LECTURE SEVEN
RESOURCEES

7.1 Introduction

It can be observed that virtually every windows application use resources. The definition for the resources such as –Menus, Icon, Cursor, Bitmap etc. is written with a few line of code in the. **RC resource script file**. The advantage of resources is that they are loaded into the memory only when required. A resource compiler (RC-Exe) is used for compiling the RC resource script file, which output a **Res file**. Further this .RC file linked with the **.exe** file at the linking process, to give the final executable file.

Resources can be used in your application program by two methods-

- First by writing the resources script file that includes definition of resources.
- Second using the resource editor.

Writing a Resource Script file

For writing a resource script file a text editor is required, after writing few lines of code in the editor which defines the particular resource, the file is saved with .RC extension. For example;

ICON.RC

```
My App  ICON  Icon.ico  //File1 singleline resource type
```

CURSOR.RC

```
My App  CURSOR  Cursor.cur  //File2 single line resource type
```

MENU.RC

```
#include "mymenu.h"
MyMenu MENU
BEGIN
POPUP"&File"
BEGIN
 MENUITEM "New" IDM_NEW
  MENUITEM "Open" IDM_OPEN
MENUITEM SEPARATOR
MENUITEM "Exit" IDM_EXIT
END
```

```
POPU "&Edit"
BEGIN

MENUITEM "Cut" IDM_CUT
  MENUITEM "Copy" IDM_COPY
MENUITEM "Paste"  IDM_PASTE
MENUITEM "Exit" IDM_EXITEND
POPUP"&Help"
BEGIN
  MENUITEM "Window Application" IDM_WINDOW
END
END
```

There are five single line resources script types:
ICON
CURSOR
BITMAP
FONT
Message Table

There are five multiple-line resource script types:
Menu
Dialog
Acceletors
Rcdata
String Table

7.2 MENUS

Every window application that interacts with the user selection, and act upon it uses menus. Generally two types of menus are used.

Main Menu
This menu is a top – level menu of the application program in which the items are visible throughout the program. Each item is associated with a particular action and acts depending upon the particular selection see fig-7.1

Fig 7.1: Top level Menu

Sub Menu

This menu is a drop-down pop-up menu, in which the items are attached with the top-level menu. These items become visible when a particular top-level menu item is selected. See figure 15.2

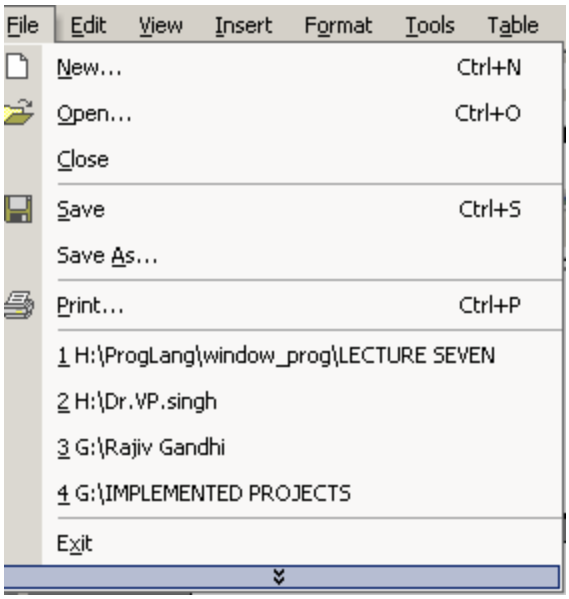

Fig 7.2 : Sub Meu

In this section we will discuss two way of creating menus for your application program. The first one is accomplished by creating your own resource file, while the second one uses the resource editor.

In addition to above specified menus of floating, stand alone pop-up menus is also available.

7.3. CREATING RESOURCE FILE (.RC)

A few lines of code can define a menu. The file should have .RC extension (so as to identify as resources file). The following example produces the menu structure shown in fig.15.3

```
#include "mymenu.h"
MyMenu MENU
BEGIN
POPUP"&File"
BEGIN
 MENUITEM "New" IDM_NEW
  MENUITEM "Open" IDM_OPEN
MENUITEM SEPARATOR
```

```
MENUITEM "Exit" IDM_EXIT
END
POPU "&Edit"
BEGIN

MENUITEM "Cut" IDM_CUT
  MENUITEM "Copy" IDM_COPY
MENUITEM "Paste"  IDM_PASTE
MENUITEM "Exit" IDM_EXITEND
POPUP"&Help"
BEGIN
  MENUITEM "Window Application" IDM_WINDOW
END
END
```

Menu header file mymenu.h

```
#define IDM_NEW1
#define IDM_OPEN2
#define IDM_EXIT3
#define IDM_CUT4
#define IDM_COPY5
#define IDM_PASTE6
#define IDM_WINDOW7
```

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

Fig 7.3 : Menu structure

It should be noted that the ampersand (&) symbol used with menu option lets you create keyboard alternatives for clicking menu items with the mouse. The characters followed by '&' is underlined in the menu; using keyboard a user can select the item by pressing the Alt Key and simultaneously pressing the key for the underlined letter.

7.4. ATTACHING MENU TO YOUR APPLICATION WINDOW

A few lines of code in the resource file (file name) is not capable of attaching menu to the application window. For this we have to attach the menu to the window's class definition in the WinMain ( ) function. Setting lpszMenu Name of the WNDCLASS structure to point to the Menu name can do this. Now it's the task of **Register Class** ( ) function to associate this menu to window application that is created by this class. The other method of attaching menu to application program window is by using **CreateWindow** ( ) and LoadMenu ( ) function. The following few line of code loads the menus.

Load Menu ( ):

Load menu loads the user menu from the resource file and returns the handle of the menu. If it is used in WinMain() function then the user menu is loaded. If it is used in wndproc() function then depending upon the value of menu handle, particular menu is loaded.

The syntax

HMENU LoadMenu (HINSTANCE hlnst, LPSTR lpMenuName);

Parameters

hlnst: The handle of the instance for the module that contains the menu definition in the resource data.

LpMenuName: A pointer to a null terminated string which is a menu name that is to be loaded. Instead, we can also use MAKEINTRESOURCE macro that identify the resource identifier.

The following lines of code in the WinMain( ) function shown below has the same effect as we have seen in fig. 15.3.

```
HMENU hmenu;/handle to the menu
Wc.lpszMenuName=NULL;//no  original menu definition
Hmenu=LoadMenu(hlnst,"hmenu");//loading menu
Hwnd=CreateWindow("MyWindow",
                                    "My Window",
WS_OVERLAPPEDWINDOW /WS_VSCROLL / WS_HSCROLL,
```

100,
100,
300,
300,
NULL
Hmenu,
Hlnst,
NULL);

7.5.MENU MESSAGE

Window send the WM_COMMAND message to WndProc(user defiend function for taking the action over the messages received) function every time the menu item is selected. But for the system menu the WS_SYSCOMMAND message is sent. Table given below are the some menu message along with brief description.

WM_COMMAND	menu messages.
WM_INITMENU	main menu is selected
WM_POPUP	popmenu is selected.
WM_MENUCHAR	activated when user selects any other shortcut key defined by & it is sent even if the menu item is grayed.
WM_MENUSELECT	it is sent even if the menu item is grayed.

Following program creates a used defined menu.

Program 7.1 creation of User defined menu

```
//Program 7.1 Creation of user defined menu

#include <windows.h>
#include "mymenu.h"

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

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE hPrev,
    LPSTR nCmd, int nShow)
{
    WNDCLASS wc;
    HWND hwnd;
    MSG Msg;
```

```
wc.cbClsExtra=NULL;
wc.cbWndExtra    = 0;
wc.hInstance     = hInst;
wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
   wc.lpfnWndProc   =WndProc;
wc.lpszMenuName  = "MyMenu";
wc.lpszClassName = "MyWindow";

if(!RegisterClass(&wc))
{
    MessageBox(NULL, "Not Registered", "SEE THAT", MB_OK);
    return 0;
}
/* HMENU  hmenu;//handle to menu
   wc.lpszClassNameMenuName=NULL;//no original menu definition
   hmenu=LoadMenu(hInst, "hmenu");//loading menu
*/
   hwnd = CreateWindow(
    "MyWindow",
    "MyWindow",
    WS_OVERLAPPEDWINDOW|WS_VSCROLL|WS_HSCROLL,
    CW_USEDEFAULT, CW_USEDEFAULT, 300,300,
    NULL, NULL, hInst, NULL);

if(hwnd == NULL)
{
    MessageBox(NULL, "Window Creation Failed!", "Error!",
       MB_ICONEXCLAMATION | MB_OK);
    return 0;
}

ShowWindow(hwnd, nShow);

while(GetMessage(&Msg, NULL, 0, 0) )
{
```

```
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }
    return Msg.wParam;
}

LRESULT CALLBACK WndProc(HWND  hWnd, UINT msg, WPARAM
wParam, LPARAM lParam)
{
    switch(msg)
    {
        case WM_COMMAND:
        {
            switch(wParam)
                    {
             case IDM_NEW:

             MessageBox(NULL,  "NEW     was clicked", "VP SINGH",
MB_OK);

        break;
        case IDM_OPEN:
            MessageBox(NULL,  "Open was clicked", "VP SINGH",
MB_OK);
        break;
        case IDM_EXIT:
            MessageBox(NULL,  "EXIT was clicked", "VP SINGH",
MB_OK);
        break;
        case IDM_CUT:
            MessageBox(NULL,  "CUT was clicked", "VP SINGH", MB_OK);
        break;
             case IDM_COPY:
            MessageBox(NULL,  "COPY was clicked", "VP SINGH",
MB_OK);
        break;

             case IDM_PASTE:
```

```
            MessageBox(NULL,  "PASTE was clicked", "VP SINGH",
MB_OK);
        break;
             case IDM_WINDOW:
            MessageBox(NULL,  "WINDOW was clicked", "VP SINGH",
MB_OK);
        break;
                    }
                    break;
              }
              case WM_DESTROY:
            MessageBox(hWnd,"Window is clsed","VP Singh",MB_OK);
                    break;
                    default;
                          return DefWindowProc(hWnd,
msg,wParam,lParam);

    }
    return 0;
}
```

7.6. ICON, CURSOR AND BITMAPS

All the icons, cursors and bitmaps are created by using an image editor provided by the Complier that is capable of creating windows 98 programs.

7.6.1. ICON

Basically Icons comes in three sizes:

Small	: 16 x 16 size
Standard	: 32 x 32 size
Large	: 48 x48 size

All the icons can be defined with a single icon file.