2023327100056

李凯涛

2025年3月26日周三

### 1. Windows 中的资源包括哪些？

Windows 中的资源是指嵌入到可执行文件中的数据，主要包括：

- 菜单（Menu）

- 图标（Icon）

- 光标（Cursor）

- 位图（Bitmap）

- 对话框（Dialog）

- 字符串表（String Table）

- 加速键表（Accelerator Table）

- 版本信息（Version Information）

- 工具栏（Toolbar）

- 自定义资源（Custom Resources）

这些资源通常在资源文件（`.rc` 文件）中定义。

---

### 2. 菜单有哪些部分组成？

菜单通常由以下部分组成：

- \*\*菜单栏\*\*：位于窗口顶部的水平条，包含顶级菜单项（如“文件”、“编辑”）。

- \*\*顶级菜单项\*\*：菜单栏上的主菜单项。

- \*\*子菜单（Popup 菜单）\*\*：顶级菜单项的下拉菜单。

- \*\*菜单项\*\*：子菜单中的具体选项（如“打开”、“保存”）。

- \*\*分隔符\*\*：用于分隔菜单项的水平线。

- \*\*快捷键\*\*：与菜单项关联的加速键（如 Ctrl+S）。

---

### 3. 怎样定义菜单？

菜单通常在资源文件（`.rc` 文件）中定义，使用 `MENU` 资源描述。例如：

```rc

IDR\_MAINMENU MENU

BEGIN

POPUP "&File"

BEGIN

MENUITEM "&New", ID\_FILE\_NEW

MENUITEM "&Open", ID\_FILE\_OPEN

MENUITEM SEPARATOR

MENUITEM "E&xit", ID\_FILE\_EXIT

END

POPUP "&Help"

BEGIN

MENUITEM "&About", ID\_HELP\_ABOUT

END

END

```

- `IDR\_MAINMENU` 是菜单的资源 ID。

- `POPUP` 定义子菜单。

- `MENUITEM` 定义菜单项。

---

### 4. Popup 是什么样的菜单？定义的形式是什么？其中的“&”的作用是什么？

- \*\*Popup 菜单\*\*：也叫弹出菜单或子菜单，是顶级菜单项的下拉菜单，通常在点击顶级菜单项时显示。

- \*\*定义形式\*\*：在资源文件中使用 `POPUP` 关键字，例如：

```rc

POPUP "&File"

BEGIN

MENUITEM "&New", ID\_FILE\_NEW

END

```

- \*\*“&”的作用\*\*：`&` 用于指定快捷键（助记符）。例如，`&File` 表示用户可以按 `Alt+F` 打开“File”菜单，`&New` 表示在“File”菜单打开后按 `N` 选择“New”。

---

### 5. 怎么样定义菜单项？

菜单项在资源文件中使用 `MENUITEM` 关键字定义，格式为：

```rc

MENUITEM "菜单项名称", 菜单项ID

```

示例：

```rc

MENUITEM "&Save", ID\_FILE\_SAVE

```

- `"&Save"`：菜单项显示的文本，`&` 指定快捷键。

- `ID\_FILE\_SAVE`：菜单项的唯一 ID。

---

### 6. 菜单的响应是通过什么消息响应的？

菜单的响应通过 `WM\_COMMAND` 消息处理：

- 当用户点击菜单项时，Windows 发送 `WM\_COMMAND` 消息到窗口过程。

- `wParam` 的低位 (`LOWORD(wParam)`) 包含菜单项的 ID。

- 在 `WndProc` 中处理：

```cpp

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case ID\_FILE\_NEW:

// 处理“新建”菜单项

break;

}

break;

```

---

### 7. 菜单项的 ID 标识符是不是唯一确定的？

是的，菜单项的 ID 标识符必须是\*\*唯一\*\*的。ID 用于区分不同的菜单项，Windows 通过 ID 确定用户选择了哪个菜单项。如果 ID 不唯一，会导致消息处理混乱。

---

### 8. 菜单项中的水平分割符用不用定义 ID 标识符？

不需要。水平分割符使用 `MENUITEM SEPARATOR` 定义，不需要 ID，因为它只是用于视觉分隔，不响应用户操作。例如：

```rc

MENUITEM SEPARATOR

```

---

### 9. 菜单资源的描述是在什么文件中做定义的？

菜单资源在\*\*资源文件\*\*（`.rc` 文件）中定义。资源文件描述了菜单、图标、对话框等资源的外观和结构。

---

### 10. 菜单资源中的 ID 标识符对应的数值在什么文件中定义？

菜单资源中的 ID 标识符对应的数值在\*\*资源头文件\*\*（通常是 `resource.h`）中定义。例如：

```cpp

#define ID\_FILE\_NEW 1001

#define ID\_FILE\_OPEN 1002

```

这些 ID 由开发者定义或由 Visual Studio 自动生成。

---

### 11. 怎样加载菜单资源？

菜单资源通过 `LoadMenu` 函数加载，并将其附加到窗口：

1. 加载菜单：

```cpp

HMENU hMenu = LoadMenu(hInstance, MAKEINTRESOURCE(IDR\_MAINMENU));

```

2. 将菜单附加到窗口：

```cpp

SetMenu(hWnd, hMenu);

```

- `hInstance`：应用程序实例句柄。

- `IDR\_MAINMENU`：菜单资源 ID。

---

### 12. 怎样禁止或激活菜单项？

使用 `EnableMenuItem` 函数：

- 禁止菜单项：

```cpp

EnableMenuItem(hMenu, ID\_FILE\_SAVE, MF\_BYCOMMAND | MF\_GRAYED);

```

- 激活菜单项：

```cpp

EnableMenuItem(hMenu, ID\_FILE\_SAVE, MF\_BYCOMMAND | MF\_ENABLED);

```

- `hMenu`：菜单句柄。

- `ID\_FILE\_SAVE`：菜单项 ID。

- `MF\_BYCOMMAND`：按 ID 操作。

- `MF\_GRAYED`：禁用并置灰；`MF\_ENABLED`：启用。

---

### 13. 怎样设置或取消选中标志？

使用 `CheckMenuItem` 函数：

- 设置选中标志：

```cpp

CheckMenuItem(hMenu, ID\_VIEW\_OPTION, MF\_BYCOMMAND | MF\_CHECKED);

```

- 取消选中标志：

```cpp

CheckMenuItem(hMenu, ID\_VIEW\_OPTION, MF\_BYCOMMAND | MF\_UNCHECKED);

```

- `MF\_CHECKED`：选中；`MF\_UNCHECKED`：取消选中。

---

### 14. 怎样增加菜单项？并举例理解该使用的函数的参数。

使用 `AppendMenu` 函数增加菜单项：

```cpp

AppendMenu(hMenu, MF\_STRING, ID\_FILE\_NEW, L"&New");

```

- `hMenu`：目标菜单句柄。

- `MF\_STRING`：菜单项类型（字符串类型）。

- `ID\_FILE\_NEW`：菜单项 ID。

- `L"&New"`：菜单项显示的文本。

---

### 15. 怎样在菜单中插入菜单项？

使用 `InsertMenu` 函数：

```cpp

InsertMenu(hMenu, 0, MF\_BYPOSITION | MF\_STRING, ID\_FILE\_OPEN, L"&Open");

```

- `hMenu`：目标菜单句柄。

- `0`：插入位置（按位置插入，0表示第一个位置）。

- `MF\_BYPOSITION | MF\_STRING`：按位置插入，类型为字符串。

- `ID\_FILE\_OPEN`：菜单项 ID。

- `L"&Open"`：菜单项文本。

---

### 16. 怎样删除菜单项？

使用 `DeleteMenu` 函数：

```cpp

DeleteMenu(hMenu, ID\_FILE\_OPEN, MF\_BYCOMMAND);

```

- `hMenu`：菜单句柄。

- `ID\_FILE\_OPEN`：要删除的菜单项 ID。

- `MF\_BYCOMMAND`：按 ID 删除。

---

### 17. 怎样修改菜单项？

使用 `ModifyMenu` 函数：

```cpp

ModifyMenu(hMenu, ID\_FILE\_NEW, MF\_BYCOMMAND | MF\_STRING, ID\_FILE\_NEW, L"&New File");

```

- `hMenu`：菜单句柄。

- `ID\_FILE\_NEW`：要修改的菜单项 ID。

- `MF\_BYCOMMAND | MF\_STRING`：按 ID 修改，类型为字符串。

- `ID\_FILE\_NEW`：新 ID（可保持不变）。

- `L"&New File"`：新文本。

---

### 18. 怎样动态创建菜单？

使用 `CreateMenu` 和 `CreatePopupMenu` 动态创建菜单：

```cpp

HMENU hMenu = CreateMenu(); // 创建主菜单

HMENU hSubMenu = CreatePopupMenu(); // 创建子菜单

AppendMenu(hSubMenu, MF\_STRING, ID\_FILE\_NEW, L"&New");

AppendMenu(hMenu, MF\_POPUP, (UINT\_PTR)hSubMenu, L"&File");

SetMenu(hWnd, hMenu); // 附加到窗口

```

---

### 19. 理解怎样在资源文件中定义加速键，常用的加速键有哪两种形式？并举例

- \*\*定义加速键\*\*：在 `.rc` 文件中使用 `ACCELERATORS` 资源定义：

```rc

IDR\_ACCELERATOR ACCELERATORS

BEGIN

"N", ID\_FILE\_NEW, VIRTKEY, CONTROL // Ctrl+N

"S", ID\_FILE\_SAVE, VIRTKEY, CONTROL // Ctrl+S

VK\_F1, ID\_HELP\_ABOUT, VIRTKEY // F1

END

```

- \*\*常用加速键形式\*\*：

1. \*\*组合键\*\*：如 `Ctrl+N`（`VIRTKEY, CONTROL`）。

2. \*\*功能键\*\*：如 `F1`（`VIRTKEY`）。

- \*\*示例\*\*：`Ctrl+N` 触发 `ID\_FILE\_NEW`，`F1` 触发 `ID\_HELP\_ABOUT`。

---

### 20. 为什么要用加速键？

加速键可以提高操作效率：

- 允许用户通过快捷键快速执行常用操作。

- 减少鼠标操作，提升用户体验。

- 适合专业用户或需要频繁操作的场景。

---

### 21. 怎样加载加速键资源？

使用 `LoadAccelerators` 函数：

```cpp

HACCEL hAccel = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDR\_ACCELERATOR));

```

- `hInstance`：应用程序实例句柄。

- `IDR\_ACCELERATOR`：加速键资源 ID。

---

### 22. 怎样翻译加速键？

在消息循环中使用 `TranslateAccelerator` 函数：

```cpp

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(hWnd, hAccel, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

```

- `hWnd`：窗口句柄。

- `hAccel`：加速键表句柄。

- `msg`：消息结构体。

- `TranslateAccelerator` 将键盘输入转换为 `WM\_COMMAND` 消息。

---