

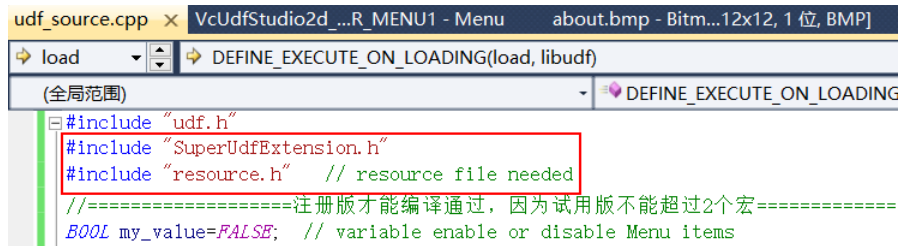
## SuperUDF 拓展库编程手册 23.2

[打开 English Version](#)

VC++ Udf Studio 将一些常用功能以第三方库的形式封装好供用户直接调用，从而大大拓展 UDF 的功能。如下图所示，如需调用拓展的第三方函数，用户只需要将如下两句语句去掉注释。

```
#include "SuperUdfExtension.h"
```

```
#include "resource.h" // resource file needed for user menu, 若插入用户菜单需要此头文件
```



### 拓展函数详解：

#### 1. void SuperUdf\_Initialize(HMODULE hLibudfDllModule)

hLibudfDllModule——udf 库的模块句柄，可用 AfxGetInstanceHandle() 获取。

返回值——void

**函数说明：**此函数负责拓展库的初始化，hLibudfDllModule 为 udf 库的模块句柄，可以使用 AfxGetInstanceHandle() 获取（参见实例一）。

**注意：**此函数必须在调用其它拓展函数前调用。较佳的调用该函数的地方是 DEFINE\_EXECUTE\_ON\_LOADING 宏中。

#### 2. int SuperUdf\_GetZoneIdByName(char\* strZoneName)

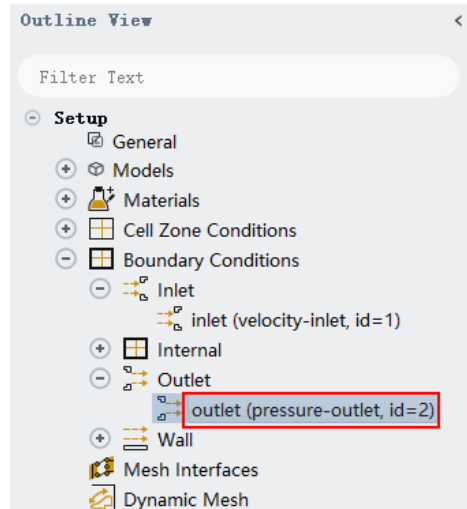
strZoneName——zone 或边界的名字

返回值——int，对应的 zone ID

**函数说明：**根据 zone 或边界的名字 strZoneName 来获取对应的 zone ID，如下图所示的 case 的 UDF 源码中调用 SuperUdf\_GetZoneIdByName("outlet") 函数将返回 2，参见实例一。

该函数主要用来解决源码通用性问题，由于 UDF 中经常会调用 Lookup\_Thread(domain, zone\_ID) 来获取 Thread，里面的 zone\_ID 是关键参数，但是会随着网格变化而变化，很多用户每次更换网格后只能手动查到对应 zone ID 后修改源码再重新编译来实现，十分不方便。有了此函数后只需要画网格时取固定名字，该函数会自动根据名字获取其 ID。

**注意：**此函数只能在 serial 或 host 上调用，node 上调用会返回-1。但可以在 serial 或 host 上获取以后再调用 host\_to\_node\_int 系列函数将值传给 node。



### 3. void SuperUdf\_Interrupt();

返回值——无

**函数说明：**该函数用来中断正在进行的稳态或非稳态迭代。你可以将该语句放在 `DEFINE_ADJUST` 或 `DEFINE_EXECUTE_AT_END` 中，这样一旦达到你要的标准你就可以马上调用该函数停止迭代计算。

### 4. HWND SuperUdf\_GetFluentMainWnd();

返回值——Fluent 的主窗体句柄

**函数说明：**获得 Fluent 的主窗体句柄。例如，`MessageBox` 函数需要父窗口输入参数，此时可以采用 `SuperUdf_GetFluentMainWnd()` 函数获取 Fluent 的主窗体句柄作为其父窗口。

**示例：**

```
DEFINE_ON_DEMAND(msgbox)
{
    HWND hFluentWnd=SuperUdf_GetFluentMainWnd();
    ::MessageBox(hFluentWnd, "Test", "Information", MB_OK); // MessageBox is a Windows API function
}
```

### 5. void SuperUdf\_Steady\_Iterate(int nTimes)

nTimes——稳态迭代的步数

返回值——无

**函数说明：**驱动 Fluent 进行稳态迭代 n 步，只可用于稳态计算，试用版只允许迭代 1 步。

**示例：**

```
SuperUdf_Steady_Iterate(1000); //drive Fluent to iterate 1000 steps
```

### 6. void SuperUdf\_ExecuteConsoleCommand(char\* strAnsiConsoleCommand)

strAnsiConsoleCommand——TUI 或 Scheme 命令

返回值——无

**函数说明：**驱动 Fluent 执行 TUI 或 Scheme 命令，试用版不可用。

**示例：**

```
SuperUdf_ExecuteConsoleCommand("/solve/dual-time-iterate 1000 20");
```

如上语句为调用 TUI 实例，可驱动 Fluent 进行非稳态迭代，1000 时间步，每步 20 次迭代。

```
SuperUdf_ExecuteConsoleCommand("(write-case \"d:\\test.cas\")");
```

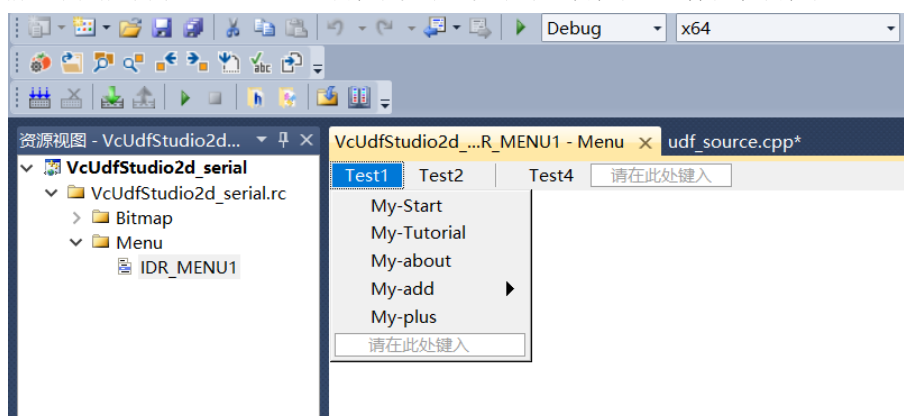
如上语句为调用 Scheme 实例，可驱动 Fluent 将当前 case 保存为 d:\\test.cas。

## 7. void SuperUdf\_AddUserMenu(UINT uMenuResourceID)

uMenuResourceID——菜单资源号

返回值——无

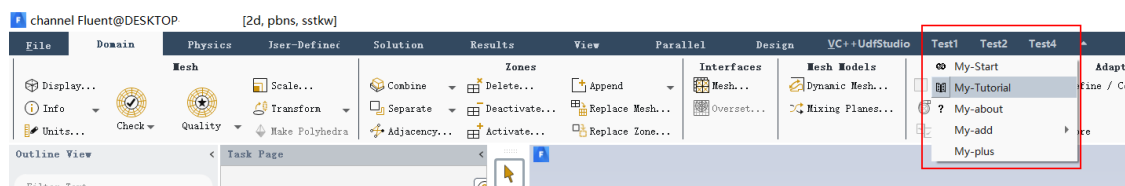
**函数说明：**在 Fluent 中插入用户菜单，当然，你首先需要在 Visual Studio 中用菜单设计器设计一个菜单<sup>[1-2]</sup>，将其资源号作为本函数的参数，则菜单的每个子菜单将被复制到 Fluent 中，插入位置为 Help 菜单之前。下图所示为 Visual Studio 中菜单设计器设计的一个菜单，包含若干子菜单。



**示例：**

```
SuperUdf_AddUserMenu(IDR_MENU1);
```

如上语句将菜单设计器中 IDR\_MENU1 的整个菜单插入到 Fluent 中，如下所示（图标需要后面的另外函数添加）。



## 8. void SuperUdf\_EnableMenuItem(UINT uTargetMenuID, BOOL bEnabled)

uTargetMenuID——子菜单的 ID 号

bEnabled——TRUE 为启用，FALSE 为变灰禁用

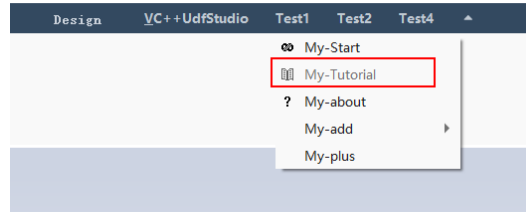
返回值——无

**函数说明：**将某个用户菜单项变灰禁止或恢复可用。

**示例：**

```
SuperUdf_EnableMenuItem(ID_TEST1_TUTORIAL, FALSE);
```

如上语句执行后，对应的 My-Tutorial 菜单变灰禁用。



9. void SuperUdf\_SetMenuBmpAndFun(MenuItemBmpAndFun menuBmpAndFuns[], ULONG nCount)  
menuBmpAndFuns——MenuItemBmpAndFun 类型的数组，MenuItemBmpAndFun 为结构体，声明如下

```
typedef struct
{
    UINT MenuItemID; //menu item ID
    HBITMAP hBitmap; //menu bitmap handle
    void(*fcn)(); // pointer to a menu item response function returning void
}MenuItemBmpAndFun;
```

其中，MenuItemID——菜单项的 ID

hBitmap——菜单图标句柄，可以用 LoadBitmap 函数加载图标资源

fcn——菜单响应函数的指针，该函数必须无返回值。

nCount——MenuItemBmpAndFun 数组的大小。

返回值——无

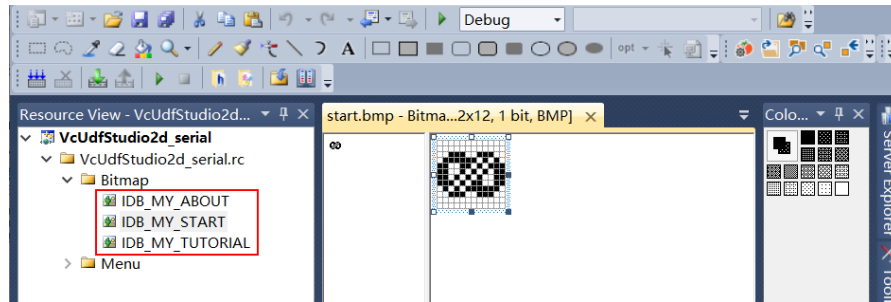
**函数说明：**设置用户菜单对应的位图和点击响应函数，图标如果不需要，可以 hBitmap 参数设为 NULL，响应函数必须设置，否则点击菜单将无响应。**注意：**必须在 SuperUdf\_AddUserMenu 之前调用。

**示例：**

```
void OnClickStart();
void OnClickTutorial();
void OnClickAbout();
void OnClick4();

DEFINE_EXECUTE_ON_LOADING(load, libudf)
{
    HMODULE hModule=AfxGetInstanceHandle();
    SuperUdf_Initialize(hModule); //Call this before any other SuperUdf functions
    MenuItemBmpAndFun bmpAndFun[]={
        {ID_TEST1_START,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_START)), OnClickStart},
        {ID_TEST1_TUTORIAL,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_TUTORIAL)), OnClickTutorial},
        {ID_TEST1_ABOUT,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_ABOUT)), OnClickAbout},
        {ID_TEST4, NULL, OnClick4} // if no bitmap, use NULL
    };
    SuperUdf_SetMenuBmpAndFun(bmpAndFun, sizeof(bmpAndFun)/sizeof(MenuItemBmpAndFun));
    SuperUdf_AddUserMenu(IDR_MENU1);
};
```

如上语句即可添加图标，图标资源号为 IDB\_MY\_START, IDB\_MY\_TUTORIAL, IDB\_MY\_ABOUT, 请在 Visual Studio 的资源管理器里面添加图标资源。OnClickStart, OnClickTutorial, OnClickAbout 分别为对应的响应函数。



#### 10. void SuperUdf\_SetMenuSelectCallBack(MENUSELECTPROC UserCallBackFunction)

UserCallBackFunction——选择菜单时触发的回调函数，MENUSELECTPROC 类型，该类型声明为  
typedef void (CALLBACK\* MENUSELECTPROC)(), 实质为空返回值的函数指针类型。

返回值——无

**函数说明:** 该函数主要用来设置选择菜单时响应回调函数，可在此回调函数中动态使菜单变灰或恢复，其它用途未经测试，请勿使用。

**注意:** 严禁在串行版 Fluent 的该回调函数中调用 Message, Message0 或者 CX\_Message 函数，可能会损坏串行版 Fluent，导致无法卸载 UDF 库。并行版本不受此限制。

**示例:**

```
void CALLBACK MenuSelectProc();
DEFINE_EXECUTE_ON_LOADING(load, libudf)
{
    HMODULE hModule=AfxGetInstanceHandle();
    SuperUdf_Initialize(hModule);
    .....
    SuperUdf_AddUserMenu(IDR_MENU1);
    SuperUdf_SetMenuSelectCallBack(MenuSelectProc);
}
void CALLBACK MenuSelectProc()
{
    if(my_value==TRUE) // my_value is a variable to judge
        SuperUdf_EnableMenuItem(ID_TEST4, TRUE);
    else
        SuperUdf_EnableMenuItem(ID_TEST4, FALSE);
}
```

## 实例一、根据 **zone** 或边界名称获取对应 ID 号

```
#include "udf.h"
#include "SuperUdfExtension.h"

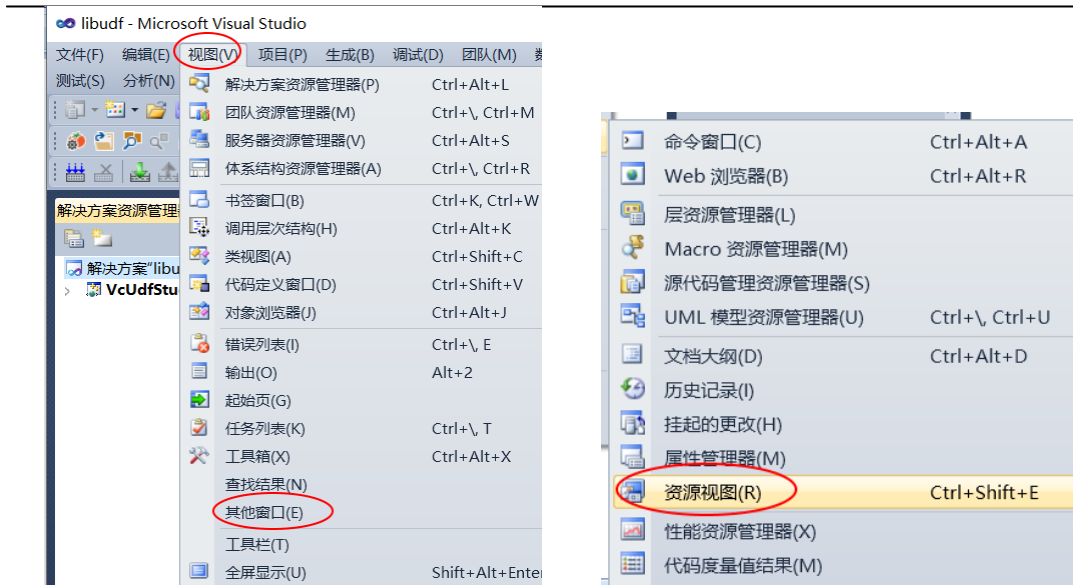
DEFINE_ON_DEMAND(GetOutletId)
{
    int outlet_id;
    face_t f;
    Thread* tf;
    Domain* domain=Get_Domain(1);
#ifdef IRP_NODE
    outlet_id=SuperUdf_GetZoneIdByName("outlet"); //get the id of zone whose name is "outlet"
#endif
    host_to_node_int_1(outlet_id);

#ifdef IRP_HOST
    if(-1==outlet_id)
        Message("Can't get the ID on myid=%d\n",myid);
    else
    {
        tf=Lookup_Thread(domain, outlet_id);
        Message("myid=%d, outlet id=%d\n",myid, outlet_id);
        begin_f_loop(f,tf)
        {
            if(PRINCIPAL_FACE_P(f,tf))
            {
                // loop over faces on "outlet"
            }
        }
        end_f_loop(f,tf)
    }
#endif
}

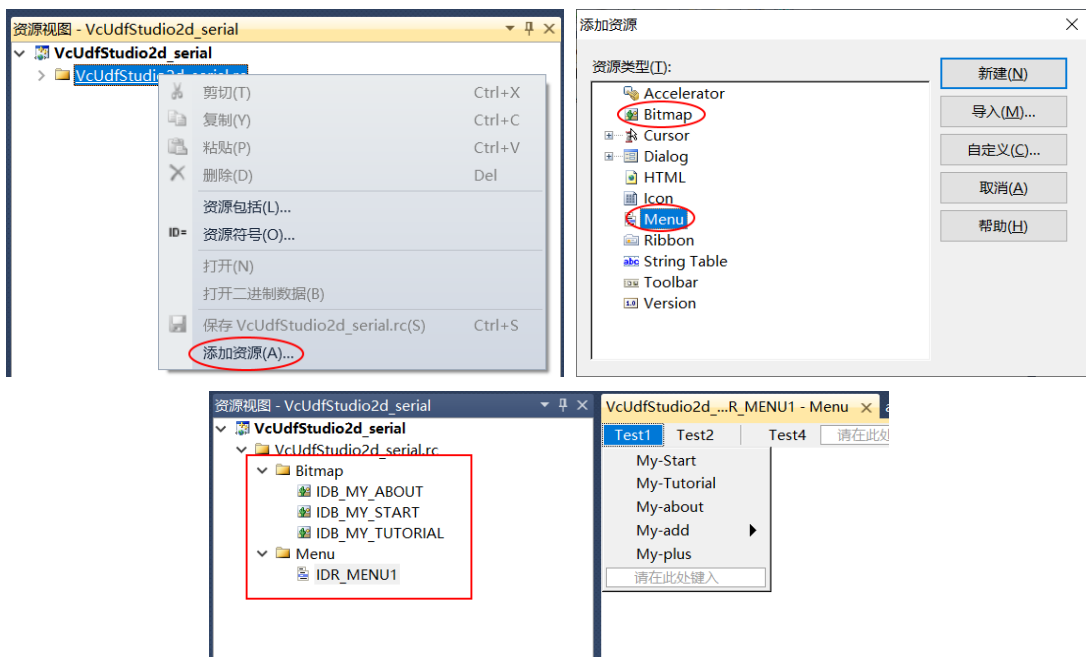
DEFINE_EXECUTE_ON_LOADING(load,libudf)
{
    SuperUdf_Initialize(AfxGetInstanceHandle());
}
```

## 实例二、用户自定义菜单的添加

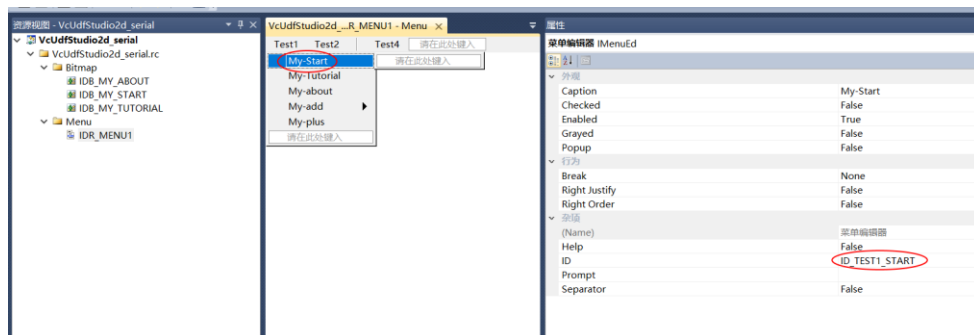
首先，打开 Visual studio 中的资源视图，如未发现，可以在“视图”菜单->“其他窗口”->“资源视图”调取出来。



进入资源视图后，利用在项目上右键弹出菜单，点击“添加资源”，首先添加三个位图，名称分别取名为 IDB\_MY\_START, IDB\_MY\_TUTORIAL, IDB\_MY\_ABOUT。可以新建也可以导入现有存在的位图，建议用黑白双色的位图，尺寸为 12\*12 像素。



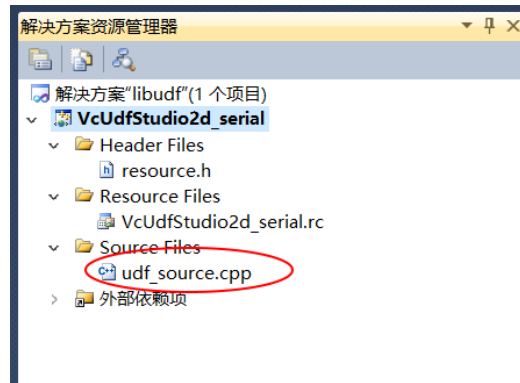
完成后再添加一个菜单资源，名字为 IDR\_MENU1，子菜单项 ID 可以在右侧菜单编辑器属性管理器中修改。如果属性管理器处于隐藏状态，可以双击菜单项使其显示。



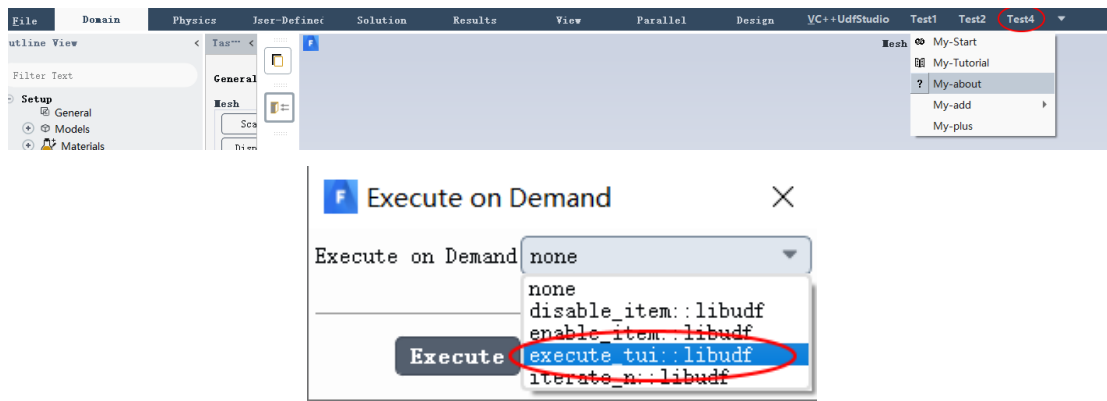
分别按下表设置菜单项的显示字符和名称

子菜单项显示字符	菜单名称
My-Start	ID_TEST1_START
My-Tutorial	ID_TEST1_TUTORIAL
My-about	ID_TEST1_ABOUT
Test4	ID_TEST4
其它	随意, 由于本例子不准备写它们响应函数, 所以随意取名即可

添加位图和菜单资源完成后, 再回到解决方案管理器, 打开 udf 源代码 “udf\_source.cpp”



将如下内容写入该文件, 编译通过后加载, 可以发现Fluent中已经出现对应菜单, 但此时Test4菜单是禁用的, 当我们手动执行 “execute\_tui::libudf” 后, 该菜单就启用了。这是因为刚开始我们把my\_value设为 FALSE, 而菜单选择事件响应函数中是根据my\_value的值来禁用或启用。当执行 “execute\_tui::libudf” 后, 该DEMAND宏中将my\_value设为TRUE, 这样我们点击菜单时自动就会启用该菜单了。



udf 源代码 “udf\_source.cpp” 内容如下:

```
#include "udf.h"
#include "SuperUdfExtension.h"
#include "resource.h" // resource file needed
//=====注册版才能编译通过, 因为试用版不能超过 2 个宏=====
BOOL my_value=FALSE; // variable enable or disable Menu items
void OnClickStart();
void OnClickTutorial();
void OnClickAbout();
void OnClick4();
void CALLBACK MenuSelectProc();

DEFINE_EXECUTE_ON_LOADING(load, libudf)
{
    HMODULE hModule=AfxGetInstanceHandle();
```



```
SuperUdf_Initialize(hModule);

MenuItemBmpAndFun bmpAndFun[]={
    {ID_TEST1_START,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_START)), OnClickStart},
    {ID_TEST1_TUTORIAL,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_TUTORIAL)), OnClickTutorial},
    {ID_TEST1_ABOUT,::LoadBitmap(hModule,MAKEINTRESOURCE(IDB_MY_ABOUT)), OnClickAbout},
    {ID_TEST4, NULL, OnClick4} // if no bitmap, use NULL
};
SuperUdf_SetMenuBmpAndFun(bmpAndFun, sizeof(bmpAndFun)/sizeof(MenuItemBmpAndFun));
SuperUdf_AddUserMenu(IDR_MENU1);
SuperUdf_SetMenuSelectCallBack(MenuSelectProc);
}

void OnClickStart()
{
    HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
    MessageBox(hFluentGUIMainWnd, "start clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

void OnClickTutorial()
{
    HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
    MessageBox(hFluentGUIMainWnd, "tutorial clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

void OnClickAbout()
{
    HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
    MessageBox(hFluentGUIMainWnd, "about clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

void OnClick4()
{
    HWND hFluentGUIMainWnd=SuperUdf_GetFluentMainWnd();
    MessageBox(hFluentGUIMainWnd, "4 clicked", "", MB_OK | MB_ICONINFORMATION | MB_TOPMOST);
}

DEFINE_ON_DEMAND(disable_item)
{
    SuperUdf_EnableMenuItem(ID_TEST4, FALSE);
    SuperUdf_EnableMenuItem(ID_TEST1_TUTORIAL, FALSE);
}

DEFINE_ON_DEMAND(enable_item)
{
    SuperUdf_EnableMenuItem(ID_TEST4, TRUE);
    SuperUdf_EnableMenuItem(ID_TEST1_TUTORIAL, TRUE);
}

DEFINE_ON_DEMAND(execute_tui)
{
    SuperUdf_ExecuteConsoleCommand("/solve/initialize/initialize-flow yes");
    my_value=TRUE;
}

DEFINE_ON_DEMAND(iterate_n)
{
    SuperUdf_Steady_Iterate(1000);
}
```

```
}  
  
void CALLBACK MenuSelectProc()  
{  
    if(my_value==TRUE) // my_value is a variable to judge  
        SuperUdf_EnableMenuItem(ID_TEST4, TRUE);  
    else  
        SuperUdf_EnableMenuItem(ID_TEST4, FALSE);  
}
```

## 参考文献:

- [1] 孙鑫,《VC++深入详解》,电子工业出版社
- [2] 佩措尔德,《Windows 程序设计(第 5 版)》,清华大学出版社