

# 文件分类应用-汉化总结

本文档主要介绍在汉化“文件分类”应用软件过程中遇到的问题，以及解决方法。

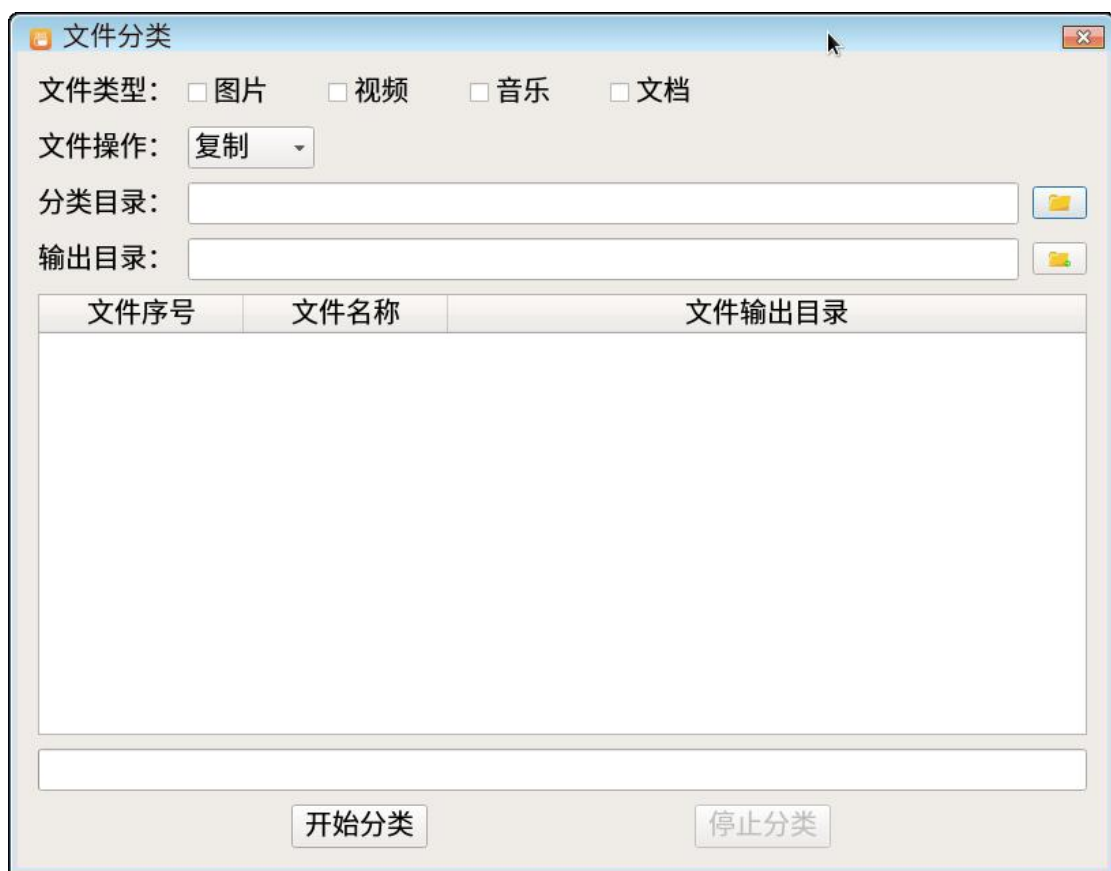
软件运行方法：

- ①：开始菜单->所有应用->系统工具->文件分类
- ②：在终端输入： `fileclassificationtool` ,然后回车运行。

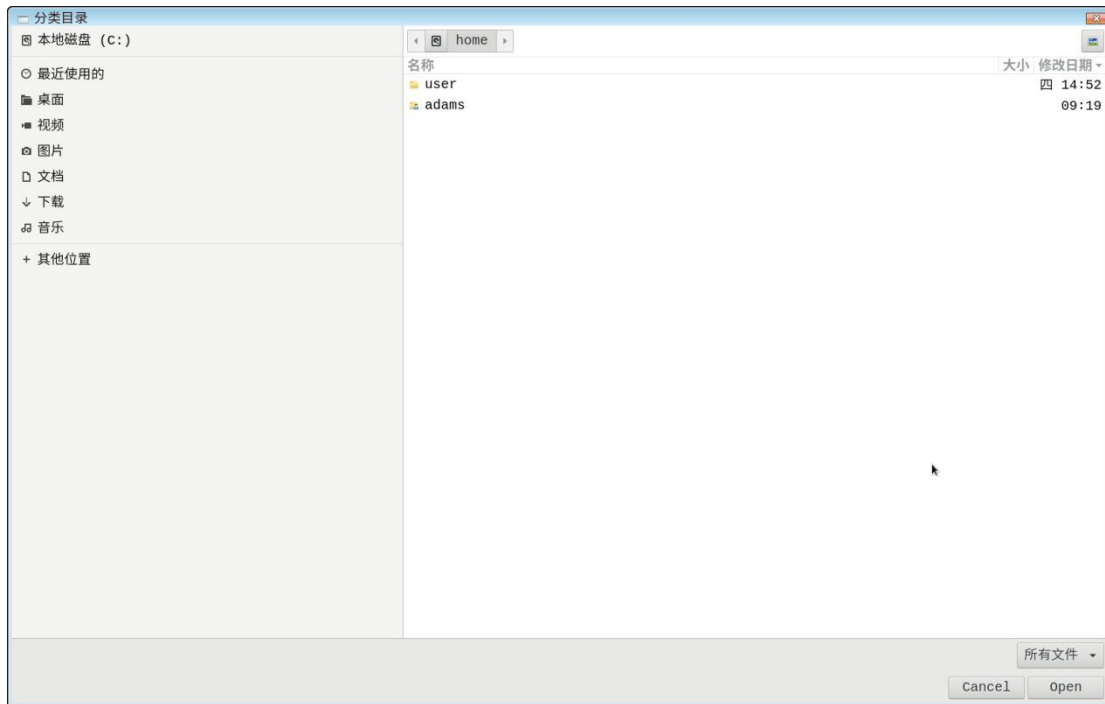
## 一、bug 描述

bug 编号： 42489（目前本人已经无权查看 Bugzilla 上编号为 42489 的 bug）。

软件运行后，点击分类/输出目录右侧带图标的按钮。



在弹出的对话框中，下面的 `Cancel` 与 `Open` 按钮没有实现汉化。如下图所示。



## 二、问题及解决方式

下载查看 `"fileclassificationtool-1.0-1.nd7.5.src.rpm"` 源码，可以定位到选择目录按钮的代码

在 `fileclassification.cpp` 文件中的: (下面是分类目录按钮的代码)

```
void FileClassification::on_pushButtonInput_clicked()
{
    QString oldpath = InputEdit->text();
    QString filePath = QFileDialog::getExistingDirectory(this,
                                                         tr("分类目录"), "/home",
                                                         QFileDialog::ShowDirsOnly |
                                                         QFileDialog::DontResolveSymlinks);
    if(!filePath.isEmpty()) {
        InputEdit->setText(filePath);
        qDebug() << "input filePath is " << filePath;
    } else {
        InputEdit->setText(oldpath);
        qDebug() << "input oldPath is " << oldpath;
    }

    if(InputEdit->text().isEmpty()) {
        InputEdit->setEnabled(true);
    } else {
        InputEdit->setEnabled(false);
    }
}
```

## 2.1 qt 基础库的情况

在上述代码中，弹窗函数是：`QFileDialog::getExistingDirectory`。该函数在 qt 基础库的 `qtbase/src/widgets/dialogs/qfiledialog.cpp` 文件中。通过查看 qt 基础库，没有注意到有对弹窗控件的汉化过程。在 `getExistingDirectory` 函数中调用了 `getExistingDirectoryUrl` 函数。

`getExistingDirectoryUrl` 函数内容如下：

```
QUrl QFileDialog::getExistingDirectoryUrl(QWidget *parent,
                                         const QString &caption,
                                         const QUrl &dir,
                                         Options options,
                                         const QStringList &supportedSchemes)
{
    QFileDialogArgs args;
    args.parent = parent;
    args.caption = caption;
    args.directory = QFileDialogPrivate::workingDirectory(dir);
    args.mode = (options & ShowDirsOnly ? DirectoryOnly : Directory);
    args.options = options;

    QFileDialog dialog(args);
    dialog.setSupportedSchemes(supportedSchemes);
    if (dialog.exec() == QDialog::Accepted)
        return dialog.selectedUrls().value(0);
    return QUrl();
}
```

上述代码中，`QFileDialogArgs` 类的定义可以在 `qfiledialog_p.h` 文件中找到。此处不在赘述。需要注意的是，在配置好一系列参数后，把 `args` 传递给了 `QFileDialog` 构造函数，而该构造函数是被重载的构造函数，属性为 `protected`。这就表明我们不能通过设置 `QFileDialogArgs` 属性的方式，重新设置弹窗的属性。

```
protected:
    QFileDialog(const QFileDialogArgs &args);
```

## 2.2 重新设置弹出窗口属性

可以通过下面的方式设置弹窗的属性等信息。

```
QFileDialog *filedialog = new QFileDialog(this);
filedialog->setAcceptMode(QFileDialog::AcceptOpen);
filedialog->setViewMode(QFileDialog::Detail);
filedialog->setFileMode(QFileDialog::Directory); //只显示目录
filedialog->setDirectory("/home");

filedialog->setOptions(QFileDialog::ShowDirsOnly |
                    QFileDialog::DontResolveSymlinks |
                    QFileDialog::DontUseNativeDialog);
```

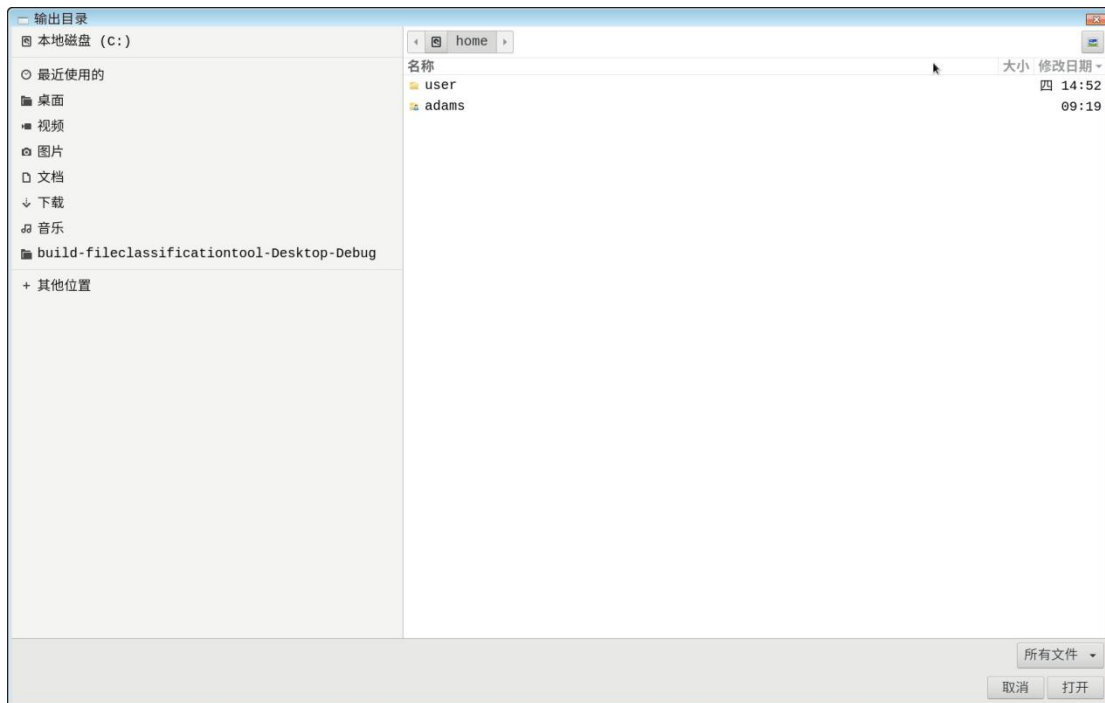
```
filedialog->setWindowTitle(tr("Output directory"));
filedialog->setLabelText(QFileDialog::Reject, tr("Cancel"));

//设置窗口大小
QDesktopWidget * desktopWidget = QApplication::desktop();
QRect rect = desktopWidget->screenGeometry();
filedialog->resize(rect.width()*0.8, rect.height()*0.8);
```

## 2.2.1 本地窗口与 qt 自己绘制的窗口

### 2.2.1.1 本地窗口

上述代码需要描述一下的是：setOptions 中的 QFileDialog::DontUseNativeDialog 选项。如果没有使用该选项，则弹出窗口为：



但是，此时会有如下警告：

**Gtk-Message: GtkDialog mapped without a transient parent. This is discouraged.**

该报错是 Gtk 的报错，翻译成汉语为：

**Gtk 消息: GtkDialog 映射而没有临时父级。 不鼓励这样做。**

在网上查了一些解决方法。都说是个警告信息，不用理会。如果需要处理的话，可以把该报错输出到/dev/null 中。我们不用该界面。

上面窗口的取消/打开按钮的汉化在 ts 文件里实现的。具体内容为：

```
<context>
  <name>QPlatformTheme</name>

  <message>
    <location filename="qplatformtheme.cpp" line="+704"/>
    <source>Open</source>
```

```

    <translation>打开</translation>
</message>
<message>
    <location filename="qplatformtheme.cpp" line="722"/>
    <source>Cancel</source>
    <translation>取消</translation>
</message>

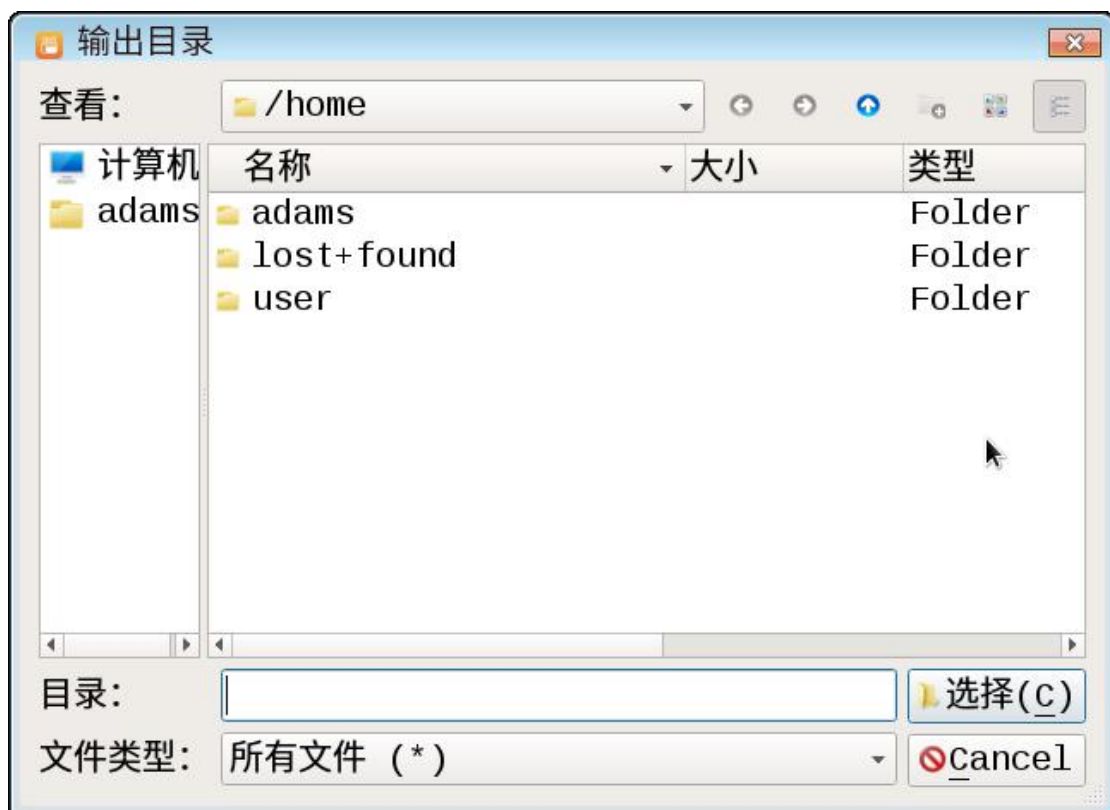
</context>

```

使用上面这段代码，可以解决 Bugzilla 中描述的 bug,但是在测试中发现了上面说的警告信息，所以转而使用下面的界面。

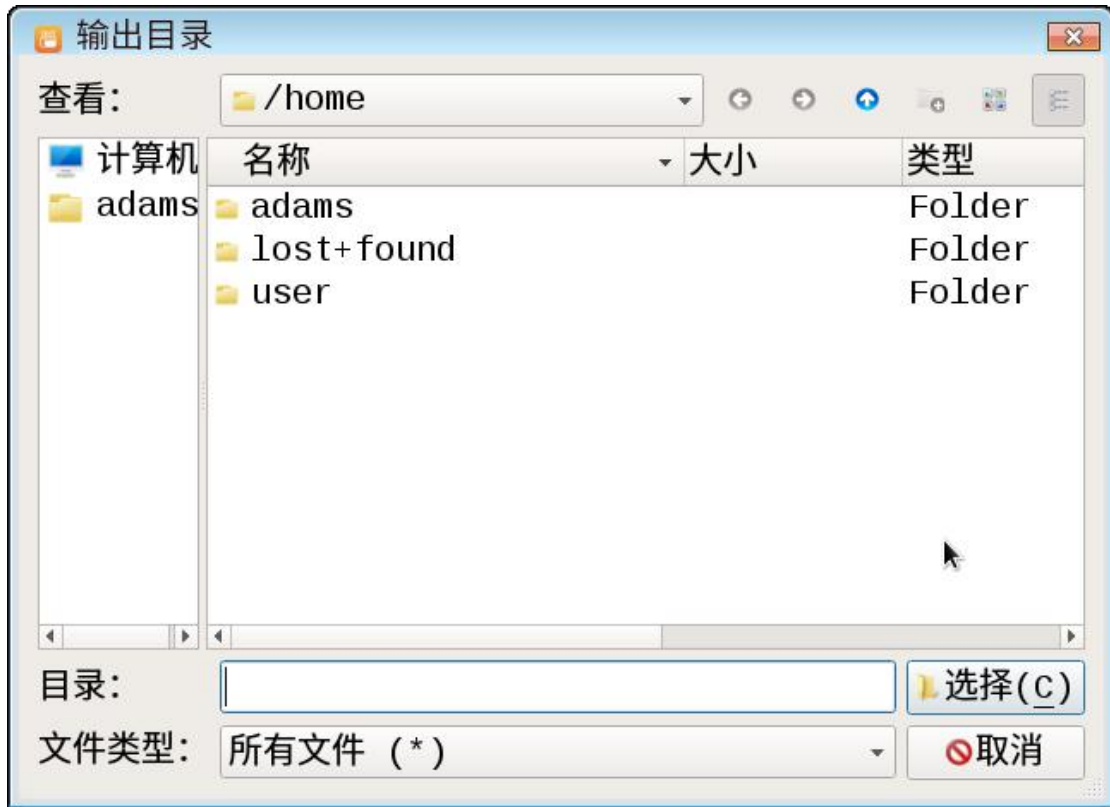
#### 2.2.1.2 qt 自己绘制的窗口

在 setOptions 中加上 QFileDialog::DontUseNativeDialog 选项，则是使用 qt 自己绘制的窗口，界面如下：



在上述界面中，下面的 Cancel 按钮没有汉化。所以我们加上了如下代码，就可以实现汉化了。

```
filedialog->setLabelText(QFileDialog::Reject, tr("Cancel"));
```

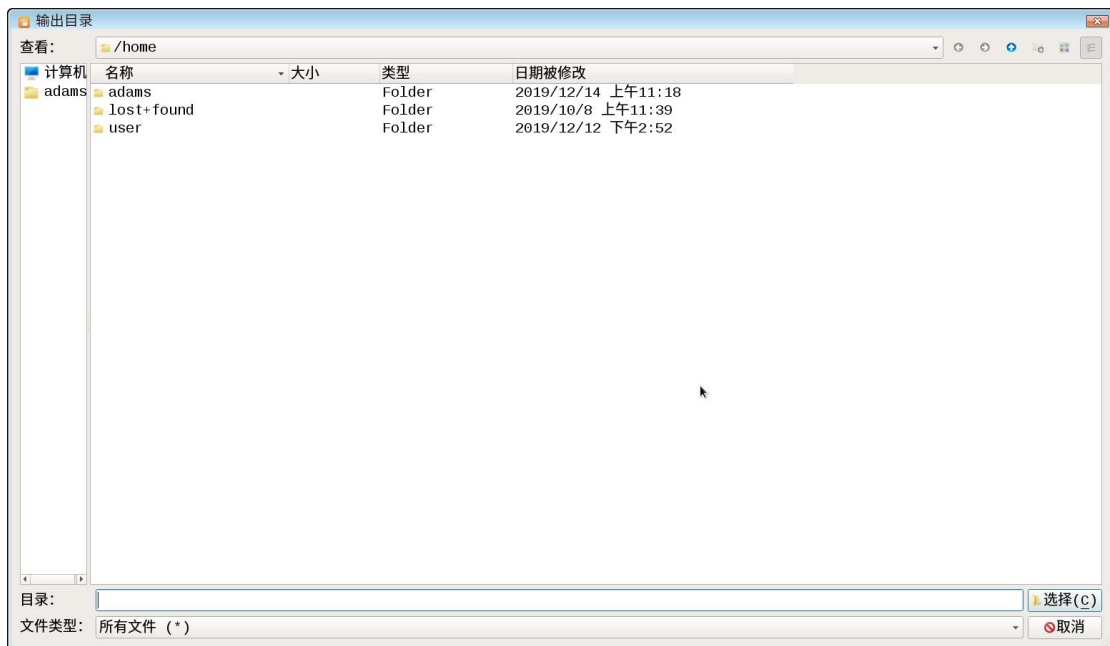


其他部件的汉化，都在下面将要介绍的 `ts` 文件中实现了。

### 2.2.2 设置弹出窗口大小

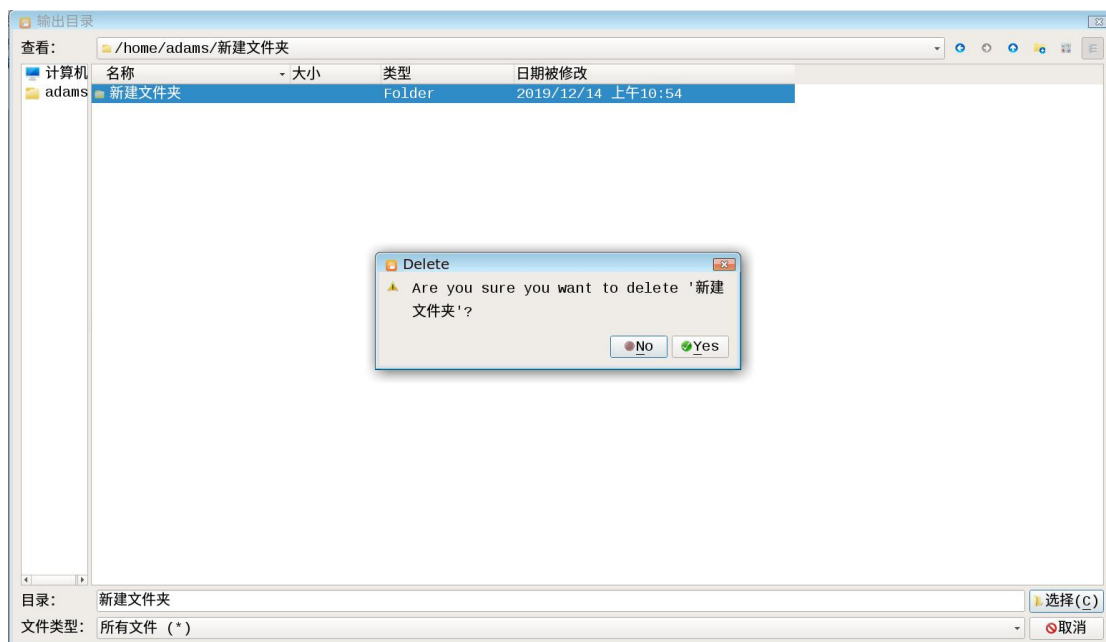
在调试过程中，发现弹出的对话框有点小，部分信息显示不全，所以使用了如下代码设置窗口的大小。

```
//设置窗口大小
QDesktopWidget * desktopWidget = QApplication::desktop();
QRect rect = desktopWidget->screenGeometry();
filedialog->resize(rect.width()*0.8, rect.height()*0.8);
```



## 2.3 部分部件汉化不完全问题

经过测试发现有如下部件汉化不完全。



总结如下：

- 1、名称/大小/类型/日期被修改中的“类型”，下面显示：“Folder”，应该显示：“文件夹”。
- 2、选中文件夹，右键选择删除选项，在弹出的对话框中没有实现汉化。

### 2.3.1 文件夹汉化问题

`Folder` 汉化的问题。经过查找资料，查看 `qt` 基础库源码，发现显示 `Folder` 对应的文件为：

```
qtbase/src/widgets/itemviews/qfileiconprovider.cpp
```

进入到 `qtbase/src/widgets/itemviews` 目录。

执行下面的命令生成 `qfileiconprovider.cpp` 文件对应的 `ts` 文件。

```
[adams@itemviews]$ lupdate-qt5 qfileiconprovider.cpp -ts qfileiconprovider_zh_CN.ts
Updating 'qfileiconprovider_zh_CN.ts'...
Found 8 source text(s) (8 new and 0 already existing)
[adams@itemviews]$
```

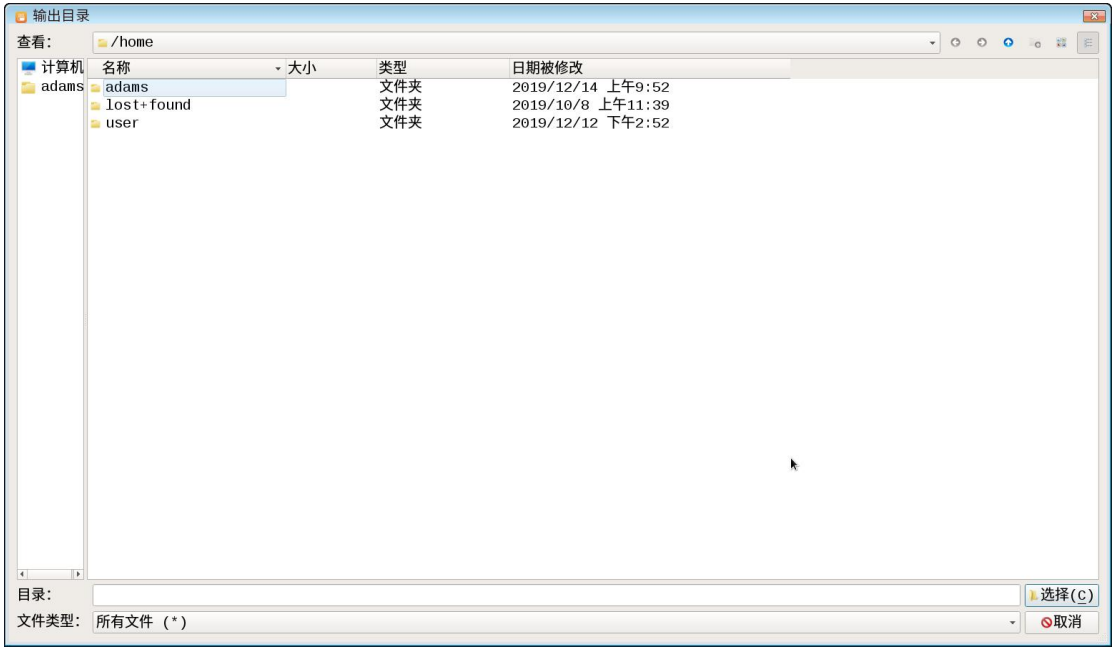
打开 `qfileiconprovider_zh_CN.ts` 文件,可以看到 `Folder` 对应的内容。

```
<context>
  <name>QFileDialog</name>

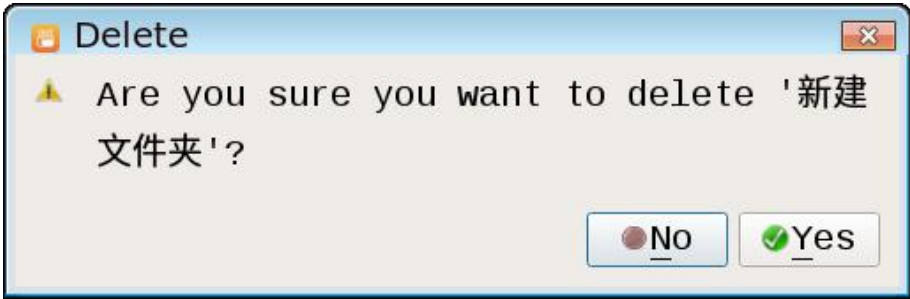
  <message>
    <location filename="qfileiconprovider.cpp" line="317"/>
    <source>Folder</source>
    <comment>All other platforms</comment>
    <translation type="unfinished"></translation>
  </message>
```

</context>

把上面代码拷贝到 qt\_zh\_CN.ts 文件的 <name>QFileDialog</name> 对应选项中。然后翻译，发布。（可以参考第三小节汉化过程）  
汉化之后效果为：



2.3.2 QMessageBox 对话框的汉化问题



2.3.2.1 标题与显示内容的汉化

对话框标题内容以及显示内容，在 qt 库中对应的文件为：

qtbase/src/widgets/dialogs/qfiledialog.cpp

按照上面同样步骤，使用 lupdate 命令生成该文件对应的 ts 文件。对应的内容为：

```
<context>
<name>QFileDialog</name>
...
<message>
    <location filename="qfiledialog.cpp" line="3587"/>
    <location filename="qfiledialog.cpp" line="3592"/>
    <source>Delete</source>
```



```

        <translation type="unfinished"></translation>
    </message>
    ...
    ...
    ...
    <message>
        <location filename="qfiledialog.cpp" line="3593"/>
        <source>Are you sure you want to delete &apos;%1&apos;?</source>
        <translation type="unfinished"></translation>
    </message>
    ...

</context>

```

把上面代码拷贝到 `qt_zh_CN.ts` 文件的 `<name>QFileDialog</name>` 对应选项中。然后翻译，发布。（可以参考第三小节汉化过程）

汉化之后效果为：



### 2.3.2.2 按钮的汉化

按钮的汉化花费的时间比较长。按钮对应的文件也是：

```
qtbase/src/widgets/dialogs/qfiledialog.cpp
```

相关代码：

```

/*!
    \internal

    Deletes the currently selected item in the dialog.
*/
void QFileDialogPrivate::_q_deleteCurrent()
{
    if (model->isReadOnly())
        return;

    QModelIndexList list = qFileDialogUi->listView->selectionModel()->selectedRows();
    for (int i = list.count() - 1; i >= 0; --i) {
        QPersistentModelIndex index = list.at(i);
        if (index == qFileDialogUi->listView->rootIndex())
            continue;
    }
}

```

```

        index = mapToSource(index.sibling(index.row(), 0));
        if (!index.isValid())
            continue;

        QString fileName = index.data(QFileSystemModel::FileNameRole).toString();
        QString filePath = index.data(QFileSystemModel::FilePathRole).toString();

        QFile::Permissions
p(index.parent().data(QFileSystemModel::FilePermissions).toInt());
#ifdef QT_CONFIG(messagebox)
        Q_Q(QFileDialog);
        if (!(p & QFile::WriteUser) && (QMessageBox::warning(q_func(),
QFileDialog::tr("Delete"),
                                QFileDialog::tr("%1' is write protected.\nDo you want
to delete it anyway?")
                                .arg(fileName),
                                QMessageBox::Yes | QMessageBox::No, QMessageBox::No)
== QMessageBox::No))
            return;
        else if (QMessageBox::warning(q_func(), QFileDialog::tr("Delete"),
                                QFileDialog::tr("Are you sure you want to delete
'%1'?")
                                .arg(fileName),
                                QMessageBox::Yes | QMessageBox::No, QMessageBox::No)
== QMessageBox::No)
            return;

        // the event loop has run, we have to validate if the index is valid because the
model might have removed it.
        if (!index.isValid())
            return;
#else
        if (!(p & QFile::WriteUser))
            return;
#endif // QT_CONFIG(messagebox)

        if (model->isDir(index) && !model->fileInfo(index).isSymLink()) {
            if (!removeDirectory(filePath)) {
#ifdef QT_CONFIG(messagebox)
                QMessageBox::warning(q, q->windowTitle(),
                                QFileDialog::tr("Could not delete directory."));
#endif
            }
        } else {
            model->remove(index);
        }
    }
}

```

由上面代码可以知道，按钮是通过 `QMessageBox::Yes` 和 `QMessageBox::No` 来显示的。没有 `tr` 修饰。一下子不知道应该怎么汉化了。在查阅了很多资料后，终于在 `qtbase/src/gui/kernel/qplatformtheme.cpp` 文件中找到相关内容。

还是按照上述方法，使用 `lupdate` 命令生成该文件对应的 `ts` 文件。对应的内容为：

```
<context>
  <name>QPlatformTheme</name>
  ...
  ...
  <message>
    <location filename="qplatformtheme.cpp" line="716"/>
    <source>&Yes</source>
    <translation type="unfinished"></translation>
  </message>
  ...
  <message>
    <location filename="qplatformtheme.cpp" line="720"/>
    <source>&No</source>
    <translation type="unfinished"></translation>
  </message>
  ...
  ...
</context>
```

把上面代码拷贝到 `qt_zh_CN.ts` 文件的 `<name>QPlatformTheme</name>` 选项中。然后翻译，发布。（可以参考第三小节汉化过程）

汉化之后效果为：



### 三 汉化过程

上面过程没有真正涉及到软件汉化的过程。`qt` 软件汉化是通过在 `main` 函数中加载某个 `.qm` 文件，在 `.qm` 文件中找到部件内容对应的汉化内容。

```
QTranslator translator;
qApp->installTranslator(&translator);
translator.load(QString("qt_zh_CN.qm"));
```

汉化的大部分工作是制作 `.qm` 的过程，另一部分工作是在源码中把需要汉化的部件，通过 `tr("")` 进行修饰。比

如：

```
LabelFileType = new QLabel(tr("File type:"));
```

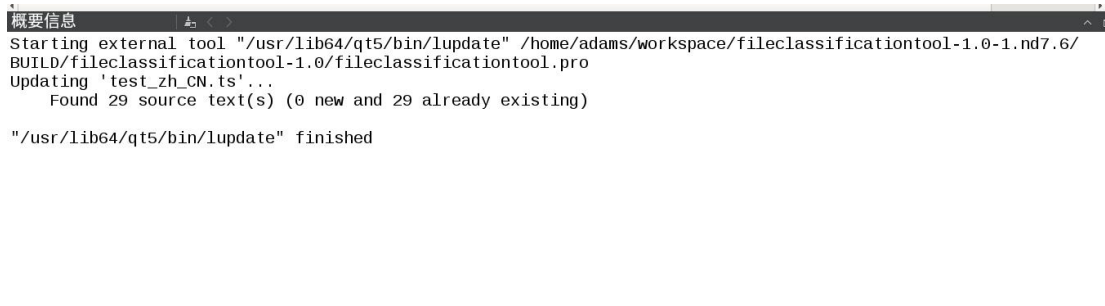
这样，LabelFileType 标签中的内容，就可以被 QT 语言家(Qt Linguist)识别获取了。

### 3.1 生成 ts 文件

想要生成自己应用软件对应的 ts 文件的话，只需要在 pro 文件中加入如下代码即可：

```
TRANSLATIONS = test_zh_CN.ts
```

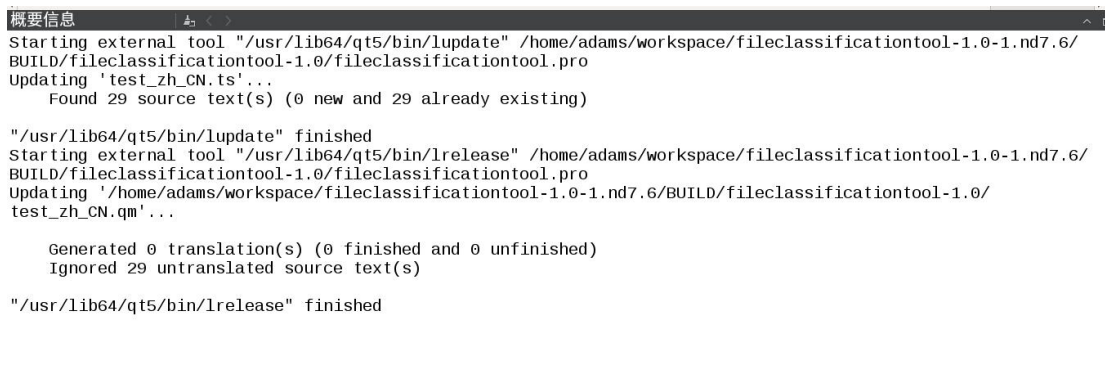
加入如上代码后，在 qtcreeator 的工具->外部->Linguist->更新翻译,可以生成对应的 ts 文件了。qtcreeator 输出信息如下：（本人的调试信息）



```
概要信息
Starting external tool "/usr/lib64/qt5/bin/lupdate" /home/adams/workspace/fileclassificationtool-1.0-1.nd7.6/
BUILD/fileclassificationtool-1.0/fileclassificationtool.pro
Updating 'test_zh_CN.ts'...
Found 29 source text(s) (0 new and 29 already existing)

"/usr/lib64/qt5/bin/lupdate" finished
```

使用 qtcreeator 的工具->外部->Linguist->发布翻译，可以生成对应的 qm 文件。我们不使用该方法，而是使用 QT 语言家(Qt Linguist) 去发布。



```
概要信息
Starting external tool "/usr/lib64/qt5/bin/lupdate" /home/adams/workspace/fileclassificationtool-1.0-1.nd7.6/
BUILD/fileclassificationtool-1.0/fileclassificationtool.pro
Updating 'test_zh_CN.ts'...
Found 29 source text(s) (0 new and 29 already existing)

"/usr/lib64/qt5/bin/lupdate" finished
Starting external tool "/usr/lib64/qt5/bin/lrelease" /home/adams/workspace/fileclassificationtool-1.0-1.nd7.6/
BUILD/fileclassificationtool-1.0/fileclassificationtool.pro
Updating '/home/adams/workspace/fileclassificationtool-1.0-1.nd7.6/BUILD/fileclassificationtool-1.0/
test_zh_CN.qm'...

Generated 0 translation(s) (0 finished and 0 unfinished)
Ignored 29 untranslated source text(s)

"/usr/lib64/qt5/bin/lrelease" finished
```

### 3.2 生成 qm 文件

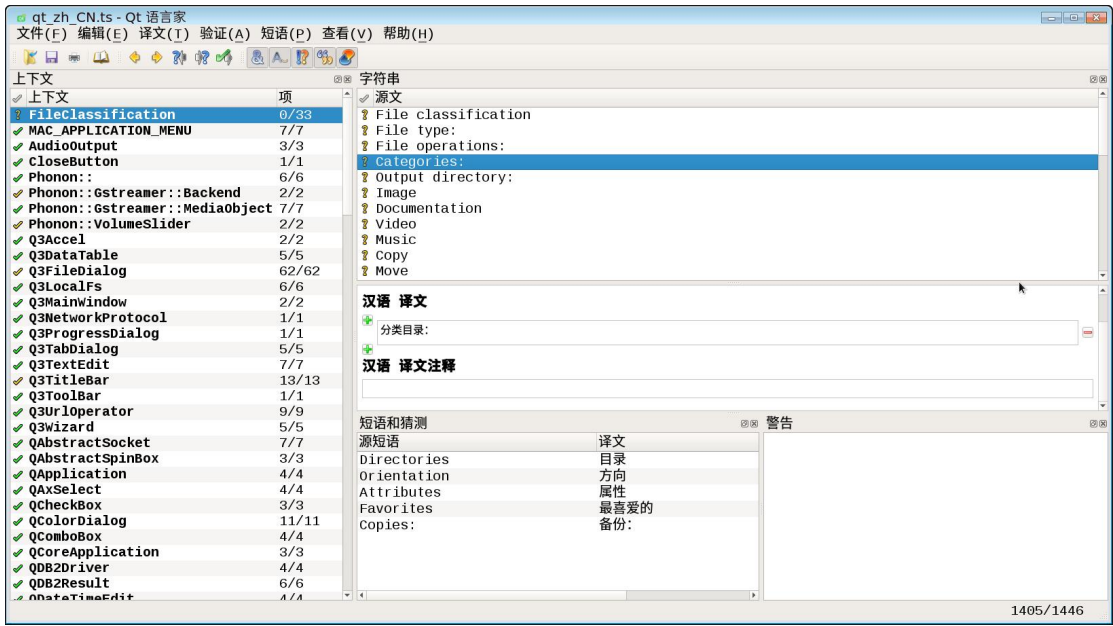
使用自己的应用软件对应的 ts 文件确实可以翻译相应的部件。但是对软件中调用的系统函数弹出的对话框就无能为力了。所以我们要把自己的 ts 文件中内容，复制到 qt 库中提供的 ts 文件中。qt 库提供的 ts 文件在：qttranslations/translation 目录中。我们使用 qt\_zh\_CN.ts 文件。

把 test\_zh\_CN.ts 文件中的下面部分内容拷贝到 qt\_zh\_CN.ts 文件中。...代表省略的内容。

```
<context>
<name>FileClassification</name>
...
...
...
</context>
```

然后打开 Qt Linguist 软件。使用该软件打开 qt\_zh\_CN.ts 文件。然后对 FileClassification 中的内容，

逐条翻译即可。



翻译完成后，点击 文件->发布 命令，在 ts 的同一目录下就生成了对应的 qm 文件。