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

# 编程任务的接口与版本控制

1、文件包含

2、第三方函数库的文件包含

## 5.2 文件包含

- 文件包含命令的作用是把指定的文件插入到该命令所处的位置上取代该命令，然后再进行编译处理，相当于将文件的内容“嵌入”到当前的源文件中一起编译。

## 5.2 文件包含

- 假定GCC安装在C:\DEV\MinGW目录中，则C:\DEV\MinGW\include文件夹中有stdio.h文件。

- 或Visual C++ 6.0安装在C:\DEV\MSVS6目录中，则C:\DEV\MSVS6\VC98\Include文件夹中有stdio.h文件。

## 5.2 文件包含

- 示例

```
1 ...
2 #include <stdio.h>
3 ...
4 int main()
5 {
6   ...
7 }
```

GCC/mingw stdio.h文件

```
1 #ifndef _STDIO_H_
2 #define _STDIO_H_
3 #include <_mingw.h>
4 #ifndef RC_INVOKED
5 #define __need_size_t
6 #define __need_NULL
7 #define __need_wchar_t
8 #define __need_wint_t
9 #include <stddef.h>
10 #define __need___va_list
11 #include <stdarg.h>
12 #endif  /* Not RC_INVOKED */
13 #define _IOREAD 1
14 #define _IOWRT  2
15 ......
```

## 5.2 文件包含

- 文件包含命令为#include，有两种命令形式：
- ①第一种形式：

```
#include <头文件名>
```

- ②第二种形式：

```
#include “头文件名”
```

## 5.2 文件包含

- （1）一个#include命令只能包含一个头文件，包含多个头文件要用多个#include命令，且每个文件包含命令占一行。通常，头文件的扩展名为.h或者.hpp。

- （2）第一种形式与第二种形式的区别是编译器查找头文件的搜索路径不一样。第一种形式仅在编译器INCLUDE系统路径中查找头文件，第二种形式先在源文件所处的文件夹（用户路径）中查找头文件，如果找不到，再在系统路径中查找。

- 一般地，如果调用标准库函数或者专业库函数包含头文件时，使用第一种形式；包含程序员自己编写的头文件时，将头文件放在源文件所处的文件夹中且使用第二种形式。

## 5.2 文件包含

- 示例

```
#include <stdio.h> //包含系统提供的标准输入输出头文件
#include "gsl.h"   //包含其他专业函数库的头文件
#include "menu.h"  //包含自己编写的头文件
```

- （3）头文件的内容通常是函数声明、全局性常量、数据类型声明、宏定义等信息，一般不包括定义，例如函数定义、变量定义等。所起的作用就是为其他程序模块提供声明性信息，而定义、函数实现代码等应放在源文件中。

## 5.2 文件包含

- （4）可以包含.C或.CPP扩展名的源文件。但在实际编程中，如果程序是由多个源文件组成，一般采用工程方式来集合，而不是使用文件包含命令。

## 5.2 文件包含

- 1. 文件包含的路径问题
- 文件包含命令中的头文件名可以写成绝对路径的形式，例如：

```
#include "C:\DEV\GSL\include\gsl_linalg.h"
#include <C:\DEV\SDL\include\SDL.h>
```

- 这时直接按该路径打开头文件，此时第一种形式或第二种形式的命令没有区别。请注意，由于文件包含命令不属于C语言语法，因此"C:\DEV\GSL\include\gsl_linalg.h"不能理解为字符串，其中的“\”不要写成“\\”。

## 5.2 文件包含

- 头文件名也可以写成相对路径的形式，例如：

```
#include <math.h>
#include <zlib\zlib.h>
#include "user.h"
#include "share\a.h"
```

- 这时的文件包含命令是相对系统INCLUDE路径或用户路径来查找头文件的。

## 5.2 文件包含

- 假设编译器系统INCLUDE路径为“C:\DEV\MinGW\include”，则

```
#include <math.h> //math.h在C:\DEV\MinGW\include
#include <zlib\zlib.h> //zlib.h在C:\DEV\MinGW\include\zlib
```

- 如果在上述路径中找不到头文件，会出现编译错误。

## 5.2 文件包含

- 假设用户路径为“D:\Devshop”，则

```
#include "user.h"
//user.h在D:\Devshop或C:\DEV\MinGW\include
#include "share\a.h“
//a.h在D:\Devshop\share或C:\DEV\MinGW\include\share
```

- 如果在上述路径中找不到头文件，会出现编译错误。

- 2．文件包含的重复包含问题
- 头文件有时需要避免重复包含（即多次包含），例如一些特定声明不能多次声明，而且重复包含增加了编译时间。这时可以采用以下两个办法之一。

## 5.2 文件包含

- （1）使用条件编译。例如：

```
1 #if !defined(_FILE1_H_C6793AB5__INCLUDED_)
2 #define _FILE1_H_C6793AB5__INCLUDED_
3
4 .......//头文件内容
5
6 #endif
```

即将头文件内容放在一个条件编译块中，第1次编译时编译条件成立，故继续往下编译，第2行使编译条件为假，这样再次编译头文件时，头文件内容就不会编译了。条件中的宏定义“_FILE1_H_C6793AB5__INCLUDED_”，为了与其他编译条件相区别，故意写得很长、写得很怪。

- （2）使用特殊预处理命令#pragma，例如：

```
1 #pragma once
2 ..... //头文件内容
```

即在头文件第1行增加这个预处理命令，它的意思是：在编译一个源文件时，只对该文件包含（打开）一次。

## 使用专业函数库的方法与步骤

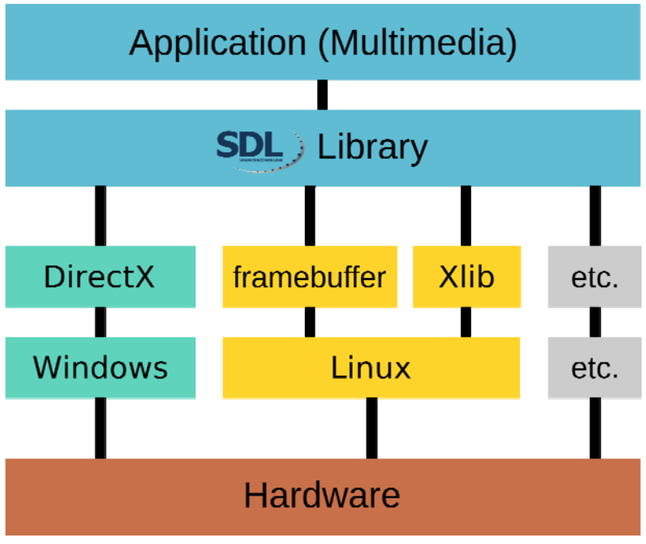
SDL（Simple DirectMedia Layer）是一套开放源代码的跨平台多媒体开发库，使用C语言写成。




## 5.2 文件包含

### 使用专业函数库的方法与步骤

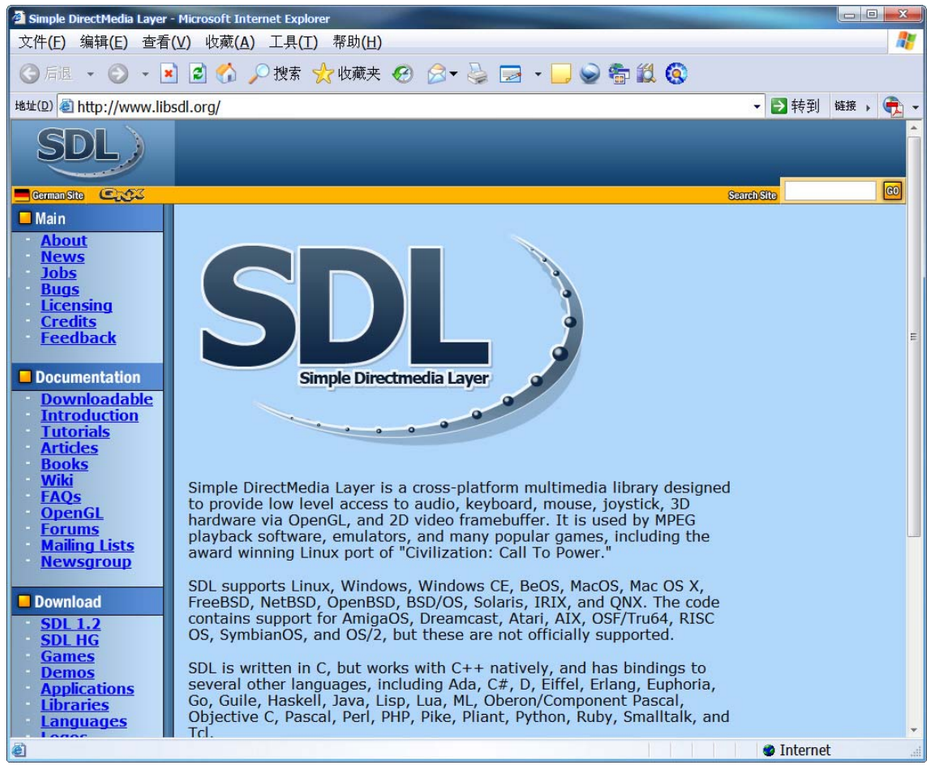
SDL提供了数种控制图像、声音、输出入的函数，让开发者只要用相同或是相似的代码就可以开发出跨多个平台（Linux、Windows、Mac OS X等）的应用软件。目前SDL多用于开发游戏、模拟器、媒体播放器等多媒体应用领域。




# 5.2 文件包含

## 使用专业函数库的方法与步骤

SDL网站 http://www.libsdl.org/



## 5.2 文件包含

### 使用专业函数库的方法与步骤

（1）从SDL网站“download”下载 Win32版本的“Runtime Libraries”，例如 SDL-1.2.14-win32.zip。

1.2.14是版本号，可能得到更高版本。

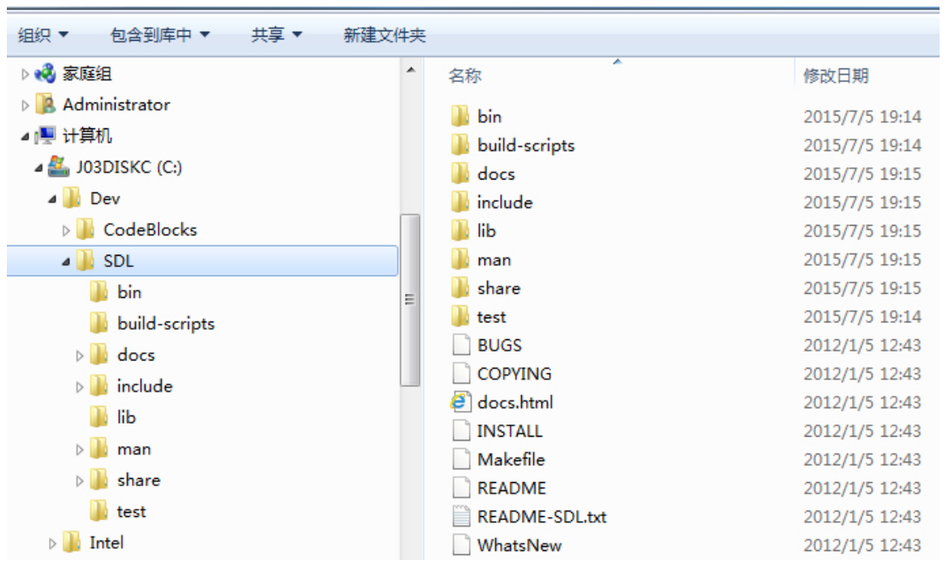
（2）再下载 Win32版本的“Development Libraries”，其中SDL-devel-1.2.14-VC6.zip用于Visual C++ 6.0，SDL-devel-1.2.14-VC8.zip用于Visual Studio 2008，SDL-devel-1.2.14-mingw32.tar.gz用于GCC。

1.2.14是版本号，可能得到更高版本。

（3）解压缩下载文件。

## 使用专业函数库的方法与步骤

得到如图所示的目录和文件结构




注意：

C:\DEV\SDL\lib

C:\DEV\SDL\include

目录

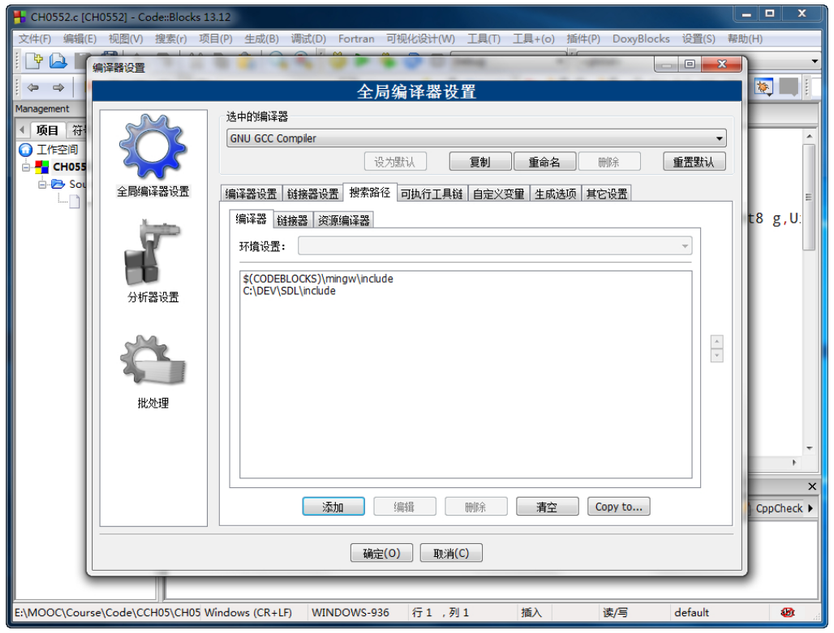
# 5.2 文件包含

## 使用专业函数库的方法与步骤

（1）单击Code::Blocks菜单“设置/编译器和调试器”。

（2）在如图所示的窗口中，选择“搜索路径/编译器”。

（3）添加“C:\DEV\SDL\include”。

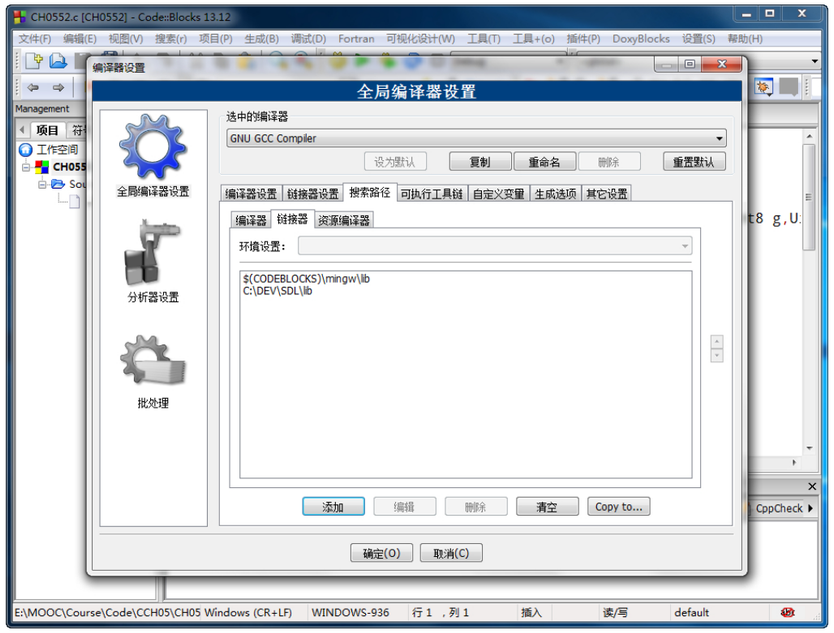


# 5.2 文件包含

## 使用专业函数库的方法与步骤

（1）在如图所示的窗口中，选择“搜索路径/链接器”。

（2）添加“C:\DEV\SDL\lib”。



## 5.2 文件包含

### 使用专业函数库的方法与步骤

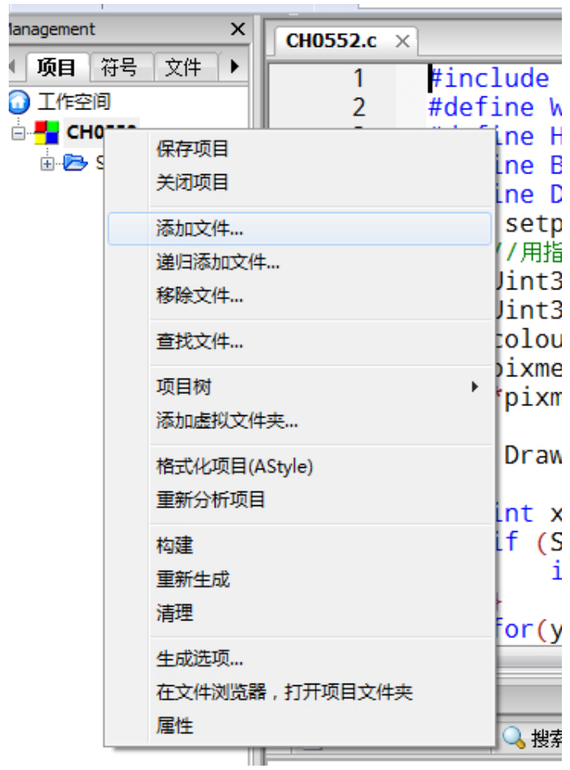
在使用SDL库的程序中加入SDL接口头文件

```
#include <SDL/SDL.h>
```

SDL接口头文件是在C:\DEV\SDL\include中SDL子目录中，因此<SDL/SDL.h>。

## 使用专业函数库的方法与步骤

如图所示，在项目工程名（如CH0552）右键，再单击“构建选项”。



# 5.2 文件包含

## 使用专业函数库的方法与步骤

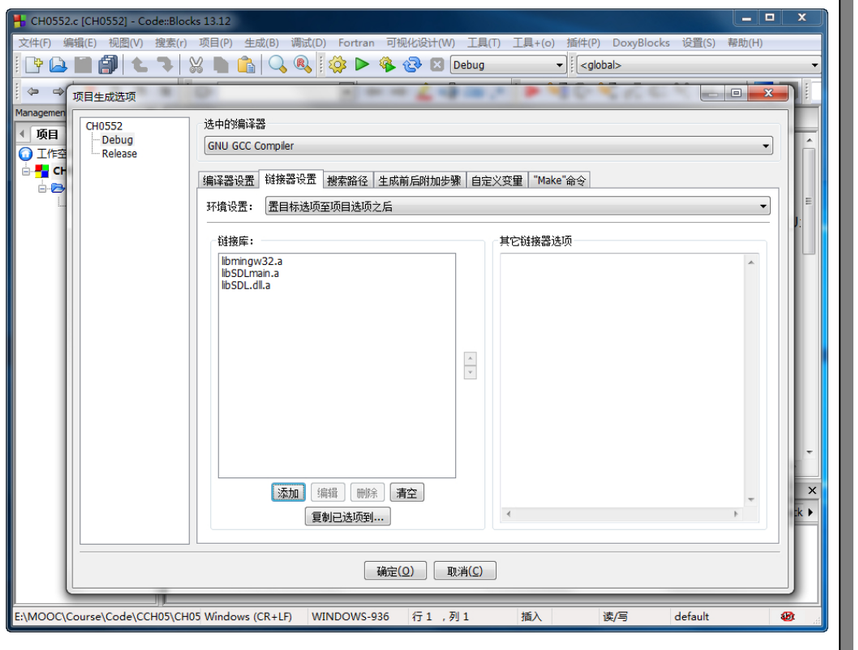
（1）如图所示，在“构建选项”对话框中选择“链接器设置”。

（2）然后按下面顺序依次添加下面库文件：

libmingw32.a

libSDLmain.a

libSDL.dll.a



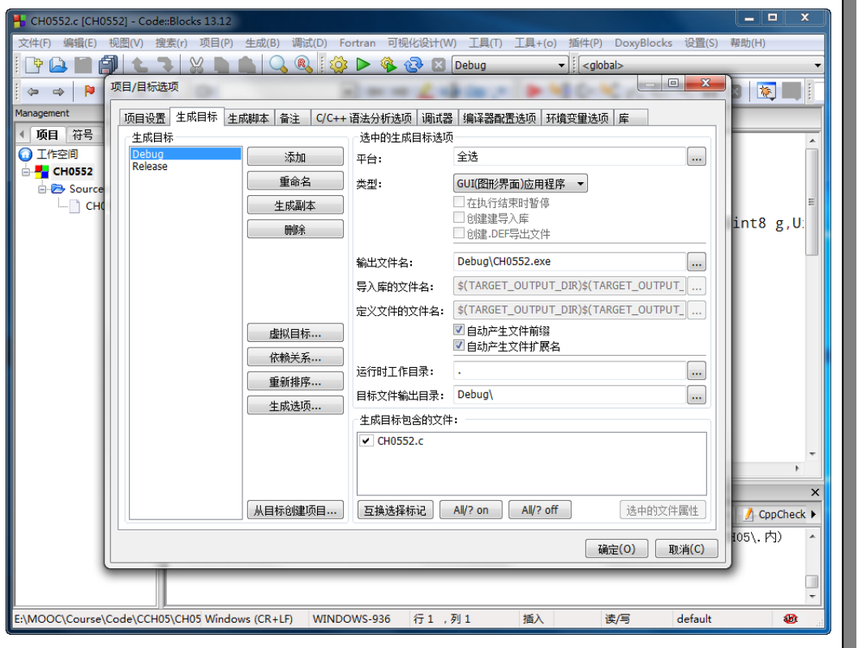
# 5.2 文件包含

## 使用专业函数库的方法与步骤

（1）然后项目工程名（如CH0552）右键，单击“属性”。

（2）如图所示，“项目属性”对话框中选择“构建目标/类型”为“GUI图形界面应用程序”。

这是SDL必须要设置的一个步骤。



## 5.2 文件包含

### 使用专业函数库的方法与步骤

至此，完成SDL开发环境的配置，程序可以使用SDL库了。

若要运行使用SDL库的程序，需要将它的“SDL.dll”动态链接库复制到

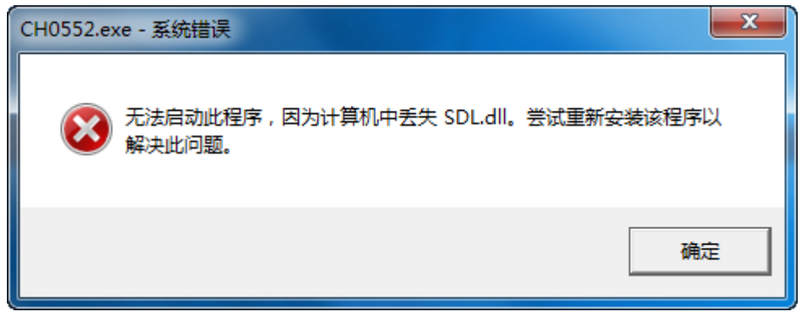
（1）程序所在的目录中；

（2）Windows/System32文件夹中。

其中（1）对每个不同的程序都要复制，（2）做1次对所有程序都有效。

## 使用专业函数库的方法与步骤

运行程序时找不到*.DLL的问题



## 5.2 文件包含

例5.52

```
  1 #include <SDL/SDL.h>
  2 #define WIDTH 640 //屏幕宽
  3 #define HEIGHT  480 //屏幕高
  4 #define BPP   4 //屏幕像素位
  5 #define DEPTH 32  //屏幕像素深度
  6 void setpixel(SDL_Surface *screen,int x,int y,Uint8 r,Uint
8 g,Uint8 b)
  7 { //用指定颜色画点
  8   Uint32 *pixmem32;
  9   Uint32 colour;
 10   colour = SDL_MapRGB(screen->format,r,g,b);
 11   pixmem32 = (Uint32*)screen->pixels+y+x;
 12   *pixmem32= colour;
 13 }
 14 void DrawScreen(SDL_Surface* screen,int h)
```

## 5.2 文件包含

例5.52

```
15 {
16   int x, y, ytimesw;
17   if (SDL_MUSTLOCK(screen)) {
18     if (SDL_LockSurface(screen)<0) return;
19   }
20   for(y=0; y<screen->h; y++) {
21     ytimesw = y*screen->pitch/BPP;
22     for(x= 0; x<screen->w; x++)
23     setpixel(screen,x,ytimesw,(x*x)/256+3*y+h,(y*y)/256+x+
h,h);
24   }
25   if (SDL_MUSTLOCK(screen)) SDL_UnlockSurface(screen);
26   SDL_Flip(screen);
27 }
28 int main(int argc, char* argv[])
```

## 5.2 文件包含

例5.52

```
 29 {
 30   SDL_Surface *screen;
 31   SDL_Event event;
 32   int keypress=0;
 33   int h=0;
 34   if (SDL_Init(SDL_INIT_VIDEO)<0 ) return 1; //SDL初始化
 35   screen=SDL_SetVideoMode(WIDTH,HEIGHT,DEPTH,SDL_FULLSCREE
N|SDL_HWSURFACE);
 36   if (!(screen)) { //SDL创建屏幕
 37     SDL_Quit();
 38     return 1;
 39   }
 40   while(!keypress) {
 41     DrawScreen(screen,h++); //绘图
 42     while(SDL_PollEvent(&event)) { //检测事件
```

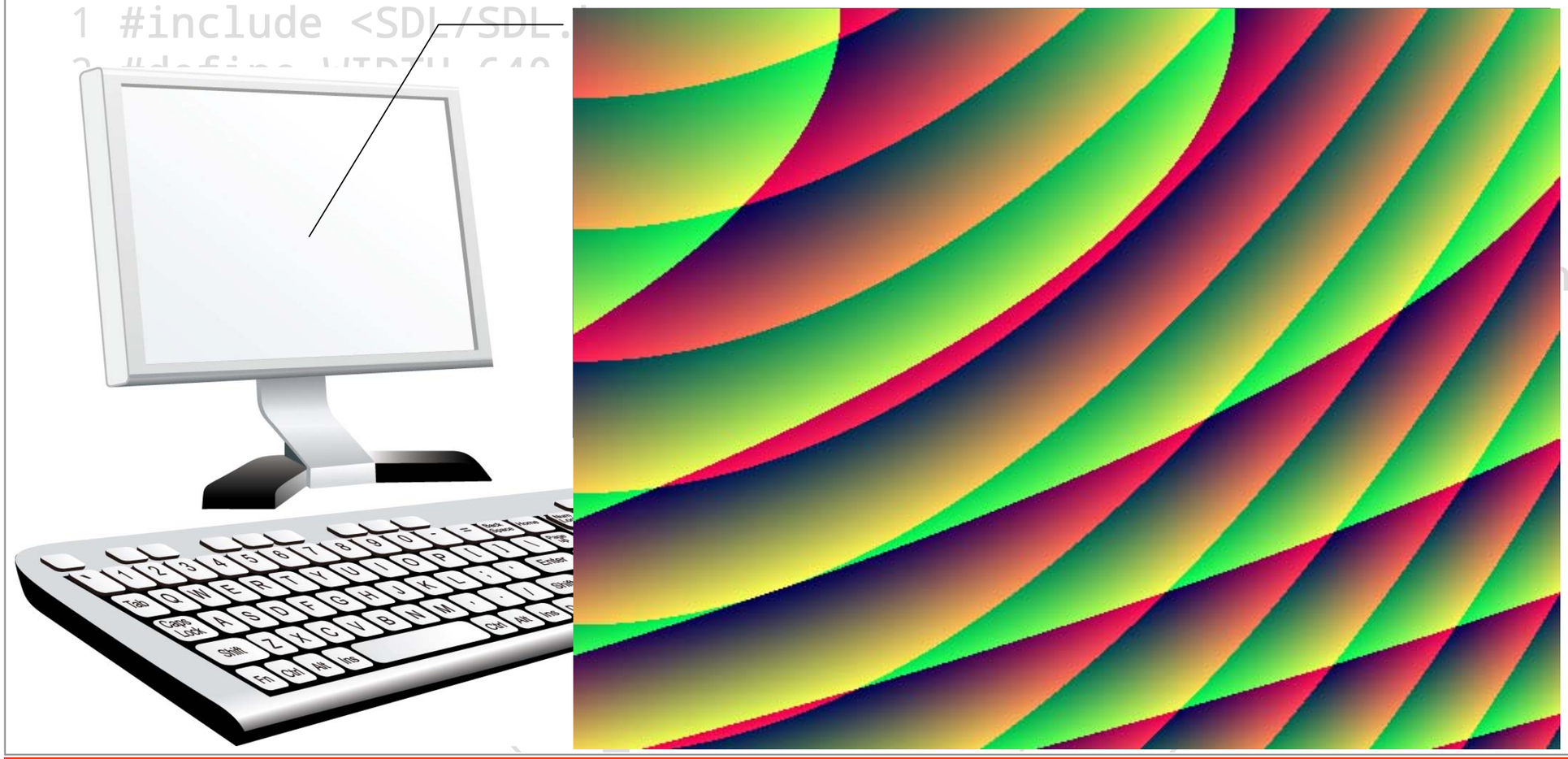
## 5.2 文件包含

例5.52

```
43          switch (event.type) { //事件类型
44            case SDL_QUIT:
45              keypress = 1;
46              break;
47            case SDL_KEYDOWN:
48              keypress = 1;
49              break;
50          }
51        }
52    }
53    SDL_Quit(); //SDL结束处理
54    return 0;
55 }
```

## 5.2 文件包含

例5.52




程序设计