QT作为一款轻量级的集成开发环境，其设计的目标是使开发人员利用QT这个应用程序框架更加快速及轻易的开发应用程序。要达到此目的，要求QT必须能够跨平台，QT能够在32位及64位的Linux，MAC OS X以及Windows上运行。

现在我们想在项目中加入一项打印功能，即使在不同的操作系统环境下，仍然能有相同的用户体验，在实现中发现QT对打印的支持不是那么友好。我们希望在跨平台的情况下，能打印各种文档，比如用户的简历，发现如果不调用对应操作系统的内核外壳打印函数，这项工作变得异常艰难。我们希望用QT自带的函数库，来实现简单的打印功能，这样就实现了一次编码，多环境使用的目的。

下面我们分别介绍这两种方法。

**一、调用内核外壳打印函数**

Linux下面相对比较容易，使用lpr –p [printer] [fileName]即可，即如果想把file.cpp文件送到打印机pl则使用lpr –p pl file.cpp即可。

Windows下，我们实现打印既定文件路径的文档时，我们调用windows内核命令执行函数，实现打印功能。具体代码如下：

|  |
| --- |
| bool printFile(const QString & filePath)  {  // create a printer  QPrinter printer;  QString printerName = printer.printerName();  // no default printer or no file return false  if( printerName.size() == 0 || filePath.size() == 0)  return false;  QFileInfo fi(filePath);  if(!fi.exists())  return false;  int ret = 0;  ret = (int)ShellExecuteW(NULL,  QString("print").toStdWString().c\_str(),  filePath.toStdWString().c\_str(),  NULL,  NULL,  SW\_HIDE);  // if return number bigger than 31 indicate succ  if (ret > SE\_ERR\_NOASSOC)  return true;  return false;  } |

**二、使用QT自定义的类和函数实现打印文件的功能**

|  |  |
| --- | --- |
| className | function |
| QAbstractPrintDialog | base implementation for print dialogs used to configure printers |
| QPageSetupDialog | configuration dialog for the page-related options on a printer |
| QPrintDialog | Dialog for specifying the printer's configuration |
| QPrintEngine | Defines an interface for how QPrinter interacts with a given printing subsystem |
| QPrintPreviewDialog | Dialog for previewing and configuring page layouts for printer output |
| QPrintPreviewWidget | Widget for previewing page layouts for printer output |
| QPrinter | Paint device that paints on a printer |
| QPrinterInfo | Gives access to information about existing printers |

QT中和打印文件相关的类如下所示，它们分别是：

* 抽象文档打印对话框类，提供配置打印机的打印对话框基本实现；
* 页面设置对话框类，和打印页面相关参数的配置对话框；
* 打印对话框类，指定打印机配置的对话框；
* 打印引擎类，定义了QPrinter类如何与打印子系统交互的接口；
* 打印预览对话框类，用来预览和配置页面布局的对话框；
* 打印预览控件类，预览页面布局的控件；
* 打印机类，指示打印机如何工作。

***QT中实现打印的方法***

在QT中如果想实现打印，首先，我们需要构造一个打印机类（QPrinter）对象， 其次，使用此对象构造一个打印对话框类（QprinterDialog）对象。如下所示：

|  |
| --- |
| QPrinter printer;  QprinterDialog \*dialog = new QprintDialog(&printer, this);  dialog->setWindowtitle(tr(“Print Document”));  if(editor->textCursor().hasSelection())  dialog->addEnabledOption(QabstractPrintDialog::PrintSelection);  if (dialog->exec() != QDialog::Accepted)  return; |

最后附上在项目中，我们如何利用QT自身类，来实现这个跨平台的打印功能。

首先，在头文件中指定槽函数：

|  |
| --- |
| private slots:  void doPrint();  void doPrintPreview();  void printPreview(QPrinter \*printer);  void createPdf();  void setUpPage();  具体实现如下：  void MainWindow::doPrint()  {  // 创建打印机对象  QPrinter printer;  // 创建打印对话框  QString printerName = printer.printerName();  if( printerName.size() == 0)  return;  QPrintDialog dlg(&printer, this);  //如果编辑器中有选中区域，则打印选中区域  if (ui->textEdit->textCursor().hasSelection())  dlg.addEnabledOption(QAbstractPrintDialog::PrintSelection);  // 如果在对话框中按下了打印按钮，则执行打印操作  if (dlg.exec() == QDialog::Accepted)  {  ui->textEdit->print(&printer);  // print the existing document by absoult path  // printFile("D:/myRSM.doc");  }  }  // 打印预览  void MainWindow::doPrintPreview()  {  QPrinter printer;  // 创建打印预览对话框  QPrintPreviewDialog preview(&printer, this);  // 当要生成预览页面时，发射paintRequested()信号  connect(&preview, SIGNAL(paintRequested(QPrinter\*)),  this,SLOT(printPreview(QPrinter\*)));  preview.exec();  }  void MainWindow::printPreview(QPrinter \*printer)  {  ui->textEdit->print(printer);  }  // 生成PDF文件  void MainWindow::createPdf()  {  QString fileName = QFileDialog::getSaveFileName(this, tr("导出PDF文件"), QString(), "\*.pdf");  if (!fileName.isEmpty()) {  // 如果文件后缀为空，则默认使用.pdf  if (QFileInfo(fileName).suffix().isEmpty())  fileName.append(".pdf");  QPrinter printer;  // 指定输出格式为pdf  printer.setOutputFormat(QPrinter::PdfFormat);  printer.setOutputFileName(fileName);  ui->textEdit->print(&printer);  }  }  // 页面设置  void MainWindow::setUpPage()  {  QPrinter printer;  QPageSetupDialog pageSetUpdlg(&printer, this);  if (pageSetUpdlg.exec() == QDialog::Accepted)  {  printer.setOrientation(QPrinter::Landscape);  }  else  {  printer.setOrientation(QPrinter::Portrait);  }  } |