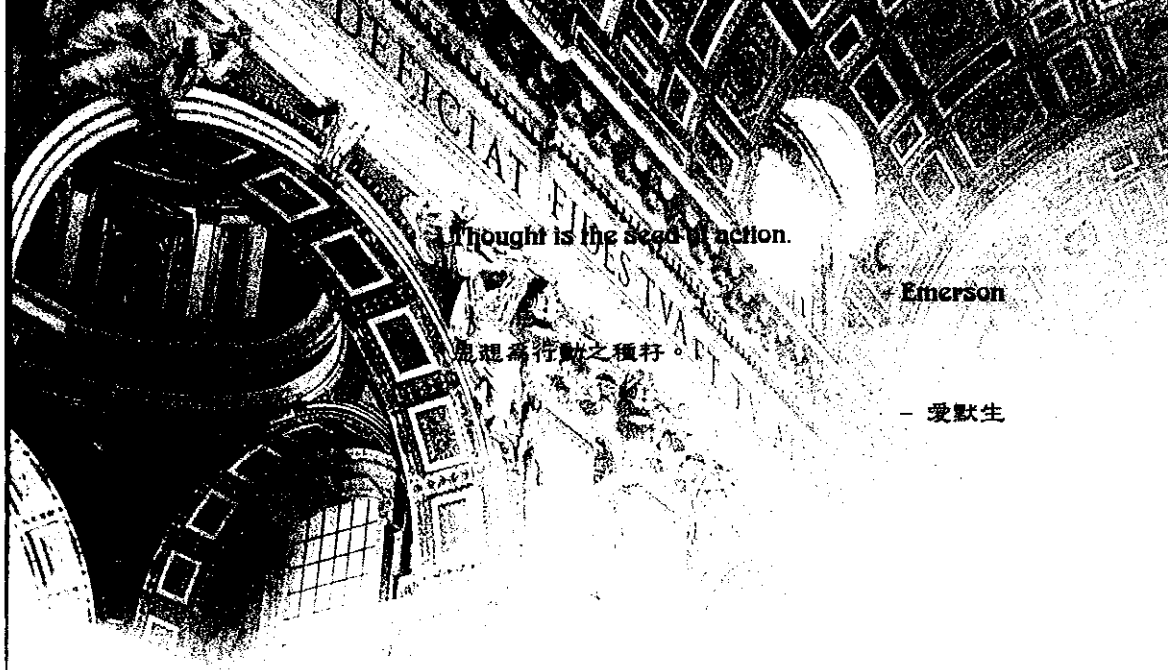


Thought is the seed of action.

– Emerson

思想為行動之種籽。

– 愛默生

8

功能表的建立

本章導讀

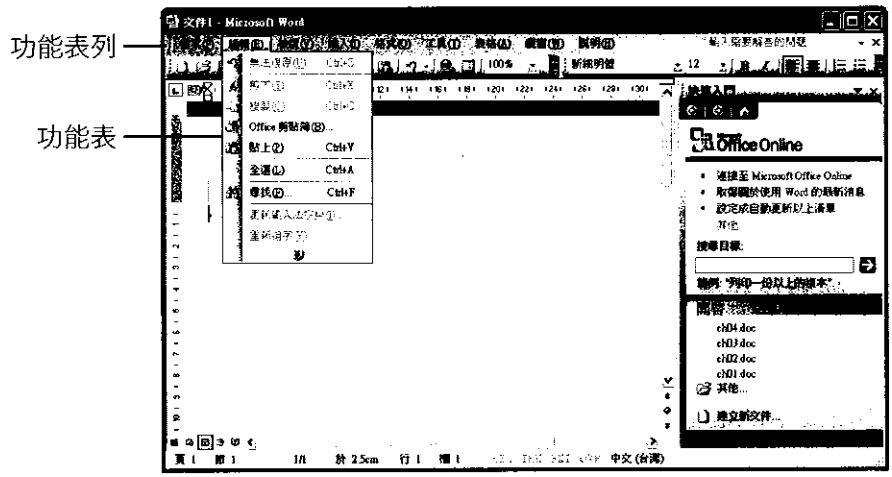
功能表是視窗應用程式中，供使用者執行命令的主要介面。功能表將應用程式提供的功能，以分類的方式置於各功能表裡。這一章除告訴您，如何利用載入資源檔的方式建立視窗的功能表，更告訴您如何利用 CMenu 物件操作功能表。

各節標題

8-1 簡介功能表

功能表的組成

功能表是視窗介面中，提供使用者選取執行指令的主要控制項。一個應用程式通常會有一個功能表列，並由好幾個功能表組成，且以分門別類的方式整理應用程式可供執行的命令，方便使用者使用。下圖為 Word 之視窗介面所提供的功能表列與功能表。



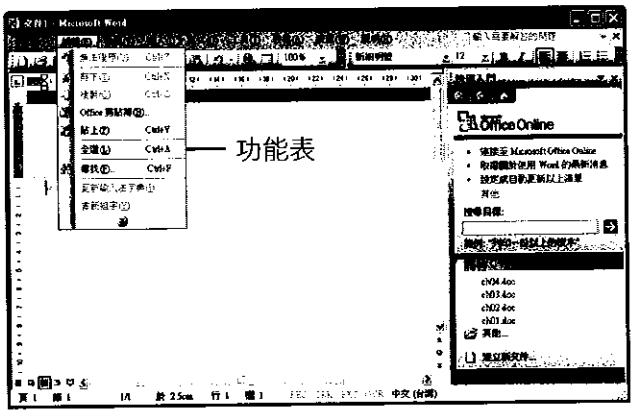
當建立應用程式的功能表資源時，功能表的每個選項，均將設定代表該選項的**識別子**。而此識別子將用於建立訊息映射項目，以連結回應選項執行動作的**訊息處理函數**。

功能表的種類

視窗介面的功能表是提供使用者選擇執行動作的主要方式之一，功能表的種類，大致有以下幾種。

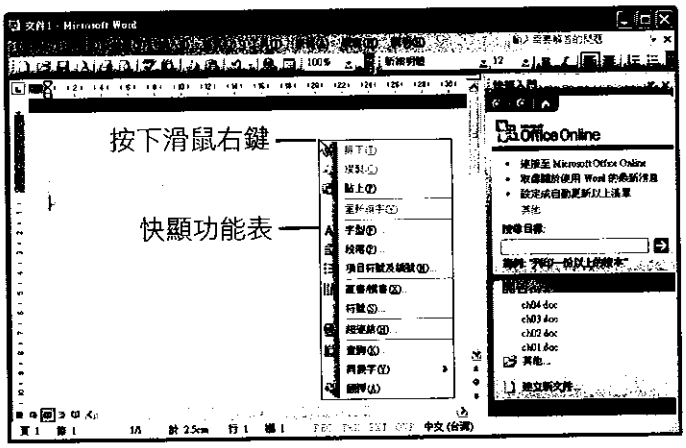
一、下拉功能表

幾乎所有視窗應用程式均提供功能表，並將程式可執行的所有指令，分門別類地置於功能表。當使用者欲執行某一命令時，只需利用滑鼠點選展開功能表，再選取選項即可。以下為 Word 的**編輯**功能表。



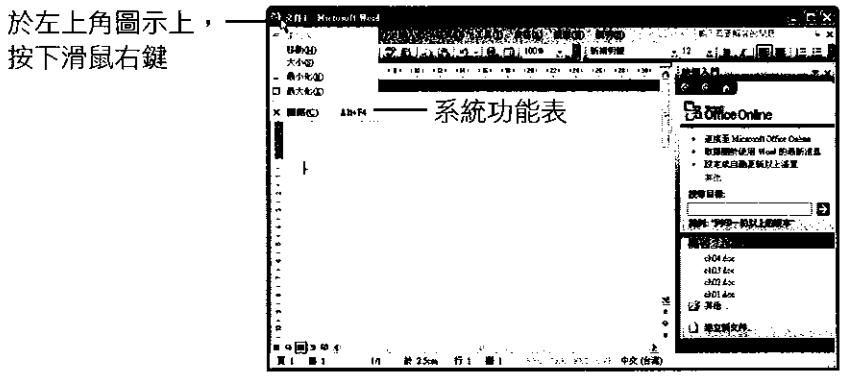
二、快顯功能表

快顯功能表為使用者於視窗上按下滑鼠右鍵後，蹦現於視窗的功能表。以下為在 Word 編輯畫面上按下滑鼠右鍵後，所產生的快顯功能表。



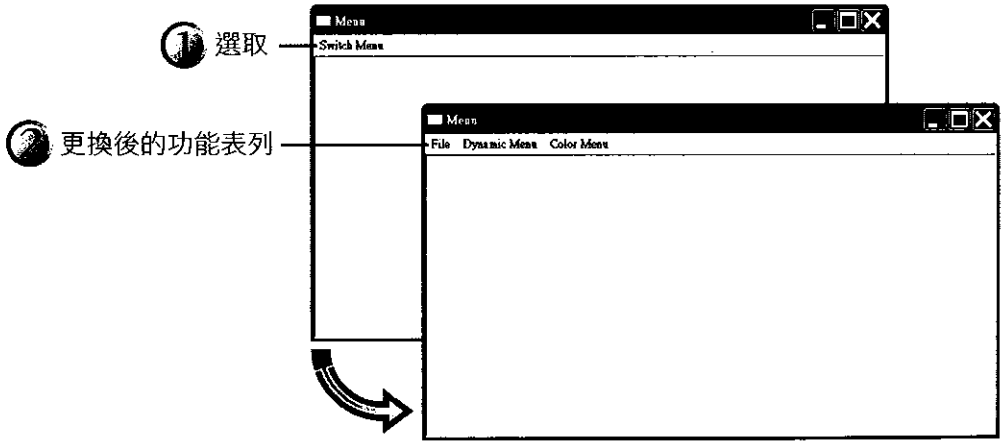
三、系統功能表

在視窗程式左上角圖示上按下滑鼠左鍵時，將出現**系統功能表**，功能表內選項將提供控制整個視窗程式的命令。下圖為 Word 的系統功能表。

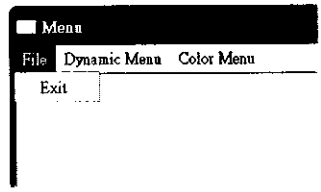


8-2 menu 程式範例

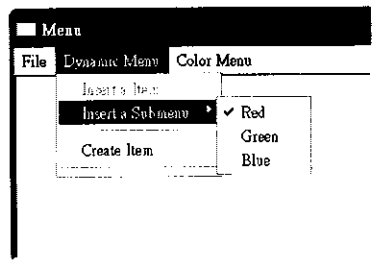
下圖為程式範例的執行結果，並同時顯示各功能表的選項。



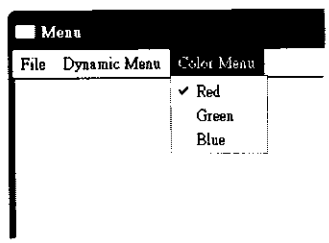
下圖為 File 功能表的選項內容。



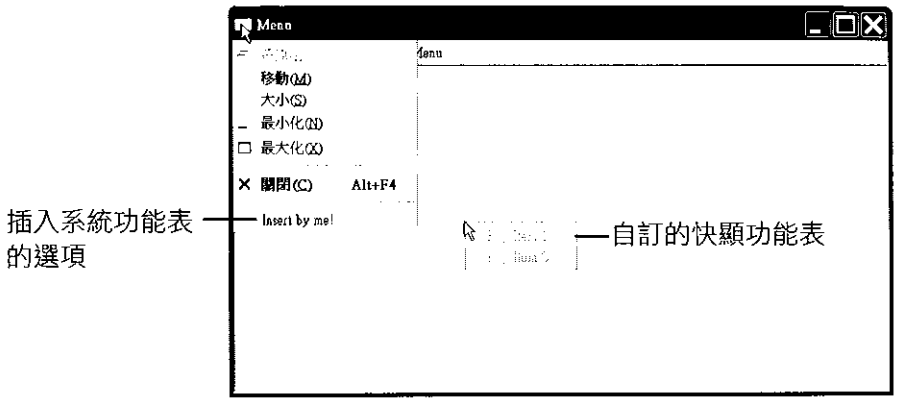
下圖為 Dynamic Menu 功能表的選項內容。



下圖為 Color Menu 功能表的選項內容。



以上各圖內所有功能表的選項，並不是全部都利用資源編輯器建立。建立視窗程式的功能表有兩種方式，一種是利用資源編輯器建立功能表，再將該資源載入應用程式。另一種是利用 CMenu 類別操作功能表。前者是建立功能表最基本方式，已於 7-2 節說明。後者將更進一步地控制功能表，並在程式執行的過程中，動態操作功能表，將於本章示範。下圖是執行本程式範例的另一重點 - **系統功能表與快顯功能表**。



本程式範例欲說明的重點如下：

1. **功能表的建立**：利用 CFrameWnd::Create()函數直接載入功能表資源。（8-3 節）
2. **功能表列的更換**：在某些情形下，應用程式的功能表列並不只有一個，必須視情形更換。此範例將示範如何更換功能表列，當程式開始執行時，功能表列將提供一個切換選項 - Switch Menu。按下該選項後，將可切換功能表。切換後的功能表將由 File、Dynamic Menu、Color Menu 三個功能表組成。（8-4 節）
3. **利用 CMenu 類別控制功能表**：利用 CMenu 修改功能表在這個程式範例裡，可以分為兩個部份說明。一是**修改功能表**，二是**動態控制功能表**。程式執行時，視窗程式內有 File、Dynamic Menu、Color Menu 三個功能表，但它們並不是全部利用功能表編輯器所建立。功能表的 Color Menu 與 Dynamic Menu 的 Insert a Item、Insert a Submenu 選項為程式執行時才動態增加的。而所謂的動態控制功能表，則是指程式執行時，動態改變的功能表選項。Dynamic Menu 功能表的 Create Item 選項，於點選後，將可建立功能表的另一個 Create Item 選項，而該選項本身則變成 Delete Item 選項。按下 Delete Item 選項後，則會刪除前面動態產生的 Create Item 選項。（8-5 節）

4. **修改系統功能表與建立快顯功能表**：於系統功能表增加選項，並建立按下滑鼠右鍵後，產生的快顯功能表。（8-6 節）

除了以上重點外，本程式範例更示範如何利用功能表控制滑鼠軌跡的顏色，以下為程式的完整內容。

menu 程式範例 - 功能表的建立與 CMenu 類別的使用

檔案位置：menu\menu.cpp

```
1    /*
2    範例檔名：menu.cpp
3    程式開發：郭尚君
4    */
5    #include <afxwin.h>
6    #include "menu.h"   //載入資源檔所使用之標頭檔
7
8    class MyFrame : public CFrameWnd
9    {
10   private:
11      CMenu MainMenu, *SysMenu, *PopMenu, * SubMenu;
12      COLORREF color;
13   public:
14      MyFrame()   //建構子
15      {
16         Create(NULL,   //產生標準視窗
17                  "Menu",  //視窗標題
18                  WS_OVERLAPPEDWINDOW|WS_VISIBLE,    //視窗樣式
19                  rectDefault,     //視窗大小
20                  NULL, //指向父視窗的指標
21                  MAKEINTRESOURCE(IDR_SMENU));//使用功能表的識別子
22
23         SysMenu = GetSystemMenu(FALSE);     //取得系統功能表
24         SysMenu->AppendMenu(MF_SEPARATOR); //插入分隔線
25         SysMenu->AppendMenu(MF_STRING, IDM_INSERT, "Insert by me!");
26         //插入選項
27         color = RGB(255,0,0); //將點的顏色預設爲紅色
28      }
29      //MyFrame::OnSwitchMenu()函數
```

```
30      afx_msg void OnSwitchMenu()
        //當 Switch Menu 選項被選取時的回應函數
31      {
32          CMenu InsMenu;//建立功能表物件
33
34          InsMenu.LoadMenu(IDR_INSMENU);      //載入功能表資源
35          MainMenu.LoadMenu(IDR_MAINMENU);  //載入功能表資源
36
37          MainMenu.AppendMenu(MF_POPUP, (UINT) InsMenu.m_hMenu,
38                                  "Color Menu");//增加一個功能表
39
40          SubMenu = MainMenu.GetSubMenu(1);//取得第 2 個功能表的指標
41
42          SubMenu->AppendMenu(MF_SEPARATOR);//插入分隔線
43          SubMenu->InsertMenu(IDM_CreateItem, MF_BYCOMMAND|MF_STRING,
44                      IDM_InsertItem, "Insert a Item"); //插入選項
45          SubMenu->InsertMenu(IDM_CreateItem,
                    MF_BYCOMMAND | MF_POPUP,
46                (UINT) InsMenu.m_hMenu, "Insert a Submenu");
                  //插入功能表
47          SetCheck(); //設定應被選取的選項
48
49          SubMenu->InsertMenu(IDM_CreateItem,
                     MF_BYCOMMAND | MF_SEPARATOR);
50                   //插入分隔線
51          InsMenu.Detach(); //將功能表資源與功能表物件分離
52
53          SetMenu(&MainMenu); //設定應用程式所使用之功能表
54      }
55      //MyFrame::OnCreateItem()函數
56      afx_msg void OnCreateItem()
57      {   //當 Dynamic Menu 功能表的 Create Item 選項被選取時的回應函數
58          MainMenu.ModifyMenu(IDM_CreateItem,MF_BYCOMMAND,
59                          IDM_DeleteItem,"Delete Item");
            //修改功能表中的 Create Item 選項爲 Delete Item
60
61          SubMenu->AppendMenu(MF_STRING, IDM_NewItem, "New Item");
62          //增加 Create item 選項
63      }
64      //MyFrame::OnDeleteItem()函數
65      afx_msg void OnDeleteItem()
```

```
66      {   //當 Dynamic Menu 功能表的 Delete Item 選項被選取時的回應函數
67          MainMenu.ModifyMenu(IDM_DeleteItem,MF_BYCOMMAND,
68                              IDM_CreateItem,"Create Item");
            //修改 Delete Item 選項爲 Create Item 選項
69
70          SubMenu->DeleteMenu(IDM_NewItem, MF_BYCOMMAND);
            //刪除 New Item 選項
71      }
72      //MyFrame::OnContextMenu()函數
73      afx_msg void OnContextMenu(CWnd* pWnd, CPoint point)
74      {   //按下滑鼠右鍵時，將執行此成員函數產生快顯功能表
75          CMenu menu;
76
77          menu.LoadMenu(IDR_POPMENU); //載入功能表資源
78
79          PopMenu = menu.GetSubMenu(0); //取得第 1 個功能表的指標
80
81          PopMenu->TrackPopupMenu(
82              TPM_CENTERALIGN|TPM_RIGHTBUTTON,point.x,point.y,
83              this); //建立快顯功能表
84          menu.Detach();
85      }
86      //MyFrame::OnRed()函數
87      afx_msg void OnRed()//當 Color Menu 中 Red 選項被選取時
88      {
89          SetUnCheck();  //取消原被打勾的選項
90          color = RGB(255,0,0);    //設定畫在畫布上的顏色
91          SetCheck();//設定應被打勾的選項
92      }
93      //MyFrame::OnGreen()函數
94      afx_msg void OnGreen()//當 Color Menu 中 Green 選項被選取時
95      {
96          SetUnCheck();
97          color = RGB(0,255,0);
98          SetCheck();
99      }
100     //MyFrame::OnBlue()函數
101     afx_msg void OnBlue() //當 Color Menu 中 Blue 選項被選取時
102     {
103         SetUnCheck();
```

```
104         color = RGB(0,0,255);
105         SetCheck();
106      }
107      //MyFrame::SetCheck()函數
108      void SetCheck() //判別 color 屬性，將 Color Menu 中對應的選項打勾
109      {
110         switch(color)
111         {
112         case RGB(255,0,0) :
113            MainMenu.CheckMenuItem(IDM_Red,MF_BYCOMMAND |
114                                   MF_CHECKED);  //將該選項打勾
115            break;
116         case RGB(0,255,0) :
117            MainMenu.CheckMenuItem(IDM_Green,
                                  MF_BYCOMMAND | MF_CHECKED);
118            break;
119         case RGB(0,0,255) :
120            MainMenu.CheckMenuItem(IDM_Blue,
               MF_BYCOMMAND | MF_CHECKED);
121            break;
122         }
123      }
124      //MyFrame::SetUnCheck()函數
125      void SetUnCheck() //取消 Color Menu 中被打勾的選項
126      {
127         switch(color)
128         {
129         case RGB(255,0,0) :
130            MainMenu.CheckMenuItem(IDM_Red,MF_BYCOMMAND |
131                     MF_UNCHECKED);//取消該選項打勾
132            break;
133         case RGB(0,255,0) :
134            MainMenu.CheckMenuItem(IDM_Green,
               MF_BYCOMMAND | MF_UNCHECKED);
135            break;
136         case RGB(0,0,255) :
137            MainMenu.CheckMenuItem(IDM_Blue,
               MF_BYCOMMAND | MF_UNCHECKED);
138            break;
139         }
140      }
141      //MyFrame::OnLButtonUp()函數
```

```
142     afx_msg void OnLButtonUp(UINT nFlags, CPoint point)
143     { ReleaseCapture(); }
        //當滑鼠左鍵放開後的回應函數，釋放滑鼠訊息接收權
144
145     //MyFrame::OnLButtonDown()函數
146     afx_msg void OnLButtonDown(UINT nFlags, CPoint point)
147     { SetCapture(); } //當滑鼠左鍵按下後的回應函數，取得滑鼠訊息接收權
148     //MyFrame::OnMouseMove()函數
149     afx_msg void OnMouseMove(UINT nFlags, CPoint point)
150     {  //當滑鼠移動時的回應函數
151        if (this == GetCapture()) //判斷滑鼠游標是否在視窗之上
152        {
153           CClientDC aDC(this); //建立一個畫布
154           aDC.SetPixel(point, color); //在畫布上畫出顏色爲 color 的點
155        }
156     }
157
158     DECLARE_MESSAGE_MAP() //宣告訊息映射表
159  };
160
161  //建立 MyFrame 類別的訊息映射表
162  BEGIN_MESSAGE_MAP(MyFrame, CFrameWnd)
163     ON_COMMAND(IDR_SwitchMenu, OnSwitchMenu)
164     ON_COMMAND(IDM_CreateItem, OnCreateItem)
165     ON_COMMAND(IDM_DeleteItem, OnDeleteItem)
166     ON_COMMAND(IDM_Red, OnRed)
167     ON_COMMAND(IDM_Green, OnGreen)
168     ON_COMMAND(IDM_Blue, OnBlue)
169     ON_WM_CONTEXTMENU() //按下滑鼠右鍵時，將呼叫 OnContextMenu
170     //回應滑鼠訊息
171     ON_WM_LBUTTONDOWN()
172     ON_WM_MOUSEMOVE()
173     ON_WM_LBUTTONUP()
174  END_MESSAGE_MAP()
175
176  class MyApp : public CwinApp //應用程式類別
177  {
178  public:
179     BOOL InitInstance() //程式進入點 ◀ 程式進入點
180     {
```

```
181         CFrameWnd *Frame = new MyFrame;
182         m_pMainWnd = Frame;
183         Frame->ShowWindow(SW_SHOW); //顯示視窗
184
185         return true;
186     }
187   } a_app; //宣告應用程式物件
```

✤ 使用資源

一、**功能表部份：** 以下是程式範例中使用的功能表資源。關於功能表的建立，請參考 7-2 節的說明。

識別子	IDR_POPMENU	IDR_MAINMENU	IDR_SMENU	IDR_INSMENU
功能表				
用途說明	快顯功能表	切換後的功能表列	程式起始時的功能表列	插入 IDR_MAINMENU 的功能表

二、**選項部份：** 下表為功能表內選項與 ID 的對照。

選項	選項	ID
IDR_POPMENU	POP Item 1	IDM_PopItem1
	POP Item 2	IDM_PopItem2
IDR_MAINMENU	Exit	ID_APP_EXIT
	Create Item	IDM_CreateItem
IDR_SMENU	Switch Menu	IDR_SwitchMenu
IDR_INSMENU	Red	IDM_Red
	Green	IDM_Green
	Blue	IDM_Blue

三、 **其他 ID**：程式範例將在功能表中插入許多選項，或修改功能表，在修改或插入選項的過程使用到的 ID 說明介紹如下表。

ID	用途
IDM_DeleteItem	當按下 Dynamic Menu 的 Create Item 選項後，將修改該選項 ID 為 IDM_CreateItem（第 48 行）。
IDM_INSERT	插入系統功能表的 Insert by me!選項的 ID(第 23 行)。
IDM_InsertItem	插入 Dynamic Menu 的 Insert Item 選項的 ID（第 77 行）。
IDM_NewItem	當按下 Dynamic Menu 的 Create Item 選項後，建立的選項的 ID（第 51 行）。

若欲察看這些 ID，請將**方案資訊區**切換至**方案總管**視窗，連按兩二資源檔，開啟**資源檢視**視窗，於執行資源檔項目資料夾上，按下滑鼠右鍵，選取快顯功能表的**資源符號**選項，即可於**資源符號**對話盒檢視方案使用的所有識別子（ID）。關於**資源符號**對話盒的說明請參考 7-7 節。

8-3 功能表的建立與設定

建立功能表第一步必須利用**資源編輯器**完成功能表列資源的建立，接著，連結資源物件與視窗程式。4-3-3 節說明了以 CMenu::LoadMenu()與 CWnd::SetMenu()函數，簡單建立 MyFrame 程式範例功能表的方法。

不過，建立功能表的方法並不需要這麼複雜，只要於視窗框架類別的建構子中，呼叫 CFrameWnd::Create()成員函數，然後將功能表的識別子傳給 Create()函數即可（關於該成員函數的說明，請參考 2-2-3 節）。

menu 程式範例便是運用這種方式建立功能表，此方法有點類似建立文件樣版物件時，將視窗程式欲使用的資源代號傳入文件樣版物件的建構子（6-2-3 節）。但由於 CFrameWnd::Create()成員函數所接受的第 6 個參數，其型態為指向字串的指標，因此，欲傳入的功能表識別子，需要利用 MAKEINTRESOURCE 轉換（第 21 行）。

```
'摘自 menu\menu.cpp 檔
14          MyFrame() //建構子
15          {
16              Create(NULL, //產生標準視窗
17                  "Menu", //視窗標題
18                  WS_OVERLAPPEDWINDOW | WS_VISIBLE, //視窗樣式
19                  rectDefault, //視窗大小
20                  NULL, //指向父視窗的指標
21                  MAKEINTRESOURCE(IDR_SMENU)); //使用功能表的識別子
...             ......
28          }
```

8-4 功能表的切換

第 16 至 21 行將把功能表的識別子（IDR_SMENU）傳入 CFrameWnd::Create()成員函數中，因此，執行程式範例後，視窗程式便會使用該功能表。當按下功能表的 Switch Menu 選項後，回應的 MyFrame::OnSwitchMenu()函數將 呼叫 CMenu::LoadMenu()函數，把 IDR_MAINMENU 功能表資源載入 MainMenu 物件（第 35 行），接著利用 CWnd::SetMenu()函數設定視窗使用的功能表為 MainMenu 功能表（第 53 行）。關於建立 CMenu 物件與功能表資源的方法，將於下節詳細說明。

```
'摘自 menu\menu.cpp 檔
16              Create(NULL, //產生標準視窗
17                  "Menu", //視窗標題
```

```
18              WS_OVERLAPPEDWINDOW | WS_VISIBLE, //視窗樣式
19              rectDefault, //視窗大小
20              NULL, //指向父視窗的指標
21              MAKEINTRESOURCE(IDR_SMENU)); //使用功能表的識別子
...          ......
29          //MyFrame::OnSwitchMenu()函數
30          afx_msg void OnSwitchMenu()
                        //當 Switch Menu 選項被選取時的回應函數
31          {
...             ......
35              MainMenu.LoadMenu(IDR_MAINMENU);    //載入功能表資源
...             ......
53              SetMenu(&MainMenu);    //設定應用程式所使用之功能表
54          }
...
161         //建立 MyFrame 類別的訊息映射表
162         BEGIN_MESSAGE_MAP(MyFrame, CFrameWnd)
163             ON_COMMAND(IDR_SwitchMenu, OnSwitchMenu)
...             ......                                    Switch Menu 選項
174         END_MESSAGE_MAP()                             的訊息回應項目
```

8-5 利用 CMenu 類別控制功能表

在程式內，若欲控制所使用的功能表時，必須建立一個 CMenu 物件，並以 CMenu::LoadMenu()函數連結 CMenu 物件與功能表資源。完成連結後，便可在程式內透過 CMenu 類別的成員函數操作功能表。以下將分成幾個主題來說明 CMenu 如何操作功能表。

功能表資源與 CMenu 的連結與分離

功能表資源的連結是利用 CMenu::LoadMenu()函數達成，menu 程式範例的 MyFrame::OnSwitchMenu()函數將分別把 IDR_MAINMENU、IDR_MAINMENU 功能表資源與 MainMenu、InsMenu 物件連結。

```
'摘自 menu\menu.cpp 檔
34          InsMenu.LoadMenu(IDR_INSMENU); //載入功能表資源
35          MainMenu.LoadMenu(IDR_MAINMENU); //載入功能表資源
```

相關函數說明如下：

BOOL CMenu::LoadMenu(LPCTSTR lpszResourceName)

BOOL CMenu::LoadMenu(UINT nIDResource)

□ 函數說明

該函數有兩種形式，一傳入功能表資源名稱，另一傳入功能表識別子。該函數若成功載入功能表資源，將傳回非零值，失敗則傳回零值。

◆ 參數說明

✓ LPCTSTR lpszResourceName
功能表名稱。

✓ UINT nIDResource
功能表的識別子。

BOOL CMenu::Attach(HMENU hMenu)

□ 函數說明

如果連結成功則傳回非零值，失敗則傳回零值。

◆ 參數說明

✓ HMENU hMenu
傳入功能表的標頭（handle）。

另一種連結功能表的方式則是利用 CMenu::Attach()函數，該函數可將已經存在的功能表連結至 CMenu 物件。當需要操作視窗物件目前所使用的功能表物件時，可以利用以下方式。

```
CMenu aMenu; //建立 CMenu 物件
HMENU aHMENU = ptrWnd->GetMenu( ); //將目前視窗所使用的功能表的標頭
aMenu.Attach(aHMENU ); //將 aMenu 物件與 aHMENU 連結
```

可以連結功能表，當然需要分離連結。為何要分離呢？這是因為若不將 CMenu 物件與相連結的功能表資源分離，就不能刪除 CMenu 物件。menu 程式範例將於 MyFrame::OnSwitchMenu()函數內連結 InsMenu 物件與 IDR_INSMENU 功能表資源。當執行 MyFrame::OnSwitchMenu()函數後，程式將刪除 InsMenu 物件。此時，若兩者資源與 CMenu 物件尚未分離程式將發生錯誤。所以，離開 MyFrame::OnSwitchMenu()函數前必須呼叫 CMenu::Detach()函數分離 InsMenu 物件與其連結的功能表資源。

```
'摘自 menu\menu.cpp 檔
51          InsMenu.Detach(); //將功能表資源與功能表物件分離
```

HMENU CMenu::Detach( )

□ **函數說明**

分離 CMenu 物件與其連結的功能表。成功將回傳該功能表的標頭，如果失敗則傳回 NULL。

功能表與選項的取得

當載入功能表列資源後，欲新增選項至特定功能表時，必須先取得該功能表的指標，然後才能操作。欲取得功能表指標時，可呼叫 CMenu::GetSubMenu()函數達成。MyFrame::OnSwitchMenu()函數中，欲將 Insert Item 選項插入 Dynamic Menu 功能表裡，因此，執行這個動作前，必須先取得 Dynamic Menu 功能表的指標。

```
'摘自 menu\menu.cpp 檔
40        SubMenu = MainMenu.GetSubMenu(1);//取得第 2 個功能表的指標
```

與 CMenu::GetSubMenu()函數類似的 CMenu::GetMenuItemID()函數，則用於取得功能表內選項的識別子，相關函數說明如下：

CMenu* CMenu::GetSubMenu(int nPos)

□ 函數說明

該函數將傳回功能表列中功能表的指標，如果該功能表不存在則傳回NULL。

◆ 參數說明

✓ int nPos
代表功能表位置的整數。功能表中第一個功能表的編號為 0，餘類推。

UINT CMenu::GetMenuItemID(int nPos)

□ 函數說明

如果成功取得該選項則傳回該選項的識別子。如果該選項為分隔線，則傳回 0，如果選項不存在則傳回-1。

◆ 參數說明

✓ int nPos
代表選項位置的整數。功能表中第一個選項的編號為 0。

選項與下一層功能表的新增

接著，將說明選項與功能表的新增。menu 程式範例的 MyFrame::OnSwitchMenu()函數將利用 CMenu::AppendMenu()函數與 CMenu::Insert()函數，新增 Color Menu 功能表（第 37 行），以及 Dynamic Menu 功能表的 Insert a Item、Insert a Submenu 選項與分隔線（第 42 至 46 行）。

```
'摘自 menu\menu.cpp 檔
29        //MyFrame::OnSwitchMenu()函數
30        afx_msg void OnSwitchMenu() //當 Switch Menu 選項被選取時的回應函數
31        {
32            CMenu InsMenu;//建立功能表物件
33
34            InsMenu.LoadMenu(IDR_INSMENU); //載入功能表資源
35            MainMenu.LoadMenu(IDR_MAINMENU); //載入功能表資源
```

```
36
37          MainMenu.AppendMenu(MF_POPUP, (UINT) InsMenu.m_hMenu,
38                                          "Color Menu"); //增加一個功能表
39
40          SubMenu = MainMenu.GetSubMenu(1); //取得第 2 個功能表的指標
41
42          SubMenu->AppendMenu(MF_SEPARATOR); //插入分隔線
43          SubMenu->InsertMenu(IDM_CreateItem,
                        MF_BYCOMMAND | MF_STRING,
44                      IDM_InsertItem, "Insert a Item"); //插入選項
45          SubMenu->InsertMenu(IDM_CreateItem,
                         MF_BYCOMMAND | MF_POPUP,
46                      (UINT) InsMenu.m_hMenu, "Insert a Submenu");
            //插入功能表
47          SetCheck(); //設定應被選取的選項
48
49          SubMenu->InsertMenu(IDM_CreateItem,
                                MF_BYCOMMAND | MF_SEPARATOR);
50                              //插入分隔線
51          InsMenu.Detach(); //將功能表資源與功能表物件分離
52
53          SetMenu(&MainMenu); //設定應用程式所使用之功能表
54      }
```

以下為 CMenu::InsertItem()函數與 CMenu::InsertMenu()函數的說明。

```
BOOL CMenu::InsertMenu( UINT nPosition, UINT nFlags,
            UINT nIDNewItem = 0, LPCTSTR lpszNewItem = NULL )

BOOL CMenu::InsertMenu( UINT nPosition, UINT nFlags,
                    UINT nIDNewItem, const CBitmap* pBmp )
```

□ **函數說明**

該函數有兩種形式，第一種可以將功能表項目加入功能表中。第二種則專門用於將 BMP 檔（圖形檔）插入功能表中。當插入成功時，將傳回非零值，反之傳回零值。

◆ **參數說明**

✓ UINT nPosition

指定一功能表項目，CMenu::InsertMenu()函數將會把功能表項目插入該項目前。指定的方式有兩種，一是指定選項在功能表中的位置，另一是指定選項的 ID。指定方式由 nFlags 參數控制。

✓ UINT nFlags

用於指定 CMenu::InsertMenu()函數的第一個參數 nPosition，其指定功能表項目的方式，以及欲插入的功能表項目與狀態將以下兩表所列之旗標來控制。下表為控制指定插入位置方式之旗標的說明：

表一、設定插入位置指定方式的旗標	
設定旗標	說　明
MF_BYPOSITION	nPosition 參數指定功能表項目的方式，是指定選項在功能表中位置。第一個功能表項目為 0，餘類推。
MF_BYCOMMAND	nPosition 參數指定功能表項目的方式，是指定選項的識別子。

指定插入功能表項目及其狀態為何的旗標共有四組旗標，欲同時使用各組旗標時，需以『|』運算子連結，但同一組旗標不可混用，下表將介紹各旗標。

表二、指定插入項目物件及狀態的旗標	
第一組：功能表項目的初始值狀態	
設定旗標	說　明
MF_DISABLED	將功能表項目設為無效
MF_ENABLED	將功能表項目設為有效
MF_GRAYED	將功能表項目的顏色設為灰色
第二組：指定插入的功能表項目	
設定旗標	說　明
MF_STRING	字串
MF_OWNERDRAW	圖形物件
MF_SEPARATOR	分隔線
MF_POPUP	功能表

表二、指定插入項目物件及狀態的旗標	
第三組：將功能表項目從插入的項目開始，從另一行排起行排起	
設定旗標	說　明
MF_MENUBARBREAK	從另一行排起時，中間以分隔線分開
MF_MENUBREAK	從另一行排起時，中間不以分隔線分開
第四組：將功能表項目打勾	
設定旗標	說　明
MF_CHECKED	將功能表項目打勾
MF_UNCHECKED	取消已打勾的功能表項目

✓ UINT nIDNewItem = 0
新功能表項目的 ID。如未輸入，預設值為 0，表無 ID。

✓ lpszNewItem = NULL
新功能表項目顯於功能表的選項名稱。如未輸入，預設值為 NULL，表無選項名稱。

✓ const CBitmap* pBmp
欲插入的圖形檔物件。

```
BOOL CMenu::AppendMenu(
UINT nFlags, UINT nIDNewItem = 0, LPCTSTR lpszNewItem = NULL )

BOOL CMenu::AppendMenu(
    UINT nFlags, UINT nIDNewItem, const CBitmap* pBmp )
```

□ **函數說明**

將項目新增至功能表最後。成功傳回非零值，反之傳回零值。

◆ **參數說明**

✓ UINT nFlags
指定增加的項目，及該項目的狀態。詳細的說明請參考前面 CMenu::InsertMenu()函數說明中，關於 nFlags 參數所使用旗標之表二的內容。

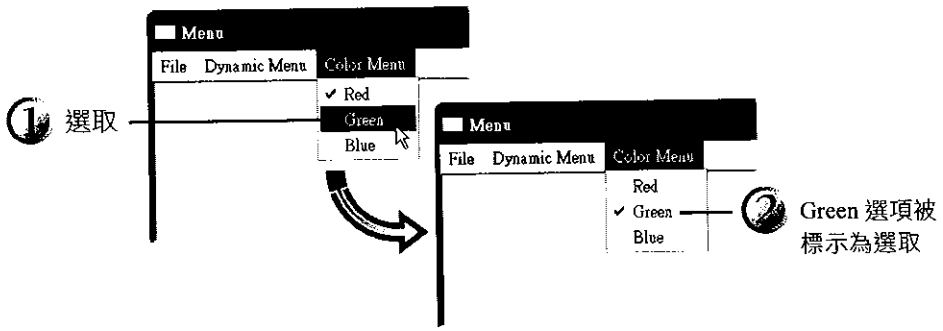
✓ UINT nIDNewItem = 0
新功能表項目的 ID。如未輸入，預設值為 0，表無 ID。

✓ LPCTSTR lpszNewItem = NULL
新功能表項目顯示於功能表的選項名稱。如未輸入，預設值為 NULL，表無選項名稱。

✓ const CBitmap* pBmp
欲插入的圖形檔物件。

滑鼠軌跡顏色的控制

本範例將利用 Color Menu 功能表控制滑鼠軌跡點的顏色，並同時利用功能表顯示目前所設定的軌跡點顏色。



因此，當使用者選擇 Color Memu 的選項後，必須執行以下三個動作。

一、**取消原先被打勾的選項**：將由 MyFrame::SetUnCheck()函數，完成這個動作。（第 89、96、103 行）

二、**依使用者的選擇更改軌跡點顏色**：直接設定 color 屬性的值。（第 90、97、104 行）

三、**將使用者選擇的選項打勾**：由 MyFrame::SetCheck()函數，完成選項的勾選動作。（第 91、98、105 行）

```
'摘自 menu\menu.cpp 檔
86      //MyFrame::OnRed()函數
87      afx_msg void OnRed()//當 Color Menu 中 Red 選項被選取時
88      {
89          SetUnCheck(); //取消原被打勾的選項
90          color = RGB(255,0,0); //設定畫在畫布上的顏色
91          SetCheck();//設定應被打勾的選項
92      }
93      //MyFrame::OnGreen()函數
94      afx_msg void OnGreen() //當 Color Menu 中 Green 選項被選取時
95      {
96          SetUnCheck();
97          color = RGB(0,255,0);
98          SetCheck();
99      }
100     //MyFrame::OnBlue()函數
101     afx_msg void OnBlue() //當 Color Menu 中 Blue 選項被選取時
102     {
103         SetUnCheck();
104         color = RGB(0,0,255);
105         SetCheck();
106     }
```

選項前的勾選符號

當 Color Menu 功能表控制選取滑鼠軌跡點顏色時，選項前勾選標記將更改，此更改動作將由 MyFrame::SetUnCheck() 函數與 MyFrame::SetCheck()函數共同完成。這兩個函數的內容如下：

```
'摘自 menu\menu.cpp 檔
107     //MyFrame::SetCheck()函數
108     void SetCheck() //判別 color 屬性，將 Color Menu 中對應的選項打勾
109     {
110         switch(color)
111         {
112         case RGB(255,0,0) :
113             MainMenu.CheckMenuItem(IDM_Red,MF_BYCOMMAND |
114                                    MF_CHECKED);//將該選項打勾
115             break;
116         case RGB(0,255,0) :
117             MainMenu.CheckMenuItem(IDM_Green,
```

```
                                        MF_BYCOMMAND|MF_CHECKED);
118            break;
119         case RGB(0,0,255) :
120            MainMenu.CheckMenuItem(IDM_Blue,
                                        MF_BYCOMMAND|MF_CHECKED);
121            break;
122         }
123      }
124      //MyFrame::SetUnCheck()函數
125      void SetUnCheck() //取消 Color Menu 中被打勾的選項
126      {
127         switch(color)
128         {
129         case RGB(255,0,0) :
130            MainMenu.CheckMenuItem(IDM_Red, MF_BYCOMMAND |
131                                    MF_UNCHECKED); //取消該選項打勾
132            break;
133         case RGB(0,255,0) :
134            MainMenu.CheckMenuItem(IDM_Green,
                                    MF_BYCOMMAND|MF_UNCHECKED);
135            break;
136         case RGB(0,0,255) :
137            MainMenu.CheckMenuItem(IDM_Blue,
                                    MF_BYCOMMAND|MF_UNCHECKED);
138            break;
139         }
140      }
```

MyFrame::SetUnCheck()函數與 MyFrame::SetCheck()函數兩者所執行的動作，其實是一樣的。都是利用 switch...case...敍述比對 color 屬性的值，然後，再決定 Color Menu 功能表中，哪一個選項前應出現或不出現勾選符號。因此，回應 Color Menu 功能表內選項的訊息處理函數中（如：回應 Red 選項的 MyFrame::OnRed() 成員函數），將先利用 MyFrame::SetUnCheck()函數依據目前 MyFrame::color 屬性的值，找出目前被打勾的選項，取消該選項前的勾選符號。再重新設定 MyFrame::color 屬性，再呼叫 MyFrame::SetCheck()函數找出需要被打勾的選項。而 MyFrame::SetCheck()函數與 MyFrame::SetUnCheck()函數中，設定選項勾選符號與取消勾選符號的動作將由 CMenu::CheckMenuItem()函數完成，此函數的詳細說明如後。

UINT CMenu::CheckMenuItem(UINT nIDCheckItem, UINT nCheck)

□ **函數說明**

用於設定或者取消功能表選項前勾選符號之函數。回傳值為該選項原來的狀態值。若原來已經有勾選符號則傳回 MF_CHECK，無勾選符號則傳回 MF_UNCHECK。

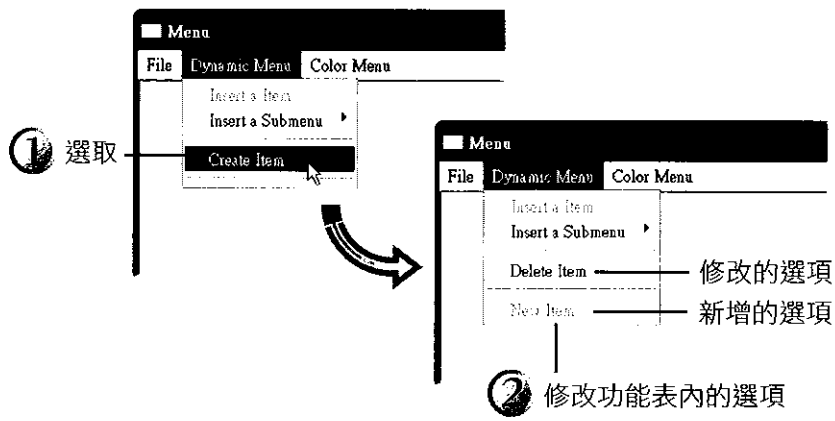
◆ **參數說明**

✓ UINT nIDCheckItem
欲設定勾選符號的選項之資源代號。

✓ UINT nCheck
指定第一個參數指定選項的方法，以及更改該選項狀態的方式為打勾或者取消打勾。指定方法的參數旗標為 MF_BYCOMMAND 或 MF_BYPOSITION，這兩個旗標的用途，請參考 CMenu::InsertMenu()函數說明中有關 nFlags 參數之說明的表一。控制選項狀態的旗標有 MF_CHECK（勾選）與 MF_UNCHECK（取消勾選）。傳入此參數時，兩種旗標需以『|』符號連結。

選項的修改與刪除

選取 Dynamic Menu 的 Create Item 選項後，將在 Dynamic Menu 功能表新增 New Item 選項，並將原先的 Create Item 選項名稱更改為 Delete Item 選項。



當選取 Create Item 選項時，將呼叫 MyFrame::OnCreateItem()函數，該函數將呼叫 CMenu::ModifyMenu()函數把被選取的 Create Item 選項之名稱改為 Delete Item 選項，ID 為 IDM_DeleteItem（第 58、59 行）。接著呼叫 CMenu::AppendMenu()函數於功能表尾端新增 New Item 選項（第 61 行）。當選取 Delete Item 選項時，將執行與 Create Item 選項相反的動作，將 Delete Item 選項名稱改回 Create Item 選項，ID 改回 IDM_CreateItem（第 67、68 行）。然後，呼叫 CMenu::DeleteMenu()函數刪除 New Item 選項（第 70 行）。

```
'摘自 menu\menu.cpp 檔
55          //MyFrame::OnCreateItem()函數
56          afx_msg void OnCreateItem()
57          { //當 Dynamic Menu 功能表的 Create Item 選項被選取時的回應函數
58             MainMenu.ModifyMenu(IDM_CreateItem,MF_BYCOMMAND,
59                                    IDM_DeleteItem,"Delete Item");
                  //修改功能表中的 Create Item 選項爲 Delete Item
60             SubMenu = MainMenu.GetSubMenu(1);
                  //取得第 2 個功能表的指標
61             SubMenu->AppendMenu(MF_STRING, IDM_NewItem, "New Item");
62                //增加 Create item 選項
63          }
64          //MyFrame::OnDeleteItem()函數
65          afx_msg void OnDeleteItem()
66          { //當 Dynamic Menu 功能表的 Delete Item 選項被選取時的回應函數
67             MainMenu.ModifyMenu(IDM_DeleteItem,MF_BYCOMMAND,
68                                 IDM_CreateItem,"Create Item");
                               //修改 Delete Item 選項爲 Create Item 選項
69             SubMenu = MainMenu.GetSubMenu(1);
               //取得第 2 個功能表的指標
70             SubMenu->DeleteMenu(IDM_NewItem, MF_BYCOMMAND);
               //刪除 New Item 選項
71          }
…
161         //建立 MyFrame 類別的訊息映射表
162         BEGIN_MESSAGE_MAP(MyFrame, CFrameWnd)
163            …
164            ON_COMMAND(IDM_CreateItem, OnCreateItem)
```

```
165             ON_COMMAND(IDM_DeleteItem, OnDeleteItem)
...             ...
174         END_MESSAGE_MAP()
```

以下將說明 CMenu::ModifyMenu()函數與 CMenu::DeleteMenu()函數：

```
BOOL CMenu::ModifyMenu( UINT nPosition, UINT nFlags,
            UINT nIDNewItem = 0, LPCTSTR lpszNewItem = NULL )

BOOL CMenu::ModifyMenu( UINT nPosition, UINT nFlags,
            UINT nIDNewItem, const CBitmap* pBmp )
```

□ 函數說明

本函數有兩種形式，第一種用於修改功能表的一般選項，如：功能表、文字… 等。第二種則專門用於修改功能表中的圖形物件。當修改功能表成功時，將傳回非零值，失敗則傳回零值。

◆ 參數說明

- ✓ UINT nPosition
  指定欲修改的功能表項目，指定的方式有兩種，一是指定選項在功能表中的位置，另一是指定選項的 ID。指定方式由 nFlags 參數控制。
- ✓ UINT nFlags
  利用旗標指定 CMenu::ModifyMenu()函數的第一個參數 nPosition，其指定功能表項目的方式為何？以及欲插入的功能表項目及其狀態為何？詳細說明請參考 CMenu::InsertMenu()函數說明中 nFlags 參數的介紹。
- ✓ UINT nIDNewItem = 0
  修改後的功能表項目 ID。如未輸入，預設值為 0，表不修改 ID。
- ✓ LPCTSTR lpszNewItem = NULL
  修改後的功能表項目名稱。如未輸入預設值為 NULL，表不修改項目名稱。
- ✓ const CBitmap* pBmp
  欲修改為 BMP 圖檔物件。

BOOL CMenu::DeleteMenu(UINT nPosition, UINT nFlags)

□ 函數說明

刪除功能表選項，成功則傳回非零值，反之傳回零值。

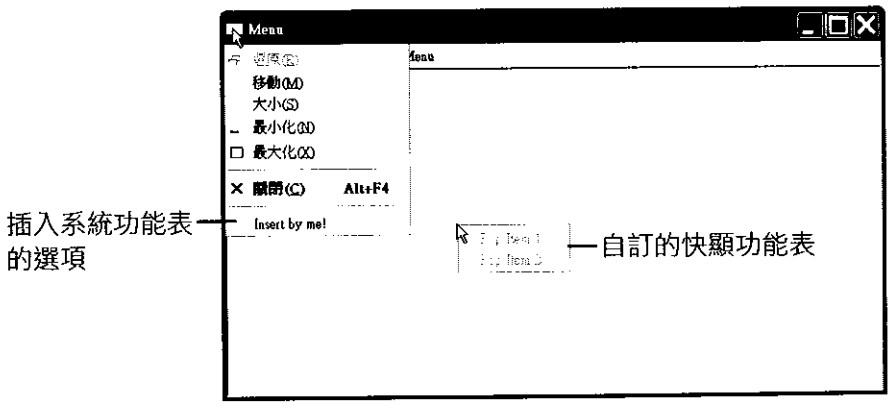
◆ 參數說明

✓ UINT nPosition
指定欲刪除的功能表項目，指定的方式有兩種，一是指定選項在功能表中的位置，另一是指定選項的 ID。指定方式由 nFlags 參數控制。

✓ UINT nFlags
利用旗標指定 CMenu::ModifyMenu()函數的第一個參數 nPosition，其指定功能表項目的方式為何？詳細的參數旗標說明請參考 CMenu::InsertMenu()函數說明中有關 nFlags 參數之說明的表一。

8-6 修改系統功能表與建立快顯功能表

menu 程式範例將示範如何在系統功能表新增一個選項，以及建立快顯功能表，如下圖所示。



系統功能表的修改

利用 MFC 建立視窗程式時，完成建立的視窗框架將具備系統功能表。若欲修改系統功能表時，必須先呼叫 CWnd::GetSystemMenu()函數取得系統功能表，再利用 8-5 節介紹的 CMenu 類別成員函數修改功能表。

menu 程式範例對於系統功能表的修改動作，執行於 MyFrame 的建構子內。第 23 行取得系統功能表的指標後，將利用 CMenu::AppendMenu()函數插入分隔線（第 24 行）與 Insert by me!選項（第 25 行）。

```
'摘自 menu\menu.cpp 檔
14        MyFrame() //建構子
15        {
…             ……
23            SysMenu = GetSystemMenu(FALSE); //取得系統功能表
24            SysMenu->AppendMenu(MF_SEPARATOR); //插入分隔線
25            SysMenu->AppendMenu(MF_STRING, IDM_INSERT, "Insertbyme!");
26            //插入選項
27            …
28        }
```

CMenu* CWnd::GetSystemMenu(BOOL bRevert) const

□ 函數說明

若傳入參數為 FALSE 時，將傳回目前執行之視窗程式的系統功能表指標。

◆ 參數說明

✓ BOOL bRevert
當傳入參數為 FALSE 時，將傳回目前執行之視窗程式的系統功能表指標。若傳入 TRUE，則傳回未定值。

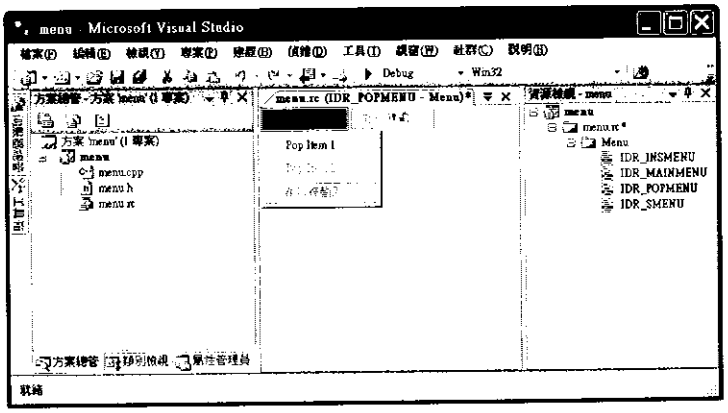
快顯功能表的建立

執行於 Windows 作業系統的視窗程式，當在視窗程式畫面按下滑鼠右鍵時，將產生快顯功能表。由 MFC 建立的視窗程式裡，當按下滑鼠右

鍵時，將產生 ON_WM_CONTEXTMENU 訊息，呼叫由衍生 CWnd 類別之類別內所覆寫的 OnContextMenu()函數。以下為此範例內，MyFrame 類別所覆寫的 OnContextMenu()函數之內容（第 73 至 85 行）。

```
'摘自 menu\menu.cpp 檔
72      //MyFrame::OnContextMenu()函數
73      afx_msg void OnContextMenu(CWnd* pWnd, CPoint point)
74      {  //按下滑鼠右鍵時，將執行此成員函數產生快顯功能表
75         CMenu menu;
76
77         menu.LoadMenu(IDR_POPMENU); //載入功能表資源
78
79         PopMenu = menu.GetSubMenu(0);//取得第 1 個功能表的指標
80
81         PopMenu->TrackPopupMenu(
82            TPM_CENTERALIGN|TPM_RIGHTBUTTON,point.x,point.y,
83            this); //建立快顯功能表
84         menu.Detach();
85      }
```

欲建立快顯功能表時，必須在該函數利用 CMenu::TrackPopupMenu() 函數建立快顯功能表（第 81 至 83 行）。需要說明的是，建立快顯功能表所需要的功能表資源，必須為沒有名稱的功能表，如下圖所示（建立方式請參考 7-2 節的說明）。



呼叫 CMenu::TrackPopupMenu()函數建立快顯功能表前，必須建立一個 CMenu 物件，並與做為快顯功能表之功能表資源連結（第 77 行），並由 CMenu::GetSubMenu()函數取得該功能表中的第一個功能表（第 79 行），也就是沒有名稱的功能表。完成上述工作後，才能呼叫 CMenu::TrackPopupMenu()函數建立快顯功能表（第 81 至 83 行）。

```
'摘自 menu\menu.cpp 檔
73          afx_msg void OnContextMenu(CWnd* pWnd, CPoint point)
74          {   //按下滑鼠右鍵時，將執行此成員函數產生快顯功能表
75              CMenu menu;
76
77              menu.LoadMenu(IDR_POPMENU); //載入功能表資源
78
79              PopMenu = menu.GetSubMenu(0); //取得第 1 個功能表的指標
80
81              PopMenu-> TrackPopupMenu(
82                  TPM_CENTERALIGN|TPM_RIGHTBUTTON,point.x,point.y,
83                  this); //建立快顯功能表
84              menu.Detach();
85          }
…           ……
161         //建立 MyFrame 類別的訊息映射表
162         BEGIN_MESSAGE_MAP(MyFrame, CFrameWnd)
…               ……
169             ON_WM_CONTEXTMENU()
                //按下滑鼠右鍵時，將呼叫 OnContextMenu
…               ……
174         END_MESSAGE_MAP()
```