第 5 章

GUI 功能函数

Octave 在实现 GUI 相关的组件时不全是基于句柄和对象实现的，所以对于不全是基于句柄和对象的功能函数，就不能按照结构体的键-值对进行理解，而是要单独分析典型的 GUI 功能函数。

5.1 文件管理

5.1.1 文件夹选择器

调用 uigetdir()函数可以调出文件夹选择器，用于选取目标文件夹。按下“确定”按钮之后，这个文件夹选择器会关闭，并且会返回选择好的目标文件夹的绝对路径。

注意：uigetdir()函数不支持选择一个文件，也不能返回文件的绝对路径。

uigetdir()函数允许不带参数调用，代码如下：

```
>> uigetdir
```

上面的代码将通过 GUI 调出一个文件夹选择器，结果如图 5-1 所示。

此外，uigetdir()函数还允许追加一个额外参数，这个参数为默认的初始路径。在指定这个参数之后，文件夹选择器在初始打开时就会打开这个文件夹。调用 uigetdir()函数，并将初始文件夹指定为/lib 的代码如下：

```
>> uigetdir('/lib')
```

因为 Windows 和 Linux 系统之下的文件路径的定义不同，两种操作系统用 uigetdir()函数返回的路径选择结果也不相同。例如，在 Windows 之下的一个示例结果如下：

```
>> dir = uigetdir
dir = C:\Octave\Octave\
```

在 Linux 之下的一个示例结果如下：

```
>> dir = uigetdir
dir = /usr/bin
```

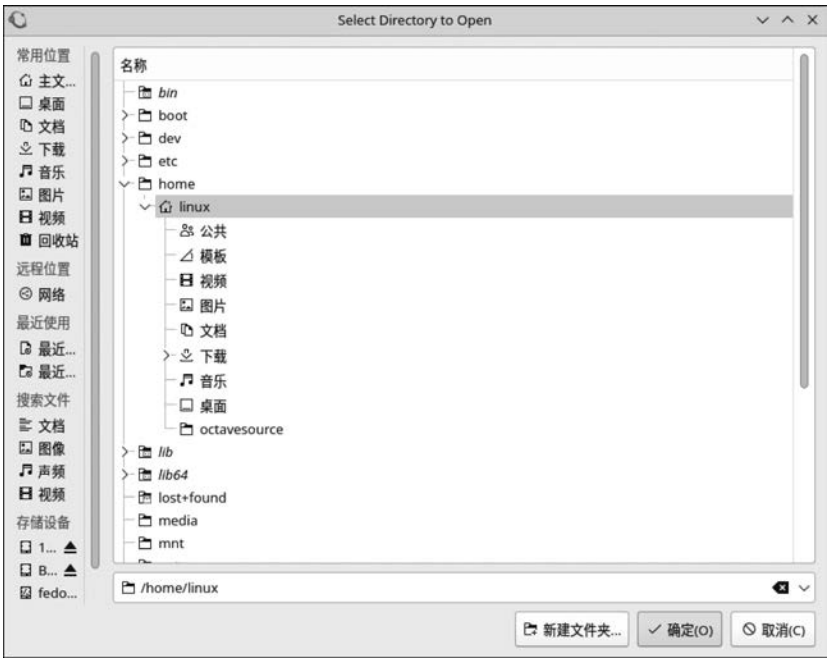

图 5-1　文件夹选择器

5.1.2　文件选择器

调用 uigetfile()函数可以调出文件选择器，用于选取目标文件。按下“确定”按钮之后，这个文件选择器会关闭，并且会返回选择好的目标文件的绝对路径。

注意：uigetfile()函数不支持选择一个文件夹，也不能返回文件夹的绝对路径。

uigetfile()函数允许不带参数调用，代码如下：

```
>> uigetfile
```

上面的代码将通过 GUI 调出一个文件选择器，结果如图 5-2 所示。

此外，uigetfile()函数还允许追加一个额外参数，这个参数作为筛选规则的用途。

使用筛选规则筛选文件类型时的规则如下：

(1) 如果所描述的文件名和这个参数匹配，文件就可以被显示出来。

(2) 如果所描述的文件名和这个参数不匹配，文件就不能被显示出来。

使用筛选规则筛选文件路径时的规则如下：

(1) 仅当所描述的文件的绝对路径包含了这个路径时，文件才可以被显示出来。

(2) 如果所描述的文件的绝对路径没有包含这个路径，文件就不能被显示出来。

使用筛选规则同时筛选文件类型和文件路径的规则如下：

(1) 仅当所描述的文件名和这个参数匹配，并且所描述的文件的绝对路径包含了这个路径时，文件才可以被显示出来。

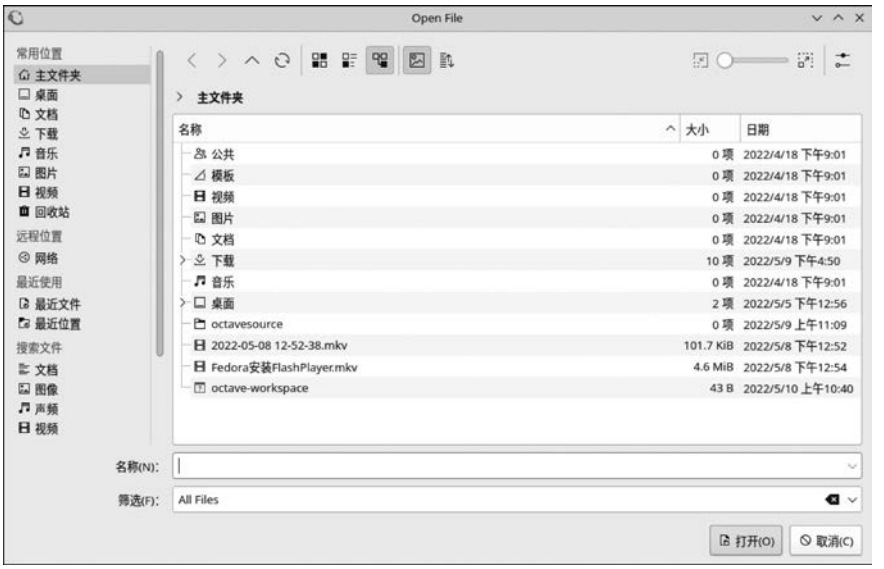

图 5-2 文件选择器

(2) 如果所描述的文件名和这个参数不匹配,文件就不能被显示出来。

(3) 如果所描述的文件的绝对路径没有包含这个路径,文件就不能被显示出来。

调用 uigetfile()函数,并限制选取的文件类型为 *.m,代码如下:

```
>> uigetfile('*.m')
```

此外,uigetfile()函数不但允许筛选一种文件类型,还允许筛选多种文件类型。调用 uigetfile()函数,并限制选取的文件类型为 *.m 和 *.txt,代码如下:

```
>> uigetfile({"*.m;*.txt"})
```

此外,uigetfile()函数还允许筛选文件路径。调用 uigetfile()函数,并将选取的文件路径限制为/lib,代码如下:

```
>> uigetfile('/lib/')
```

此外,uigetfile()函数还允许同时筛选文件类型和文件路径。调用 uigetfile()函数,并将选取的文件类型限制为 *.m,并将选取的文件路径限制为/lib,代码如下:

```
>> uigetfile({'*.m;/lib/'})
```

注意:这里只有以/结尾的字符串才被视为路径。不以/结尾的字符串,并且不符合筛选文件类型写法的字符串均视为筛选规则提示。

此外,uigetfile()函数还允许追加筛选规则提示,用于提示其他用户对于某个文件筛选器的筛选规则的解释,代码如下:

```
>> uigetfile({"*.m;*.txt",'My Programming Components'})
```

上面的代码标注了 My Programming Components 字符串，这个字符串会被原封不动地显示在文件筛选器对应的筛选规则提示的开头。追加提示字符串的效果如图 5-3 所示。

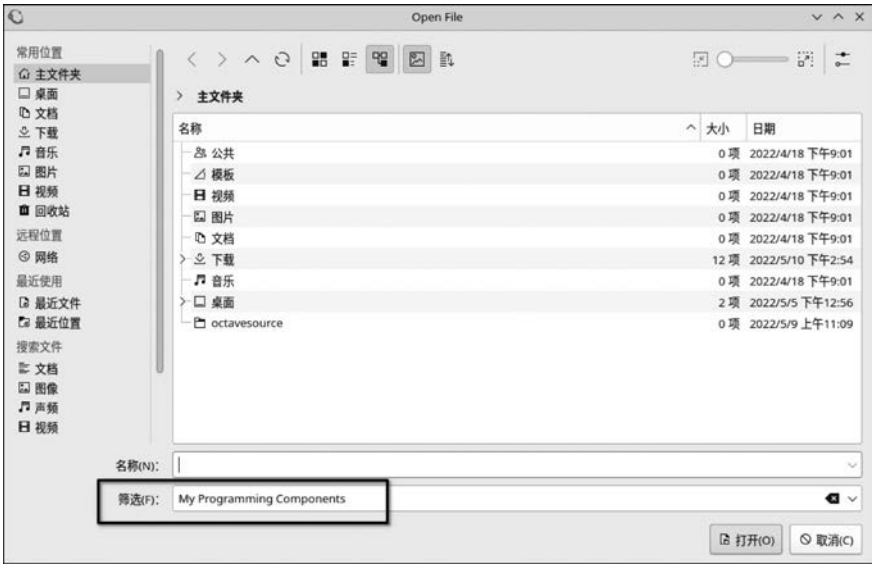

图 5-3　指定筛选规则提示的文件选择器

此外，uigetfile()函数还允许将筛选规则和筛选规则提示配对使用。配对使用的规则如下：

(1) 仅当参数类型是元胞时，筛选规则才能和筛选规则提示配对使用。

(2) 元胞的每行都属于一个分组。

对于多条筛选规则的写入，此函数可以参照一条参数的写法，将输入参数看作一个元胞，直接增加这个元胞的一行，然后在新的一行中写入新的筛选规则和筛选规则提示，即可完成多条筛选规则的输入，代码如下：

```
>> uigetfile({"*.m,*.txt","My Programming Components";'*.doc;*.docx','My documents'})
```

此外，uigetfile()函数还支持其他自定义设置。uigetfile()函数支持的属性名称和含义对照表如表 5-1 所示。

表 5-1　uigetfile()函数支持的属性名称和含义对照表

属性名称	含　义
Position	文件选择器左上角第 1 像素的位置
MultiSelect	开启或关闭多选功能

5.1.3　文件保存器

调用 uiputfile()函数可以调出文件保存器，用于将数据存放到外部文件。uiputfile()函数允许不带参数调用，代码如下：

```
>> uiputfile
```

上面的代码将通过 GUI 调出一个文件保存器，结果如图 5-4 所示。

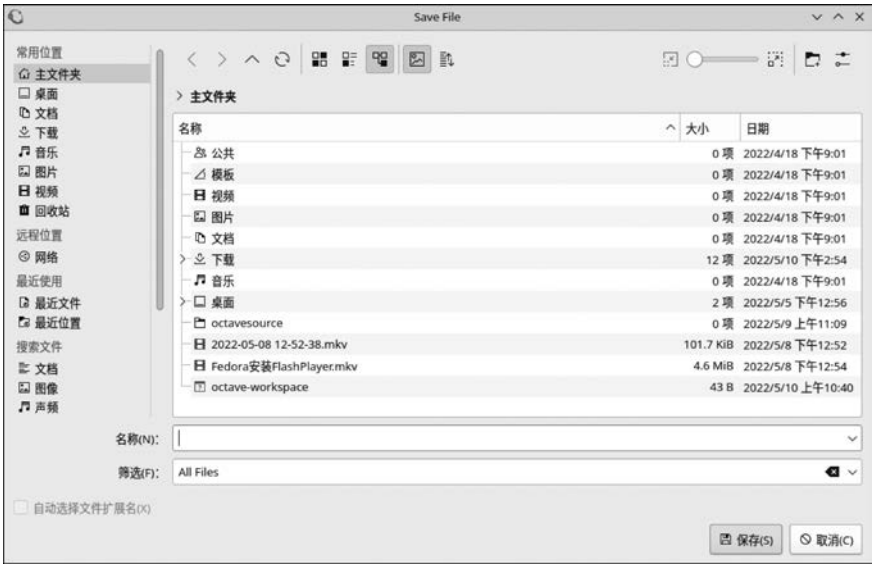

图 5-4 文件保存器

此外，uiputfile()函数还允许追加一个额外参数，这个参数作为筛选规则的用途。uiputfile()函数的筛选规则和 uigetfile()函数相同。

调用 uiputfile()函数，并将选取的文件类型限制为 *.m，代码如下：

```
>> uiputfile('*.m')
```

此外，uiputfile()函数不但允许筛选一种文件类型，还允许筛选多种文件类型。调用 uiputfile()函数，并将选取的文件类型限制为 *.m 和 *.txt，代码如下：

```
>> uiputfile({"*.m, *.txt"})
```

此外，uiputfile()函数还允许筛选文件路径。调用 uiputfile()函数，并将选取的文件路径限制为/lib，代码如下：

```
>> uiputfile('/lib/')
```

注意：这里只有以/结尾的字符串才被视为路径。不以/结尾的字符串，并且不符合筛选文件类型写法的字符串均视为筛选规则提示。

此外，uiputfile()函数还允许追加筛选规则提示，用于提示其他用户对于某个文件筛选器的筛选规则的解释，代码如下：

```
>> uiputfile({"*.m, *.txt",'My Programming Components'})
```

对于多条筛选规则的写入，此函数可以参照一条参数的写法，将输入参数看作一个元胞，直接增加这个元胞的一行，然后在新的一行中写入新的筛选规则和筛选规则提示，即可完成多

条筛选规则的输入，代码如下：

```
>> uiputfile({"*.m, *.txt",'My Programming Components';'*.doc; *.docx','My documents'})
```

5.2 对话框

5.2.1 错误对话框

调用 errordlg()函数可以启动一个带有错误图标的对话框。在对话框中，错误的图标被放置于对话框的左半部分。

errordlg()函数允许不带参数调用，代码如下：

```
>> errordlg
```

默认的错误对话框如图 5-5 所示。

此外，errordlg()函数还允许追加两个额外参数，其中，第 1 个参数为错误提示语句，第 2 个参数为对话框的标题栏文字，代码如下：

```
>> errordlg({"Error occurred:", "Error reason: Assertion 404"},"Error!")
```

上面的代码是一个指定错误警告和错误原因的对话框示例，并且这个对话框的标题为字符串 Error!。代码运行后的结果如图 5-6 所示。

图 5-5 默认的错误对话框

图 5-6 指定错误警告和错误原因的错误对话框

此外，errordlg()函数还支持信息框的自定义设置。在进行设置时，使用键-值对的形式传入自定义选项即可。

5.2.2 帮助对话框

调用 helpdlg()函数可以启动一个带有帮助图标的对话框。在对话框中，帮助的图标被放置于对话框的左半部分。

helpdlg()函数允许不带参数调用，代码如下：

```
>> helpdlg
```

默认的帮助对话框如图 5-7 所示。

此外，helpdlg()函数还允许追加一个额外参数，此时在生成的提示框中将额外指定提示信息。在帮助对话框中显示文字 Help Text 的代码如下：

```
>> helpdlg('Help Text')
```

代码运行后的结果如图 5-8 所示。

图 5-7 默认的帮助对话框

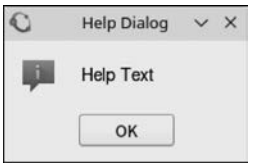

图 5-8 指定提示信息的帮助对话框

此外,也可以向 helpdlg()函数传入多行矩阵,实现多行文本的显示。在帮助对话框中同时显示三行文字(one 、two 和 three)的代码如下:

```
>> a = ["one";"two";"three"];
>> helpdlg(a)
```

代码运行后的结果如图 5-9 所示。

此外,helpdlg()函数还允许追加第 2 个额外参数,这个参数代表对话框的标题栏文字。指定帮助对话框的标题栏文字为 Help Title 的代码如下:

```
>> a = ["one";"two";"three"];
>> helpdlg(a,"Help Title")
```

代码运行后的结果如图 5-10 所示。

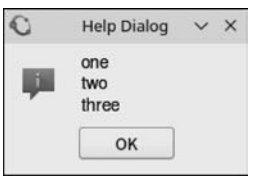

图 5-9 显示多行文本的帮助对话框

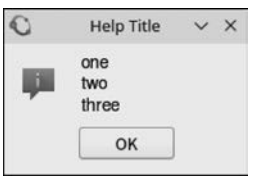

图 5-10 指定标题栏文字的帮助对话框

此外,helpdlg()函数还支持信息框的自定义设置。在进行设置时,使用键-值对的形式传入自定义选项即可。

5.2.3 文本框对话框

调用 inputdlg()函数可以启动一个带有文本框的对话框。调用 inputdlg()函数时,至少需要传入一个参数,这个参数代表输入框上方的提示文字。将文本框对话框的提示文字指定为 123 的代码如下:

```
>> inputdlg('123')
```

代码运行后的结果如图 5-11 所示。

图 5-11 指定提示文字的文本框对话框

此外，inputdlg()函数还允许追加第2个额外参数，这个参数代表文本框对话框的标题栏文字。将文本框对话框的标题栏文字指定为IP Title的代码如下：

```
>> inputdlg('123','IP Title')
```

代码运行后的结果如图5-12所示。

此外，inputdlg()函数还允许追加第3个额外参数，这个参数代表文本框的高度。将文本框的高度指定为2行文本的代码如下：

```
>> inputdlg('123','IP Title',2)
```

此外，inputdlg()函数还允许追加第4个额外参数，这个参数代表在输入框内默认填入的文字。将文本框输入框内默认填入的文字指定为456的代码如下：

```
>> inputdlg('123','IP Title',2,{'456'})
```

代码运行后的结果如图5-13所示。

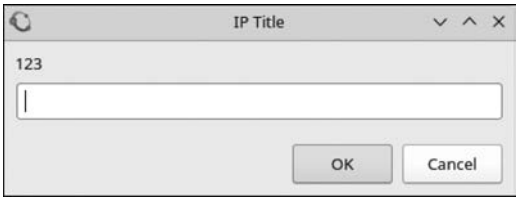

图5-12　指定标题栏文字的文本框对话框

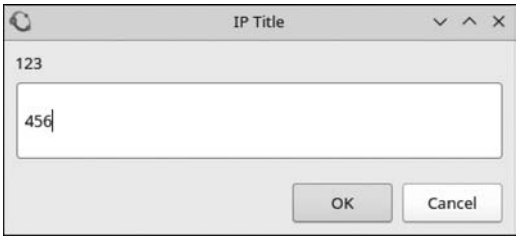

图5-13　指定默认填入的文字的文本框对话框

5.2.4　列表对话框

调用listdlg()函数可以启动一个带有列表选项的对话框。listdlg()函数使用键-值对的形式传入选项。调用listdlg()函数时，至少需要传入一对键-值对。将列表项目指定为Option 1和Option 2的代码如下：

```
>> listdlg("ListString", {"Option 1","Option 2"})
```

代码运行后的结果如图5-14所示。

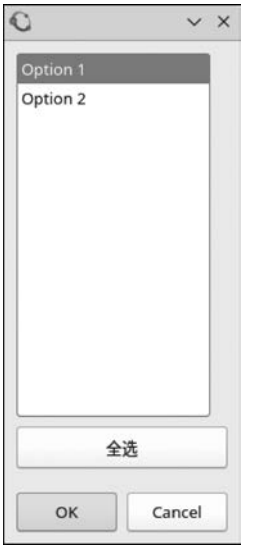

图5-14　指定列表项目的列表对话框

listdlg()函数支持的属性名称和含义对照表如表5-2所示。

表5-2　listdlg()函数支持的属性名称和含义对照表

属性名称	含　义
ListString	列表项目
SelectionMode	单选或多选的选项
ListSize	列表的显示像素大小
InitialValue	列表初始选中的项目序号
Name	标题栏标题
PromptString	显示在列表上方的提示文字
OKString	“确定”按钮的替换提示文字
CancelString	“取消”按钮的替换提示文字

5.2.5 信息框

信息框也叫 Message Box。在 Octave 中，错误对话框和警告对话框也支持信息框的选项。

调用 msgbox()函数可以启动一个用于显示信息的对话框。调用 msgbox()函数时，至少需要传入一个参数，这个参数代表信息框中的信息文字。将信息框的信息文字指定为 123 的代码如下：

```
>> msgbox('123')
```

代码运行后的结果如图 5-15 所示。

msgbox()函数还允许追加第 2 个额外参数，这个参数代表信息框的标题栏文字。将信息框的标题栏文字指定为 MSGBOX Title 的代码如下：

```
>> msgbox('123','MSGBOX Title')
```

代码运行后的结果如图 5-16 所示。

图 5-15 指定信息文字的信息框

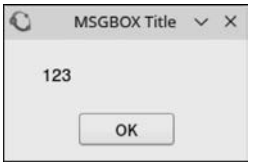

图 5-16 指定标题栏文字的信息框

msgbox()函数还允许追加其他键-值对参数。msgbox()函数支持的属性名称和含义对照表如表 5-3 所示。

表 5-3 msgbox()函数支持的属性名称和含义

属性名称	含义	属性名称	含义
windowstyle	信息框的风格	Interpreter	显示在列表上方的提示文字
non-modal	默认的信息框行为，是默认值	tex	使用 tex 风格解释器，是默认值
modal	阻止用户进行 UI 界面元素的交互操作	none	使用纯文本解释器
		latex	使用 LaTeX 风格解释器
replace	不会生成新的相同信息框替换已有的相同信息框		

5.2.6 询问对话框

调用 questdlg()函数可以启动一个用于显示问题的对话框。调用 questdlg()函数时，至少需要传入一个参数，这个参数代表询问对话框中的问题文字。将询问对话框的问题文字指定为 Question Text 的代码如下：

```
>> questdlg('Question Text')
```

代码运行后的结果如图 5-17 所示。

questdlg()函数还允许追加第 2 个额外参数，这个参数代表询问对话框的标题栏文字。

将询问对话框的标题栏文字指定为 QUESTION Title 的代码如下：

```
>> questdlg('Question Text','QUESTION Title')
```

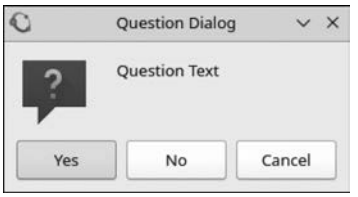

图 5-17　指定问题文字的询问对话框

代码运行后的结果如图 5-18 所示。

questdlg()函数还允许追加第 3 个额外参数，这个参数代表询问对话框的默认选项。将询问对话框的默认选项指定为 No 的代码如下：

```
>> questdlg('Question Text','QUESTION Title', 'No')
```

questdlg()函数还允许分别指定是选项的文字、否选项的文字和取消选项的文字。将询问对话框的问题文字指定为 Question Text，将询问对话框的标题栏文字指定为 QUESTION Title，分别指定是选项的文字为“是”、否选项的文字为“否”、取消选项的文字为“取消”，并且将询问对话框的默认选项指定为否的代码如下：

```
>> questdlg('Question Text','QUESTION Title',...
'是','否','取消','否')
```

代码运行后的结果如图 5-19 所示。

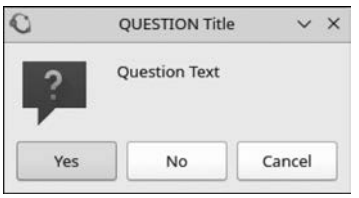

图 5-18　指定标题栏文字的询问对话框

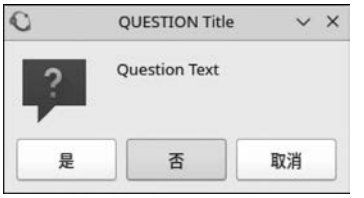

图 5-19　指定默认选项和选项文字的询问对话框

此外，questdlg()函数支持至多 3 个按钮，并且函数的返回值会随着是选项的文字、否选项的文字和取消选项的文字的改变而改变。例如在上面的代码中，如果单击“是”按钮，则 questdlg()函数的返回值也将为是；如果单击“否”按钮，则 questdlg()函数的返回值也将为否。单击“否”按钮时的返回值如下：

```
ans = 否
```

5.2.7　警告对话框

调用 warndlg()函数可以启动一个带有警告图标的对话框。在对话框中，警告的图标被放置于对话框的左半部分。

warndlg()函数允许不带参数调用，代码如下：

```
>> warndlg
```

默认的警告对话框如图 5-20 所示。

warndlg()函数还允许追加一个额外参数，此时在生成的警告对话框中将额外附带警告信息。在警告对话框中指定警告信息 Warn Text 的代码如下：

```
>> warndlg('Warn Text')
```

代码运行后的结果如图 5-21 所示。

也可以向 warndlg()函数中传入多行矩阵,实现多行文本的显示。在警告对话框中同时显示三行文字(one 、two 和 three)的代码如下:

```
>> a = ["one";"two";"three"];
>> warndlg(a)
```

代码运行后的结果如图 5-22 所示。

图 5-20 默认的警告对话框

图 5-21 指定警告信息的警告对话框

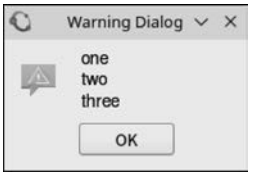

图 5-22 显示多行文字的警告对话框

此外,warndlg()函数还支持信息框的自定义设置。在进行设置时,使用键-值对的形式传入自定义选项即可。

5.2.8 自定义对话框

调用 dialog()函数可以启动一个可以增加控制对象的自定义对话框。dialog()函数允许不带参数调用,代码如下:

```
>> d = dialog
d =  - 88.243
```

代码运行后的结果如图 5-23 所示。

dialog()函数允许使用键-值对的形式传入选项。创建一个背景颜色为红色的自定义对话框的代码如下:

```
>> d = dialog('color', 'red')
d =  - 88.341
```

代码运行后的结果如图 5-24 所示。

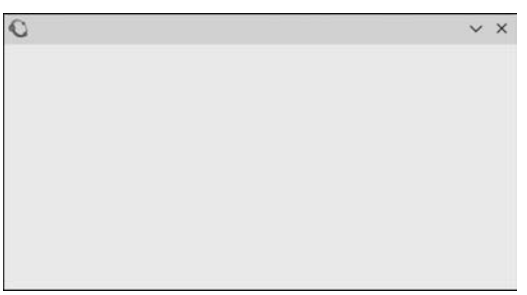

图 5-23 默认的自定义对话框

图 5-24 背景颜色为红色的自定义对话框

dialog()函数所支持修改的属性详见默认图像对象属性表格。

注意：不是所有的图像对象属性都能在对话框中被修改。

5.3 进度条

调用 waitbar()函数可以持续生成进度条。waitbar()函数至少需要传入一个参数，这个参数是一个 0～1 的数字，代表当前进度。将进度条的进度指定为 0.5432 的代码如下：

```
>> waitbar(0.5432)
```

代码运行后的结果如图 5-25 所示。

waitbar()函数允许追加传入一个额外参数，这个参数代表的是进度条上方的信息。将进度条的信息指定为 WAITBAR msg 的代码如下：

```
>> waitbar(0.5432, 'WAITBAR msg')
```

代码运行后的结果如图 5-26 所示。

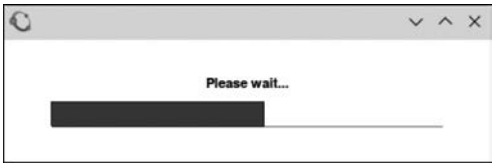

图 5-25 指定进度的进度条

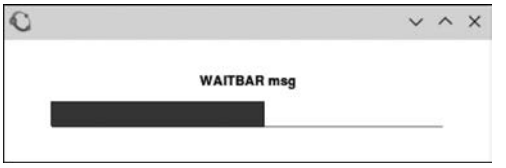

图 5-26 指定信息的进度条

waitbar()函数允许追加传入 createcancelbtn 额外参数和一个函数句柄，此时这个参数将在进度条下方生成一个取消按钮。在进度条下方生成一个取消按钮，并使进度条每次单击按钮后都计算 1+2 的结果的代码如下：

```
>> waitbar(0.5432, 'WAITBAR msg', 'createcancelbtn', '@sum([1 2])')
>> ans = 3
ans = 3
```

代码运行后的结果如图 5-27 所示。

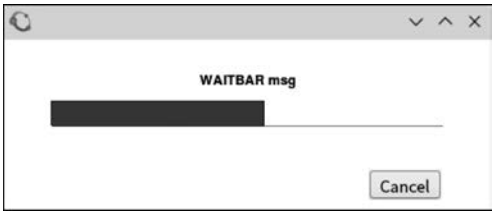

图 5-27 指定信息、带取消按钮的进度条

waitbar()函数允许使用键-值对的形式传入选项。waitbar()函数所支持修改的属性详见默认图像对象属性表格。

注意：不是所有的图像对象属性都能在进度条中被修改。

5.4 字体选择器

调用 uisetfont()函数可以启动一个用于选择字体的对话框，被称为字体选择器。uisetfont()函数允许不带参数调用，代码如下：

```
>> uisetfont
```

代码运行后的结果如图 5-28 所示。

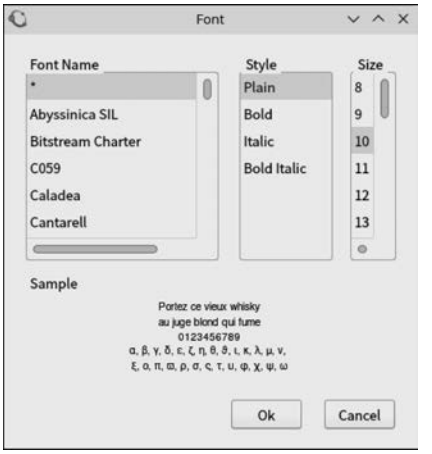

图 5-28 字体选择器

uisetfont()函数还允许追加一个额外参数，这个参数可以被视为一个图形句柄。此时若单击 OK 按钮将设置这个图形句柄中 FontName、FontWeight、FontAngle、FontUnits 和/或 FontSize 选项。调用 uisetfont()函数选择当前轴对象的句柄的字体的代码如下：

```
>> uisetfont(gca)
```

这个参数还可以被视为一个结构体。通过传入结构体参数的方式，可以设置字体选择器的初始选项。通过结构体方式同时设置 FontName、FontWeight、FontAngle、FontUnits 和 FontSize 的初始选项的代码如下：

```
>> font_setting = struct;
>> font_setting.FontName = '*';
>> font_setting.FontAngle = 'normal';
>> font_setting.FontUnits = 'pixel';
>> font_setting.FontWeight = 'normal';
>> font_setting.FontSize = 10;
>> uisetfont(font_setting)
```

uisetfont()函数还允许追加一个放在最后的额外参数，这个参数代表的是字体选择器的标题栏文字。将字体选择器的标题栏文字指定为 123 的代码如下：

```
>> uisetfont('123')
```

5.5 查询或设置 GUI 数据

调用 guidata()函数可以查询或设置 GUI 的用户自定义数据。guidata()函数至少需要传入一个参数，代表要查询或设置的图形句柄。查询默认控制对象的用户自定义数据的代码如下：

```
>> h = uicontrol
h =  -42.338
>> guidata(h)
```

guidata()函数允许追加传入一个额外参数,此时这个参数代表要设置的数据的值。将默认控制对象的用户自定义数据设置为 data1 和 data2 的代码如下:

```
>> h = uicontrol
h =  -42.612
>> guidata(h, {'data1', 'data2'})
```

5.6 查询 GUI 相关句柄

调用 guihandles()函数可以查询或设置 GUI 的用户自定义数据。guihandles()函数需要传入一个参数,代表要查询的图形句柄。查询默认菜单对象的相关句柄的代码如下:

```
>> f = figure
f =  1
>> uit = uitoolbar(f)
uit =  -42.899
>> uic = uicontextmenu(f)
uic =  -41.366
>> uim = uimenu(uic)
uim =  -40.882
>> guihandles(uim)
ans =

    scalar structure containing the fields:

        __default_toolbar__ =  -51.925
        __default_button_text__ =  -45.273
        __default_button_zoomout__ =  -47.122
        __default_button_zoomin__ =  -48.866
        __default_button_rotate__ =  -49.623
        __default_button_pan__ =  -50.719
        __default_menu__Tools =  -64.462
        zoom_off =  -52.184
        zoom_on =  -53.282
        rotate3d =  -54.167
        no_pan_rotate =  -55.203
        pan_yon =  -56.626
        pan_xon =  -57.176
        pan_on =  -58.127
        off =  -61.014
        on =  -62.161
        toggle =  -63.120
        __default_menu__Edit =  -70.394
        __default_menu__File =  -75.912
```

5.7 GUI 功能查询

调用 have_window_system 函数可以查询操作系统的 GUI 功能是否可用。have_window_system 函数在调用时不允许传入参数,代码如下:

```
>> have_window_system
```

5.8 GUI 运行模式查询

调用 isguirunning 函数可以查询操作系统的 GUI 功能是否运行在 GUI 模式之下。isguirunning 函数在调用时不允许传入参数，代码如下：

```
>> isguirunning
```

(1) 如果当前的 Octave 实例运行在 GUI 模式之下，则 isguirunning 函数将返回 1。

(2) 如果当前的 Octave 实例运行在 CLI 模式之下，则 isguirunning 函数将返回 0。

5.9 精确移动窗口

调用 movegui()函数可以将窗口精确地移动到特定的坐标上。movegui()函数至少需要传入一个参数，代表要移动的图形句柄。移动默认控制对象所在的窗口的代码如下：

```
>> uic = uicontrol
uic =  -42.035
>> movegui(uic)
```

这个参数还可以被认为是要移动的目标坐标，此时 movegui()函数会隐式地调用 gcbf 函数或 gcf 函数，然后隐式地获取某个 GUI 控件的句柄，并移动那个句柄所在的窗口。将窗口移动到坐标(100,200)的代码如下：

```
>> movegui([100,200])
```

方位坐标还支持以方位参数的方式传入。方位参数是一系列预设的方位坐标，以参数形式给出，使用更加方便。movegui()函数支持的方位参数如表 5-4 所示。

表 5-4 movegui()函数支持的方位参数

参 数	含 义	参 数	含 义
north	移至屏幕上侧	northwest	移至屏幕左上角
south	移至屏幕下侧	southeast	移至屏幕右下角
east	移至屏幕右侧	southwest	移至屏幕左下角
west	移至屏幕左侧	center	移至屏幕中心
northeast	移至屏幕右上角	onscreen	移至默认位置

此函数还可以额外传入第 2 个参数，此时第 1 个参数被认为是 GUI 对象的句柄，第 2 个参数可以被认为是要移动的目标坐标，代码如下：

```
>> a = figure;
>> movegui(a,[100,600])
```

第 2 个参数还可以被认为是一系列被忽略的事件。该事件参数为一个结构体，用于在回调函数被调用之前做出额外判断，代码如下：

```
>> a = figure;
>> movegui(a,b)
```

此函数还可以额外传入第 3 个参数，此时第 1 个参数被认为是 GUI 对象的句柄，第 2 个

参数被认为是一系列被忽略的事件，第3个参数被认为是要移动的目标坐标，代码如下：

```
>> a = figure;
>> movegui(a,b,[100,600])
```

5.10 变量编辑器

调用openvar()函数可以根据变量名打开变量编辑器，然后可以在打开的编辑器中编辑对应的变量。openvar()函数需要传入一个参数，代表要编辑的变量，代码如下：

```
>> a = 1;
>> openvar('a')
```

代码运行后的结果如图5-29所示。

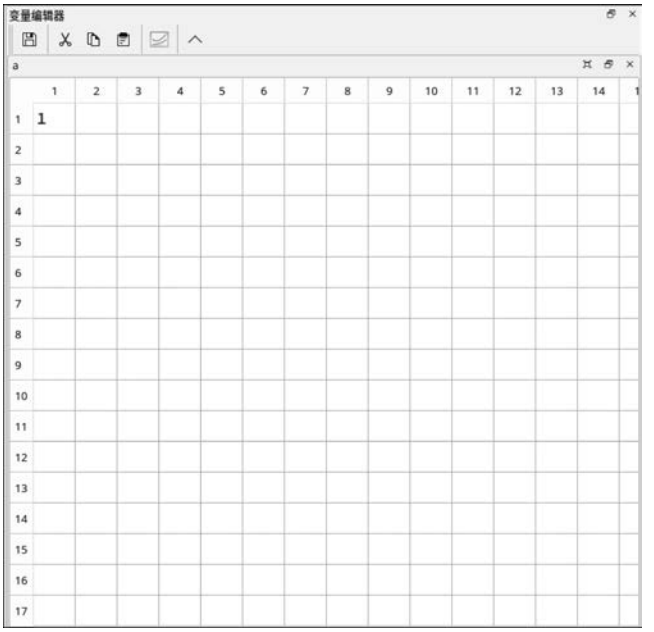

图5-29 变量编辑器

5.11 暂停与恢复GUI之外的程序执行

5.11.1 暂停GUI之外的程序执行

uiwait()函数用于暂停GUI控件之外的程序执行。直到这个GUI控件的句柄被删除，或者手动调用uiresume()函数才可以恢复GUI控件之外的程序执行，代码如下：

```
>> a = figure;
>> uiwait(a)
```

此函数还可以不加参数而直接调用uiwait()函数：

(1) 如果Octave含有GUI控件，则uiwait()函数会隐式地调用gcbf函数或gcf函数，然后隐式地获取某个GUI控件的句柄，GUI控件之外的程序被暂停执行。

(2) 如果 Octave 不含有 GUI 控件,则 uiwait()函数会直接返回。

此函数还可以传入两个参数,此时第 1 个参数被认为是 GUI 控件的句柄,第 2 个参数被认为是暂停时间,并且时间以秒为单位,代码如下:

```
>> a = figure;
>> uiwait(a,2)
```

暂停时间原则上必须是一个大于 1 的数字,而且,如果暂停时间是一个浮点数,则真正的暂停时间将向下取整。

如果暂停时间是一个小于 1 的数字,则真正的暂停时间将用 1 代替,并报警告如下:

```
warning: waitfor: TIMEOUT value must be >= 1, using 1 instead
warning: called from
    uiwait at line 72 column 7
```

5.11.2 恢复暂停的程序

uiresume()函数用于恢复被 uiwait()函数暂停的程序。调用 uiresume()函数时,此函数需要传入一个参数,这个参数被认为是 GUI 控件的句柄,代码如下:

```
>> uiresume(a)
```

5.11.3 可自动恢复的暂停

waitfor()函数用于暂停 GUI 控件之外的程序执行,直到该 GUI 控件满足某个条件。

调用 waitfor()函数时,此函数至少需要传入一个参数,这个参数被认为是 GUI 控件的句柄,此时 waitfor()函数会在 GUI 控件的句柄被删除时自动解除暂停,代码如下:

```
>> a = figure;
>> waitfor(a)
```

此函数还可以额外传入第 2 个参数,此时第 1 个参数被认为是 GUI 控件的句柄,第 2 个参数被认为是句柄的键参数。当 GUI 控件的句柄被删除时,或者当 GUI 控件的句柄的键参数对应的值参数发生改变(例如 GUI 控件的选中状态改变)时,自动解除 waitfor()函数的暂停,代码如下:

```
>> a = figure;
>> waitfor(a,'selected'})
```

此函数还可以额外传入第 3 个参数,此时第 1 个参数被认为是 GUI 控件的句柄,第 2 个参数被认为是句柄的键参数,第 3 个参数被认为是句柄的值参数。当 GUI 控件的句柄被删除时,或者当 GUI 控件的句柄的键参数对应的值参数符合预期(例如 GUI 控件被选中)时,自动解除 waitfor()函数的暂停,代码如下:

```
>> a = figure;
>> waitfor(a,'selected','on')
```

此函数还可以额外传入等待时间。如果传入了 timeout 参数,则紧随其后的参数将被认为是等待时间,代码如下:

```
>> a = figure;
>> waitfor(a,'timeout',1)
```