lab1：显示系统的进程，注意如何写入字符串

```
#include <windows.h>
#include <stdlib.h>
#include <cstdlib>
#include <stdio.h>
#include <tlhelp32.h>
//*************************************************************************************
*********

BOOL InitApplication(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow);
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam);

int WINAPI WinMain(HINSTANCE hInstance,     // 入口函数
                   HINSTANCE,
                   LPSTR     lpCmdLine,
                   int       nCmdShow  )
{
    if (!InitApplication(hInstance))       // 应用初始化
        return FALSE;

    if (!InitInstance(hInstance,nCmdShow)) // 实例初始化
        return FALSE;

    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0))   // 消息循环
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int)msg.wParam;
}

//*************************************************************************************
*********

BOOL InitApplication(HINSTANCE hInstance)   // 应用初始化
{
    WNDCLASS  wc;  // Data structure of the window class

    wc.style            = CS_HREDRAW|CS_VREDRAW;
    wc.lpfnWndProc      = (WNDPROC)MainWndProc;  // Name of the Window Function
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance        = hInstance;
    wc.hIcon            = LoadIcon (NULL, IDI_APPLICATION);
    wc.hCursor          = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground    = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName     = NULL;
```

```
    wc.lpszClassName    = TEXT("2020MPADLab1: Windows进程列表 作者学号：10185102153
姓名：汪子凡");  // Name of the window class

    return RegisterClass(&wc);
}

//*************************************************************************************
*********

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  // 实例初始化
{
    HWND hWnd = CreateWindow(TEXT("2020MPADLab1: Windows进程列表 作者学号：
10185102153 姓名：汪子凡"),     // Name of the window class
                             TEXT("2020MPADLab1: Windows进程列表 作者学号：
10185102153 姓名：汪子凡"), // Title of the window
                             WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT,
                             CW_USEDEFAULT,
                             1300,
                             600,
                             NULL,
                             NULL,
                             hInstance,
                             NULL                                        );
    if (!hWnd) return FALSE;

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    return TRUE;
}

//*************************************************************************************
*********

// 窗口过程函数

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
    LPCTSTR ID = TEXT("ID");   // 显示的内容
    LPCTSTR NAME = TEXT("Name");

    PAINTSTRUCT ps;
    HDC hdc;

    switch (message) {

        case WM_PAINT:  // 窗口客户区得刷新
        {

            hdc = BeginPaint (hWnd, &ps);
            for(int i = 10; i < 400 * 3; i += 400)                  //输出第一行
            {
                TextOut(hdc, i, 10, ID,lstrlen(ID));
                TextOut(hdc, i + 100, 10, NAME,lstrlen(NAME));
                printf("%d %d\n", i, i + 100);
            }
```

```
            PROCESSENTRY32 pe32;
            HANDLE hProcessSnap = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS,
0);
            pe32.dwSize = sizeof(pe32);
            BOOL bMore = Process32First(hProcessSnap, &pe32);

            int cnt = 0;
            TCHAR String_ID[20];
            LPCTSTR String_File;
            while (bMore)
            {
                //printf("%06x\n", (unsigned)pe32.th32ProcessID);
                wsprintf(String_ID, TEXT("%06x"),
(unsigned)pe32.th32ProcessID);

                String_File = pe32.szExeFile;
                TextOut(hdc, cnt%3*400 + 10, (cnt/3+2)*20,
String_ID,lstrlen(String_ID));
                TextOut(hdc, cnt%3*400 + 110, (cnt/3+2)*20,
String_File,lstrlen(String_File));

                bMore = Process32Next(hProcessSnap, &pe32);
                cnt++;
            }
            CloseHandle(hProcessSnap);


            EndPaint(hWnd, &ps);

            return 0;
        }
        case WM_DESTROY: // 窗口关闭

            PostQuitMessage(0);

            return 0;

       default:  // 缺省消息的处理

           return DefWindowProc(hWnd, message, wParam, lParam);
   }

}
```

lab2: wx展现lab1

```
# -*- coding: utf8 -*-

import wx
import psutil

class MyFrame(wx.Frame):
```

```
  def __init__(self):
    wx.Frame.__init__(self, None, -1, u"2020MPADLab2: Windows/Linux进程列表 作者学号: 10185102153 姓名:汪子凡", size = (1200, 800))
    panel = wx.Panel(self, -1)

    icon = wx.Icon(name = "icon1.ico", type = wx.BITMAP_TYPE_ICO)
    self.SetIcon(icon)

    for i in range(4):
      wx.StaticText(panel, -1, u"进程ID:    模块名:", pos = (i * 300, 10))
    cnt = 0
    ACCESS_DENIED=''
    for pid in sorted(psutil.pids()):
      try:
        p = psutil.Process(pid)
        pinfo = p.as_dict(ad_value=ACCESS_DENIED)
        wx.StaticText(panel, -1, u"%05x      %s"%(pid,pinfo['name']), pos = (cnt % 4 * 300, (cnt // 4 + 1) * 20 + 10))
        cnt += 1
      except psutil.NoSuchProcess: pass

if __name__ == '__main__':
  app = wx.App()
  frame = MyFrame()
  frame.Show(True)
  app.MainLoop()
```

lab3，4:修改图标，动态生成菜单，设置菜单属性，设置加速键，消息框


```
#include <windows.h>
#include <stdio.h>
#include "resource.h"

LPCTSTR showIcon[3] = {"当前使用的图标是：图标1", "当前使用的图标是：图标2", "当前使用的图标是：图标3"};
LPCTSTR showMesg[5] = {"显示信息1", "显示信息2", "显示信息3", "显示信息4", "
        "};
int id_icon[] = {IDI_ICON1, IDI_ICON2, IDI_ICON3};
HWND hWnd;
```

```
//*************************************************************************************
*********
BOOL InitApplication(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow);
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam);

int WINAPI WinMain(HINSTANCE hInstance,     // 入口函数
                   HINSTANCE,
                   LPSTR     lpCmdLine,
                   int       nCmdShow  )
{
    if (!InitApplication(hInstance))       // 应用初始化
        return FALSE;

    if (!InitInstance(hInstance,nCmdShow)) // 实例初始化
        return FALSE;

    MSG msg;
    HACCEL haccel =
LoadAccelerators(hInstance,MAKEINTRESOURCE(IDR_ACCELERATOR1));           //Load
进加速键

    while (GetMessage(&msg, NULL, 0, 0))   // 消息循环
    {
        if(!TranslateAccelerator(hWnd, haccel, &msg))
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int)msg.wParam;
}

//*************************************************************************************
*********

BOOL InitApplication(HINSTANCE hInstance)   // 应用初始化
{
    WNDCLASS  wc;  // Data structure of the window class

    wc.style            = CS_HREDRAW|CS_VREDRAW;
    wc.lpfnWndProc      = (WNDPROC)MainWndProc;  // Name of the Window Function
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance        = hInstance;
    //wc.hIcon            = LoadIcon (NULL, IDI_APPLICATION);
    wc.hIcon            = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));
    wc.hCursor          = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground    = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName     = MAKEINTRESOURCE(IDR_MENU1);               //这个宏将资源
的ID表示转化为字符串表示，或者直接用字符串
    wc.lpszClassName    = TEXT("My1stWClass");  // Name of the window class

    return RegisterClass(&wc);
}

//*************************************************************************************
*********
```

```
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  // 实例初始化
{
    hWnd = CreateWindow(TEXT("My1stWClass"),     // Name of the window class
                        TEXT("2020MPADLab3: Windows资源使用(1) 姓名：汪子凡 学
号：10185102153"), // Title of the window
                        WS_OVERLAPPEDWINDOW,                    //窗口风格
                        CW_USEDEFAULT,                        //初始位置
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,                         //宽高
                        CW_USEDEFAULT,
                        NULL,                                   //父窗口句柄
                        NULL,                                   //菜单句柄，
需要的是句柄,可以在两个地方加载菜单，两个都有，以这个地方为准
                        hInstance,
                        NULL                                        );
    if (!hWnd) return FALSE;

    HMENU hMenuMain = GetMenu(hWnd);
    HMENU hMenuFile = CreatePopupMenu();
      //动态生成File下拉菜单
    InsertMenu(hMenuMain, 0, MF_BYPOSITION|MF_STRING|MF_POPUP, (UINT)hMenuFile,
"&File");  //将File菜单插入主菜单
    AppendMenu(hMenuFile, MF_STRING, ID_FILE_EXIT,
(LPTSTR)"Exit\tCtrl+Shift+Delete");     //在File菜单下插入子菜单(并赋予ID)

     //ID_FILE_EXIT在rc文件中定义了加速键
    CheckMenuRadioItem(hMenuMain,ID_40002,ID_40004, ID_40002, MF_BYCOMMAND);
        //初始选中第一个图标状态

    ShowWindow(hWnd, nCmdShow);                                 //产生
VM_SHOWWINDOW消息和VM_SIZE消息
    UpdateWindow(hWnd);                                         //产生VM_PAINT
消息

    HDC hdc = GetDC(hWnd);
    TextOut(hdc, 10, 10, showIcon[0], lstrlen(showIcon[0]));
            //显示文字
    ReleaseDC(hWnd, hdc);

    return TRUE;
}

//*************************************************************************
*********

// 窗口过程函数

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
    HDC hdc;                            //设备上下文句柄
    //PAINTSTRUCT ps;
    static HINSTANCE hInst;
    switch (message) {
       case WM_CREATE:                                  //获取应用的句柄
         {
             hInst = ((LPCREATESTRUCT)lParam)-> hInstance;
```

```
        }

        /*case WM_PAINT:
        {
           hdc = BeginPaint(hWnd, &ps);
           TextOut(hdc, 500, 500, showMesg[0], lstrlen(showMesg[0]));
           EndPaint(hWnd, &ps);                    //用这两个函数可以告诉系统需要重
画的地方已经重画，而getDC不行，会一直重复命令
        }*/

        case WM_INITMENU:
        {
          // HMENU hMenuMain = (HMENU)wParam;
        }

        case WM_COMMAND:                                        //若是COMMAND消息
        {
           HMENU hMenuMain = GetMenu(hWnd);
           switch(LOWORD(wParam))
           {
               printf("%d\n", LOWORD(wParam));
               case ID_40002:                                   //若是选中图标下的按
键
               case ID_40003:
               case ID_40004:
               {
                   printf("%d\n", LOWORD(wParam));
                   int d = LOWORD(wParam) - ID_40002;                    //
计算一下要改成几个图标(从0开始)
                   //if(!(GetMenuState(hMenuMain,LOWORD(wParam), MF_BYCOMMAND)
& MF_CHECKED))      //如果这个图标未被选中
                   {
                       printf("123");
                    if(MessageBox (hWnd, TEXT("确认要修改图标吗？"), TEXT("确认"),
 MB_YESNO|MB_ICONQUESTION ) == IDYES )   //消息框进行确认
                    {
                       CheckMenuRadioItem(hMenuMain,ID_40002,ID_40004,
LOWORD(wParam), MF_BYCOMMAND);           //先改变菜单的选中状态，再改变图标
                       SetClassLong(hWnd, GCL_HICON, (long)LoadIcon(hInst,
MAKEINTRESOURCE(id_icon[d])));
                       if(d == 2)
     //若是第三个图标被选中，灰化
                       {
                           EnableMenuItem(hMenuMain, 2,
MF_GRAYED|MF_BYPOSITION);
                           DrawMenuBar(hWnd);
                       }
                       else
    //若不是第三个图标，不灰化
                       {
                           EnableMenuItem(hMenuMain, 2,
MF_ENABLED|MF_BYPOSITION);
                           DrawMenuBar(hWnd);
                       }
                       hdc = GetDC(hWnd);
    //更新文字
                       TextOut(hdc, 10, 10, showIcon[d],
lstrlen(showIcon[d]));
```

```
                        ReleaseDC(hWnd, hdc);
                    }
                }
                return 0;
            }

            case ID_40005:                                          //若是要打印消
息
            case ID_40006:
            case ID_40007:
            case ID_40008:
            {
                int d2 = LOWORD(wParam) - ID_40005;
  //计算要打印第几个消息(从0开始)
                if(!(GetMenuState(hMenuMain,LOWORD(wParam), MF_BYCOMMAND) &
MF_CHECKED))        //先检查一开始是否已被选中
                {
                    CheckMenuItem(hMenuMain, LOWORD(wParam), MF_CHECKED);
              //先改变菜单的选中状态，再改变文字
                    hdc = GetDC(hWnd);
                    TextOut(hdc, 100 + d2 / 2 * 100, 100 + d2 % 2 * 100,
showMesg[d2], lstrlen(showMesg[d2]));
                    ReleaseDC(hWnd, hdc);
                }
                else
      //若是已被选中，则让消息消失
                {
                    CheckMenuItem(hMenuMain, LOWORD(wParam), MF_UNCHECKED);
                    hdc = GetDC(hWnd);
                    TextOut(hdc, 100 + d2 / 2 * 100, 100 + d2 % 2 * 100,
showMesg[4], lstrlen(showMesg[4]));
                    ReleaseDC(hWnd, hdc);
                }
                return 0;
            }

            case ID_40009:                                          //处理
关于消息
            {
                MessageBox(hWnd, TEXT("2020MPADLab3: Windows资源使用(1)\n图
标、菜单、动态菜单、加速键、消息框\n作者学号：10185102153 姓名：汪子凡\n"),
                           TEXT("2020MPAD"), MB_ICONEXCLAMATION|MB_OK);
                return 0;
            }

            case ID_FILE_EXIT:                                      //处理动
态生成的Exit消息
            {
                printf("11111");
                PostQuitMessage(0);
                return 0;
            }
            return 0;
        }
    }

     //case WM_DESTROY: // 窗口关闭
    //     printf("11111111");
```

```
        //    PostQuitMessage(0);                    //设置为0表示程序正常结束

            return 0;

       default:  // 缺省消息的处理

           return DefWindowProc(hWnd, message, wParam, lParam);
   }

}
```

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ 生成的包含文件。
// 供 Resource.rc 使用
//
#define IDI_ICON1                       101
#define IDI_ICON2                       102
#define IDI_ICON3                       103
#define IDR_MENU1                       104
#define IDR_ACCELERATOR1                105
#define ID_FILE_EXIT                    40001
#define ID_40002                        40002
#define ID_40003                        40003
#define ID_40004                        40004
#define ID_40005                        40005
#define ID_40006                        40006
#define ID_40007                        40007
#define ID_40008                        40008
#define ID_40009                        40009
#define ID_ACCELERATOR40023             40023

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        106
#define _APS_NEXT_COMMAND_VALUE         40024
#define _APS_NEXT_CONTROL_VALUE         1001
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "winresrc.h"

////////////////////////////////////////////////////////////////////////////
```

```
#undef APSTUDIO_READONLY_SYMBOLS

/////////////////////////////////////////////////////////////////////////////
// 中文(简体，中国) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_CHS)
LANGUAGE LANG_CHINESE, SUBLANG_CHINESE_SIMPLIFIED
#pragma code_page(936)

#ifdef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE
BEGIN
    "#include ""winresrc.h""\r\n"
    "\0"
END

3 TEXTINCLUDE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED


/////////////////////////////////////////////////////////////////////////////
//
// Icon
//

// Icon with lowest ID value placed first to ensure application icon
// remains consistent on all systems.
IDI_ICON1               ICON                    "icon1.ico"

IDI_ICON2               ICON                    "icon2.ico"

IDI_ICON3               ICON                    "icon3.ico"


/////////////////////////////////////////////////////////////////////////////
//
// Menu
//

IDR_MENU1 MENU
BEGIN
    POPUP "图标(&I)"
    BEGIN
```

```
        MENUITEM "图标(&X)\tCtrl+X",               ID_40002
        MENUITEM "图标(&Y)\tCtrl+Y",               ID_40003
        MENUITEM "图标(&Z)\tCtrl+Z",               ID_40004
    END
    POPUP "显示(&D)"
    BEGIN
        MENUITEM "显示(&1)\tCtrl+1",               ID_40005
        MENUITEM "显示(2)\tCtrl+2",                ID_40006
        MENUITEM "显示(3)\tCtrl+3",                ID_40007
        MENUITEM "显示(4)\tCtrl+4",                ID_40008
    END
    POPUP "关于(&A)"
    BEGIN
        MENUITEM "程序信息(&I)\tF1",                ID_40009
    END
END


/////////////////////////////////////////////////////////////////////////////
//
// Accelerator
//

IDR_ACCELERATOR1 ACCELERATORS
BEGIN
    "^X",           ID_40002,               ASCII,  NOINVERT
    "^Y",           ID_40003,               ASCII,  NOINVERT
    "^Z",           ID_40004,               ASCII,  NOINVERT
    "1",            ID_40005,               VIRTKEY, CONTROL, NOINVERT
    "2",            ID_40006,               VIRTKEY, CONTROL, NOINVERT
    "3",            ID_40007,               VIRTKEY, CONTROL, NOINVERT
    "4",            ID_40008,               VIRTKEY, CONTROL, NOINVERT
    VK_F1,          ID_40009,               VIRTKEY, NOINVERT
    VK_DELETE,      ID_FILE_EXIT,           VIRTKEY, SHIFT, CONTROL, NOINVERT
END

#endif    // 中文(简体，中国) resources
/////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
/////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//


/////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

```
# -*- coding: utf8 -*-

import wx
```

```
class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, u"2020MPADLab2: Windows/Linux进程列表 作者学号: 10185102153 姓名:汪子凡", size = (1000, 700))       #调用基类的构造函数

    self.menuBar = wx.MenuBar()                 #创建一个菜单栏
    panel = wx.Panel(self, -1)                  #建立一个面板，上面可以放各种部件
    self.choose_icon = 0

    self.showMessage = []                       #创建一个静态文本框，来打印4个显示消息
    self.showMessage.append(wx.StaticText(panel, -1, "", pos = (400, 400)))
    self.showMessage.append(wx.StaticText(panel, -1, "", pos = (600, 400)))
    self.showMessage.append(wx.StaticText(panel, -1, "", pos = (400, 600)))
    self.showMessage.append(wx.StaticText(panel, -1, "", pos = (600, 600)))

    self.showIcon = wx.StaticText(panel, -1, "当前显示的是：图标1", pos = (100, 100))     #用此文本框来显示当前用了哪个图标
    self.showIcon.SetFont(wx.Font(10, wx.SWISS, wx.NORMAL, wx.BOLD, False))
        #这一行可以改变文本框的字体
    icon = wx.Icon(name = "icon1.ico", type = wx.BITMAP_TYPE_ICO)
        #一开始先加载入图标1
    self.SetIcon(icon)


    #接下来创建四个菜单标题(popmenu)，先用wx.Menu()创建，然后用Append方法添加菜单项，并将每个菜单项的选中与一个操作进行bind，最后将该菜单标题添加到菜单栏中
    Pop_File = wx.Menu()
    Pop_File.Append(wx.ID_EXIT, u"&Exit\tCtrl+Shift+Delete", u"退出程序")    #\t后面直接定义了加速键
    self.Bind(wx.EVT_MENU, self.OnClose, id = wx.ID_EXIT)
    self.menuBar.Append(Pop_File, u"&File")                        #将该菜单标题添加到菜单栏

    Pop_Icon = wx.Menu()
    Pop_Icon.Append(100, u"图标1(&X)\tCtrl+X", u"选择图标1", wx.ITEM_RADIO)
#注意最后一个参数，表示这三个图标只能选择其中一个
    Pop_Icon.Append(101, u"图标2(&Y)\tCtrl+Y", u"选择图标2", wx.ITEM_RADIO)
    Pop_Icon.Append(102, u"图标3(&Z)\tCtrl+Z", u"选择图标3", wx.ITEM_RADIO)
    self.Bind(wx.EVT_MENU_RANGE,self.OnIcon,id=100,id2=102)
  #注意这个绑定方法，可以将3种不同的图标选中响应到一个去
    self.menuBar.Append(Pop_Icon, u"图标(I&)")
    self.menuBar.Check(100, True)
  #一开始是默认选择图标1状态

    Pop_Show = wx.Menu()
    Pop_Show.Append(110, u"显示(&1)\tCtrl+1", u"显示信息1", wx.ITEM_CHECK)
 #注意最后一个参数，表示这三个图标可以多选
    Pop_Show.Append(111, u"显示(&2)\tCtrl+2", u"显示信息2", wx.ITEM_CHECK)
    Pop_Show.Append(112, u"显示(&3)\tCtrl+3", u"显示信息3", wx.ITEM_CHECK)
    Pop_Show.Append(113, u"显示(&4)\tCtrl+4", u"显示信息4", wx.ITEM_CHECK)
    self.Bind(wx.EVT_MENU_RANGE,self.OnShow,id=110,id2=113)
    self.menuBar.Append(Pop_Show, u"显示(&D)")

    Pop_About = wx.Menu()
    Pop_About.Append(114, u"程序信息(&I)\tF1")
    self.Bind(wx.EVT_MENU, self.OnMessage, id = 114)
    self.menuBar.Append(Pop_About, u"关于(&A)")
```

```
        self.SetMenuBar(self.menuBar)
    #添加菜单栏到面板




    def OnClose(self, event):
        self.Close()

    def OnIcon(self, event):
                #与SDK方法不同，wx点击菜单项会自动选中,点击已经选中(单选)的，则不会进入这个函数
        print(1)
                #在对话框里的控件都是点击即选中
        IconName = ["icon1.ico", "icon2.ico", "icon3.ico"]
        contents = [u"当前显示的是：图标1",u"当前显示的是：图标2",u"当前显示的是：图标3"]
        iSelection = event.GetId() - 100
              #确定要改变到哪一个图标
        returns = wx.MessageBox(u"确定要修改吗？","确认",wx.YES_NO|wx.ICON_QUESTION)
            #进行消息框确认
        if returns==wx.YES:
            self.choose_icon = iSelection
            icon = wx.Icon(name = IconName[iSelection], type = wx.BITMAP_TYPE_ICO)
                #改变图标并改变输出的文字
            self.showIcon.SetLabel(contents[iSelection])
            self.SetIcon(icon)
            if(iSelection == 2):
                #选中第三个图标后要将第三个菜单标题灰化
                self.GetMenuBar().EnableTop(2,False)
            else:
                self.GetMenuBar().EnableTop(2,True)
        self.menuBar.Check(self.choose_icon + 100, True)

    def OnShow(self, event):
        ShowName=[u"显示1",u"显示2",u"显示3",u"显示4"]
        iSelection = event.GetId() - 110
        if event.IsChecked():
            self.showMessage[iSelection].SetLabel(ShowName[iSelection])
        else:
            self.showMessage[iSelection].SetLabel("")
        #self.Refresh()

    def OnMessage(self, event):
        wx.MessageBox(u"2020MPADLab3: Windows资源使用(1)\n图标、菜单、动态菜单、加速键、消息框\n作者学号：10185102153 姓名：汪子凡\n", "2020MPADLab4", wx.OK |
wx.ICON_INFORMATION,self)



if __name__ == '__main__':
    app = wx.App()                  #创建一个应用
    frame = MyFrame()               #创建一个窗口
    frame.Show(True)
    app.MainLoop()
```

lab5，6: 模式对话框，无模式对话框，通用对话框，显示位图，改变光标，切换语言，字符串资源


```
#include <windows.h>
#include <stdio.h>
#include "resource.h"

TCHAR buffer[60];
HWND hWnd, hDlgModeless = 0;
int is_check[3], is_choose;

//*****************************************************************************
*********
BOOL InitApplication(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow);
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam);
BOOL CALLBACK ModelDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam);
BOOL CALLBACK ModelessDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam);

int WINAPI WinMain(HINSTANCE hInstance,     // 入口函数
                   HINSTANCE,
                   LPSTR     lpCmdLine,
                   int       nCmdShow  )
{
    if (!InitApplication(hInstance))       // 应用初始化
        return FALSE;

    if (!InitInstance(hInstance,nCmdShow)) // 实例初始化
        return FALSE;

    MSG msg;
    HACCEL haccel = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDR_ACCELERATOR1));           //Load进加速键
```

```
    while (GetMessage(&msg, NULL, 0, 0))   // 消息循环
    {
        if((!IsWindow(hDlgModeless) || !IsDialogMessage(hDlgModeless, &msg)) &&
(!TranslateAccelerator(hWnd, haccel, &msg)))
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int)msg.wParam;
}

//*************************************************************************************
*********

BOOL InitApplication(HINSTANCE hInstance)   // 应用初始化
{
    WNDCLASS  wc;  // Data structure of the window class

    wc.style            = CS_HREDRAW|CS_VREDRAW;
    wc.lpfnWndProc      = (WNDPROC)MainWndProc;  // Name of the Window Function
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance        = hInstance;
    //wc.hIcon            = LoadIcon (NULL, IDI_APPLICATION);
    wc.hIcon            = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));
    wc.hCursor          = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground    = (HBRUSH)GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName     = MAKEINTRESOURCE(IDR_MENU_CH);                //这个宏将资
源的ID表示转化为字符串表示，或者直接用字符串
    wc.lpszClassName    = TEXT("My1stWClass");  // Name of the window class

    return RegisterClass(&wc);
}

//*************************************************************************************
*********

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  // 实例初始化
{
    hWnd = CreateWindow(TEXT("My1stWClass"),     // Name of the window class
                        TEXT("2020MPADLab5: Windows资源使用(2)  姓名：汪子凡 学
号：10185102153"), // Title of the window
                        WS_OVERLAPPEDWINDOW,                    //窗口风格
                        CW_USEDEFAULT,                        //初始位置
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,                         //宽高
                        CW_USEDEFAULT,
                        NULL,                                   //父窗口句柄
                        NULL,                                   //菜单句柄，
需要的是句柄,可以在两个地方加载菜单，两个都有，以这个地方为准
                        hInstance,
                        NULL                                        );
    if (!hWnd) return FALSE;

    ShowWindow(hWnd, nCmdShow);                                  //产生
VM_SHOWWINDOW消息和VM_SIZE消息
    UpdateWindow(hWnd);                                          //产生VM_PAINT
消息
```

```
    return TRUE;
}

//*****************************************************************************
*********

// 窗口过程函数

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
    HDC hdc;                            //设备上下文句柄
    PAINTSTRUCT ps;
    static HINSTANCE hInst;
    static HMENU hMenuMain;
    static int id_bitmap[3] = {IDB_BITMAP1, IDB_BITMAP2, IDB_BITMAP3},
Bitmap_width, Bitmap_height;
    static int language_choose, cursor_choose, id_text;                    //记录
相应的语言选择，光标选择，以及这时字符串表的起始位置

    switch (message)
    {
       case WM_CREATE:                              //获取应用的句柄
         {
             hInst = ((LPCREATESTRUCT)lParam)-> hInstance;
             language_choose = ID_LANGUAGE_CH;
             id_text = IDS_CHINESE1;
             cursor_choose = 0;

             HBITMAP hbitmap = LoadBitmap(hInst, MAKEINTRESOURCE(id_bitmap[0]));
             BITMAP bitmap;
             GetObject(hbitmap, sizeof(BITMAP), &bitmap);          //将位图有关信
息放入bitmap
             Bitmap_width = bitmap.bmWidth;
             Bitmap_height = bitmap.bmHeight;
             DeleteObject(hbitmap);
             break;
         }

         case WM_PAINT:
         {
            hdc = BeginPaint(hWnd, &ps);
            LoadString(hInst, id_text + cursor_choose, buffer, 60);      //在字符
串表里获取字符串
            TextOut(hdc, 10, 10, buffer, lstrlen(buffer));

            HDC hmemDC = CreateCompatibleDC(hdc);            //内存设备描述表
            HBITMAP hbitmap;
            for(int i = 0; i < 3; i++)                    //依次检查三个复选和单选位
图哪些要显示
            {
                if(is_check[i])
                {
                    hbitmap = LoadBitmap(hInst, MAKEINTRESOURCE(id_bitmap[i]));
                    SelectObject(hmemDC, hbitmap);
                    BitBlt(hdc, 100, 50 + 200*i, Bitmap_width, Bitmap_height,
hmemDC, 0, 0, SRCCOPY);
```

```
                    //DELETEDC(hMemDC);
                    DeleteObject(hbitmap);
                }
            }
            hbitmap = LoadBitmap(hInst, MAKEINTRESOURCE(id_bitmap[is_choose]));
            SelectObject(hmemDC, hbitmap);
            BitBlt(hdc, 400, 50, Bitmap_width, Bitmap_height, hmemDC, 0, 0,
SRCCOPY);

            DeleteObject(hbitmap);
            DeleteDC(hmemDC);                          //释放资源
            EndPaint(hWnd, &ps);                       //用这两个函数可以告诉系统需要重
画的地方已经重画，而getDC不行，会一直重复命令
        }

        case WM_INITMENU:                          //显示勾选内容
        {
            hMenuMain = (HMENU)wParam;
            CheckMenuRadioItem(hMenuMain, ID_CUSSOR1, ID_CUSSOR3, cursor_choose
+ ID_CUSSOR1, MF_BYCOMMAND);
            CheckMenuRadioItem(hMenuMain, ID_LANGUAGE_CH, ID_LANGUAGE_EG,
language_choose, MF_BYCOMMAND);
        }

        case WM_COMMAND:                                        //若是COMMAND消息
        {
            HMENU hMenuMain = GetMenu(hWnd);
            switch(LOWORD(wParam))
            {
                printf("%d\n", LOWORD(wParam));
                case ID_CUSSOR1:
                case ID_CUSSOR2:
                case ID_CUSSOR3:
                {
                    int iSelection = LOWORD(wParam) - ID_CUSSOR1;
                    if(!(GetMenuState(hMenuMain,LOWORD(wParam), MF_BYCOMMAND) &
MF_CHECKED))
                    {
                        CheckMenuRadioItem(hMenuMain,ID_CUSSOR1,ID_CUSSOR3,
LOWORD(wParam), MF_BYCOMMAND);
                        if(iSelection == 0)
                            SetClassLong(hWnd, GCL_HCURSOR,
(long)LoadCursor(NULL,IDC_ARROW));
                        else if(iSelection == 1)
                            SetClassLong(hWnd, GCL_HCURSOR,
(long)LoadCursor(NULL,IDC_CROSS));
                        else
                            SetClassLong(hWnd, GCL_HCURSOR,
(long)LoadCursor(hInst, MAKEINTRESOURCE(IDC_CURSOR1)));
                        cursor_choose = iSelection;
                        DrawMenuBar(hWnd);
                        InvalidateRect(hWnd, NULL, TRUE);
                    }
                    break;
                }

                case ID_DIALOG_A:
 //创建模式对话框
```

```
                    DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG1),hWnd,
ModelDlgProc);
                    break;
                case ID_DIALOG_B:
                    if(!IsWindow(hDlgModeless))
 //创建无模式对话框(防止重复打开)
                    {
                       hDlgModeless = CreateDialog(hInst,
MAKEINTRESOURCE(IDD_DIALOG2), hWnd, ModelessDlgProc);
                       ShowWindow(hDlgModeless, SW_SHOW);
                    }
                   break;
                case ID_DIALOG_C:
                {
                    OPENFILENAME dlgfile;
                    TCHAR szFile[300];
                    ZeroMemory(&dlgfile, sizeof(dlgfile));
                    dlgfile.lStructSize = sizeof(dlgfile);      //确定结构的大小
                    dlgfile.hwndOwner = hWnd;                  //指定它的父窗口，如
果为NULL，表示通用对话框
                    dlgfile.lpstrFile = szFile;                 //用于保存文件的完
整路径及文件名
                    dlgfile.lpstrFile[0] = '\0';
                    dlgfile.nMaxFile = sizeof(szFile);         //指示上面结构的大小
                    dlgfile.lpstrFilter = ("All Files(*.*)\0*.*\0Python
source(*.py)\0*.py\0C++ Files(*.cpp)\0*.cpp\0\0");
                    dlgfile.nFilterIndex = 1;
                    dlgfile.lpstrFileTitle = NULL;              //用于保存文件名
                    dlgfile.nMaxFileTitle = 0;
                    dlgfile.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;
                    dlgfile.lpstrTitle = ("打开");
                    if(GetOpenFileName(&dlgfile))
                        MessageBox(hWnd, dlgfile.lpstrFile, "文件名",
MB_ICONINFORMATION|MB_OK);
                    break;
                }

                case ID_LANGUAGE_CH:
                    if(language_choose == ID_LANGUAGE_CH)
                            break;
                    language_choose = ID_LANGUAGE_CH;
                    SetMenu(hWnd, LoadMenu(hInst,
MAKEINTRESOURCE(IDR_MENU_CH)));
                    DestroyMenu(hMenuMain);
                    id_text = IDS_CHINESE1;
                    InvalidateRect(hWnd, NULL, TRUE);
                    break;
                case ID_LANGUAGE_EG:
                    if(language_choose == ID_LANGUAGE_EG)
                            break;
                    language_choose = ID_LANGUAGE_EG;
                    SetMenu(hWnd, LoadMenu(hInst,
MAKEINTRESOURCE(IDR_MENU_EG)));
                    DestroyMenu(hMenuMain);
                    id_text = IDS_ENGLISH1;
                    InvalidateRect(hWnd, NULL, TRUE);
                    break;
```

```
                case ID_MESSAGESHOW:
                    MessageBox (hWnd, TEXT ("2020MPADLab5(SDK)资源2\n光标,对话框,
位图，字符串\n\n作者学号: 10185102153 姓名: 汪子凡"),
                    "2020MPADlab5", MB_ICONINFORMATION|MB_OK);
                    break;
                case ID_FILE_EXIT:
                    PostQuitMessage(0);
                    return 0;
            }
            break;
         }


        case WM_DESTROY: // 窗口关闭
            PostQuitMessage(0);                    //设置为0表示程序正常结束
            return 0;

    }
    return DefWindowProc(hWnd, message, wParam, lParam);

}

BOOL CALLBACK ModelDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
    static int id_check[3] = {IDC_CHECK1,IDC_CHECK2,IDC_CHECK3};
    static int tmp_check[3];

    switch (message)
    {
        case WM_INITDIALOG:
        {
            for(int i = 0; i < 3; i++)
               if(is_check[i])
                    CheckDlgButton(hDlg, id_check[i], BST_CHECKED);
            for(int i = 0; i < 3; i++)
                tmp_check[i] = 0;

            SetFocus(GetDlgItem(hDlg, IDOK1));          //设置默认焦点
            return FALSE;
        }

        case WM_COMMAND:
        {
            switch (LOWORD(wParam))
            {
                case IDOK1:
                    for(int i = 0; i < 3; i++)
                        is_check[i] = (is_check[i] + tmp_check[i]) % 2;
                    InvalidateRect(GetParent(hDlg), NULL, TRUE);      //表示刷新整
个客户区(原来的要清空)
                    EndDialog(hDlg, 0);
                    return TRUE;

                case IDCANCEL1:
                    EndDialog(hDlg, 0);
                    return TRUE;
```

```
                case IDC_CHECK1:
                case IDC_CHECK2:
                case IDC_CHECK3:
                {
                    int iSelection = LOWORD(wParam) - IDC_CHECK1;
                    tmp_check[iSelection] ^= 1;
                    return TRUE;
                }
            }
        }

        case WM_DESTROY:
        {
            EndDialog(hDlg, 0);
            return TRUE;
        }
    }
    return FALSE;
}

BOOL CALLBACK ModelessDlgProc(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
    static int id_choose[3] = {IDC_RADIO1, IDC_RADIO2, IDC_RADIO3};
    static int tmp_choose;

    switch (message)
    {
        case WM_INITDIALOG:
        {
            CheckRadioButton(hDlg, IDC_RADIO1, IDC_RADIO3,
id_choose[is_choose]);
            tmp_choose = is_choose;
            return FALSE;
        }

        case WM_COMMAND:
        {
            switch (LOWORD(wParam))
            {
                case IDOK2:
                    if(is_choose != tmp_choose)
                    {
                        is_choose = tmp_choose;
                        InvalidateRect(GetParent(hDlg), NULL, TRUE);      //表示
刷新整个客户区(原来的要清空)
                    }
                    DestroyWindow(hDlg);
                    return TRUE;

                case IDCANCEL2:
                    DestroyWindow(hDlg);
                    return TRUE;

                case IDC_RADIO1:
                case IDC_RADIO2:
                case IDC_RADIO3:
                {
```

```
                    tmp_choose = LOWORD(wParam) - IDC_RADIO1;
                    return TRUE;
                }
            }
        }

        case WM_CLOSE:
        {
            DestroyWindow(hDlg);
            return TRUE;
        }
    }
    return FALSE;
}
```

```
# -*- coding: utf8 -*-

import wx
import os
from String import StringTable                                  #字符串资源

class ModalDialog(wx.Dialog):                                   #定义一个模式对话框
类
    def __init__(self, parent):
        wx.Dialog.__init__(self, parent, -1, u"复选对话框",pos=(500,300),size=
(500, 300))             #调用基类的构造函数
        panel = wx.Panel(self)
                    #创建画板
        self.okButton = wx.Button(panel, wx.ID_OK, u"确定", pos=(30, 200),size=
(130,30))              #创建button控件
        self.okButton.SetDefault()
                    #设置输入焦点
        self.cancelButton = wx.Button(panel, wx.ID_CANCEL, u"取消",pos=(250,
200),size=(130,30))
        self.check = []
        self.check.append(wx.CheckBox(panel,500,u"复选位图1",pos=(30,50),size=
(130,40)))              #创建CheckBox控件(用来复选)
        self.check.append(wx.CheckBox(panel,501,u"复选位图2",pos=(30,100),size=
(130,40)))
        self.check.append(wx.CheckBox(panel,502,u"复选位图3",pos=(30,150),size=
(130,40)))
        self.Bind(wx.EVT_CHECKBOX, self.OnCheckBoxs, id=500,id2=502)
                    #绑定，注意控件所对应的函数
        self.Bind(wx.EVT_BUTTON, self.OnOk, self.okButton)
        self.Bind(wx.EVT_BUTTON, self.OnCancel, self.cancelButton)

        self.tmp = self.GetParent().choose_bitmap1[:]
                    #对复选按钮按照原来状态初始化
        for i in range(3):
            if(self.tmp[i] == 1):
                self.check[i].SetValue(True)

    def OnCheckBoxs(self, evt):
            iSelection = evt.GetId() - 500
```

```
            self.tmp[iSelection] ^= 1

    def OnCancel(self, evt):
                #cancel的话之前改变是无效的

            self.Destroy()

    def OnOk(self, evt):
            for i in range(3):
                self.GetParent().choose_bitmap1[i] = self.tmp[i]
            self.GetParent().Refresh()
                 #刷新客户区
            self.Destroy()


class ModalessDialog(wx.Dialog):                                #定义一个无模式对话框类
    def __init__(self,parent):
        wx.Dialog.__init__(self, parent, -1, u"单选对话框",size=(400, 300),pos=
(500,300))       #调用基类的构造函数
        panel = wx.Panel(self)
              #创建画板
        self.okButton = wx.Button(panel, wx.ID_OK, u"确认", pos=(30, 200),size=
(130,40))
        self.okButton.SetDefault()
        self.cancelButton = wx.Button(panel, wx.ID_CANCEL, u"取消",pos=(250,
200),size=(130,40))
        self.Bind(wx.EVT_BUTTON, self.OnOK, self.okButton)
        self.Bind(wx.EVT_BUTTON, self.OnCancel, self.cancelButton)
        self.radio = []
        self.radio.append(wx.RadioButton(panel, 600, u"单选位图1",pos=
(30,50),size=(130,40)))          #创建RadioButton控件(用来单选)
        self.radio.append(wx.RadioButton(panel, 601, u"单选位图2",pos=
(30,100),size=(130,40)))
        self.radio.append(wx.RadioButton(panel, 602, u"单选位图3",pos=
(30,150),size=(130,40)))
        self.Bind(wx.EVT_RADIOBUTTON, self.OnRadio, id=600, id2= 602)
                        #绑定，注意控件所对应的函数

        self.tmp = self.GetParent().choose_bitmap2
                    #对单选按钮按照原来状态初始化
        self.radio[self.tmp].SetValue(True)

    def OnRadio(self, evt):
            iSelection = evt.GetId() - 600
            self.tmp = iSelection

    def OnCancel(self, evt):
         #cancel的话之前改变是无效的
            self.GetParent().choose_open = 0
            self.Destroy()

    def OnOK(self, evt):
            self.GetParent().choose_open = 0
            self.GetParent().choose_bitmap2 = self.tmp
            self.GetParent().Refresh()
            self.Destroy()
```

```
class MyFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, u"2020MPADLab6: Windows资源使用(2)  姓名：汪子凡 学号：10185102153", size = (1000, 700))      #调用基类的构造函数

        self.choose_language = 0            #0表示中文，1表示英文
        self.choose_cursor = 0              #0表示箭头，1表示十字，2表示自定义
        self.choose_bitmap1 = [0, 0, 0]     #0表示不选，1表示选
        self.choose_bitmap2 = 0
        self.choose_open = 0                #0表示无模式对话框未打开，1表示已经打开，避免重复打开

        self.Bind(wx.EVT_PAINT, self.OnPaint)                          #将paint事件与函数OnPaint绑定

        #创建一个中文菜单栏
        self.CH_menuBar = wx.MenuBar()
        #接下来创建四个菜单标题(popmenu)，先用wx.Menu()创建，然后用Append方法添加菜单项，并将每个菜单项的选中与一个操作进行bind，最后将该菜单标题添加到菜单栏中
        CH_Pop_File = wx.Menu()
        CH_Pop_File.Append(wx.ID_EXIT, u"&Exit\tCtrl+Shift+Delete", u"退出程序")  #\t后面直接定义了加速键
        self.Bind(wx.EVT_MENU, self.OnClose, id = wx.ID_EXIT)
        self.CH_menuBar.Append(CH_Pop_File, u"&File")                       #将该菜单标题添加到菜单栏

        CH_Pop_Dialog = wx.Menu()
        CH_Pop_Dialog.Append(100, u"模式对话框(&M)...\tCtrl+A", u"打开模式对话框")

        CH_Pop_Dialog.Append(101, u"无模式对话框(&L)...\tCtrl+B", u"打开无模式对话框")
        CH_Pop_Dialog.Append(102, u"文件对话框(&F)...\tCtrl+C", u"打开通用对话框")
        self.Bind(wx.EVT_MENU_RANGE,self.OnDialog,id=100,id2=102)
        #注意这个绑定方法
        self.CH_menuBar.Append(CH_Pop_Dialog, u"对话框(&I)")

        CH_Pop_Cursor = wx.Menu()
        CH_Pop_Cursor.Append(200, u"光标1(箭头)\tCtrl+1", u"使用箭头光标", wx.ITEM_RADIO)        #注意最后一个参数，表示这三个光标是单选
        CH_Pop_Cursor.Append(201, u"光标2(十字)\tCtrl+2", u"使用十字光标", wx.ITEM_RADIO)
        CH_Pop_Cursor.Append(202, u"光标3(自定义)\tCtrl+3", u"使用自定义光标", wx.ITEM_RADIO)
        self.Bind(wx.EVT_MENU_RANGE,self.CursorChange,id=200,id2=202)
        self.CH_menuBar.Append(CH_Pop_Cursor, u"光标类型(&C)")

        CH_Pop_Language = wx.Menu()
        CH_Pop_Language.Append(300, u"中(Chinses)\tCtrl+Shift+C", u"切换至中文", wx.ITEM_RADIO)        #注意最后一个参数，表示这两个语言是单选
        CH_Pop_Language.Append(301, u"英(English)\tCtrl+Shift+D", u"切换至英文", wx.ITEM_RADIO)
        self.Bind(wx.EVT_MENU_RANGE,self.ChangeLanguage,id=300,id2=301)
        self.CH_menuBar.Append(CH_Pop_Language, u"语言(&L)")

        CH_Pop_About = wx.Menu()
        CH_Pop_About.Append(400, u"程序信息(&I)\tF1")
```

```
        self.Bind(wx.EVT_MENU, self.OnMessage, id = 400)
        self.CH_menuBar.Append(CH_Pop_About, u"关于(&A)")

        self.SetMenuBar(self.CH_menuBar)


        #创建一个英文菜单栏
        self.EG_menuBar = wx.MenuBar()
        #接下来创建四个菜单标题(popmenu)，先用wx.Menu()创建，然后用Append方法添加菜单项，并将每个菜单项的选中与一个操作进行bind，最后将该菜单标题添加到菜单栏中
        EG_Pop_File = wx.Menu()
        EG_Pop_File.Append(wx.ID_EXIT, u"&Exit", u"退出程序")    #\t后面直接定义了加速键
        self.EG_menuBar.Append(EG_Pop_File, u"&File")                        #将该菜单标题添加到菜单栏

        EG_Pop_Dialog = wx.Menu()
        EG_Pop_Dialog.Append(100, u"Modal Dialog...\tCtrl+A", u"打开模式对话框")

        EG_Pop_Dialog.Append(101, u"Modaless Dialog...\tCtrl+B", u"打开无模式对话框")
        EG_Pop_Dialog.Append(102, u"File Dialog...\tCtrl+C", u"打开通用对话框")
        #self.Bind(wx.EVT_MENU_RANGE,self.OnDialog,id=100,id2=102)
       #由于中英文菜单对应的菜单项赋予了同一个ID，所以不需要重复绑定
        self.EG_menuBar.Append(EG_Pop_Dialog, u"&Dialog")

        EG_Pop_Cursor = wx.Menu()
        EG_Pop_Cursor.Append(200, u"Cursor1(ARROW)\tCtrl+1", u"使用箭头光标", wx.ITEM_RADIO)
        EG_Pop_Cursor.Append(201, u"Cursor2(CROSS)\tCtrl+2", u"使用十字光标", wx.ITEM_RADIO)
        EG_Pop_Cursor.Append(202, u"Cursor3(UserDefined)\tCtrl+3", u"使用自定义光标", wx.ITEM_RADIO)
        self.EG_menuBar.Append(EG_Pop_Cursor, u"&Cursor")

        EG_Pop_Language = wx.Menu()
        EG_Pop_Language.Append(300, u"Chinese(中)\tCtrl+Shift+C", u"切换至中文", wx.ITEM_RADIO)
        EG_Pop_Language.Append(301, u"English(英)\tCtrl+Shift+D", u"切换至英文", wx.ITEM_RADIO)
        self.EG_menuBar.Append(EG_Pop_Language, u"Language")

        EG_Pop_About = wx.Menu()
        EG_Pop_About.Append(400, u"Program Information\tF1")
        self.EG_menuBar.Append(EG_Pop_About, u"&About")

        #创建位图资源
        self.bmpList = []
        self.bmpList.append(wx.Image(name = "bitmap1.bmp", type = wx.BITMAP_TYPE_BMP).ConvertToBitmap())
        self.bmpList.append(wx.Image(name = "bitmap2.bmp", type = wx.BITMAP_TYPE_BMP).ConvertToBitmap())
        self.bmpList.append(wx.Image(name = "bitmap3.bmp", type = wx.BITMAP_TYPE_BMP).ConvertToBitmap())

        icon = wx.Icon(name = "icon1.ico", type = wx.BITMAP_TYPE_ICO)
             #创建icon资源
        self.SetIcon(icon)
```

```
    def OnPaint(self,evt):
        dc = wx.PaintDC(self)
        for i in range(3):
            if(self.choose_bitmap1[i] == 1):
                dc.DrawBitmap(self.bmpList[i] , 400, 200 * i + 50)
#省略了MemDC的简便方式加载位图资源
        dc.DrawBitmap(self.bmpList[self.choose_bitmap2] , 600, 50)
        dc.SetFont(wx.Font(15, wx.SWISS, wx.NORMAL, wx.BOLD, False))
        dc.DrawText(StringTable[self.choose_language][self.choose_cursor],
25,50)            #显示光标信息

    def OnClose(self,evt):
            self.Close()

    def CursorChange(self, evt):
        iSelection = evt.GetId() - 200
        if(iSelection == self.choose_cursor):
      #若是光标不用切换
            return
        if(iSelection == 0):
            self.choose_cursor = 0
            frame.SetCursor(wx.Cursor(wx.CURSOR_ARROW))
      #注意加载光标资源的方法
        elif(iSelection == 1):
            self.choose_cursor = 1
            frame.SetCursor(wx.Cursor(wx.CURSOR_CROSS))
        else:
            self.choose_cursor = 2
            MyCursor=wx.Cursor(cursorName="cursor1.cur",type=wx.BITMAP_TYPE_CUR)
            frame.SetCursor(MyCursor)
        self.Refresh()

    def ChangeLanguage(self, evt):
        iSelection = evt.GetId() - 300
        if(iSelection == self.choose_language):                                  #若
是语言不需要切换
            return
        if(iSelection == 0):
            self.choose_language = 0
            self.SetMenuBar(self.CH_menuBar)                                    #注意改
变菜单资源的方法
            self.CH_menuBar.Check(300, True)                                    #注意需
要将对应的语言单选和光标单选恢复初始值
            self.CH_menuBar.Check(self.choose_cursor + 200, True)
            self.SetTitle(u"2020MPADLab6: Windows资源使用(2)  姓名：汪子凡 学号：
10185102153")       #改变标题
            self.Refresh()                                                      #重新刷
新客户区，以便重新加载光标的显示文字切换
        else:
            self.choose_language = 1
            self.SetMenuBar(self.EG_menuBar)
            self.EG_menuBar.Check(301, True)
            self.EG_menuBar.Check(self.choose_cursor + 200, True)
            self.SetTitle(u"2020MPADLab6: Windows Resources(2)  Name：汪子凡  ID：
10185102153")
            self.Refresh()
```

```
    def OnDialog(self, evt):
        iSelection = evt.GetId() - 100
        if(iSelection == 0 ):
            dlg = ModalDialog(self)
            dlg.ShowModal()                                          #创建模式对话框，
使用showModal()方法
        elif(iSelection == 1 and self.choose_open == 0):
            self.choose_open = 1
            dlg1 = ModalessDialog(self)
            dlg1.Show()                                              #创建无模式对话
框，使用show()方法
        else:
            wil=u"All Files(*.*)|*.*|C++ files(*.cpp)|*.cpp|Python
source(*.py)|*.py"              #创建通用对话框
            if(self.choose_language ==0):
                Dlg=wx.FileDialog(None, u"", os.getcwd(), wil, style =
wx.FD_OPEN)
            else:
                Dlg=wx.FileDialog(None, u"", os.getcwd(), wil, style =
wx.FD_OPEN)
            if(Dlg.ShowModal() == wx.ID_OK):
                flag = 0
                i = 0
                for line in open(Dlg.GetPath(),"r"):
                    tmp = line.strip().split(' ')
                    print(tmp)
                Dlg.Destroy()
                if(self.choose_language == 0):
                    wx.MessageBox(text, StringTable[self.choose_language][3],
wx.OK, self)


            else:
                Dlg.Destroy()

    def OnMessage(self,evt):
        wx.MessageBox(u"2020MPADLab6(WX)资源2\n光标,对话框，位图，字符串\n\n作者学号:
10185102153 姓名: 汪子凡", "2020MPADLab6(WX)" ,wx.OK | wx.ICON_INFORMATION, self)



if __name__ == '__main__':
  app = wx.App()                    #创建一个应用
  frame = MyFrame()                 #创建一个窗口
  frame.Show(True)
  app.MainLoop()
```

lab7: 绘制beizer曲线，三角形，以及保存文件

```
#include <windows.h>
#include <stdio.h>
#include "resource.h"
```

```
HWND hWnd;
POINT point[4], point2[4];
int nState = 0, nState_triangle = 0;
char myfile[300];
char buffer[3][200], tmp_read[200];
//*************************************************************************
*********
BOOL InitApplication(HINSTANCE hInstance);
BOOL InitInstance(HINSTANCE hInstance, int nCmdShow);
LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam);
BOOL CALLBACK ModelDlgProc1(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam);
BOOL CALLBACK ModelDlgProc2(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam);

int WINAPI WinMain(HINSTANCE hInstance,     // 入口函数
                   HINSTANCE,
                   LPSTR     lpCmdLine,
                   int       nCmdShow  )
{
    if (!InitApplication(hInstance))       // 应用初始化
        return FALSE;

    if (!InitInstance(hInstance,nCmdShow)) // 实例初始化
        return FALSE;

    MSG msg;
    HACCEL haccel = LoadAccelerators(hInstance,
MAKEINTRESOURCE(IDR_ACCELERATOR1));           //Load进加速键

    while (GetMessage(&msg, NULL, 0, 0))   // 消息循环
    {
        if(!TranslateAccelerator(hWnd, haccel, &msg))
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }

    return (int)msg.wParam;
}

//*************************************************************************
*********

BOOL InitApplication(HINSTANCE hInstance)   // 应用初始化
{
    WNDCLASS  wc;  // Data structure of the window class

    wc.style            = CS_HREDRAW|CS_VREDRAW;
    wc.lpfnWndProc      = (WNDPROC)MainWndProc;  // Name of the Window Function
    wc.cbClsExtra       = 0;
    wc.cbWndExtra       = 0;
    wc.hInstance        = hInstance;
    //wc.hIcon            = LoadIcon (NULL, IDI_APPLICATION);
    wc.hIcon            = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_ICON1));
    wc.hCursor          = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground    = (HBRUSH)GetStockObject(WHITE_BRUSH);
```

```
    wc.lpszMenuName     = MAKEINTRESOURCE(IDR_MENU1);                //这个宏将资源
的ID表示转化为字符串表示，或者直接用字符串
    wc.lpszClassName    = TEXT("My1stWClass");  // Name of the window class

    return RegisterClass(&wc);
}

//*************************************************************************
*********

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)  // 实例初始化
{
    hWnd = CreateWindow(TEXT("My1stWClass"),     // Name of the window class
                        TEXT("2020MPADLab7: 绘制图形 姓名：汪子凡 学号：
10185102153"), // Title of the window
                        WS_OVERLAPPEDWINDOW,                    //窗口风格
                        CW_USEDEFAULT,                        //初始位置
                        CW_USEDEFAULT,
                        CW_USEDEFAULT,                         //宽高
                        CW_USEDEFAULT,
                        NULL,                                    //父窗口句柄
                        NULL,                                    //菜单句柄，
需要的是句柄,可以在两个地方加载菜单，两个都有，以这个地方为准
                        hInstance,
                        NULL                                          );
    if (!hWnd) return FALSE;

    ShowWindow(hWnd, nCmdShow);                                  //产生
VM_SHOWWINDOW消息和VM_SIZE消息
    UpdateWindow(hWnd);                                          //产生VM_PAINT
消息

    return TRUE;
}

//*************************************************************************
*********

// 窗口过程函数

LRESULT CALLBACK MainWndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM
lParam)
{
    HDC hdc;                        //设备上下文句柄
    PAINTSTRUCT ps;
    static HINSTANCE hInst;
    static HMENU hMenuMain;

    static LPCTSTR showMesg[8] = {"Beizer曲线绘制：",
                                  "1.按下鼠标左键，即确定端点P0，可以拖动，放开左键确定端点
P3",
                                  "2.再次按下左键，可以拖动，放开左键确定控制点P1(直线P0P1
与曲线相切)",
                                  "3.再次按下左键，可以拖动，放开左键确定控制点P2(直线P2P3
与曲线相切)",
                                  "三角形绘制：",
                                  "1.按下鼠标左键，即确定顶点P0，可以拖动，放开左键确定端点
P1",
```

```
                                "2.再次按下左键，可以拖动，放开左键确定控制点P2",
                                "绘制完成！可以重新绘制或者保存"};
    static int choose = 0;
    static BOOL fErase=FALSE;
    int xPos,yPos;

    switch (message)
    {
        case WM_CREATE:                             //获取应用的句柄
        {
             hInst = ((LPCREATESTRUCT)lParam)-> hInstance;

             FILE *fp = fopen("Recently.txt", "r");
//读入最近浏览的文件
             fscanf(fp, "%s %s %s", buffer[0], buffer[1], buffer[2]);
             fclose(fp);
             DrawMenuBar(hWnd);
             break;
       }

       case WM_PAINT:
       {
            hdc = BeginPaint(hWnd, &ps);
            if(choose == 0)                 //若是画Beizer曲线
            {
                TextOut(hdc, 10, 10, showMesg[0], lstrlen(showMesg[0]));
                TextOut(hdc, 10, 30, showMesg[1], lstrlen(showMesg[1]));
                TextOut(hdc, 10, 50, showMesg[2], lstrlen(showMesg[2]));
                TextOut(hdc, 10, 70, showMesg[3], lstrlen(showMesg[3]));
                if(nState == 2)                                         //若
是只完成了第一步
                {
                    MoveToEx(hdc,point[0].x,point[0].y,NULL);
                    LineTo(hdc,point[3].x,point[3].y);
                }
                else if(nState > 2)                                      //
已完成第一步之后
                {
                    TextOut(hdc, 10, 100, showMesg[7], lstrlen(showMesg[7]));
                    PolyBezier(hdc,point,4);
                }
            }
            else                                //若是画三角形
            {
                TextOut(hdc, 10, 10, showMesg[4], lstrlen(showMesg[4]));
                TextOut(hdc, 10, 30, showMesg[5], lstrlen(showMesg[5]));
                TextOut(hdc, 10, 50, showMesg[6], lstrlen(showMesg[6]));
                if(nState_triangle == 2)                                //若是
只完成了第一步
                {
                    MoveToEx(hdc, point2[0].x, point2[0].y, NULL);
                    LineTo(hdc, point2[1].x, point2[1].y);
                }
                else if(nState_triangle > 2)
                {
                    TextOut(hdc, 10, 100, showMesg[7], lstrlen(showMesg[7]));
                    MoveToEx(hdc, point2[0].x, point2[0].y, NULL);
                    LineTo(hdc, point2[1].x, point2[1].y);
```

```
                    LineTo(hdc, point2[2].x, point2[2].y);
                    LineTo(hdc, point2[0].x, point2[0].y);
                }
            }
            EndPaint(hWnd, &ps);                        //用这两个函数可以告诉系统需要重
画的地方已经重画，而getDC不行，会一直重复命令

            break;
        }

        case WM_INITMENU:                           //显示勾选内容
        {
            hMenuMain = (HMENU)wParam;                              //显示是三角形还是
曲线
            CheckMenuRadioItem(hMenuMain, ID_CHANGE_BEIZER, ID_CHANGE_TRIANGLE,
choose + 40007, MF_BYCOMMAND);

            ModifyMenu(hMenuMain, ID_RECENTFILES_1, MF_BYCOMMAND,
ID_RECENTFILES_1, buffer[0]);
            ModifyMenu(hMenuMain, ID_RECENTFILES_2, MF_BYCOMMAND,
ID_RECENTFILES_2, buffer[1]);
            ModifyMenu(hMenuMain, ID_RECENTFILES_3, MF_BYCOMMAND,
ID_RECENTFILES_3, buffer[2]);
            break;
        }

        case WM_LBUTTONDOWN:
            xPos = (signed short)LOWORD(lParam);   yPos = (signed
short)HIWORD(lParam);            //强制类型转换保证往左和上不会出bug
            SetCapture(hWnd);
            hdc = GetDC(hWnd);
            SetROP2(hdc,R2_NOTXORPEN);
            if(choose == 0)
                switch (nState)
                {
                    case 0:
                        point[0].x = xPos;  point[0].y = yPos;
//获取端点P0
                        nState = 1;
                        break;
                    case 2:
                        MoveToEx(hdc,point[0].x,point[0].y,NULL);            //
消除原来的直线
                        LineTo(hdc,point[3].x,point[3].y);
                        point[1].x=xPos;   point[1].y=yPos;
                        point[2].x=xPos;   point[2].y=yPos;
                        PolyBezier(hdc,point,4);
                        nState=3;
                        fErase=TRUE;
                        break;
                    case 4:
                        PolyBezier(hdc,point,4);
                        point[2].x=xPos;   point[2].y=yPos;
                        PolyBezier(hdc,point,4);
                        nState=5;
                        fErase=TRUE;
                        break;
                }
```

```
            else
                switch(nState_triangle)
                {
                    case 0:
                        point2[0].x = xPos;  point2[0].y = yPos;
                        nState_triangle =1;
                        break;
                    case 2:
                        point2[2].x = xPos;   point2[2].y = yPos;
                        point2[2].x = xPos;   point2[2].y = yPos;

                        MoveToEx(hdc, point2[0].x, point2[0].y, NULL);
                        LineTo(hdc, point2[2].x, point2[2].y);
                        MoveToEx(hdc, point2[1].x, point2[1].y, NULL);
                        LineTo(hdc, point2[2].x, point2[2].y);

                        nState_triangle = 3;
                        fErase = TRUE;
                        break;

                }
            ReleaseDC(hWnd,hdc);
            break;

        case WM_LBUTTONUP:
            xPos = (signed short)LOWORD(lParam);   yPos = (signed
short)HIWORD(lParam);
            fErase = FALSE;
            if(choose == 0)
            {
                switch (nState)
                {
                    case 1:
                        point[3].x = xPos;   point[3].y = yPos;
                        nState=2;
                        break;
                    case 3:
                        point[1].x = xPos;   point[1].y = yPos;
                        nState=4;
                        break;
                    case 5:
                        point[2].x = xPos;   point[2].y = yPos;
                        nState = 6;                                          //状态结
束，置nstate为6
                        InvalidateRect(hWnd, NULL, TRUE);
                        break;
                }
            }
            else
            {
                switch(nState_triangle)
                {
                    case 1:
                        point2[1].x = xPos;   point2[1].y = yPos;
                        nState_triangle = 2;
                        break;
                    case 3:
                        point2[2].x = xPos;   point2[2].y = yPos;
```

```
                        nState_triangle = 4;                                    //
状态结束，置 nState_triangle为4
                        InvalidateRect(hWnd, NULL, TRUE);
                        break;
                }
            }
           ReleaseCapture();
           break;

        case WM_MOUSEMOVE:
            xPos = (signed short)LOWORD(lParam);   yPos = (signed
short)HIWORD(lParam);
            if(wParam&MK_LBUTTON)
            {
                hdc=GetDC(hWnd);
                SetROP2(hdc,R2_NOTXORPEN);
                if(choose == 0)                                 //bizer曲线
                {
                    switch (nState)
                    {
                        case 1:
                            if(fErase)                  //第一次划线的时候fErase还
是0，没有线可以擦
                            {
                                MoveToEx(hdc,point[0].x,point[0].y,NULL);
                                LineTo(hdc,point[3].x,point[3].y);
                            }
                            point[3].x=xPos;    point[3].y=yPos;
       //划线
                            MoveToEx(hdc,point[0].x,point[0].y,NULL);
                            LineTo(hdc,point[3].x,point[3].y);
                            break;
                        case 3:
                            if(fErase)  PolyBezier(hdc,point,4);
                            point[1].x=xPos;     point[1].y=yPos;
                            point[2].x=xPos;     point[2].y=yPos;
                            PolyBezier(hdc,point,4);
                            break;
                        case 5:
                            if(fErase)  PolyBezier(hdc,point,4);
                            point[2].x=xPos;     point[2].y=yPos;
                            PolyBezier(hdc,point,4);
                            break;
                    }
              }
              else
              {
                  switch(nState_triangle)
                  {
                        case 1:
                            if(fErase)                  //第一次划线的时候fErase还
是0，没有线可以擦
                            {
                                MoveToEx(hdc,point2[0].x,point2[0].y,NULL);
                                LineTo(hdc,point2[1].x,point2[1].y);
                            }
                            point2[1].x = xPos;    point2[1].y = yPos;
             //划线
```

```
                            MoveToEx(hdc, point2[0].x, point2[0].y, NULL);
                            LineTo(hdc,point2[1].x, point2[1].y);
                            break;
                        case 3:
                            if(fErase)
                            {
                                MoveToEx(hdc, point2[0].x, point2[0].y, NULL);
                                LineTo(hdc, point2[2].x, point2[2].y);
                                MoveToEx(hdc, point2[1].x, point2[1].y, NULL);
                                LineTo(hdc, point2[2].x, point2[2].y);
                            }
                            point2[2].x = xPos;    point2[2].y = yPos;

                            MoveToEx(hdc, point2[0].x, point2[0].y, NULL);
                            LineTo(hdc, point2[2].x, point2[2].y);
                            MoveToEx(hdc, point2[1].x, point2[1].y, NULL);
                            LineTo(hdc, point2[2].x, point2[2].y);
                            break;

                    }
                }
                fErase=TRUE;
                ReleaseDC(hWnd,hdc);
            }
            break;


          case WM_COMMAND:                                          //若是COMMAND消息
          {
             HMENU hMenuMain = GetMenu(hWnd);
             switch(LOWORD(wParam))
             {
                 case ID_CHANGE_BEIZER:
                 case ID_CHANGE_TRIANGLE:                          //更改曲线类型
                 {
                     int iSelection = LOWORD(wParam) - 40007;
                     if(iSelection == choose)
                         break;
                     if(MessageBox (hWnd, TEXT("确认要修改画图类型吗？"), TEXT("确
认"),  MB_YESNO|MB_ICONQUESTION ) == IDYES )   //消息框进行确认
                     {
                         CheckMenuRadioItem(hMenuMain,ID_CHANGE_BEIZER,
ID_CHANGE_TRIANGLE, LOWORD(wParam), MF_BYCOMMAND);
                         choose = iSelection;
                         fErase = false;  nState = nState_triangle = 0;
                         InvalidateRect(hWnd, NULL, true);
                     }
                     break;
                 }

                 case ID_SETTINGS_REPAINT:                          //重新绘制
                     if(MessageBox (hWnd, TEXT("确认要重新绘制吗？"), TEXT("确认"),
MB_YESNO|MB_ICONQUESTION ) == IDYES )
                     {
                         fErase = false;  nState = nState_triangle = 0;
                         InvalidateRect(hWnd, NULL, true);
                     }
                     break;
```

```
                case ID_SETTINGS_MODIFY:
                    if(choose == 0)
                    {
                        DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG1),hWnd,
ModelDlgProc1);
                    }
                    else
                    {
                        DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG2),hWnd,
ModelDlgProc2);
                    }
                    break;

                case ID_HELP_LAB7HELP:
                    MessageBox (hWnd, TEXT ("本次作业可以绘制两种曲线(用鼠标画或者输入
坐标),实现了基本的文件保存新建打开功能(以txt格式)"),
                    "2020MPADlab7", MB_ICONINFORMATION|MB_OK);
                    break;

                case ID_HELP_ABOUT:
                    MessageBox (hWnd, TEXT ("2020MPADLab7 绘图\n\n作者学号:
10185102153 姓名: 汪子凡"),
                    "2020MPADlab7", MB_ICONINFORMATION|MB_OK);
                    break;

                case ID_FILE_NEW:
                case ID_FILE_OPEN:
                case ID_FILE_SAVE:
                case ID_FILE_SAVEAS:
                {
                    OPENFILENAME dlgfile;
                    TCHAR szFile[300];
                    ZeroMemory(&dlgfile, sizeof(dlgfile));
                    dlgfile.lStructSize = sizeof(dlgfile);      //确定结构的大小
                    dlgfile.hwndOwner = hWnd;                  //指定它的父窗口，如
果为NULL，表示通用对话框
                    dlgfile.lpstrFile = szFile;                 //用于保存文件的完
整路径及文件名
                    dlgfile.lpstrFile[0] = '\0';
                    dlgfile.nMaxFile = sizeof(szFile);         //指示上面结构的大小
                    dlgfile.lpstrFilter = ("txt source(*.txt)\0*.txt\0");
                    dlgfile.nFilterIndex = 1;
                    dlgfile.lpstrFileTitle = NULL;              //用于保存文件名
                    dlgfile.nMaxFileTitle = 0;
                    dlgfile.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

                    int tmp_array[4][2], tmp_choose;
                    if(LOWORD(wParam) == ID_FILE_OPEN)
                    {
                        dlgfile.lpstrTitle = ("打开");
                        if(GetOpenFileName(&dlgfile))
                        {
                            FILE *fp = fopen(dlgfile.lpstrFile, "r");
    //打开文件后先读取文件信息
                            if(fscanf(fp, "%d %d %d %d %d %d %d %d %d",
&tmp_choose, &tmp_array[0][0], &tmp_array[0][1], &tmp_array[1][0],
```

```
                            &tmp_array[1][1], &tmp_array[2][0], &tmp_array[2]
[1], &tmp_array[3][0], &tmp_array[3][1]) == 9)
                            {
                                sprintf(myfile, "%s", dlgfile.lpstrFile);
         //更改当前的myfile
                                choose = tmp_choose;
                                if(choose == 0)
                                {
                                    for(int i = 0; i < 4; i++)
                                        point[i].x = tmp_array[i][1], point[i].y
= tmp_array[i][0];
                                    nState = 6;
                                }
                                else
                                {
                                    for(int i = 0; i < 3; i++)
                                        point2[i].x = tmp_array[i][1],
point2[i].y = tmp_array[i][0];
                                    nState_triangle = 4;
                                }
                            }
                            else                                                //
若是文件内信息读取格式不正确
                                MessageBox (hWnd, TEXT ("打开失败"), "提示",
MB_ICONINFORMATION|MB_OK);
                            fclose(fp);
                        }
                        else                                                    //
若是文件没有打开
                             MessageBox (hWnd, TEXT ("打开失败"), "提示",
MB_ICONINFORMATION|MB_OK);
                    }

                    else if(LOWORD(wParam) == ID_FILE_NEW)
                    {
                        dlgfile.lpstrTitle = ("新建");
                        if(GetSaveFileName(&dlgfile))
                        {
                            sprintf(myfile, "%s", dlgfile.lpstrFile);
                            char b[]=".txt", *pa = myfile;            //创建文件时
需要加上.txt后缀
                            while(*pa) pa++;
                            sprintf(pa, "%s", b);
                            printf("%s", myfile);
                            FILE *fp = fopen(myfile, "w");
                            fclose(fp);
                            choose = 0;  nState = 0;  fErase = FALSE;
//默认新建的文件为bizer曲线
                        }
                    }

                    else if(LOWORD(wParam) == ID_FILE_SAVE && myfile[0] != '\0')
    //若是保存文件
                    {
                        if(nState < 6 && nState_triangle < 4)
                        {
                            MessageBox (hWnd, TEXT ("保存失败! 图形未绘制完成"), "提
示", MB_ICONINFORMATION|MB_OK);
```

```
                            break;
                        }
                        FILE *fp = fopen(myfile, "w");
                        if(choose == 0)
                            fprintf(fp, "0\n%ld %ld\n%ld %ld\n%ld %ld\n%ld %ld",
point[0].y, point[0].x, point[1].y, point[1].x,
                            point[2].y, point[2].x, point[3].y, point[3].x);
                        else
                            fprintf(fp, "1\n%ld %ld\n%ld %ld\n%ld %ld\n%ld
%ld",point2[0].y, point2[0].x, point2[1].y, point2[1].x,
                            point2[2].y, point2[2].x, point2[3].y, point2[3].x);
                        fclose(fp);
                    }

                    else
                    {
                        if(GetSaveFileName(&dlgfile))
                        {
                            char b[]=".txt", *pa = myfile;                 //要加
后缀
                            sprintf(myfile, "%s", dlgfile.lpstrFile);
                            while(*pa)
                                pa++;
                            sprintf(pa, "%s", b);
                            FILE *fp = fopen(myfile, "w");
                            if(choose == 0)
                                fprintf(fp, "0\n%ld %ld\n%ld %ld\n%ld %ld\n%ld
%ld",point[0].y, point[0].x, point[1].y, point[1].x,
                                point[2].y, point[2].x, point[3].y, point[3].x);
                            else
                                fprintf(fp, "1\n%ld %ld\n%ld %ld\n%ld %ld\n%ld
%ld",point2[0].y, point2[0].x, point2[1].y, point2[1].x,
                                point2[2].y, point2[2].x, point2[3].y,
point2[3].x);
                            fclose(fp);
                        }
                    }

                    int cur = 0;
                    char tmp_buffer[2][200];
                    for(int i = 0; cur < 2 && i < 3; i++)
                    {
                         if(strcmp(myfile, buffer[i]) != 0)
                            strcpy(tmp_buffer[cur++], buffer[i]);
                    }
                    strcpy(buffer[0], myfile);
                    strcpy(buffer[1], tmp_buffer[0]);
                    strcpy(buffer[2], tmp_buffer[1]);
                    for(int i = 0; i < 3; i++) printf("%s\n", buffer[i]);
                    InvalidateRect(hWnd, NULL, TRUE);
                    DrawMenuBar(hWnd);
                    break;
                }

                case ID_RECENTFILES_1:
                case ID_RECENTFILES_2:
                case ID_RECENTFILES_3:
                {
```

```
                    FILE *fp = fopen(buffer[LOWORD(wParam) - ID_RECENTFILES_1],
"r");
                    int tmp_array[4][2], tmp_choose;
                    if(fp)
                    {
                        if(fscanf(fp, "%d %d %d %d %d %d %d %d %d", &tmp_choose,
&tmp_array[0][0], &tmp_array[0][1], &tmp_array[1][0],
                        &tmp_array[1][1], &tmp_array[2][0], &tmp_array[2][1],
&tmp_array[3][0], &tmp_array[3][1]) == 9)
                        {
                            sprintf(myfile, "%s", buffer[LOWORD(wParam) -
ID_RECENTFILES_1]);
                            choose = tmp_choose;
                            if(choose == 0)
                            {
                                for(int i = 0; i < 4; i++)
                                    point[i].x = tmp_array[i][1], point[i].y =
tmp_array[i][0];
                                nState = 6;
                            }
                            else
                            {
                                for(int i = 0; i < 3; i++)
                                    point2[i].x = tmp_array[i][1], point2[i].y =
tmp_array[i][0];
                                nState_triangle = 4;
                            }
                        }
                        else
                            MessageBox (hWnd, TEXT ("打开失败"), "提示",
MB_ICONINFORMATION|MB_OK);
                        fclose(fp);
                    }
                    else
                        MessageBox (hWnd, TEXT ("打开失败"), "提示",
MB_ICONINFORMATION|MB_OK);
                    InvalidateRect(hWnd, NULL, TRUE);
                    break;
                }
            }
            break;
         }

        case WM_DESTROY: // 窗口关闭
        {
            for(int i = 0; i < 3; i++) printf("在退出中： %s\n", buffer[i]);
            FILE *FP = fopen("Recently.txt", "w");
            fprintf(FP, "%s\n%s\n%s", buffer[0], buffer[1], buffer[2]);
            fclose(FP);
            PostQuitMessage(0);                    //设置为0表示程序正常结束
            return 0;
        }

    }
    return DefWindowProc(hWnd, message, wParam, lParam);

}
```

```
BOOL CALLBACK ModelDlgProc1(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
    static int tmp_point[4][2];
    switch (message)
    {
        case WM_INITDIALOG:
        {
            for(int i = 0; i < 3; i++)
                tmp_point[i][0] = point[i].x, tmp_point[i][1] = point[i].y;
            //GetDlgItem(hDlg, IDC_EDIT1)->SetWindowText(0);
            break;
        }

        case WM_COMMAND:
        {
            switch (LOWORD(wParam))
            {
                case OK_B:
                {
                    BOOL k = TRUE, *test = &k;
                    tmp_point[0][0] = GetDlgItemInt(hDlg, IDC_EDIT1, test,
TRUE);
                    tmp_point[1][0] = GetDlgItemInt(hDlg, IDC_EDIT2, test,
TRUE);
                    tmp_point[2][0] = GetDlgItemInt(hDlg, IDC_EDIT3, test,
TRUE);
                    tmp_point[3][0] = GetDlgItemInt(hDlg, IDC_EDIT4, test,
TRUE);
                    tmp_point[0][1] = GetDlgItemInt(hDlg, IDC_EDIT5, test,
TRUE);
                    tmp_point[1][1] = GetDlgItemInt(hDlg, IDC_EDIT6, test,
TRUE);
                    tmp_point[2][1] = GetDlgItemInt(hDlg, IDC_EDIT7, test,
TRUE);
                    tmp_point[3][1] = GetDlgItemInt(hDlg, IDC_EDIT8, test,
TRUE);

                    printf("%d\n", k);
                    if(k)
                    {
                        nState = 6;
                        for(int i = 0; i < 4; i++)
                            point[i].x = tmp_point[i][1], point[i].y =
tmp_point[i][0];
                        InvalidateRect(GetParent(hDlg), NULL, TRUE);      //表示
刷新整个客户区(原来的要清空)
                    }
                    else
                    {
                        MessageBox (hDlg, TEXT ("修改失败!(请输入合法的整数)"), "提
示", MB_ICONINFORMATION|MB_OK);
                        return TRUE;
                    }
                    EndDialog(hDlg, 0);
                    return TRUE;
                }
```

```
                case CANCEL_B:
                    EndDialog(hDlg, 0);
                    return TRUE;
            }

        }
        break;

        case WM_DESTROY:
        {
            EndDialog(hDlg, 0);
            return TRUE;
        }
    }
    return FALSE;
}

BOOL CALLBACK ModelDlgProc2(HWND hDlg, UINT message, WPARAM wParam, LPARAM
lParam)
{
    static int tmp_point[3][2];
    switch (message)
    {
        case WM_INITDIALOG:
        {
            for(int i = 0; i < 3; i++)
                tmp_point[i][0] = point2[i].x, tmp_point[i][1] = point2[i].y;
            //GetDlgItem(hDlg, IDC_EDIT1)->SetWindowText(0);
            break;
        }

        case WM_COMMAND:
        {
            switch (LOWORD(wParam))
            {
                case IDOK:
                {
                    BOOL k = TRUE, *test = &k;
                    tmp_point[0][0] = GetDlgItemInt(hDlg, IDC_T_EDIT1, test,
TRUE);
                    tmp_point[1][0] = GetDlgItemInt(hDlg, IDC_T_EDIT2, test,
TRUE);
                    tmp_point[2][0] = GetDlgItemInt(hDlg, IDC_T_EDIT3, test,
TRUE);
                    tmp_point[0][1] = GetDlgItemInt(hDlg, IDC_T_EDIT4, test,
TRUE);
                    tmp_point[1][1] = GetDlgItemInt(hDlg, IDC_T_EDIT5, test,
TRUE);
                    tmp_point[2][1] = GetDlgItemInt(hDlg, IDC_T_EDIT6, test,
TRUE);

                    if(k)
                    {
                        nState_triangle = 4;
                        for(int i = 0; i < 3; i++)
                            point2[i].x = tmp_point[i][1], point2[i].y =
tmp_point[i][0];
```

```
                        InvalidateRect(GetParent(hDlg), NULL, TRUE);      //表示刷新整个客户区(原来的要清空)
                    }
                    else
                    {
                        MessageBox (hDlg, TEXT ("修改失败!(请输入合法的整数)"), "提示", MB_ICONINFORMATION|MB_OK);
                        return TRUE;
                    }
                    EndDialog(hDlg, 0);
                    return TRUE;
                }

                case IDCANCEL:
                    EndDialog(hDlg, 0);
                    return TRUE;
            }

        }
        break;

        case WM_DESTROY:
        {
            EndDialog(hDlg, 0);
            return TRUE;
        }
    }
    return FALSE;
}
```

```
# -*- coding: utf8 -*-

import wx
import os




class MyFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, u"2020MPADLab8: 绘制图形(2)  姓名：汪子凡 学号：10185102153", size = (1000, 700))      #调用基类的构造函数


        self.Bind(wx.EVT_PAINT, self.OnPaint)                          #将paint事件与函数OnPaint绑定

        #创建一个菜单栏
        self.menuBar = wx.MenuBar()
        #接下来创建菜单标题(popmenu)，先用wx.Menu()创建，然后用Append方法添加菜单项，并将每个菜单项的选中与一个操作进行bind，最后将该菜单标题添加到菜单栏中
        Pop_File = wx.Menu()
```

```
        Pop_File.Append(100, u"&New\tCtrl+N", u"c创建新文件")   #\t后面直接定义了加速
键
        self.Bind(wx.EVT_MENU, self.OnNew, id = 100)
        Pop_File.Append(101, u"&Open\tCtrl+O", u"打开新文件")
        self.Bind(wx.EVT_MENU, self.OnOpen, id = 101)
        Pop_File.Append(102, u"&Save\tCtrl+S", u"保存文件")
        self.Bind(wx.EVT_MENU, self.OnSave, id = 102)
        Pop_File.Append(103, u"&Save as...\tCtrl+Shift+S", u"另存为")
        self.Bind(wx.EVT_MENU, self.OnSave_as, id = 103)
        Pop_Recently = wx.Menu()
        Pop_Recently.Append(104, " ", u" ")
        Pop_Recently.Append(105, " ", u" ")
        Pop_Recently.Append(106, " ", u" ")
        self.Bind(wx.EVT_MENU, self.OnRecently, id = 104, id2 = 106)
        Pop_File. AppendSubMenu(Pop_Recently, u"&Recently File")    #recently
files要创建二级菜单
        self.menuBar.Append(Pop_File, u"&File")                     #将该菜单标题
添加到菜单栏

        Pop_Settings = wx.Menu()
        Pop_Change = wx.Menu()
        Pop_Change.Append(107, u"Beizer", u"Beizer", wx.ITEM_RADIO)
        Pop_Change.Append(108, u"Triangle", u"Triangle", wx.ITEM_RADIO)
        self.Bind(wx.EVT_MENU, self.OnChoose, id = 107, id2 = 108)
        Pop_Settings. AppendSubMenu(Pop_Change, u"&Change")
        Pop_Settings.Append(109, u"Repaint", u"重新绘制")
        self.Bind(wx.EVT_MENU, self.OnRepaint, id = 109)
        self.menuBar.Append(Pop_Settings, u"Settings")

        Pop_Help = wx.Menu()
        Pop_Help.Append(110, u"Lab7 help", u"Lab7 help")
        self.Bind(wx.EVT_MENU, self.OnHelp, id = 110)
        Pop_Help.Append(111, u"About", u"About")
        self.Bind(wx.EVT_MENU, self.OnAbout, id = 111)
        self.menuBar.Append(Pop_Help, u"Help")

        self.SetMenuBar(self.menuBar)

        icon = wx.Icon(name = "icon1.ico", type = wx.BITMAP_TYPE_ICO)
             #创建icon资源
        self.SetIcon(icon)

        self.choose = 0
        self.state = 0
        self.fErase = False
        self.filename = None
        self.point = [[0, 0] for i in range(4)]
        self.Bind(wx.EVT_LEFT_DOWN, self.OnMouseLeftDown)
        self.Bind(wx.EVT_LEFT_UP, self.OnMouseLeftUp)
        self.Bind(wx.EVT_MOTION, self.OnMouseMove)

        self.showMesg= ["Beizer曲线绘制：",
                                "1.按下鼠标左键，即确定端点P0，可以拖动，放开左键确定端点
P3",
                                "2.再次按下左键，可以拖动，放开左键确定控制点P1(直线P0P1
与曲线相切)",
                                "3.再次按下左键，可以拖动，放开左键确定控制点P2(直线P2P3
与曲线相切)",
```

```
                                    "三角形绘制：",
                                    "1.按下鼠标左键，即确定顶点P0，可以拖动，放开左键确定端点
P1",
                                    "2.再次按下左键，可以拖动，放开左键确定控制点P2",
                                    "绘制完成！可以重新绘制或者保存",
                                    "当前还没有打开任何文件,可以打开一个txt文件或者新建一
个"]
        self.buffer = []
        for line in open("data.txt","r"):
                self.buffer.append(line.strip())
        for i in  range(3):
            self.menuBar.FindItemById(104 + i).SetText(self.buffer[i])


    def OnPaint(self,evt):
        dc = wx.PaintDC(self)
        dc.SetPen(wx.Pen("black", 5))
        if(self.choose == 0):
            dc.DrawText(self.showMesg[0], 10, 10)
            dc.DrawText(self.showMesg[1], 10, 30)
            dc.DrawText(self.showMesg[2], 10, 50)
            dc.DrawText(self.showMesg[3], 10, 70)
            if(self.state == 2):
                dc.DrawLine(self.point[0][0], self.point[0][1], self.point[3]
[0], self.point[3][1])
            elif(self.state > 2):
                dc.DrawText(self.showMesg[7], 10, 100)
                dc.DrawSpline(self.point)
        else:
            dc.DrawText(self.showMesg[4], 10, 10)
            dc.DrawText(self.showMesg[5], 10, 30)
            dc.DrawText(self.showMesg[6], 10, 50)
            if(self.state == 2):
                dc.DrawLine(self.point[0][0], self.point[0][1], self.point[1]
[0], self.point[1][1])
            elif(self.state > 2):
                dc.DrawText(self.showMesg[7], 10, 100)
                dc.DrawLine(self.point[0][0], self.point[0][1], self.point[2]
[0], self.point[2][1])
                dc.DrawLine(self.point[2][0], self.point[2][1], self.point[1]
[0], self.point[1][1])
                dc.DrawLine(self.point[0][0], self.point[0][1], self.point[1]
[0], self.point[1][1])
        if(self.filename == None):                                      #显示现
在的绘图地址
            dc.DrawText(self.showMesg[8], 10, 600)
        else:
            dc.DrawText(self.filename, 10, 600)
        self.menuBar.Check(self.choose + 107, True)                      #显示菜
单的绘图格式

    def OnMouseLeftDown(self, evt):
        p = evt.GetPosition()
        dc = wx.ClientDC(self)
        dc.SetPen(wx.Pen("black", 5))
        dc.SetLogicalFunction(wx.INVERT)                #设置绘图模式
        if(self.choose == 0):
            if(self.state == 0):
```

```
                self.point[0][0] = p.x
                self.point[0][1] = p.y
                self.state = 1
            elif(self.state == 2):
                dc.DrawLine(self.point[0][0], self.point[0][1], self.point[3]
[0], self.point[3][1])
                self.point[1][0] = p.x
                self.point[1][1] = p.y
                self.point[2][0] = p.x
                self.point[2][1] = p.y
                dc.DrawSpline(self.point)
                self.state = 3
                self.fErase = True
            elif(self.state == 4):
                dc.DrawSpline(self.point)
                self.point[2][0] = p.x
                self.point[2][1] = p.y
                dc.DrawSpline(self.point)
                self.state = 5
                self.fErase = True
        else:
            if(self.state == 0):
                self.point[0][0] = p.x
                self.point[0][1] = p.y
                self.state = 1
            elif(self.state == 2):
                self.point[2][0] = p.x
                self.point[2][1] = p.y
                dc.DrawLine(self.point[0][0], self.point[0][1], self.point[2]
[0], self.point[2][1])
                dc.DrawLine(self.point[1][0], self.point[1][1], self.point[2]
[0], self.point[2][1])
                self.state = 3
                self.fErase = True


    def OnMouseLeftUp(self, evt):
        p = evt.GetPosition()
        self.fErase = False
        dc = wx.ClientDC(self)
        dc.SetPen(wx.Pen("black", 5))
        dc.SetLogicalFunction(wx.INVERT)                #设置绘图模式
        if(self.choose == 0):
            if(self.state == 1):
                self.point[3][0] = p.x
                self.point[3][1] = p.y
                self.state = 2
            elif(self.state == 3):
                self.point[1][0] = p.x
                self.point[1][1] = p.y
                self.state = 4
            elif(self.state == 5):
                self.point[2][0] = p.x
                self.point[2][1] = p.y
                self.state = 6
                self.Refresh()
        else:
            if(self.state == 1):
```

```
                self.point[1][0] = p.x
                self.point[1][1] = p.y
                self.state = 2
            elif(self.state == 3):
                self.point[2][0] = p.x
                self.point[2][1] = p.y
                self.state = 4
                self.Refresh()

    def OnMouseMove(self, evt):
        p = evt.GetPosition()
        dc = wx.ClientDC(self)
        dc.SetPen(wx.Pen("black", 5))
        dc.SetLogicalFunction(wx.INVERT)
        if evt.LeftIsDown():
            if(self.choose == 0):
                if(self.state == 1):
                    if(self.fErase):
                        dc.DrawLine(self.point[0][0], self.point[0][1],
self.point[3][0], self.point[3][1])
                    self.point[3][0] = p.x
                    self.point[3][1] = p.y
                    dc.DrawLine(self.point[0][0], self.point[0][1],
self.point[3][0], self.point[3][1])
                elif(self.state == 3):
                    if(self.fErase):
                        dc.DrawSpline(self.point)
                    self.point[1][0] = p.x
                    self.point[1][1] = p.y
                    self.point[2][0] = p.x
                    self.point[2][1] = p.y
                    dc.DrawSpline(self.point)
                elif(self.state == 5):
                    if(self.fErase):
                        dc.DrawSpline(self.point)
                    self.point[2][0] = p.x
                    self.point[2][1] = p.y
                    dc.DrawSpline(self.point)
            else:
                if(self.state == 1):
                    if(self.fErase):
                        dc.DrawLine(self.point[0][0], self.point[0][1],
self.point[1][0], self.point[1][1])
                    self.point[1][0] = p.x
                    self.point[1][1] = p.y
                    dc.DrawLine(self.point[0][0], self.point[0][1],
self.point[1][0], self.point[1][1])
                elif(self.state == 3):
                    if(self.fErase):
                        dc.DrawLine(self.point[0][0], self.point[0][1],
self.point[2][0], self.point[2][1])
                        dc.DrawLine(self.point[1][0], self.point[1][1],
self.point[2][0], self.point[2][1])
                    self.point[2][0] = p.x
                    self.point[2][1] = p.y
                    dc.DrawLine(self.point[0][0], self.point[0][1],
self.point[2][0], self.point[2][1])
```

```
                    dc.DrawLine(self.point[1][0], self.point[1][1],
self.point[2][0], self.point[2][1])

            self.fErase = True

    def OnNew(self,evt):
        wil=u"TXT source(*.txt)|*.txt"              #创建通用对话框
        Dlg=wx.FileDialog(None, u"新建", os.getcwd(), wil, style = wx.FD_OPEN)
        if(Dlg.ShowModal() == wx.ID_OK):
            self.filename = Dlg.GetPath()
            if(self.filename[-4:-1] != ".tx"):
                self.filename += ".txt"
            print(self.filename)
            self.Refresh()
            self.state = 0
            self.Erase = False
            self.choose = 0
            open(self.filename, "w")


    def ModifyMenu(self):
        tmp = self.buffer[0]
        self.buffer[0] = self.filename
        self.menuBar.FindItemById(104).SetText(self.buffer[0])
        if(tmp == self.buffer[0]):
            return
        for i in range(1, 3):
            if(self.buffer[i] == self.buffer[0]):
                self.buffer[i] = tmp
                self.menuBar.FindItemById(104 + i).SetText(self.buffer[i])
                break
            tmp1 = tmp
            tmp = self.buffer[i]
            self.buffer[i] = tmp1
            self.menuBar.FindItemById(104 + i).SetText(self.buffer[i])
        with open("data.txt",'w') as f:
            for i in range(3):
                f.write(str(self.buffer[i]) + "\n")

    def Open(self, tmp):
        self.choose = int(tmp[0][0])
        for i in range(4):
            self.point[i][0] = int(tmp[i+1][0])
            self.point[i][1] = int(tmp[i+1][1])
        self.state = 10
        fErase = True
        self.ModifyMenu()
        self.Refresh()

    def OnOpen(self,evt):
        wil=u"Txt source(*.txt)|*.txt"              #创建通用对话框
        Dlg=wx.FileDialog(None, u"打开", os.getcwd(), wil, style = wx.FD_OPEN)

        if(Dlg.ShowModal() == wx.ID_OK):
            flag = 0                                                #判断打开文件是否数
据格式正确
            tmp = []
            for line in open(Dlg.GetPath(),"r"):
```

```
                tmp.append(line.strip().split(' '))
            if(len(tmp) == 5 and len(tmp[0]) == 1 and tmp[0][0].isdigit()):
                for i in range(1, 5):
                    if(len(tmp[i]) == 2 and tmp[i][0].isdigit() and tmp[i]
[1].isdigit()):
                        if(i == 4):
                            flag = 1
            if(flag == 0):
                wx.MessageBox(u"读取文件失败", "提示" ,wx.OK | wx.ICON_INFORMATION,
self)

            else:                                                   #读取数据
                self.filename = Dlg.GetPath()
                self.Open(tmp)
                self.Refresh()
            Dlg.Destroy()
        else:
            Dlg.Destroy()


    def OnSave(self,evt):
        if((self.choose == 0 and self.state < 6) or (self.choose == 1 and
self.state < 4)):
           wx.MessageBox(u"当前没有绘制完成", "提示" ,wx.OK | wx.ICON_INFORMATION,
self)
        elif(self.filename != None):
            self.save()
        else:
            self.OnSave_as(evt)

    def OnSave_as(self,evt):
        if((self.choose == 0 and self.state < 6) or (self.choose == 1 and
self.state < 4)):
           wx.MessageBox(u"当前没有绘制完成", "提示" ,wx.OK | wx.ICON_INFORMATION,
self)
           return
        wil=u"Txt source(*.txt)|*.txt"              #创建通用对话框
        Dlg=wx.FileDialog(None, u"另存为", os.getcwd(), wil, style = wx.FD_SAVE)
        if(Dlg.ShowModal() == wx.ID_OK):
            self.filename = Dlg.GetPath()
            if(self.filename[-4:-1] != ".tx"):
                self.filename += ".txt"
            print(self.filename)
            self.save()
            Dlg.Destroy()
        else:
            Dlg.Destroy()

    def save(self):
        with open(self.filename,'w') as f:
            f.write(str(self.choose) + "\n")
            for i in range(4):
                f.write(str(self.point[i][0]) + " " + str(self.point[i][1]) +
"\n")
        self.ModifyMenu()
        self.Refresh()
```

```
    def OnRecently(self,evt):
        tmp_file = self.buffer[evt.GetId() - 104]
        flag = 0                                                    #判断打开文件是否数据格式
正确
        tmp = []
        for line in open(tmp_file,"r"):
            tmp.append(line.strip().split(' '))
        if(len(tmp) == 5 and len(tmp[0]) == 1 and tmp[0][0].isdigit()):
            for i in range(1, 5):
                if(len(tmp[i]) == 2 and tmp[i][0].isdigit() and tmp[i]
[1].isdigit()):
                    if(i == 4):
                        flag = 1
        if(flag == 0):
            wx.MessageBox(u"读取文件失败", "提示" ,wx.OK | wx.ICON_INFORMATION,
self)

        else:                                                       #读取数据
            self.filename = tmp_file
            self.Open(tmp)
            self.Refresh()

    def OnChoose(self,evt):
         i = evt.GetId()  - 107
         if(i != self.choose):
             self.choose = i
             self.state = 0
             self.fErase = False
             self.Refresh()

    def OnRepaint(self,evt):
        self.state = 0
        self.fErase = False
        self.Refresh()

    def OnHelp(self,evt):
        wx.MessageBox(u"本次作业实现了基本的文件保存新建打开功能(以txt格式)",
"2020MPADlab8" ,wx.OK | wx.ICON_INFORMATION, self)

    def OnAbout(self,evt):
        wx.MessageBox(u"2020MPADLab8 绘图\n\n作者学号: 10185102153 姓名: 汪子凡",
"2020MPADlab8" ,wx.OK | wx.ICON_INFORMATION, self)

if __name__ == '__main__':
  app = wx.App()                #创建一个应用
  frame = MyFrame()             #创建一个窗口
  frame.Show(True)
  app.MainLoop()
```

lab9: **右键显示出关于字体的通用对话框，显示古诗**

```
#include<windows.h>
#include<stdlib.h>
#include<string.h>
```

```
long WINAPI WndProc(HWND hWnd,UINT iMessage,UINT wParam,LONG lParam);
BOOL InitWindowsClass(HINSTANCE hInstance);
BOOL InitWindows(HINSTANCE hInstance,int nCmdShow);

BOOL InitWindowsClass(HINSTANCE hInstance)//初始化窗口类
{
        WNDCLASS WndClass;
        WndClass.cbClsExtra=0;
        WndClass.cbWndExtra=0;
        WndClass.hbrBackground=(HBRUSH)(GetStockObject(WHITE_BRUSH));
        WndClass.hCursor=LoadCursor(NULL,IDC_ARROW);
        WndClass.hIcon=LoadIcon(NULL,"END");
        WndClass.hInstance=hInstance;
        WndClass.lpfnWndProc=WndProc;
        WndClass.lpszClassName="WinText";
        WndClass.lpszMenuName=NULL;
        WndClass.style=CS_HREDRAW|CS_VREDRAW;
        return RegisterClass(&WndClass);
}

BOOL InitWindows(HINSTANCE hInstance,int nCmdShow) //初始化窗口
{
    HWND hWnd;
        hWnd=CreateWindow("WinText",  //生成窗口
                        "文本显示示例程序",
                        WS_OVERLAPPEDWINDOW,
                        CW_USEDEFAULT,
                        0,
                        CW_USEDEFAULT,
                        0,
                        NULL,
                        NULL,
                        hInstance,
                        NULL);
        if(!hWnd)
            return FALSE;
        ShowWindow(hWnd,nCmdShow);//显示窗口
        UpdateWindow(hWnd);
        return TRUE;
}
//主函数
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,LPSTR lpCmdLine,
int nCmdShow)
{
    MSG Message;
    if(!InitWindowsClass(hInstance))        return FALSE;
    if(!InitWindows(hInstance,nCmdShow))    return FALSE;
    while(GetMessage(&Message,0,0,0))       //消息循环
        {TranslateMessage(&Message);
         DispatchMessage(&Message);
        }
    return Message.wParam;
}

long WINAPI WndProc(HWND hWnd,UINT iMessage,UINT wParam,LONG lParam)
{ static long nXChar,nCaps,nYChar;
  int pointx,pointy,i,j;
  HDC hDC;                        //定义指向设备环境的句柄
```

```
  TEXTMETRIC tm;                  //存放字体各种属性的结构体变量
  PAINTSTRUCT PtStr;              //指向包含绘图信息的结构体变量
  char *textbuf[4]={{"故人西辞黄鹤楼"}, {"烟花三月下扬州"},{"孤帆远影碧空尽"}, {"唯见长
江天际流"}};
  switch(iMessage)                        //处理消息
  { case WM_CREATE:                       //处理窗口创建消息
     hDC=GetDC(hWnd) ;                    //获取当前设备表句柄
     GetTextMetrics(hDC,&tm);             //获取字体信息
     nXChar=tm.tmAveCharWidth;                //获取字符宽度
     nYChar=tm.tmHeight+tm.tmExternalLeading;   //字符高度
     nCaps=(tm.tmPitchAndFamily&1?3:2)*nXChar/2;    //字间距
     ReleaseDC(hWnd,hDC); return 0;      //释放当前设备句柄
    case WM_PAINT:                        //处理重画消息
    hDC=BeginPaint(hWnd,&PtStr);          //开始绘图
     for(i=4;i>0;i--)
     {for(j=0;j<7;j++)                    //输出文本
        {  pointx=100+i*nXChar*5;   pointy=50+j*(nYChar+nCaps);
           TextOut(hDC,pointx,pointy,textbuf[4-i]+j*2,2);
        }
     }
     EndPaint(hWnd,&PtStr); return 0;        //结束绘图
    case WM_DESTROY: //结束应用程序
     PostQuitMessage(0);    return 0;
    default:
        return(DefWindowProc(hWnd,iMessage,wParam,lParam));
  }
}
```

quiz1: 弹出式菜单

```
#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

HINSTANCE hInst ;
TCHAR     szAppName[] = TEXT ("PopMenu") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND     hwnd ;
     MSG      msg ;
     WNDCLASS wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, szAppName) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = MAKEINTRESOURCE(POPMENU) ;
```

```
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("Failed to RegisterClass! "),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hInst = hInstance ;

     hwnd = CreateWindow (szAppName, TEXT ("Popup Menu Demonstration"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HMENU hMenu ;
     static int   idColor [5] = { WHITE_BRUSH,  LTGRAY_BRUSH, GRAY_BRUSH,
                                  DKGRAY_BRUSH, BLACK_BRUSH } ;
     static int   iSelection = IDM_BKGND_WHITE ;
     POINT        point ;

     switch (message)
     {
     case WM_CREATE:
          hMenu = LoadMenu (hInst, MAKEINTRESOURCE(POPMENU) ) ;
          hMenu = GetSubMenu (hMenu, 0) ;
          return 0 ;

     case WM_RBUTTONUP:
          point.x = LOWORD (lParam) ;
          point.y = HIWORD (lParam) ;
          ClientToScreen (hwnd, &point) ;

          TrackPopupMenu (hMenu, TPM_RIGHTBUTTON, point.x, point.y,
                          0, hwnd, NULL) ;
          return 0 ;

     case WM_COMMAND:
          switch (LOWORD (wParam))
          {
          case IDM_FILE_NEW:
          case IDM_FILE_OPEN:
          case IDM_FILE_SAVE:
```

```
          case IDM_FILE_SAVE_AS:
          case IDM_EDIT_UNDO:
          case IDM_EDIT_CUT:
          case IDM_EDIT_COPY:
          case IDM_EDIT_PASTE:
          case IDM_EDIT_CLEAR:
               MessageBeep (0) ;
               return 0 ;

          case IDM_BKGND_WHITE:         // Note: Logic below
          case IDM_BKGND_LTGRAY:        //   assumes that IDM_WHITE
          case IDM_BKGND_GRAY:          //   through IDM_BLACK are
          case IDM_BKGND_DKGRAY:        //   consecutive numbers in
          case IDM_BKGND_BLACK:         //   the order shown here.

               CheckMenuItem (hMenu, iSelection, MF_UNCHECKED) ;
               iSelection = LOWORD (wParam) ;
               CheckMenuItem (hMenu, iSelection, MF_CHECKED) ;

               SetClassLong (hwnd, GCL_HBRBACKGROUND, (LONG)
                    GetStockObject
                         (idColor [LOWORD (wParam) - IDM_BKGND_WHITE])) ;

               InvalidateRect (hwnd, NULL, TRUE) ;
               return 0 ;

          case IDM_APP_ABOUT:
               MessageBox (hwnd, TEXT ("Popup Menu Demonstration Program\n")
                                 TEXT (" "),
                           szAppName, MB_ICONINFORMATION | MB_OK) ;
               return 0 ;

          case IDM_APP_EXIT:
               SendMessage (hwnd, WM_CLOSE, 0, 0) ;
               return 0 ;

          case IDM_APP_HELP:
               MessageBox (hwnd, TEXT ("Help not yet implemented!"),
                           szAppName, MB_ICONEXCLAMATION | MB_OK) ;
               return 0 ;
          }
          break ;

     case WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

```
#_*_coding:utf-8_*_

import wx
class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, u"My Frame", size=(800, 500))
```

```
    # Create the menubar
    self.myMenu = wx.Menu()                         #将原来的MenuBar改成Menu

    # add a menu
    menu = wx.Menu()

    # add an item to the menu, using \tKeyName automatically
    # creates an accelerator, the third param is some help text
    # that will show up in the statusbar
#    self.IdCommand = wx.NewId()
#    menu.Append(self.IdCommand, u"命令(&R)\tCtrl+R", "This the text in the
Statusbar")
    # bind the menu event to an event handler
#    self.Bind(wx.EVT_MENU, self.OnCommand, id=self.IdCommand)

    self.IdCommand=menu.Append(-1, u"命令(&R)\tCtrl+R", "This the text in the
Statusbar")
    self.Bind(wx.EVT_MENU, self.OnCommand, self.IdCommand)

    menu.AppendSeparator()

    menu.Append(wx.ID_EXIT, u"E&xit\tAlt-X", u"Exit this simple sample")
    # bind the menu event to an event handler
    self.Bind(wx.EVT_MENU, self.OnClose, id=wx.ID_EXIT)

    # and put the menu on the menubar
    self.myMenu.AppendSubMenu(menu, u"&File")             #将函数换成AppendSubMenu

    self.color = wx.Menu()
    # Radio items
    self.color.Append(201, u"White", u"", wx.ITEM_RADIO)
    self.color.Append(202, u"Gray", u"", wx.ITEM_RADIO)
    self.color.Append(203, u"Black", u"", wx.ITEM_RADIO)
#    self.Bind(wx.EVT_MENU, self.OnColor,id=201)
#    self.Bind(wx.EVT_MENU, self.OnColor,id=202)
#    self.Bind(wx.EVT_MENU, self.OnColor,id=203)
    self.Bind(wx.EVT_MENU_RANGE, self.OnColor,id=201,id2=203)

    self.myMenu.AppendSubMenu(self.color, u"&Color")     #将函数换成AppendSubMenu
    self.myMenu.Check(202,True)                             #这里需要改变
    self.SetBackgroundColour(u"Gray")

    control = wx.Menu()
    # Check items
    control.Append(301, u"Enable", u"Enable/Disable BG change", wx.ITEM_CHECK)
    self.myMenu.AppendSubMenu(control, u"Con&trol")
#将函数换成AppendSubMenu
    self.Bind(wx.EVT_MENU, self.OnControl,id=301)
    self.myMenu.Check(301,True)                          #这里需要改变

    self.changeable = True

    # and another menu
    menu = wx.Menu()

    IdAbout = menu.Append(-1, u"&About\tF1", u"Help tip")

    # bind the menu event to an event handler
```

```
    self.Bind(wx.EVT_MENU, self.OnHelp, IdAbout)

    # and put the menu on the menubar
    self.myMenu.AppendSubMenu(menu, u"&Help")                #将函数换成
AppendSubMenu
    #self.SetMenuBar(self.menuBar)

    self.CreateStatusBar()
    self.Bind(wx.EVT_PAINT, self.OnPaint)

    self.Bind(wx.EVT_CONTEXT_MENU, self.OnContextMenu)          #这里需要添加事件

  def OnPaint(self, evt):
    dc=wx.PaintDC(self)
    # draw something in client area
    evt.Skip()

  def OnCommand(self, evt):
    wx.MessageBox(u"Sorry,运行命令 not implemented yet! ",
           "Message",
           wx.OK | wx.ICON_EXCLAMATION, self)

  def OnColor(self, evt):
    item = self.myMenu.FindItemById(evt.GetId())          #这里需要改变
#    text = item.GetText()
    text = item.GetItemLabel()
    wx.MessageBox(u"You selected item '%s'" % text,
           u"Color Menu", wx.OK | wx.ICON_INFORMATION, self)
    if self.changeable:
      self.SetBackgroundColour(text)
      self.Refresh()
    else:
      dc=wx.ClientDC(self)
      dc.SetTextForeground(u'red')
      dc.DrawText(u"不能改变BG!",100,50)

  def OnControl(self, evt):
    self.changeable = evt.IsChecked()
#    self.GetMenuBar().Enable(self.IdCommand, self.changeable)
    self.myMenu.Enable(self.IdCommand.GetId(), self.changeable)               #这
里需要改变

  def OnHelp(self, evt):                                                  #这里需要改变

    wx.MessageBox(u"2020MPADQuiz1(WX)PopMenu\n\n作者学号： 10185102153 姓名： 汪子
凡", "2020MPADQuiz1(WX)", wx.OK | wx.ICON_INFORMATION, self)

  def OnClose(self, evt):
    self.Close()

  def OnContextMenu(self, evt):                          #这里需要添加
      pos = evt.GetPosition()
      pos = frame.ScreenToClient(pos)
      frame.PopupMenu(self.myMenu, pos)

if __name__ == u'__main__':
  app = wx.App()
  frame = MyFrame()
```

```
  frame.Show(True)
  app.MainLoop()
```

quiz2:建立与客户区大小始终相同的一个编辑控件(60%)

2. **输入SHIFT+F1显示作者信息(10%)**
3. **在编辑控件中输入功能键CTRL F1-F12时实际输入**
   并显示的是1-12月份英文单词(20%)
4. **单击右键显示字体选择通用对话框 (10%)**（窗口子类化）

```
#include <windows.h>
#include <stdio.h>
#define ID_EDIT    1

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);
WNDPROC OldList ;
static HWND hwndEdit, HWnd;
LRESULT CALLBACK ListProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("EditDemo") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND     hwnd ;
     MSG      msg ;
     WNDCLASS wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("Failed to RegisterClass! "),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, szAppName,
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
    HWnd =  hwnd;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
```

```
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{

     switch (message)
     {
     case WM_CREATE :
          hwndEdit = CreateWindow (TEXT ("edit"), NULL,
   //创建一个Edit控件
                         WS_CHILD | WS_VISIBLE | WS_HSCROLL | WS_VSCROLL |
                                   WS_BORDER | ES_LEFT | ES_MULTILINE |
                                   ES_AUTOHSCROLL | ES_AUTOVSCROLL,
                         0, 0, 0, 0, hwnd, (HMENU) ID_EDIT,
                         ((LPCREATESTRUCT) lParam) -> hInstance, NULL) ;
        OldList = (WNDPROC) SetWindowLong (hwndEdit, GWL_WNDPROC,
                                                (LPARAM) ListProc) ;
          return 0 ;

     case WM_SETFOCUS :
          SetFocus (hwndEdit) ;
          return 0 ;

     case WM_SIZE :
          MoveWindow (hwndEdit, 0, 0, LOWORD (lParam), HIWORD (lParam), TRUE) ;
          return 0 ;

     case WM_COMMAND :
          if (LOWORD (wParam) == ID_EDIT)
               if (HIWORD (wParam) == EN_ERRSPACE ||
                         HIWORD (wParam) == EN_MAXTEXT)

                    MessageBox (hwnd, TEXT ("Edit control out of space."),
                                szAppName, MB_OK | MB_ICONSTOP) ;

          return 0 ;

     case WM_DESTROY :
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}

LRESULT CALLBACK ListProc (HWND hwnd, UINT message,
                           WPARAM wParam, LPARAM lParam)
{
    static int choose_Shift = 0, choose_Ctrl = 0;
    CHOOSEFONT cf;
    static LOGFONT lf;        // logical font structure
    static DWORD rgbCurrent;  // current text color
    static HFONT hfont;
```

```
    int iSelection;
    static LPTSTR Month[12] = {TEXT("January"), TEXT("February"), TEXT("March"),
                                    TEXT("April"), TEXT("May"), TEXT("June"),
                                    TEXT("July"), TEXT("August"),
TEXT("September"),
                                    TEXT("October"), TEXT("November"),
TEXT("December") };
    static LPTSTR m;
    switch (message)
    {
        case WM_KEYDOWN:
        {
            switch(LOWORD(wParam))
            {
                case VK_F1:
                case VK_F2:
                case VK_F3:
                case VK_F4:
                case VK_F5:
                case VK_F6:
                case VK_F7:
                case VK_F8:
                case VK_F9:
                case VK_F10:
                case VK_F11:
                case VK_F12:
                {
                    iSelection = LOWORD(wParam) - VK_F1;
                    if(iSelection == 0 && choose_Shift)
                    {
                        MessageBox(hwnd, TEXT ("2020MPADQuiz2(SDK)\nEdit\n\n作者
学号:10185102153 姓名:汪子凡"),
                    TEXT("2020MPADlQuiz2"), MB_ICONINFORMATION|MB_OK);
                        break;
                    }
                    if(choose_Ctrl)
                    {
                        printf("%s\n", Month[iSelection]);
                        //GetDlgItemText(hwnd, 0, m, 100);
                        //printf("%s", m);
                        SetDlgItemText(HWnd, ID_EDIT, Month[iSelection]);
                    }
                    break;
                }
                case 17:
                    choose_Ctrl = 1;
                    break;
                case 16:
                    choose_Shift = 1;
                    break;
            }
            break;
        }
        case WM_KEYUP:
        {
            switch(LOWORD(wParam))
                case 17:
                    choose_Ctrl = 0;
```

```
                    break;
                case 16:
                    choose_Shift = 0;
                    break;
            break;
        }
        case WM_RBUTTONUP:
        {
        ZeroMemory(&cf, sizeof(cf));
        cf.lStructSize = sizeof (cf);
        cf.hwndOwner = hwnd;
        cf.lpLogFont = &lf;
        cf.rgbColors = rgbCurrent;
        cf.Flags = CF_SCREENFONTS | CF_EFFECTS;

        if (ChooseFont(&cf)==TRUE)
        {
            //hfont = CreateFontIndirect(cf.lpLogFont);
            //rgbCurrent = cf.rgbColors;
            //InvalidateRect(hWnd, NULL, TRUE);
            return TRUE;
        }
        break;
        }
    }
    return CallWindowProc (OldList, hwnd, message, wParam, lParam) ;
}
```

quiz3:写一个包含author, lower, f函数的.c/.cpp 文件用于建立DLL(60%)
建立一个Windows DLL，供MPADQuiz3DLLTest.py调用(30%)
建立一个Linux SO，供MPADQuiz3DLLTest.py调用(10%)

```
//MAPDQuiz3DLL.c
#include <windows.h>
#ifdef __cplusplus
#define EXPORT extern "C" __declspec (dllexport)
#else
#define EXPORT __declspec (dllexport)
#endif

EXPORT WINAPI int sum(int* a, int n);
EXPORT WINAPI const char* author(int i);
EXPORT WINAPI int f(int x, int *a);
EXPORT WINAPI char* lower(char* s);


BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)
{
     return TRUE ;
}

EXPORT const char* WINAPI author(int i)
{
    const char *s1 = "10185102153", *s2 = "汪子凡";
```

```
    if(i == 1) return s1;
    return s2;
}

EXPORT int WINAPI f(int x, int *a)
{
    return a[0]*x*x + a[1]*x + a[2];
}

EXPORT char* WINAPI lower(char* s)
{
    for(int i = 0; s[i]; i++)
        if(s[i] >= 'A' && s[i] <= 'Z')
            s[i] = s[i] - 'A' + 'a';
    return s;
}
```

```
//MAPDQuiz3DLL_linux.c
const char* author(int i)
{
    const char *s1 = "10185102153", *s2 = "汪子凡";
    if(i == 1) return s1;
    return s2;
}

int f(int x, int *a)
{
    return a[0]*x*x + a[1]*x + a[2];
}

 char*  lower(char* s)
{
    for(int i = 0; s[i]; i++)
        if(s[i] >= 'A' && s[i] <= 'Z')
            s[i] = s[i] - 'A' + 'a';
    return s;
}
```

写一个包含author, iSort函数的.c/.cpp文件(40%)
建立Windows LabTest.DLL(10%)和LabTest.so(10%)
建立一个WX程序(Windows/Linux):
3.1 调用author显示作者信息(10%)，输入一个List，调用iSort, 显示排序前后的点连接而成的线(10%)
3.2 输入符号公式等，对公式解方程，因式分解，化简，展开，计算极限，计算定积分(至少2个功能，20%)
只能在Windows/Linux中的一个环境中运行 -5%

```
#include <windows.h>
#ifdef __cplusplus
#define EXPORT extern "C" __declspec (dllexport)
#else
#define EXPORT __declspec (dllexport)
#endif


EXPORT WINAPI const char* author(int i);
EXPORT WINAPI void iSort(int *a, int n, int ascending);


BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)
{
     return TRUE ;
}

EXPORT const char* WINAPI author(int i)
{
    const char *s1 = "10185102153", *s2 = "汪子凡";
    if(i == 1) return s1;
    return s2;
}



EXPORT void iSort(int *a, int n, int ascending)
{
    int i, j, cur, tmp;
    if(ascending == 1)
    {
        for(i = 0; i < n - 1; i++)
        {
            cur = i;
            for(j = i + 1; j < n; j++)
                if(a[cur] > a[j])
                    cur = j;
            tmp = a[cur];
            a[cur] = a[i];
            a[i] = tmp;
        }
    }
    else
    {
        for(i = 0; i < n - 1; i++)
        {
            cur = i;
            for(j = i + 1; j < n; j++)
                if(a[cur] < a[j])
                    cur = j;
            tmp = a[cur];
            a[cur] = a[i];
            a[i] = tmp;
        }
    }
}
```

wxDIALOG

```
#_*_coding:utf-8_*_

import wx
class SubclassDialog(wx.Dialog):
  def __init__(self):
    wx.Dialog.__init__(self, None, -1, 'Dialog Subclass',size=(300, 100))
    okButton = wx.Button(self, wx.ID_OK, "OK", pos=(25, 15))
    okButton.SetDefault()
    cancelButton = wx.Button(self, wx.ID_CANCEL, "Cancel",pos=(125, 15))

class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, "Dialog DEMO", size=(800, 600))
    icon = wx.Icon(name="About1.ico", type=wx.BITMAP_TYPE_ICO)
    self.SetIcon(icon)
    self.menuBar = wx.MenuBar()
    menu = wx.Menu()
    self.OpenDialog = menu.Append(-1, u"打开对话框(&O)\tCtrl+O")
    self.Bind(wx.EVT_MENU, self.OnOpenDialog, self.OpenDialog)
    menu.AppendSeparator()
    menu.Append(wx.ID_EXIT, "E&xit\tAlt-X", "Exit this sample program")
    self.Bind(wx.EVT_MENU, self.OnClose, id=wx.ID_EXIT)
    self.menuBar.Append(menu, "&File")
    self.SetMenuBar(self.menuBar)

  def OnClose(self, evt):
    self.Close()

  def OnOpenDialog(self, evt):
    dialog = SubclassDialog()
    result = dialog.ShowModal()
    if result == wx.ID_OK:
        wx.MessageBox("Dialog selected button '%s'" % 'OK',
           "Dialog Result", wx.OK | wx.ICON_INFORMATION, self)
    else:
        wx.MessageBox("Dialog selected button '%s'" % 'CANCEL',
           "Dialog Result", wx.OK | wx.ICON_INFORMATION, self)
    dialog.Destroy()

if __name__ == '__main__':
  app = wx.App()
  frame = MyFrame()
  frame.Show(True)
  app.MainLoop()
```

wxGetValue

```
#_*_coding:utf-8_*_
```

```
import wx
class SubclassDialog(wx.Dialog):
  def __init__(self):
    wx.Dialog.__init__(self, None, -1, 'Dialog Subclass',size=(300, 200))
    self.value = ''
    self.textCtrl = wx.TextCtrl(self, -1,self.value,pos=(25, 15))
    okButton = wx.Button(self, wx.ID_OK, "OK", pos=(25, 115))
    okButton.SetDefault()
    cancelButton = wx.Button(self, wx.ID_CANCEL, "Cancel",pos=(125, 115))
    self.Bind(wx.EVT_BUTTON, self.OnOK, okButton)

  def OnOK(self, evt):
    self.value = self.textCtrl.GetValue()
    evt.Skip()  # this is neccesary to complete default action of OK button

  def getValue(self):
   return  self.value

class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, "Dialog DEMO", size=(800, 600))
    icon = wx.Icon(name="About1.ico", type=wx.BITMAP_TYPE_ICO)
    self.SetIcon(icon)
    self.menuBar = wx.MenuBar()
    menu = wx.Menu()
    self.OpenDialog = menu.Append(-1, u"打开对话框(&O)\tCtrl+O")
    self.Bind(wx.EVT_MENU, self.OnOpenDialog, self.OpenDialog)
    menu.AppendSeparator()
    menu.Append(wx.ID_EXIT, "E&xit\tAlt-X", "Exit this simple sample")
    self.Bind(wx.EVT_MENU, self.OnClose, id=wx.ID_EXIT)
    self.menuBar.Append(menu, "&File")
    self.SetMenuBar(self.menuBar)

  def OnClose(self, evt):
    self.Close()

  def OnOpenDialog(self, evt):
    dialog = SubclassDialog()
    result = dialog.ShowModal()
    wx.MessageBox("Dialog value is '%s'" % dialog.getValue(),
           "Dialog Result", wx.OK | wx.ICON_INFORMATION, self)
    dialog.Destroy()

if __name__ == '__main__':
  app = wx.App()
  frame = MyFrame()
  frame.Show(True)
  app.MainLoop()
```

wxGenericDialog.py

```
#_*_coding:utf-8_*_
```

```
import wx,os
class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, u"通用对话框DEMO", size=(800, 600))
    icon = wx.Icon(name="About1.ico", type=wx.BITMAP_TYPE_ICO)
    self.SetIcon(icon)
    self.menuBar = wx.MenuBar()
    menu = wx.Menu()
    self.OpenFileDialog = menu.Append(-1, u"打开文件对话框(&O)\tShift+O")
    self.Bind(wx.EVT_MENU, self.OnOpenFileDialog, self.OpenFileDialog)
    self.OpenColorDialog = menu.Append(-1, u"打开颜色对话框(&C)\tShift+C")
    self.Bind(wx.EVT_MENU, self.OnOpenColorDialog, self.OpenColorDialog)
    self.OpenFontDialog = menu.Append(-1, u"打开字体对话框(&C)\tShift+F")
    self.Bind(wx.EVT_MENU, self.OnOpenFontDialog, self.OpenFontDialog)

    menu.AppendSeparator()
    menu.Append(wx.ID_EXIT, "E&xit\tAlt-X", "Exit this simple sample")
    self.Bind(wx.EVT_MENU, self.OnClose, id=wx.ID_EXIT)
    self.menuBar.Append(menu, "&File")
    self.SetMenuBar(self.menuBar)
    self.text = None
    self.color=None
    self.font=None
    self.Bind(wx.EVT_PAINT, self.OnPaint)

  def OnPaint(self, evt):
    dc=wx.PaintDC(self)
    if self.color: dc.SetTextForeground(self.color)
    if self.font: dc.SetFont(self.font)
    if self.text: dc.DrawText(self.text,10,10)

  def OnClose(self, evt):
    self.Close()

  def OnOpenFileDialog(self, evt):
    wildcard = "Python source (*.py)|*.py|" \
               "Compiled Python (*.pyc)|*.pyc|" \
               "All files (*.*)|*.*"
    dialog = wx.FileDialog(None, "Choose a file", os.getcwd(),
                           "", wildcard, wx.FD_OPEN)
    if dialog.ShowModal() == wx.ID_OK:
       self.text = u"所选文件名："+dialog.GetPath()
    dialog.Destroy()
    self.Refresh()


  def OnOpenColorDialog(self, evt):
    dialog = wx.ColourDialog(None)
    dialog.GetColourData().SetChooseFull(True)
    if dialog.ShowModal() == wx.ID_OK:
      self.color = dialog.GetColourData().GetColour().Get()
    dialog.Destroy()
    self.Refresh()


  def OnOpenFontDialog(self, evt):
    dialog = wx.FontDialog(None, wx.FontData())
    if dialog.ShowModal() == wx.ID_OK:
```

```
      data = dialog.GetFontData()
      self.font = data.GetChosenFont()
      self.color = data.GetColour()
    dialog.Destroy()
    self.Refresh()

if __name__ == '__main__':
  app = wx.App()
  frame = MyFrame()
  frame.Show(True)
  app.MainLoop()
```

wxBitmap

```
import wx
class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, "Bitmap Demo(Native)", size=(800, 600))
    self.bmp = wx.Image(name="bitmap1.bmp",
type=wx.BITMAP_TYPE_BMP).ConvertToBitmap()
    self.bmpSizeX, self.bmpSizeY = self.bmp.GetWidth(), self.bmp.GetHeight()
    self.Bind(wx.EVT_PAINT, self.OnPaint)

  def OnPaint(self,evt):
    clientSizeX,clientSizeY = self.GetClientSize()
    dc = wx.PaintDC(self)
    dcMem = wx.MemoryDC()
    dcMem.SelectObject(self.bmp)
    for y in range(30,clientSizeY,self.bmpSizeY*2):
      for x in range(30,clientSizeX,self.bmpSizeX*2):
        dc.Blit(x, y, self.bmpSizeX, self.bmpSizeY, dcMem, 0, 0, wx.COPY)

if __name__ == '__main__':
  app = wx.App()
  frame = MyFrame()
  frame.Show(True)
  app.MainLoop()
//可以用更简单的        dc.DrawBitmap(Bitmap,x,y) ，隐藏内存设备描述表的细节
```

keydemo

```
import wx,sys
if sys.platform.startswith("win32"):
  import win32con
class MyFrame(wx.Frame):
  def __init__(self):
    wx.Frame.__init__(self, None, -1, "KeyDemo", size=(800, 600))
    self.Bind(wx.EVT_KEY_DOWN, self.OnKeyDown)
    self.Bind(wx.EVT_KEY_UP, self.OnKeyUp)
    self.Bind(wx.EVT_CHAR, self.OnChar)
    self.Bind(wx.EVT_PAINT, self.OnPaint)
    self.Bind(wx.EVT_ACTIVATE, self.OnActivate)
    self.count = 0
    self.x = 50
```

```
    self.SetFocus()
    print(self.FindFocus())

  def OnActivate(self, event):
    print("Activate",)
    state = event.GetActive()
    print(state)
    event.Skip()


  def OnPaint(self, event):
    dc=wx.PaintDC(self)
    key='wx.WXK_LEFT'
    keycode=wx.WXK_LEFT
    dc.DrawText(" WXKEYVALUE  %25s  :   %04d" % (key,keycode),10,10)
    if sys.platform.startswith("win32"):
      winkey='win32con.VK_LEFT'
      winkeycode=win32con.VK_LEFT
      dc.DrawText("WINKEYVALUE %25s  :   %04d" % (winkey,winkeycode),10,30)

  def OnKeyDown(self, event):
    if self.count>=10:
      self.count=0
      self.Refresh()
    dc=wx.ClientDC(self)
    keycode = event.GetKeyCode()
    self.count+=1
    self.y = 100+(self.count%11)*15
    dc.DrawText("%d: KEYDOWN:%d" % (self.count,keycode),self.x,self.y+60)
    event.Skip()

  def OnKeyUp(self, event):
    dc=wx.ClientDC(self)
    keycode = event.GetKeyCode()
    self.count+=1
    self.y = 100+(self.count%11)*15
    dc.DrawText("%d: KEYUP:%d" % (self.count,keycode),self.x,self.y+260)
    event.Skip()

  def OnChar(self, event):
    dc=wx.ClientDC(self)
    keycode = event.GetKeyCode()
    self.y = 100+(self.count%11)*15
    dc.DrawText("%d: CHAR :%d" % (self.count,keycode),self.x+200,self.y+60)

if __name__ == '__main__':
  app = wx.App()
  frame = MyFrame()
  frame.Show(True)
  app.MainLoop()
```

用鼠标消息实现滚动条

```
#include <windows.h>
#include "sysmets.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
```

```
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR szAppName[] = TEXT ("SysMets");
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("Fialed to RegisterClass!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Get System Metrics  -  use mouse
wheel"),
                          WS_OVERLAPPEDWINDOW | WS_VSCROLL | WS_HSCROLL,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static int  cxChar, cxCaps, cyChar, cxClient, cyClient, iMaxWidth ;
     static int  iDeltaPerLine, iAccumDelta ;     // for mouse wheel logic
     HDC         hdc ;
     int         i, x, y, iVertPos, iHorzPos, iPaintBeg, iPaintEnd ;
     PAINTSTRUCT ps ;
     SCROLLINFO  si ;
     TCHAR       szBuffer[10] ;
     TEXTMETRIC  tm ;
     ULONG       ulScrollLines ;                  // for mouse wheel logic

     switch (message)
     {
     case WM_CREATE:
```

```
          hdc = GetDC (hwnd) ;

          GetTextMetrics (hdc, &tm) ;
          cxChar = tm.tmAveCharWidth ;
          cxCaps = (tm.tmPitchAndFamily & 1 ? 3 : 2) * cxChar / 2 ;
          cyChar = tm.tmHeight + tm.tmExternalLeading ;

          ReleaseDC (hwnd, hdc) ;

               // Save the width of the three columns

          iMaxWidth = 40 * cxChar + 22 * cxCaps ;

               // Fall through for mouse wheel information

     case WM_SETTINGCHANGE:
          SystemParametersInfo (SPI_GETWHEELSCROLLLINES, 0, &ulScrollLines, 0) ;

               // ulScrollLines usually equals 3 or 0 (for no scrolling)
               // WHEEL_DELTA equals 120, so iDeltaPerLine will be 40

          if (ulScrollLines)
               iDeltaPerLine = WHEEL_DELTA / ulScrollLines ;
          else
               iDeltaPerLine = 0 ;

          return 0 ;

     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;

               // Set vertical scroll bar range and page size

          si.cbSize = sizeof (si) ;
          si.fMask  = SIF_RANGE | SIF_PAGE ;
          si.nMin   = 0 ;
          si.nMax   = NUMLINES - 1 ;
          si.nPage  = cyClient / cyChar ;
          SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;

               // Set horizontal scroll bar range and page size

          si.cbSize = sizeof (si) ;
          si.fMask  = SIF_RANGE | SIF_PAGE ;
          si.nMin   = 0 ;
          si.nMax   = 2 + iMaxWidth / cxChar ;
          si.nPage  = cxClient / cxChar ;
          SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
          return 0 ;

     case WM_VSCROLL:
               // Get all the vertical scroll bar information

          si.cbSize = sizeof (si) ;
          si.fMask  = SIF_ALL ;
          GetScrollInfo (hwnd, SB_VERT, &si) ;
```

```
                // Save the position for comparison later on

          iVertPos = si.nPos ;

          switch (LOWORD (wParam))
          {
          case SB_TOP:
                si.nPos = si.nMin ;
                break ;

          case SB_BOTTOM:
                si.nPos = si.nMax ;
                break ;

          case SB_LINEUP:
                si.nPos -= 1 ;
                break ;

          case SB_LINEDOWN:
                si.nPos += 1 ;
                break ;

          case SB_PAGEUP:
                si.nPos -= si.nPage ;
                break ;

          case SB_PAGEDOWN:
                si.nPos += si.nPage ;
                break ;

          case SB_THUMBTRACK:
                si.nPos = si.nTrackPos ;
                break ;

          default:
                break ;
          }
                // Set the position and then retrieve it.  Due to adjustments
                //   by Windows it may not be the same as the value set.

          si.fMask = SIF_POS ;
          SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
          GetScrollInfo (hwnd, SB_VERT, &si) ;

                // If the position has changed, scroll the window and update it

          if (si.nPos != iVertPos)
          {
                ScrollWindow (hwnd, 0, cyChar * (iVertPos - si.nPos),
                                    NULL, NULL) ;
                UpdateWindow (hwnd) ;
          }
          return 0 ;

     case WM_HSCROLL:
                // Get all the vertical scroll bar information

          si.cbSize = sizeof (si) ;
```

```
          si.fMask  = SIF_ALL ;

               // Save the position for comparison later on

          GetScrollInfo (hwnd, SB_HORZ, &si) ;
          iHorzPos = si.nPos ;

          switch (LOWORD (wParam))
          {
          case SB_LINELEFT:
               si.nPos -= 1 ;
               break ;

          case SB_LINERIGHT:
               si.nPos += 1 ;
               break ;

          case SB_PAGELEFT:
               si.nPos -= si.nPage ;
               break ;

          case SB_PAGERIGHT:
               si.nPos += si.nPage ;
               break ;

          case SB_THUMBPOSITION:
               si.nPos = si.nTrackPos ;
               break ;

          default:
               break ;
          }
               // Set the position and then retrieve it.  Due to adjustments
               //   by Windows it may not be the same as the value set.

          si.fMask = SIF_POS ;
          SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
          GetScrollInfo (hwnd, SB_HORZ, &si) ;

               // If the position has changed, scroll the window

          if (si.nPos != iHorzPos)
          {
               ScrollWindow (hwnd, cxChar * (iHorzPos - si.nPos), 0,
                             NULL, NULL) ;
          }
          return 0 ;

     case WM_KEYDOWN :
          switch (wParam)
          {
          case VK_HOME :
               SendMessage (hwnd, WM_VSCROLL, SB_TOP, 0) ;
               break ;

          case VK_END :
               SendMessage (hwnd, WM_VSCROLL, SB_BOTTOM, 0) ;
               break ;
```

```
          case VK_PRIOR :
               SendMessage (hwnd, WM_VSCROLL, SB_PAGEUP, 0) ;
               break ;

          case VK_NEXT :
               SendMessage (hwnd, WM_VSCROLL, SB_PAGEDOWN, 0) ;
               break ;

          case VK_UP :
               SendMessage (hwnd, WM_VSCROLL, SB_LINEUP, 0) ;
               break ;

          case VK_DOWN :
               SendMessage (hwnd, WM_VSCROLL, SB_LINEDOWN, 0) ;
               break ;

          case VK_LEFT :
               SendMessage (hwnd, WM_HSCROLL, SB_PAGEUP, 0) ;
               break ;

          case VK_RIGHT :
               SendMessage (hwnd, WM_HSCROLL, SB_PAGEDOWN, 0) ;
               break ;
          }
          return 0 ;

     case WM_MOUSEWHEEL:
          if (iDeltaPerLine == 0)
               break ;

          iAccumDelta += (short) HIWORD (wParam) ;     // 120 or -120

          while (iAccumDelta >= iDeltaPerLine)
          {
               SendMessage (hwnd, WM_VSCROLL, SB_LINEUP, 0) ;
               iAccumDelta -= iDeltaPerLine ;
          }

          while (iAccumDelta <= -iDeltaPerLine)
          {
               SendMessage (hwnd, WM_VSCROLL, SB_LINEDOWN, 0) ;
               iAccumDelta += iDeltaPerLine ;
          }

          return 0 ;

     case WM_PAINT :
          hdc = BeginPaint (hwnd, &ps) ;

               // Get vertical scroll bar position

          si.cbSize = sizeof (si) ;
          si.fMask  = SIF_POS ;
          GetScrollInfo (hwnd, SB_VERT, &si) ;
          iVertPos = si.nPos ;

               // Get horizontal scroll bar position
```

```
          GetScrollInfo (hwnd, SB_HORZ, &si) ;
          iHorzPos = si.nPos ;

               // Find painting limits

/*          iPaintBeg = max (0, iVertPos + ps.rcPaint.top / cyChar) ;
          iPaintEnd = min (NUMLINES - 1,
                           iVertPos + ps.rcPaint.bottom / cyChar) ;

*/
          iPaintBeg = 0 ;
          iPaintEnd =NUMLINES - 1;

          for (i = iPaintBeg ; i <= iPaintEnd ; i++)
          {
               x = cxChar * (1 - iHorzPos) ;
               y = cyChar * (i - iVertPos) ;

               TextOut (hdc, x, y,
                        sysmetrics[i].szLabel,
                        lstrlen (sysmetrics[i].szLabel)) ;

               TextOut (hdc, x + 22 * cxCaps, y,
                        sysmetrics[i].szDesc,
                        lstrlen (sysmetrics[i].szDesc)) ;

               SetTextAlign (hdc, TA_RIGHT | TA_TOP) ;

               TextOut (hdc, x + 22 * cxCaps + 40 * cxChar, y, szBuffer,
                        wsprintf (szBuffer, TEXT ("%5d"),
                             GetSystemMetrics (sysmetrics[i].iIndex))) ;

               SetTextAlign (hdc, TA_LEFT | TA_TOP) ;
          }

          EndPaint (hwnd, &ps) ;
          return 0 ;

     case WM_DESTROY :
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

wx.static

```
import wx
class StaticTextFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, -1, 'Static Text Example', size=(400,
300))
        panel = wx.Panel(self, wx.ID_ANY)
```

```
        wx.StaticText(panel, wx.ID_ANY, u"This is an example of 静态文本框", (100,
10))
        rev = wx.StaticText(panel, wx.ID_ANY, "Static Text With Reversed
Colors", (100, 30))
        rev.SetForegroundColour('white')
        rev.SetBackgroundColour('black')
        center = wx.StaticText(panel, wx.ID_ANY, "align center", (100, 50),
(160, -1), wx.ALIGN_CENTER)
        center.SetForegroundColour('white')
        center.SetBackgroundColour('black')
        right = wx.StaticText(panel, wx.ID_ANY, "align right", (100, 70),  (160,
-1), wx.ALIGN_RIGHT)
        right.SetForegroundColour('white')
        right.SetBackgroundColour('black')
        str = u"You can also change the font."
        text = wx.StaticText(panel, wx.ID_ANY, str, (20, 100))
        text.SetFont(wx.Font(18, wx.DECORATIVE, wx.ITALIC, wx.NORMAL))
        wx.StaticText(panel, wx.ID_ANY, "Your text\ncan be split\n"
                "over multiple lines\n\neven blank ones", (20,150))
        wx.StaticText(panel, -1, "Multi-line text\ncan also\n"
                "be right aligned\n\neven with a blank", (220,150),
style=wx.ALIGN_RIGHT)
        self.Centre()

if __name__ == '__main__':
    app = wx.App()
    frame = StaticTextFrame()
    frame.Show()
    app.MainLoop()
```

colors.c

```
#include <windows.h>

LRESULT CALLBACK WndProc    (HWND, UINT, WPARAM, LPARAM) ;
LRESULT CALLBACK ScrollProc (HWND, UINT, WPARAM, LPARAM) ;

int     idFocus ;
WNDPROC OldScroll ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR szAppName[] = TEXT ("Colors") ;
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
```

```
     wndclass.hbrBackground = CreateSolidBrush (0) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("Failed to RegisterClass!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Color Scroll"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage  (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static COLORREF crPrim[3] = { RGB (255, 0, 0), RGB (0, 255, 0),
                                   RGB (0, 0, 255) } ;
     static HBRUSH  hBrush[3], hBrushStatic ;
     static HWND    hwndScroll[3], hwndLabel[3], hwndValue[3], hwndRect ;
     static int     color[3], cyChar ;
     static RECT    rcColor ;
     static TCHAR * szColorLabel[] = { TEXT ("Red"), TEXT ("Green"),
                                       TEXT ("Blue") } ;
     HINSTANCE      hInstance ;
     int            i, cxClient, cyClient ;
     TCHAR          szBuffer[10] ;

     switch (message)
     {
     case WM_CREATE :
          hInstance = (HINSTANCE) GetWindowLong (hwnd, GWL_HINSTANCE) ;

               // Create the white-rectangle window against which the
               // scroll bars will be positioned. The child window ID is 9.

          hwndRect = CreateWindow (TEXT ("static"), NULL,
                                   WS_CHILD | WS_VISIBLE | SS_WHITERECT,
                                   0, 0, 0, 0,
                                   hwnd, (HMENU) 9, hInstance, NULL) ;

          for (i = 0 ; i < 3 ; i++)
          {
                    // The three scroll bars have IDs 0, 1, and 2, with
```

```
                    // scroll bar ranges from 0 through 255.

               hwndScroll[i] = CreateWindow (TEXT ("scrollbar"), NULL,
                                             WS_CHILD | WS_VISIBLE |
                                             WS_TABSTOP | SBS_VERT,
                                             0, 0, 0, 0,
                                             hwnd, (HMENU) i, hInstance, NULL) ;


               {   SCROLLINFO si;
                   si.cbSize=sizeof(SCROLLINFO);
                   si.fMask=SIF_POS|SIF_RANGE;
                   si.nMin=0;
                   si.nMax=255;
                   si.nPos=0;

                   SetScrollInfo(hwndScroll[i],SB_CTL,&si,FALSE);
               }
/*
           SetScrollRange (hwndScroll[i], SB_CTL, 0, 255, FALSE) ;
           SetScrollPos   (hwndScroll[i], SB_CTL, 0, FALSE) ;
*/
                    // The three color-name labels have IDs 3, 4, and 5,
                    // and text strings "Red", "Green", and "Blue".

               hwndLabel [i] = CreateWindow (TEXT ("static"), szColorLabel[i],
                                             WS_CHILD | WS_VISIBLE | SS_CENTER,
                                             0, 0, 0, 0,
                                             hwnd, (HMENU) (i + 3),
                                             hInstance, NULL) ;

                    // The three color-value text fields have IDs 6, 7,
                    // and 8, and initial text strings of "0".

               hwndValue [i] = CreateWindow (TEXT ("static"), TEXT ("0"),
                                             WS_CHILD | WS_VISIBLE | SS_CENTER,
                                             0, 0, 0, 0,
                                             hwnd, (HMENU) (i + 6),
                                             hInstance, NULL) ;

               OldScroll = (WNDPROC) SetWindowLong (hwndScroll[i],
   //设置新的窗口过程函数，并获得原来的
                                             GWL_WNDPROC, (LONG) ScrollProc) ;

               hBrush[i] = CreateSolidBrush (crPrim[i]) ;
          }

          hBrushStatic = CreateSolidBrush (
                              GetSysColor (COLOR_BTNHIGHLIGHT)) ;

          cyChar = HIWORD (GetDialogBaseUnits ()) ;
          return 0 ;

     case WM_SIZE :
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;

          SetRect (&rcColor, cxClient / 2, 0, cxClient, cyClient) ;
```

```
          MoveWindow (hwndRect, 0, 0, cxClient / 2, cyClient, TRUE) ;

          for (i = 0 ; i < 3 ; i++)
          {
               MoveWindow (hwndScroll[i],
                           (2 * i + 1) * cxClient / 14, 2 * cyChar,
                           cxClient / 14, cyClient - 4 * cyChar, TRUE) ;

               MoveWindow (hwndLabel[i],
                           (4 * i + 1) * cxClient / 28, cyChar / 2,
                           cxClient / 7, cyChar, TRUE) ;

               MoveWindow (hwndValue[i],
                           (4 * i + 1) * cxClient / 28,
                           cyClient - 3 * cyChar / 2,
                           cxClient / 7, cyChar, TRUE) ;
          }
          SetFocus (hwnd) ;
          return 0 ;

     case WM_SETFOCUS :
          SetFocus (hwndScroll[idFocus]) ;
          return 0 ;

     case WM_VSCROLL :
          i = GetWindowLong ((HWND) lParam, GWL_ID) ;

          switch (LOWORD (wParam))
          {
          case SB_PAGEDOWN :
               color[i] += 15 ;
                                             // fall through
          case SB_LINEDOWN :
               color[i] = min (255, color[i] + 1) ;
               break ;

          case SB_PAGEUP :
               color[i] -= 15 ;
                                             // fall through
          case SB_LINEUP :
               color[i] = max (0, color[i] - 1) ;
               break ;

          case SB_TOP :
               color[i] = 0 ;
               break ;

          case SB_BOTTOM :
               color[i] = 255 ;
               break ;

          case SB_THUMBPOSITION :
          case SB_THUMBTRACK :
               color[i] = HIWORD (wParam) ;
               break ;

          default :
```

```
                break ;
          }

          {   SCROLLINFO si;
              si.cbSize=sizeof(SCROLLINFO);
              si.fMask=SIF_POS;
              si.nPos=color[i];
              SetScrollInfo(hwndScroll[i],SB_CTL,&si,TRUE);
           }
/*
          SetScrollPos  (hwndScroll[i], SB_CTL, color[i], TRUE) ;
*/
          wsprintf (szBuffer, TEXT ("%i"), color[i]) ;
          SetWindowText (hwndValue[i], szBuffer) ;

          DeleteObject ((HBRUSH)
               SetClassLong (hwnd, GCL_HBRBACKGROUND, (LONG)
                    CreateSolidBrush (RGB (color[0], color[1], color[2])))) ;

          InvalidateRect (hwnd, &rcColor, TRUE) ;
          return 0 ;

     case WM_CTLCOLORSCROLLBAR :
          i = GetWindowLong ((HWND) lParam, GWL_ID) ;
          return (LRESULT) hBrush[i] ;

     case WM_CTLCOLORSTATIC :
          i = GetWindowLong ((HWND) lParam, GWL_ID) ;

          if (i >= 3 && i <= 8)    // static text controls
          {
               SetTextColor ((HDC) wParam, crPrim[i % 3]) ;
               SetBkColor ((HDC) wParam, GetSysColor (COLOR_BTNHIGHLIGHT));
               return (LRESULT) hBrushStatic ;
          }
          break ;

     case WM_SYSCOLORCHANGE :
          DeleteObject (hBrushStatic) ;
          hBrushStatic = CreateSolidBrush (GetSysColor (COLOR_BTNHIGHLIGHT)) ;
          return 0 ;

     case WM_DESTROY :
          DeleteObject ((HBRUSH)
               SetClassLong (hwnd, GCL_HBRBACKGROUND, (LONG)
                    GetStockObject (WHITE_BRUSH))) ;

          for (i = 0 ; i < 3 ; i++)
               DeleteObject (hBrush[i]) ;

          DeleteObject (hBrushStatic) ;
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}

LRESULT CALLBACK ScrollProc (HWND hwnd, UINT message,
```

```
                              WPARAM wParam, LPARAM lParam)
{
     int id = GetWindowLong (hwnd, GWL_ID) ;

     switch (message)
     {
     case WM_KEYDOWN :
          if (wParam == VK_TAB)
               SetFocus (GetDlgItem (GetParent (hwnd),
                    (id + (GetKeyState (VK_SHIFT) < 0 ? 2 : 1)) % 3)) ;
          break ;

     case WM_SETFOCUS :
          idFocus = id ;
          break ;
     }
     return CallWindowProc (OldScroll, hwnd, message, wParam, lParam) ;
}
```