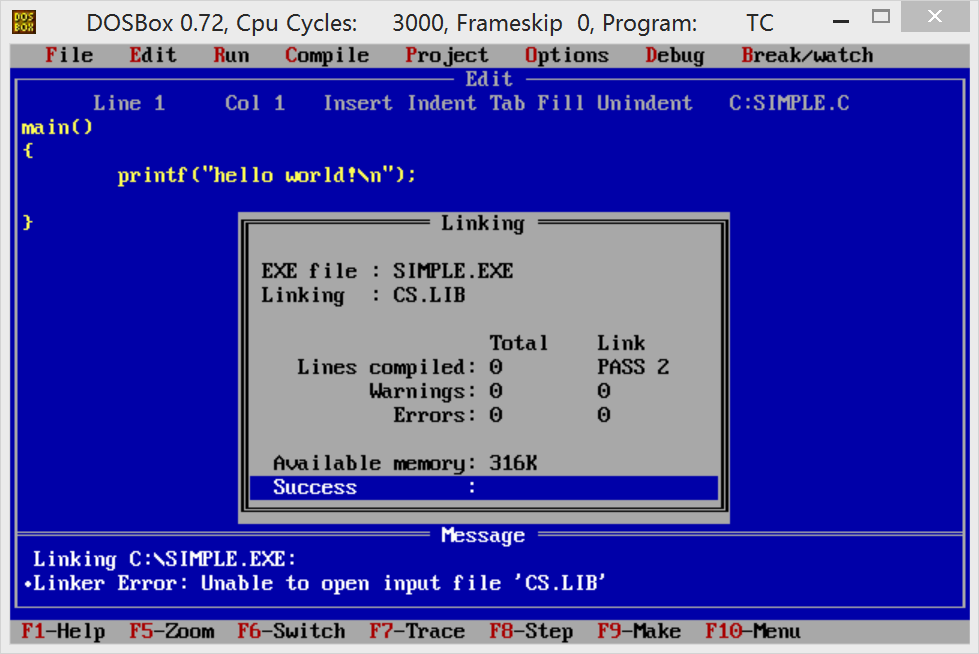
****

拷贝：

****

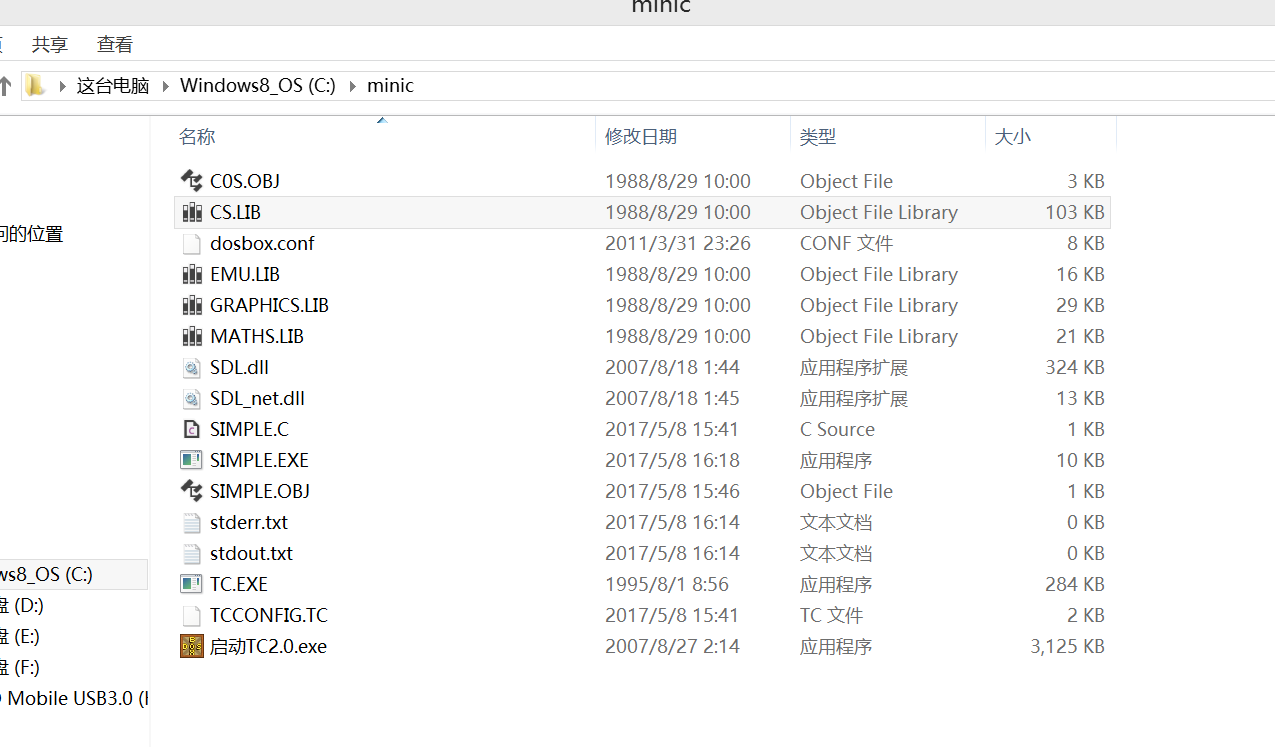
再次连接，结果出现“新挑战”：

****

效仿以上步骤，直到出现下图情况：

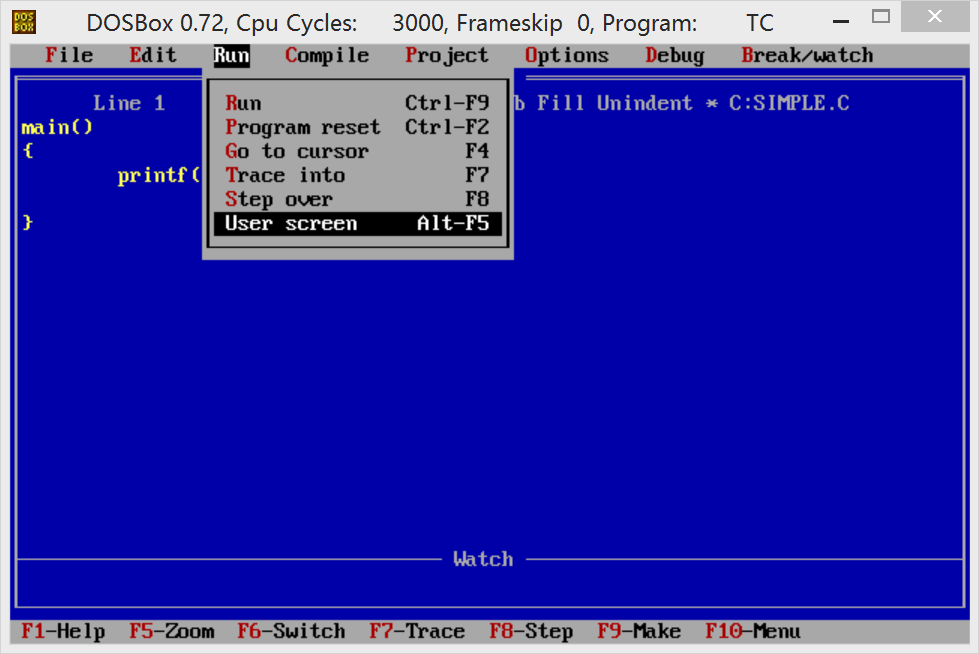
出现“Success”，意为连接成功。

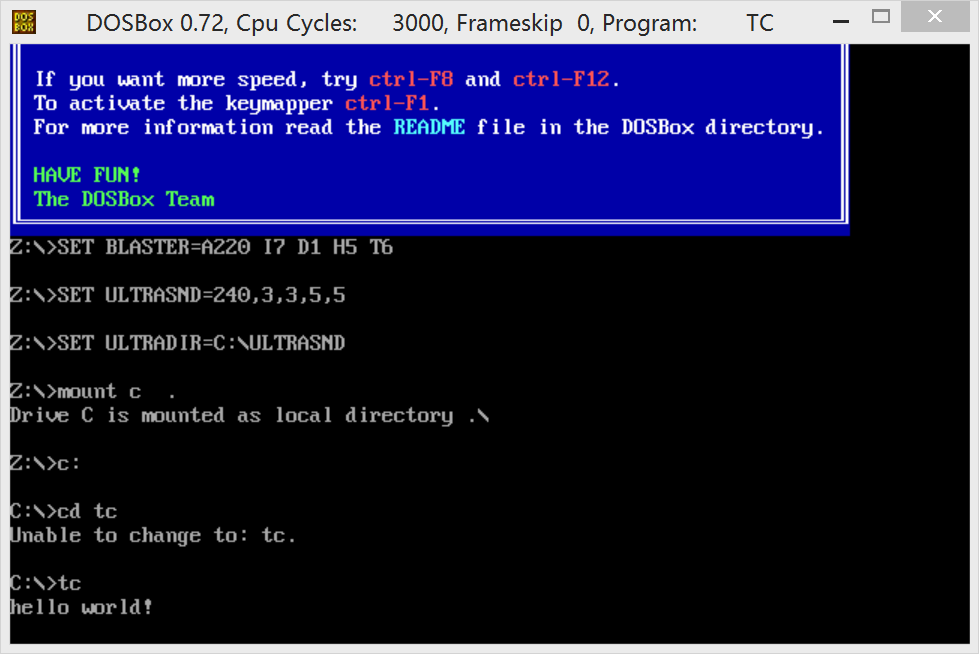
至此，再次检查**C:\minic** :



如图，已生成“TC.EXE”文件此过程中一共搬运了“C0S.OBJ、CS.LIB、EMU.LIB、GRAPHICS.LIB、MATHS.LIB”,而且提示消息有前有后，这个顺序是按照什么规则排的？或者这一点并没有必要了解？存疑。

最后运行一下，看看效果



****

**Hello world！**

1. **已思考研究并已解决问题汇总**

**1>.生成“TC.EXE”文件此过程中搬运了“C0S.OBJ、CS.LIB、EMU.LIB、GRAPHICS.LIB、MATHS.LIB”,各自是啥作用？**

“每一个库文件都相当于一个小模块，支持一种扩展。其实，一个lib文件是obj文件的集合。当然，其中还夹杂着其他一些辅助信息，目的是为了让编译器能够准确找到对应的obj文件。我们可以通过tlib.exe（在tc2.0下的根目录）来对lib文件进行操作，你可以把自己生成的obj文件通过tlib命令加入到一个lib文件中，也可以把lib文件内的obj文件进行删除操作，还可以把内部的obj文件给提取出来。明白了lib文件的大致结构以及对它的具体操作，在学习C语言的过程中，就会又多了一个切入点对C语言具体实现进行研究。”

经过上网查找，初步找到的各个文件的作用如下，待亲自佐证：

**C0s.obj**文件里包含了main函数的定义，找不到这个文件，就不能连接；

**TC.EXE** 集成开发环境

**C0?.OBJ** 不同模式启动代码

**C?.LIB** 不同模式运行库

**GRAPHICS.LIB** 图形库

**EMU.LIB** 8087仿真库

**MATHS.LIB**数学函数库

其中上面的?分别为:

**T** Tiny(微型模式)

**S** Small(小模式)

**C** Compact(紧凑模式)

**M** Medium(中型模式)

**L** Large(大模式)

**H** Huge(巨大模式)

几点解释:

（1）一般我们常使用的是TC.exe，实际上，Turbo C包有两种编译器，集成开发环境下的叫做TC.exe和命令行方式下的图形界面运行环境叫做TCC.exe（需要额外的graphics.lib文件提供支持）

（2）此程序中调用了printf函数，大家都知道调用C语言的库函数需要添加头文件，这里却不添加也可以，因为在 cs.lib中已经添加了printf函数的具体实现。

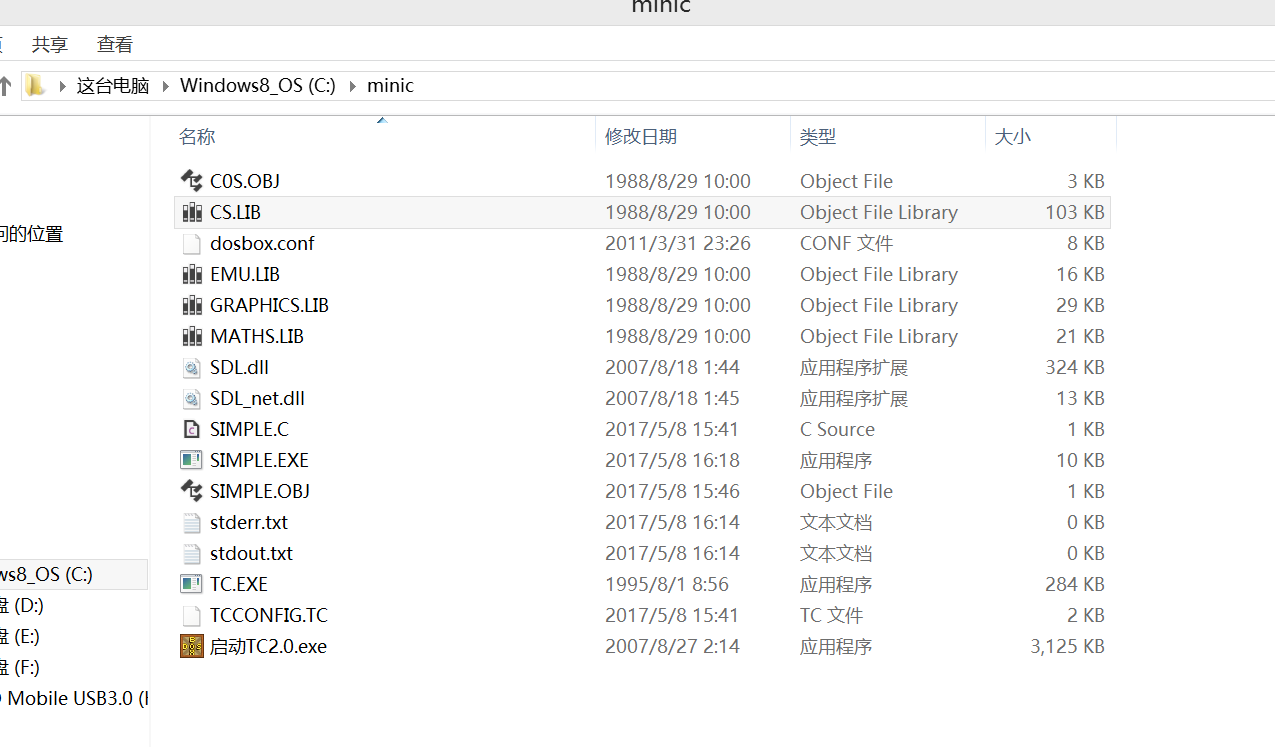
C 语言中提供了6种编译模式，这6种模式是：

微模式（Tiny），小模式（Small），中模式（Medium），紧凑模式（Compact），大模式（Large）和巨模式（Huge）。它们之间的关系如下图所示。用户可以按照自己的程序大小及需要进行选择。  
　　　　　　│ 小程序　　 │ 大程序  
　　━━━━┿━━━━━━┿━━━━━━━━  
　　 小数据 │ 微，小　　 │ 中  
　　 大数据 │ 紧凑　　　 │ 大，巨

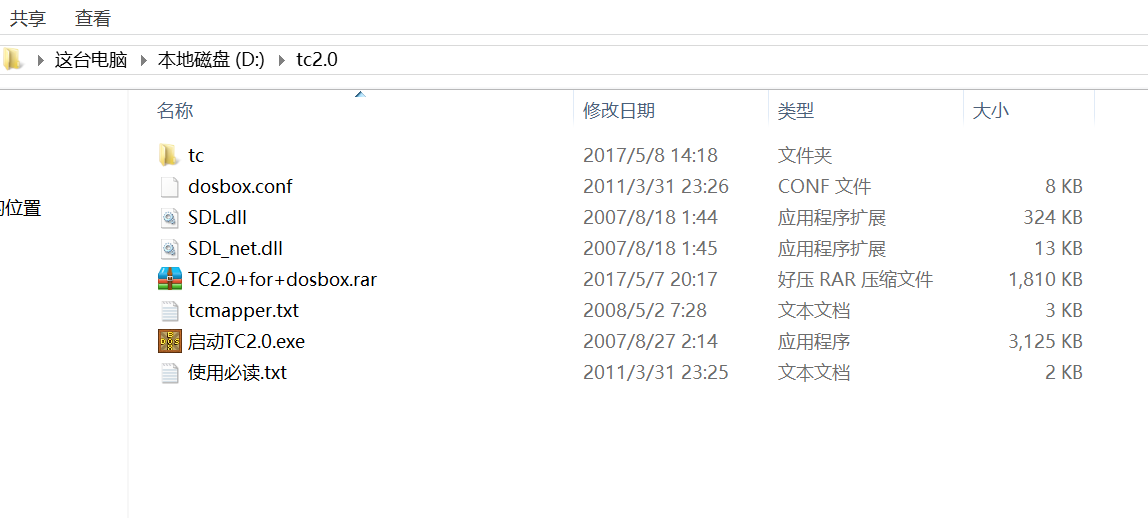
所谓小程序就是指程序只有一个程序段，大小不超过64KB，缺省的码（函数）指针是near（近程指针）。所谓大程序就是指程序只有多个程序段，每个程序段不超过64KB，但总程序量可超过64KB，缺省的码指针是far（远程指针）。小数据就是指数据只有一个数据段，缺省的数据指针是near。大数据就是指数据有多个数据段，缺省的数据指针是far。

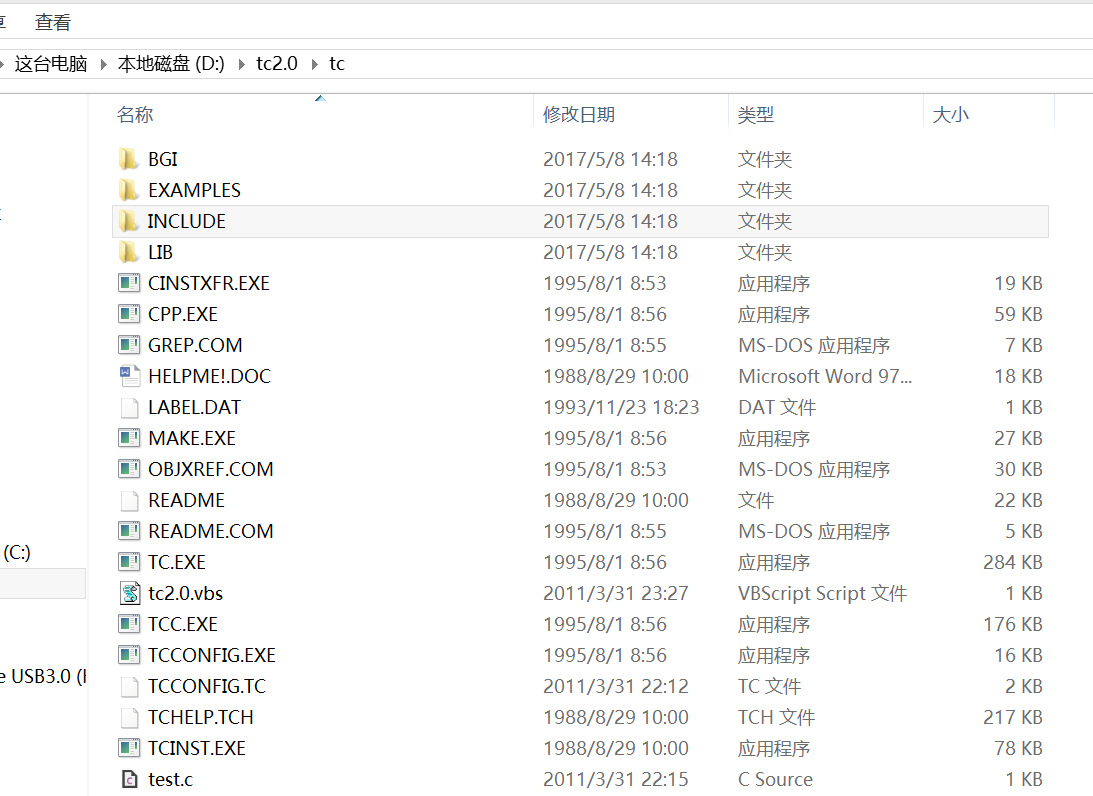
**2>. 程序在编译运行时，源码所需要的库（静态库和动态库）及头文件编译器是去哪找的？（库及头文件的查找）**

通过本实验过程，比较**C:\minic** 与**D:\tc2.0**：

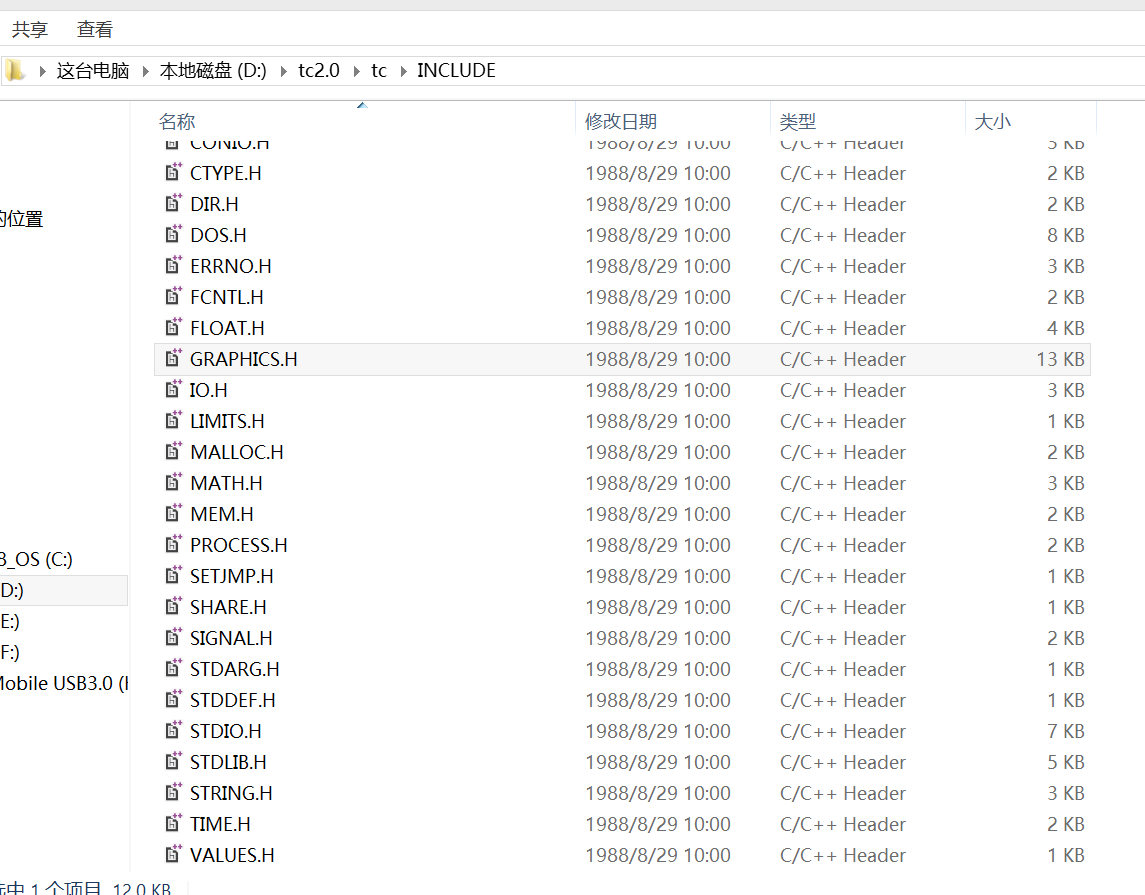


在**C:\minic**中，各个拷贝过来的\*.obj、\*.lib文件都是相对于simple.c、simple.obj文件而言，处于同一目录下，这是一种情况，此情况经过验证可以使simple.exe最终生成，并正确执行。

****

****

在**D:\tc2.0\tc**中，有INCLUDE、LIB文件夹等，本次实验所拷贝文件均出自LIB文件夹，而此次没有用到的INCLUDE文件夹内，打开之后，可以看到多个头文件，如图：

****

很明显，这又是另一种方式；

通过上网查找，发现还有一种，如下：

在turboc.CFG中可以指定tcc可以用来搜索的库文件的位置。但是用TC2.0修改路径不会保存在turboc.CFG中，而是生成另一个配置文件。（待佐证，可排到未解决问题中）

1. **已思考研究并未解决问题汇总**

1>.生成“TC.EXE”文件此过程中一共搬运了“C0S.OBJ、CS.LIB、EMU.LIB、GRAPHICS.LIB、MATHS.LIB”,而且提示消息有前有后，这个顺序是按照什么规则排的？或者这一点并没有必要了解？存疑。

1. **研究感想（心得体会）**

以前不懂什么是“一花一世界”，现在应该也不算懂，不过好在了解了许多。王爽老师的三个一课程设计的题目，让人做了之后总是很有启发，有一种醍醐灌顶的感觉，比对一下学校内的老师授课方式以及我个人不太理解的排课方式，高下立判。“师者，所以传道授业解惑也”，如果大部分老师都能摒弃教给学生的那些似是而非的知识，一步脚印一个坑，大胆的把“计算机是什么？编程怎么玩”讲得“一丝不挂”。那么，该是一件多么好的事！