Programação gráfica com Qt - Parte2

Continuando com o tutorial "Programação gráfica com Qt", vamos montar os nossos próprios widgets (isso inclui nossas próprias janelas).

Apenas lembrando: na parte anterior, para colocar mais de um widget na mesma janela, nós criamos um widget básico e indicamos que os demais widgets pertenceriam a este widget básico:

QWidget main_widget(0);
QLabel label1("Olá mundo!!!", &main_widget);
QLabel label2("Bom dia!", &main_widget);

Mas esta não é a maneira correta de se criar janelas. O certo é aproveitar o poder da programação orientada a objetos. Podemos criar um widget com os componentes da janela, além do código para fazer essa janela funcionar... Vamos a um exemplo prático.

Como vimos, sempre que queremos usar um widget diferente, precisamos colocar um #include com a interface do widget no nosso programa. Para widgets personlizados também é necessário. Por isso, vamos criar o nosso mywidget.h:

```
 1: /*****************************************************
 2:  * Programação gráfica com Qt
 3:  *
 4:  * Interface de MyWidget
 5:  *
 6:  * Programa: mywidget.h
 7:  *****************************************************/
 8:
 9: #ifndef MYWIDGET_H
10: #define MYWIDGET_H
11:
12: #include <qwidget.h>
13:
14: class QPushButton;
15: class QLineEdit;
16: class QLabel;
17:
18: class MyWidget : public QWidget {
19: public:
20:     QLabel* c_campo;
21:     QLineEdit* campo;
22:     QPushButton* fechar;
23:
24:     MyWidget(QWidget* parent=0, const char* name=0, WFlags fl=0);
25: };
26:
27: #endif // MYWIDGET_H
```

Agora vamos ao nosso velho passo a passo:

9: #ifndef MYWIDGET_H
10: #define MYWIDGET_H
..: (...)
27: #endif // MYWIDGET_H

Toda a nossa interface está entre esses dois trechos de código. Isso é necessário para evitar que aconteçam dois includes para o mesmo .h. Assim, quando acontecer o primeiro include para o nosso mywidget.h ele vai passar pela linha 9, pois não foi definido ainda o MYWIDGET_H. Porém, ele faz isso na linha seguinte. Se houver um segundo include para o mywidget.h, a linha 9 não vai deixar que a classe MyWidget seja declarada novamente.

12: #include <qwidget.h>

Como a nossa classe será baseada no widget básico, precisamos incluí-lo.

14: class QPushButton; 15: class QLineEdit; 16: class QLabel;

A nossa janela terá botão, caixa de texto e um label. Aqui apenas declaramos que vamos usar esses widgets. Na implementação do MyWidget fazemos os includes. Obviamente, esses class QAlgumaCoisa podem ser substituidos por #include <qalgumacoisa.h>, mas como nem sempre esses componentes serão chamados por outro programa que não a implementação do próprio widget, é conveniente colocarmos os includes apenas lá. Isso poupa serviço do compilador.

18: class MyWidget : public QWidget {

Aqui criamos a nossa classe MyWidget, baseada em QWidget.

19: public:
20: QLabel* c_campo;
21: QLineEdit* campo;
22: QPushButton* fechar;

Declaramos os widgets que vamos colocar dentro do nosso (ou da nossa janela). É uma classe como qualquer outra em C++. Assim, você pode colocar outras variáveis aqui, como um contador, um flag. Lembre-se, por estar abaixo de public: (linha 19), qualquer variável aqui pode ser acessada de qualquer ponto do programa que faça uso do nosso widget. Para fazer com que o acesso seja restrito ao widget, use protected: ou private:.

24: MyWidget(QWidget* parent=0, const char* name=0, WFlags fl=0);

Essa é a declaração do constructor do nosso widget. Ele será chamado sempre que o nosso widget for criado. Veremos sua implementação no mywidget.cpp.

25: };

Parece ridículo eu ter chamado a atenção para esta linha, mas nunca esqueçam do ponto-e-vírgula após o fecha-chave. Alguns compiladores não mostram

mensagens de erro claras quanto a este erro.

Implementação

Agora vamos à impelentação do nosso MyWidget. Grave-o como mywidget.cpp:

```
 1: /****************************************************
 2:  * Programação gráfica com Qt
 3:  *
 4:  * Implementação de MyWidget
 5:  *
 6:  * Programa: mywidget.cpp
 7:  ****************************************************/
 8:
 9: #include "mywidget.h"
10: #include <qpushbutton.h>
11: #include <qlineedit.h>
12: #include <qlabel.h>
13:
14: MyWidget::MyWidget(QWidget* parent, const char* name, WFlags fl)
15:         : QWidget(parent, name, fl)
16: {
17:     setGeometry(100, 100, 300, 200);
18:
19:     c_campo=new QLabel("Campo:", this);
20:     c_campo->move(20, 20);
21:
22:     campo=new QLineEdit(this);
23:     campo->setGeometry(20, 45, 260, 25);
24:
25:     fechar=new QPushButton("&Fechar", this);
26:     fechar->setGeometry(230, 150, 50, 30);
27: }
```

Bom, já temos a implementação. Isso já é suficiente para mostrara a janela, porém, ela ainda não faz nada. Vejamos no passo a passo o que fizemos:

9: #include "mywidget.h"
10: #include <qpushbutton.h>
11: #include <qlineedit.h>
12: #include <qlabel.h>

Primeiro incluímos a interface do MyWidget, para podermos implementá-lo. Em seguida incluimos a interface dos widgets que vamos usar, no caso QPushButton, QLineEdit e QLabel.

14: MyWidget::MyWidget(QWidget* parent, const char* name, WFlags fl)
15: : QWidget(parent, name, fl)

Aqui começa o nosso constructor. Veja que ele recebe três argumentos: parent, name e fl. Esses três argumentos são passados para o constructor do QWidget,

que é a base do MyWidget.

17: setGeometry(100, 100, 300, 200);

Com o setGeometry nós posicionamos e dimensionamos a janela.

19: c_campo=new QLabel("Campo:", this);
20: c_campo->move(20, 20);
21:
22: campo=new QLineEdit(this);
23: campo->setGeometry(20, 45, 260);
24:
25: fechar=new QPushButton("&Fechar", this);
26: fechar->setGeometry(230, 150, 50, 30);

E nas linhas 19-26 nós criamos os widgets que vão compor a nossa janela e os posicionamos na tela. Veja que todos eles pertencem a "this", no caso, o nosso widget. this se refere sempre à classe que estamos criando ou implementando.

Bem fácil, não?

Agora falta o main.cpp, que é o que vai chamar a nossa janela. Podemos usar o hello.cpp que criamos na primeira parte, apenas substituindo alguns nomes (grave-o como main.cpp):

```
 1: /*****************************************************
 2:  * Programação gráfica com Qt
 3:  *
 4:  * Programa: main.cpp
 5:  *****************************************************/
 6:
 7: #include <qapplication.h>
 8: #include "mywidget.h"
 9:
10: int main(int argc, char *argv[])
11: {
12:     QApplication a(argc, argv);
13:
14:     MyWidget main_widget(0);
15:     a.setMainWidget(&main_widget);
16:     main_widget.show();
17:
18:     a.exec();
19: }
```

Pronto! Agora é só compilar tudo...

g++ -I $QTDIR/include -c main.cpp
g++ -I $QTDIR/include -c mywidget.cpp
g++ -L $QTDIR/lib -lqt -o mywidget main.o mywidget.o

E rodar:

./mywidget

Veja o resultado:

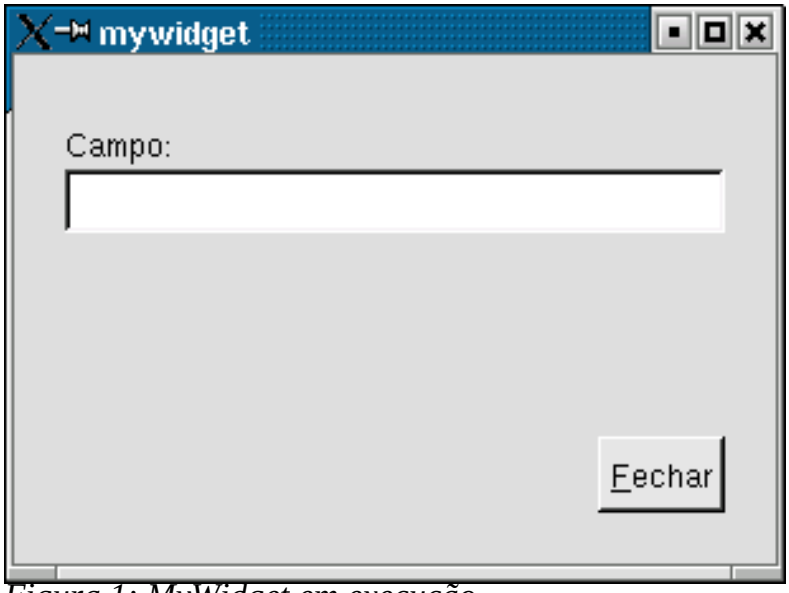

Figura 1: MyWidget em execução

É isso aí... na próxima parte faremos esse botão "Fechar" funcionar. Veremos o funcionamento dos "signals" e "slots" da Qt (conhecido como eventos no Delphi e VB).

Até lá...

Autor: Ricardo Vaz Mannrich
Email:
husk-le@comlinux.com.br