

2020/03/31_第2课_C语言命令行编译、include、main、printf函数的使用

笔记本: C

创建时间: 2020/3/31 星期二 20:04

作者: ileemi

- [C语言的特点](#)
 - [丰富的数据类型](#)
 - [结构化的控制语句](#)
 - [高效的目标代码](#)
 - [可移植性好](#)
- [命令行编译](#)
 - [编译脚本的使用](#)
- [C库函数的使用](#)
 - [include](#)
 - [main\(int argc, char argv\[\], char* envp\[\]\)](#)
 - [printf](#)

C语言的特点

丰富的数据类型

C具有整型、浮点型、字符型、数组类型、指针类型、结构体类型、共用体类型等多种数据类型。其中C语言的指针类型功能强大，灵活运用方便

结构化的控制语句

C语言的控制结构语句符合结构化程序设计的要求，并且用函数作为程序模块，使得结构清晰，可读性好，易于调试。

高效的目标代码

主要看方法：算法 -> 数据结构 -> 编译器

可移植性好

指的是源代码可移植性好，不是可执行文件的可移植性好

命令行编译

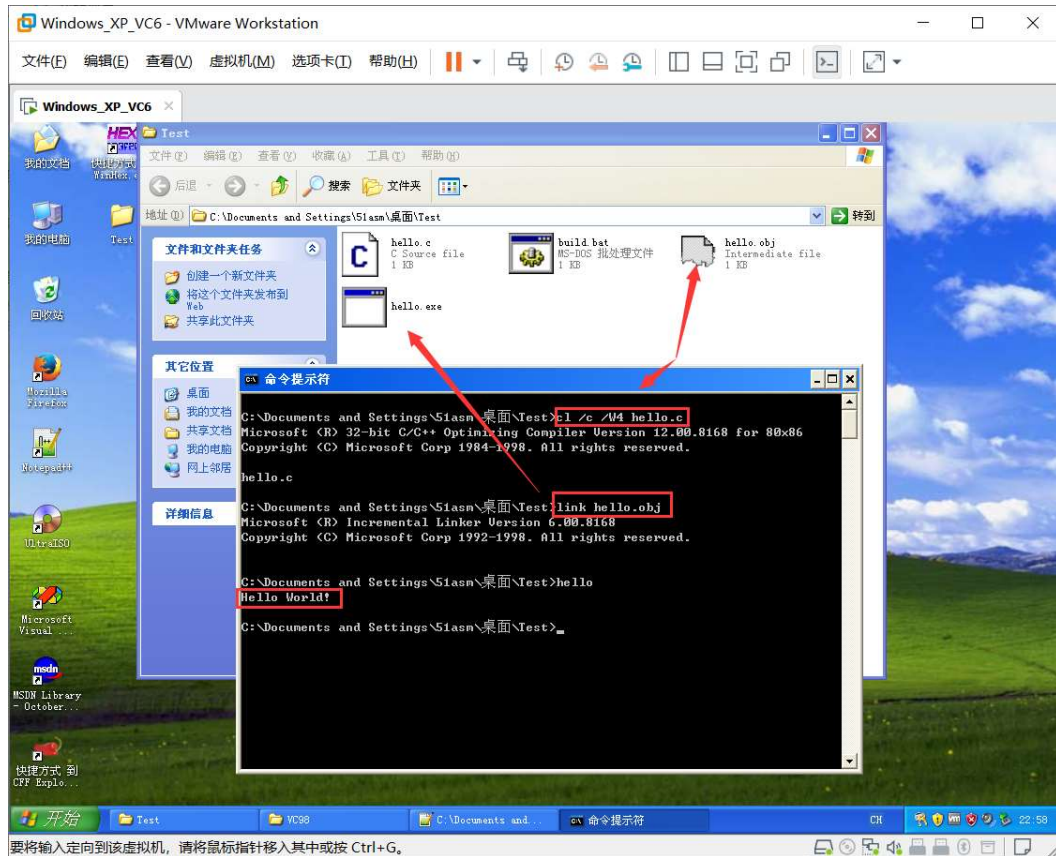
打开IDE的安装目录，例如VC6.0，在其安装目录VC98->Bin下可以看到：

cl.exe->编译器

link.exe->链接器

编译器，可以将.c文件生成.obj文件，编译器的工作就是将人类阅读的文本代码转换成计算机理解的二进制格式代码

链接器，从指定的obj文件中抽出二进制代码，数据以及相关所需信息，按约定的操作系统中的执行文件的格式打造一个符合要求的执行文件。



在使用编译器的时候可以添加相应的参数，例如：

cl /c /W4 /WX hello.c

其中的 /c 表示：只编译，不链接

/W4：设置警告等级（默认 n=1）

/WX：将警告视为错误

编译脚本的使用

创建.bat文件，添加下面的代码，将其放在存放.c的文件夹内，双击运行即可

```
del *.obj
```

```
del *.exe
```

```
cl /c /W4 hello.c
```

```
link hello.obj
```

hello.exe

pause

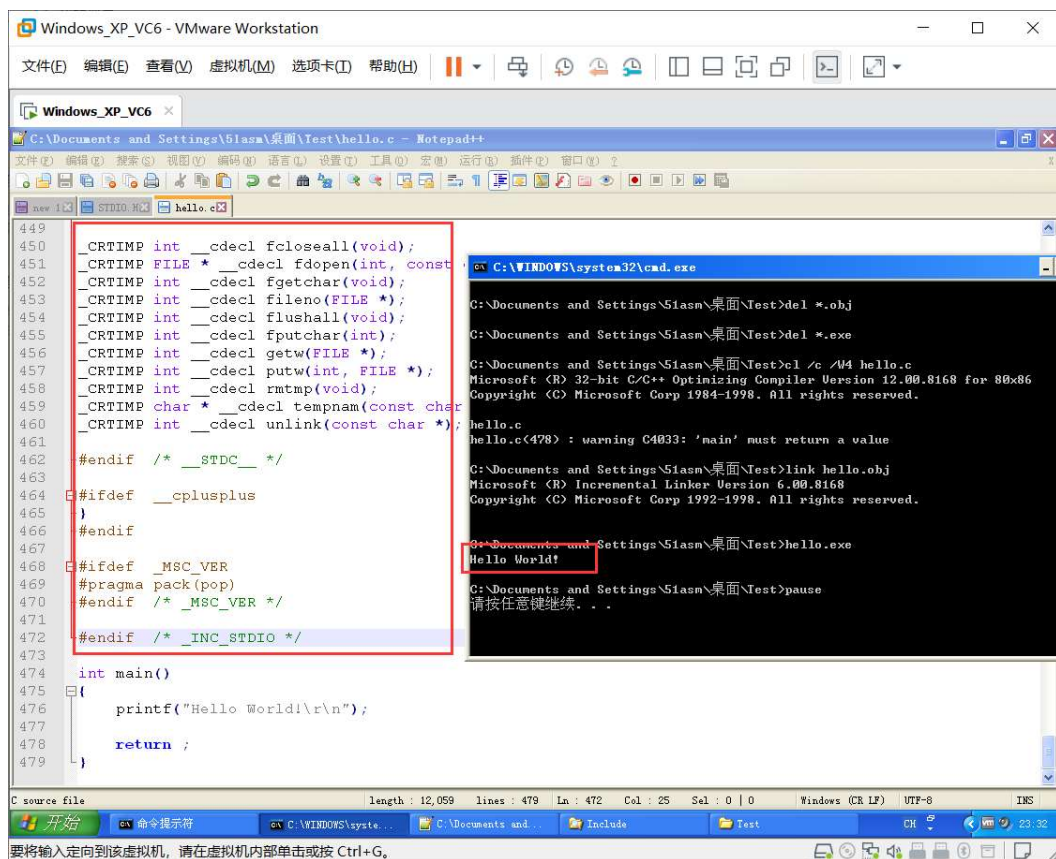
C库函数的使用

include <stdio.h>

包含头文件，将程序所需的代码通过include复制到程序中，主要工作就是复制粘贴的工作。

<> 和 "" 的区别就是查找顺序的不太，使用 <> 程序会优先找环境变量，去IDE的安装路径下查找所需的头文件。

例如将官方库 stdio.h 头文件内的所有内容复制粘贴到 .c 文件内的int main()前面，编译、链接效果是一样的。



还可以使用 cl /c /P hello.c

/P ->预处理到 .i 文件 ,该文件为说明信息，不能通过编译的

main(int argc, char argv[], char* envp[])

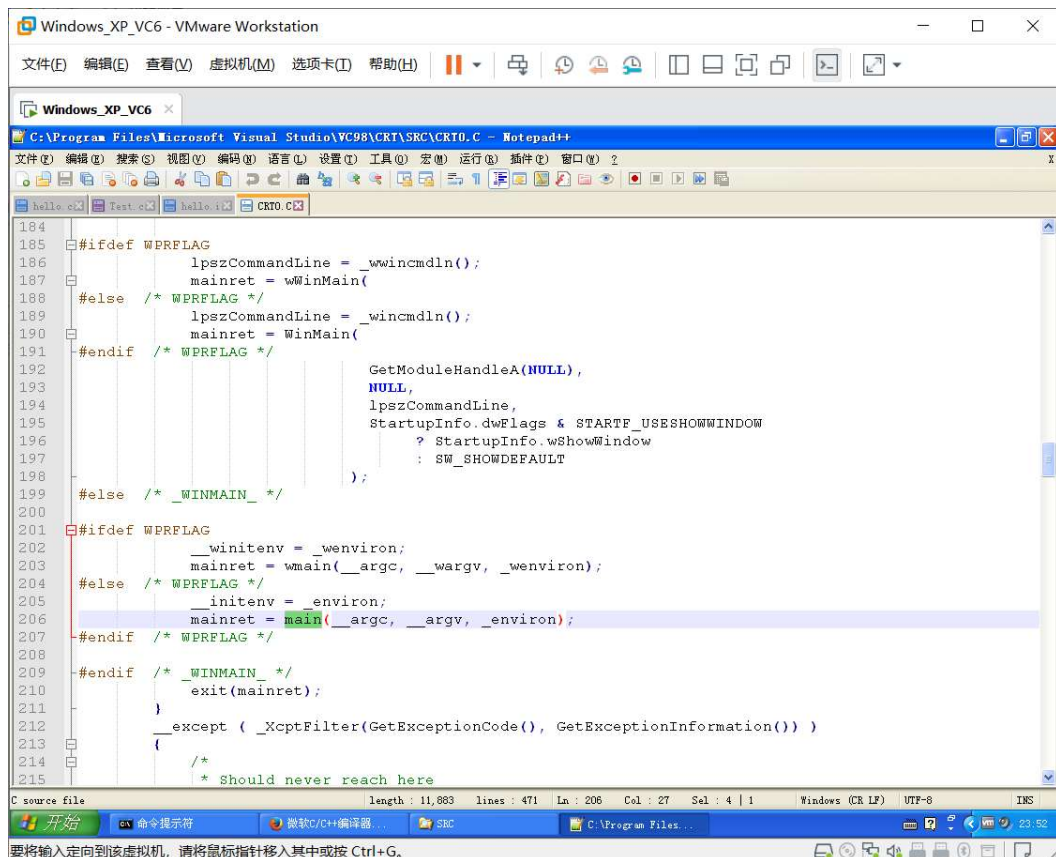
main函数为：程序员的可控入口

main函数参数的来历：

获取版本信息->初始化堆->初始化多线程环境->初始化io信息->获得命令行->获得环境变量->格式化命令行->格式化环境变量->初始化浮点xie处理器->初始化官方的

全局变量->初始化用户的全局变量（这里的用于实行对于编译器的用户，所谓的程序猿）->获得进程的初始信息->之后根据字符编码进行选择

C:\Program Files\Microsoft Visual Studio\VC98\CRT\SRC\CRT0.C



```
184
185 #ifdef WPRFLAG
186     lpszCommandLine = _wincmdln();
187     mainret = wWinMain(
188 #else /* WPRFLAG */
189     lpszCommandLine = _wincmdln();
190     mainret = WinMain(
191 #endif /* WPRFLAG */
192     GetModuleHandleA(NULL),
193     NULL,
194     lpszCommandLine,
195     StartupInfo.dwFlags & STARTF_USESHOWWINDOW
196     ? StartupInfo.wShowWindow
197     : SW_SHOWDEFAULT
198 );
199 #else /* _WINMAIN_ */
200
201 #ifdef WPRFLAG
202     _winitenv = _wenviron;
203     mainret = wmain(__argc, __wargv, _wenviron);
204 #else /* WPRFLAG */
205     _initenv = _environ;
206     mainret = main(__argc, __argv, _environ);
207 #endif /* WPRFLAG */
208
209 #endif /* _WINMAIN_ */
210     exit(mainret);
211 }
212
213 except ( _XcptFilter(GetExceptionCode(), GetExceptionInformation()) )
214 {
215     /* Should never reach here
216
C source file length: 11,883 lines: 471 Ln: 206 Col: 27 Sel: 4 | 1 Windows (CR LF) UTF-8 INS
```

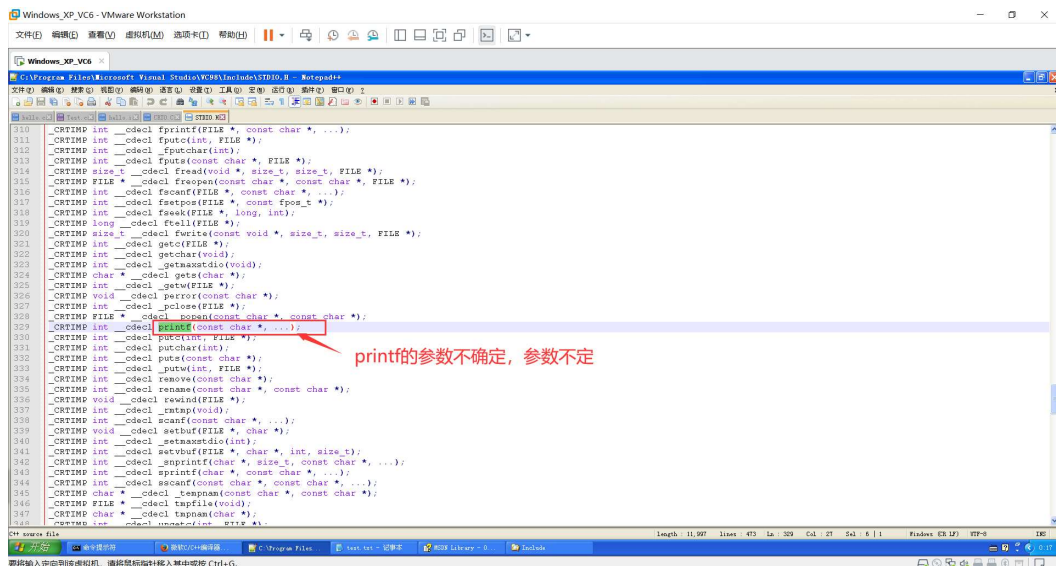
printf

格式化输出到标准输出设备，标准输出设备到默认显示器，参数不定
printf就是完成与操作系统的交互

3环 Ring3 R3 用户态 目态

0环 Ring0 R0 系统态 管态 内核态

可在官方函数的stdio.h中进行查看，代码实现在LIBC.LIB



```
310 _CRTIMP int __cdecl printf(FILE *, const char *, ...);
311 _CRTIMP int __cdecl fprintf(FILE *, const char *, ...);
312 _CRTIMP int __cdecl fputc(int, FILE *);
313 _CRTIMP int __cdecl fputchar(int);
314 _CRTIMP size_t __cdecl fread(void *, size_t, size_t, FILE *);
315 _CRTIMP FILE * __cdecl freopen(const char *, const char *, FILE *);
316 _CRTIMP int __cdecl fseek(FILE *, const char *, ...);
317 _CRTIMP int __cdecl fseeko(FILE *, const char *, ...);
318 _CRTIMP int __cdecl fseek(FILE *, long, int);
319 _CRTIMP long __cdecl ftell(FILE *);
320 _CRTIMP size_t __cdecl fwrite(const void *, size_t, size_t, FILE *);
321 _CRTIMP int __cdecl getc(FILE *);
322 _CRTIMP int __cdecl getc_unlocked(FILE *);
323 _CRTIMP int __cdecl _getchar_n(void);
324 _CRTIMP char * __cdecl _getchar_n(char *, rsize_t, FILE *);
325 _CRTIMP int __cdecl _getchar_n(FILE *);
326 _CRTIMP void __cdecl perror(const char *);
327 _CRTIMP int __cdecl _pclose(FILE *);
328 _CRTIMP FILE * __cdecl _popen(const char *, const char *);
329 _CRTIMP int __cdecl _putenv(const char *, ...);
330 _CRTIMP int __cdecl _putenv_s(const char *, ...);
331 _CRTIMP int __cdecl _putenv_l(const char *, ...);
332 _CRTIMP int __cdecl _putenv_t(const char *, ...);
333 _CRTIMP int __cdecl _putenv_w(const char *, ...);
334 _CRTIMP int __cdecl _remove(const char *);
335 _CRTIMP int __cdecl _rename(const char *, const char *);
336 _CRTIMP void __cdecl _rmdir(FILE *);
337 _CRTIMP int __cdecl _setbuf(FILE *, char *);
338 _CRTIMP int __cdecl _setbuffer(FILE *, char *, rsize_t);
339 _CRTIMP int __cdecl _setvbuf(FILE *, char *, int, rsize_t);
340 _CRTIMP int __cdecl _sprintf(FILE *, const char *, ...);
341 _CRTIMP int __cdecl _sprintf_s(FILE *, const char *, ...);
342 _CRTIMP int __cdecl _sscanf(const char *, const char *, ...);
343 _CRTIMP int __cdecl _scanf(const char *, const char *, ...);
344 _CRTIMP int __cdecl _topen(const char *, const char *);
345 _CRTIMP FILE * __cdecl _tmpfile(void);
346 _CRTIMP char * __cdecl _tmpnam(char *);
347 _CRTIMP int __cdecl _tmpnam_s(char *, rsize_t);
348 _CRTIMP int __cdecl _tmpnam_l(char *, rsize_t);
```

变量就是对地址命名的，变量不做从初始化，其值为上一次使用过的这段内存的残留值

printf、scanf等多参函数是没有条件做类型检查的

数据保留区禁止写入数据，一般会报Cx0000 0005，内存访问写入异常