西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

C程序设计

Programming in C

1011014

主讲：姜学锋，计算机学院

编写规模化程序

3、规模化程序的组织

4、开发工具的项目管理功能

4.10.5 提高编译速度

- C程序代码的运行效率是很高的，但它的编译速度在众多程序语言中算是慢的。一个C程序总是要反复编译、运行、调试才得到正确结果，所以，无论是多文件结构或是单文件程序，是大规模程序或是小型程序，提高编译速度是必需的。

4.10.5 提高编译速度

- 提高编译速度首先要完善程序的代码结构。例如精简头文件、接口与实现完全分离、高度模块化。如果是单个文件且代码量较大时，为了一点小改动就要重新对大面积代码编译，实在是不划算。高度模块化就是低耦合，就是尽可能的减少相互依赖。
- 例如函数与函数之间降低相互依赖，则这两个函数就可以分离到不同的文件中；文件与文件之间，一个头文件的变化，尽量不要引起其他文件的重新编译。

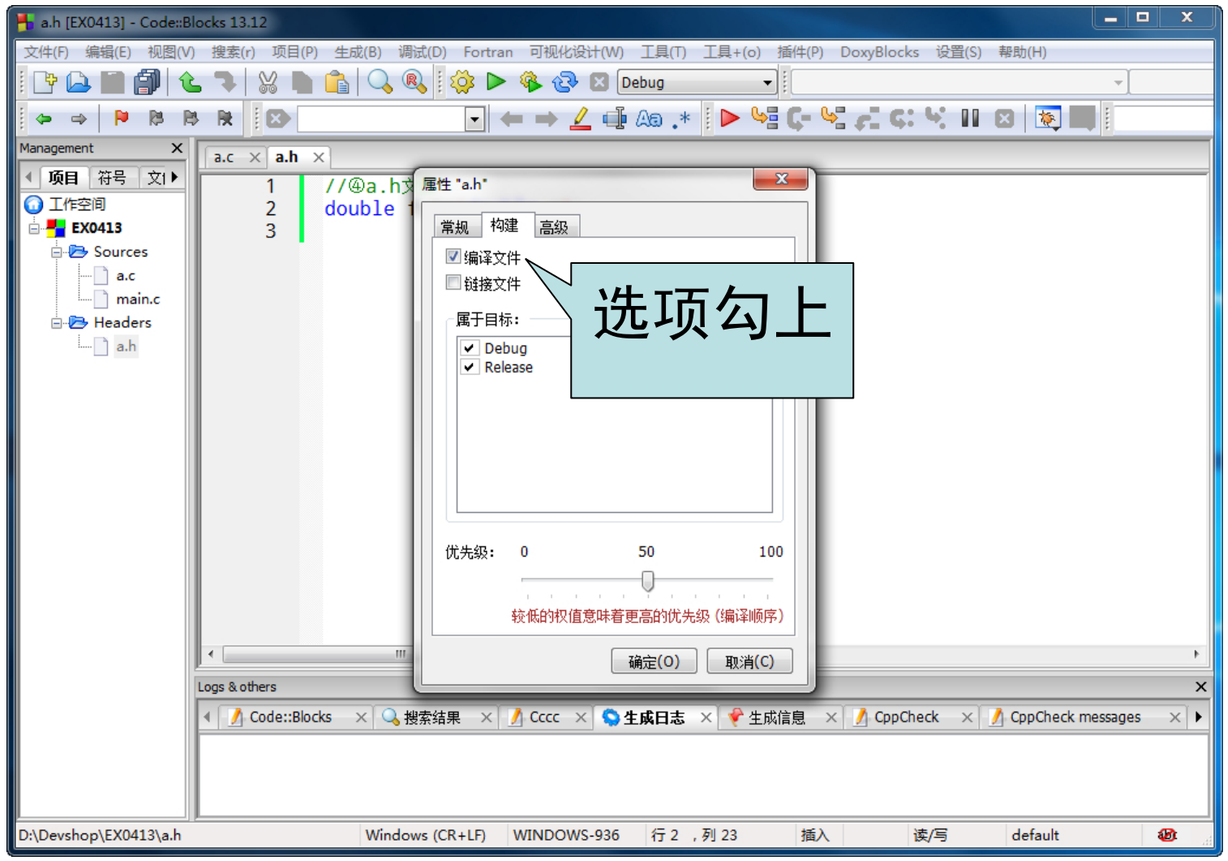
4.10.5 提高编译速度

- 提高编译速度还可以利用编译器的功能，提高编译器工作效率。现今的编译器一般都有下面的功能：
- （1）预编译头文件
- 即使很小的程序都会使用头文件的，特别是那些标准库的头文件，代码量甚至超过程序本身。预编译头文件是指编译器第一次编译文件时，将头文件的编译结果缓存下来，如果下次编译时发现没有新的则它将直接使用缓存结果，不用对头文件再次编译。

4.10.5 提高编译速度

- GCC预编译头文件方法
- 1．打开工程文件，在工程文件浏览中，在要设置为预编译的头文件上，点击鼠标右键，选择“属性（properties）”，得到多页面的属性框，选择“构建(build)”页面，将“编译文件（complie file）”选项勾上（默认状态为没有勾选）。

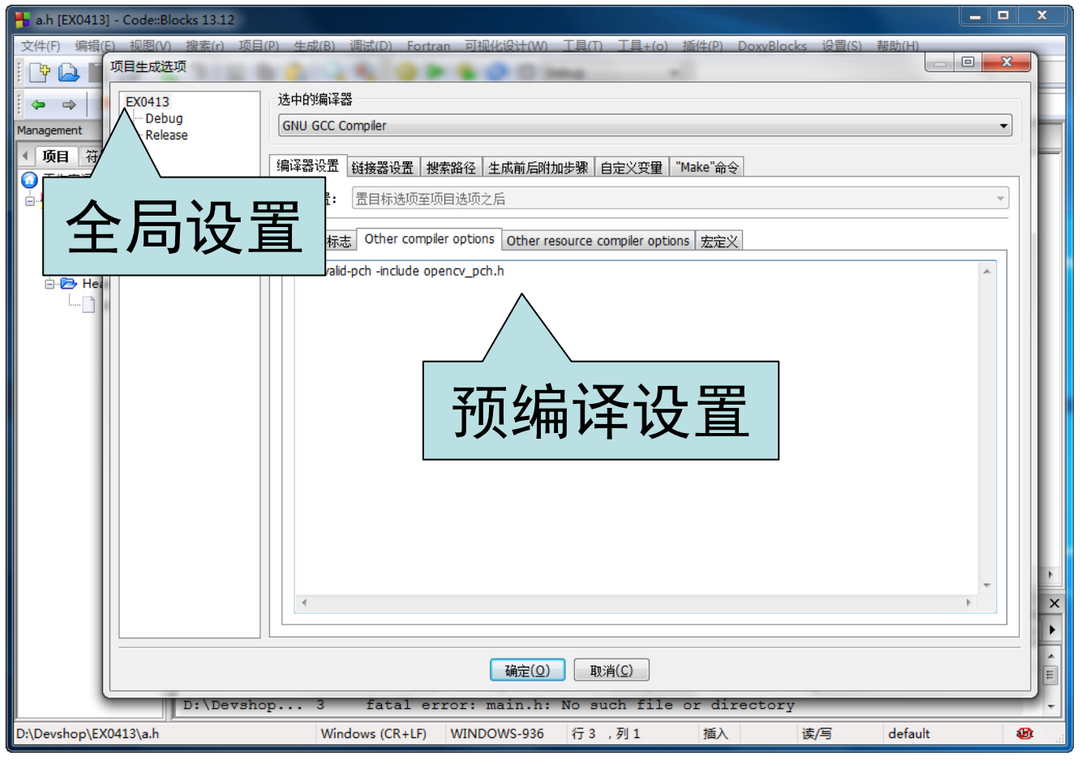
4.10.5 提高编译速度


a.h [EX0413] - Code::Blocks 13.12
文件(F) 编辑(E) 视图(V) 搜索(r) 项目(P) 生成(B) 调试(D) Fortran 可视化设计(W) 工具(T) 工具+(o) 插件(P) DoxyBlocks 设置(S) 帮助(H)
Debug
Management
项目 符号
工作空间
EX0413
Sources
a.c
main.c
Headers
a.h
a.c
a.h
1 //④a.h
2 double
3
属性 "a.h"
常规 构建 高级
编译文件
链接文件
属于目标:
Debug
Release
选项勾上
优先级: 0 50 100
较低的权值意味着更高的优先级（编译顺序）
确定(O) 取消(C)
Logs & others
Code::Blocks 搜索结果 Cccc 生成日志 生成信息 CppCheck CppCheck messages
D:\Devshop\EX0413\a.h
Windows (CR+LF)
WINDOWS-936
行 2 ，列 23
插入
读/写
default

4.10.5 提高编译速度

- GCC预编译头文件方法
- 2．菜单->工程（project）->构建选项（build options），得到一个多页面属性框，选择工程全局设置（project name），然后在“编译器设置（complier setting）”页面中，选择“其它选项（other option）”页面，在该页面的文本编辑框中加入-Winvalid-pch 和 -include 头文件名（例如：-include opencv_pch.h）

4.10.5 提高编译速度



-Winvalid-pch 是对预编译头文件有效性进行检查并提示。-include xxx.h 是所有cpp文件隐含了包含xxx.h文件，即在cpp文件中没有include xxx.h，一样可使用xxx.h中声明的内容。

4.10.5 提高编译速度

- VC预编译头文件方法
- 1．创建需要预编译的头文件，比如wx_pch.h，头文件中务必有防止重复包含的宏，例如：

```
#ifndef STDAFX_H
#define STDAFX_H
//......
#endif
```

4.10.5 提高编译速度

- VC预编译头文件方法
- 2．加入.cpp文件，在文件属性中，取消LINK，只保留编译。（因这个文件只生成pch文件，不生成obj文件）。提高编译优先级为2，默认值为50，使得该文件先于其他任何文件进行编译，并且为该文件指定单独编译配置：
- $compiler /nologo $options $includes /c $file /Ycwx_pch.h

4.10.5 提高编译速度

- VC预编译头文件方法
- 3．在build option中加入编译选项：
- /FI "wx_pch.h"
- /Yu "wx_pch.h"
- /Fp "wx_pch.pch"

/FI：命名强制包含文件，相当于MinGW（gcc）-include

/Yu：使用指定的头文件

/Fp：指定预编译头文件的名称，最好为debug和release分别指定，否则切换debug和release时会重编译头文件

4.10.5 提高编译速度

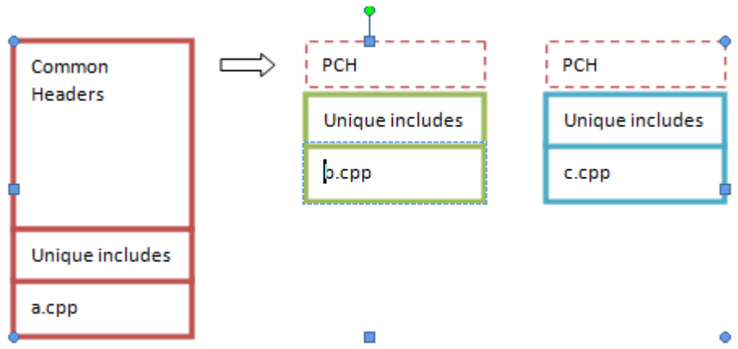
- VC预编译头文件方法
- 4．使用预编译头文件后，如果程序所链接的库发生变化，而这些库是预编译头文件需要用到的，那么会出现下面类似的错误：

```
vc100.pdb不是创建此预编译头时使用的pdb文件。
```

- 如果出现这种情况，需要对整个项目（当前配置如debug）rebuild就好了。

4.10.5 提高编译速度

- （2）增量编译
- 增量编译是指编译器编译得到目标代码后，如果下次编译时发现文件并未经过修改，则使用前次的目标代码而无需再次编译。在多文件结构程序中，调试修改一般是集中少数文件上，所以增量编译节省的编译时间是可观的。



4.10.5 提高编译速度

- （3）编译缓存
- 增量编译是比较目标代码和源文件的时间来决定是否要重新编译。编译缓存是指将前面的编译结果缓存下来，根据文件的内容来判断是否重新编译。例如对一个文件输入一个没有作用的空格会导致增量编译失效，但编译缓存能判别出增加的这个空格是否足以需要重新编译。

4.11 函数应用程序举例

【例4.13】

编写sin(x)、cos(x)、sqrt(x)数学函数演示程序。

T：编写main

A：编写sin

B：编写cos

C：编写sqrt

4.11 函数应用程序举例

例题分析

多文件结构程序的编写和步骤。

（1）首先计划在主程序文件main.c中编写main函数，功能是输出一个小型菜单让用户选择哪个函数要运算，然后提示输入x，再根据菜单选择调用函数得到计算结果。

（2）其次计划将三个数学函数的计算安排在三个不同的函数中实现，且将这三个函数安排到三个文件（a.c、b.c、c.c）中编写。

4.11 函数应用程序举例

例题分析

（3）main函数要调用这三个文件中的函数，需要函数声明，具体做法是将三个文件对应地写出头文件来，头文件的内容是三个函数原型，在main.c中包含。

（4）由于计算角度的sin(x)、cos(x)需要将x转换，用到π，所以对应main.c写出头文件main.h，包含π的符号常量，供其他两个文件包含。

4.11 函数应用程序举例

例4.13

```
1  //①main.c文件
2  #include <stdio.h>
3  #include "a.h"
4  #include "b.h"
5  #include "c.h"
6  int main()
7  {
8     int n;
9     double x;
10    printf("1. sin(x)\n2. cos(x)\n3. sqrt(x)\ninput select
(1-3):"); //菜单
11    scanf("%d",&n); //选择
12    printf("input x:");
13    scanf("%lf",&x); //输入计算数据
14    switch (n) { //根据n分别调用a.c、b.c、c.c的函数
```

4.11 函数应用程序举例

例4.13

```
15      case 1 : printf("sin=%lf\n",fsin(x)); break;
16      case 2 : printf("cos=%lf\n",fcos(x)); break;
17      case 3 : printf("sqrt=%lf\n",fsqrt(x)); break;
18    }
19    return 0;
20 }
```

4.11 函数应用程序举例

例4.13

```
1 //②main.h文件
2 #define PI 3.1415926
```

4.11 函数应用程序举例

例4.13

```
1 //③a.c文件
2 #include <math.h>
3 #include "main.h"
4 double fsin(double x)
5 {
6   return sin(x*PI/180.0);
7 }
```

4.11 函数应用程序举例

例4.13

```
1 //④a.h文件
2 double fsin(double x);
```

4.11 函数应用程序举例

例4.13

```
1 //⑤b.c文件
2 #include <math.h>
3 #include "main.h"
4 double fcos(double x)
5 {
6   return cos(x*PI/180.0);
7 }
```

4.11 函数应用程序举例

例4.13

```
1 //⑥b.h文件
2 double fcos(double x);
```

4.11 函数应用程序举例

例4.13

```
1 //⑦c.c文件
2 #include <math.h>
3 double fsqrt(double x)
4 {
5   return sqrt(x);
6 }
```

4.11 函数应用程序举例

例4.13

```
1 //⑧c.h文件
2 double fsqrt(double x);
```

4.11 函数应用程序举例

例4.13

程序运行屏幕

```
1. sin(x)
2. cos(x)
3. sqrt(x)
input select(1-3):2
input x:60
cos=0.500000
```

2↙
60↙


程序设计