

# Einführung in Qt

- Empfohlene Version 4
- Empfohlene Pakete:
  - qt4-default
  - qt4-doc-html
- Werkzeuge:
  - designer (Designtool)
  - qtcreator (Ide)
  - qmake

# Einführung in Qt

- Empfohlene Version 4
- Empfohlene Pakete:
  - qt4-default
  - qt4-doc-html
- Werkzeuge:
  - designer (Designtool)
  - qtcreator (Ide)
  - qmake

# Einführung in Qt

## Beispiel 1

```
#include <QApplication>
#include <QPushButton>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

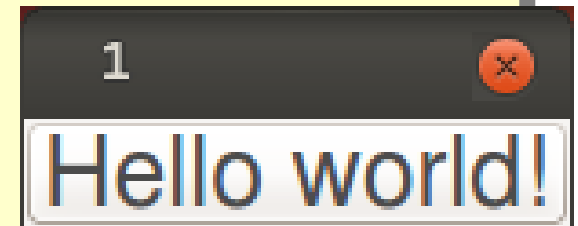
    QPushButton hello( "Hello world!", 0 );
    hello.resize( 100, 30 );

    hello.show();
    return a.exec();
}
```

Parent Widget  
Hier null, weil  
Hauptfenster

# Einführung in Qt

- Verzeichnis anlegen
- Quelltext erfassen und speichern (xxx.cpp)
- Buildschritte:
  - `qmake -project`
  - `qmake`
  - `make`
- Qmake ist nach Änderung von Quellfilenamen oder Hinzufügen von Quellfiles auszuführen.
- Make nach jeder gespeicherten Änderung des Quelltextes.



# Einführung in Qt

- Ein Objekt QApplication beschreibt die gesamte Applikation.
- Ein Objekt QPushButton beschreibt einen Button
- Der Button wird hier zum MainWindow.
- QPushButton->QAbstractButton->QWidget
- show - Methode von QWidget
- Der Button führt zu keiner Aktion, wenn er angeklickt wird
- Eine Beschreibung aller Klassen ist zu finden unter:  
<file:///usr/share/qt4/doc/html/qapplication.html>

# Einführung in Qt

## Beispiel 2


```
#include <QApplication>
#include <QPushButton>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    QPushButton quit( "Quit", 0 );
    quit.resize( 150, 45 );
    quit.setFont( QFont( "Times", 24, QFont::Bold ) );

    QObject::connect(&quit, SIGNAL(clicked()), &a, SLOT(quit()));

    quit.show();
    return a.exec();
}
```



Hier wird die  
Funktionalität  
ergänzt

# Einführung in Qt

## Signals and Slots

Eine Komponente (QWidget) wird mit einer Funktionalität verbunden.

```
QObject::connect      // Funktion, die Widgets
(                    // mit Funktion verbindet
    &quit,            // Widget mit dem etwas passiert
    SIGNAL(clicked()), // Welches Signal wird ausgewertet
    &a,               // Welches Objekt ist zuständig
    SLOT(quit())      // Was soll passieren
);
```

# Einführung in Qt

## Beispiel 3

```
#include <QApplication>
#include <QPushButton>

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

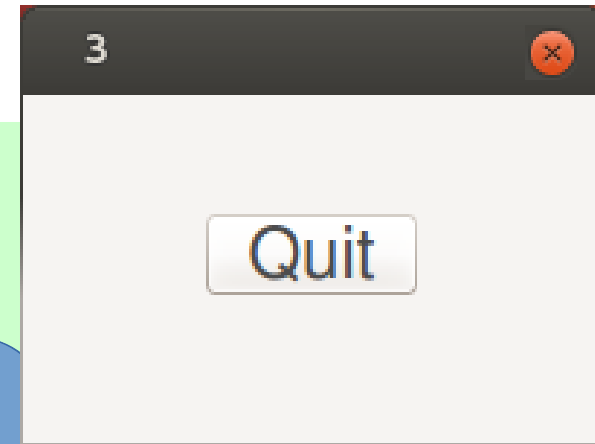
    QWidget w;
    w.resize( 200, 120 );

    QPushButton quit( "Quit", &w );
    quit.move( 62, 40 );
    quit.resize( 75, 30 );
    quit.setFont(QFont("Helvetica",24,QFont::Normal ) );

    QObject::connect(&quit,SIGNAL(clicked()),&a, SLOT(quit()));

    w.show();
    return a.exec();
}
```

Das neue  
Hauptwidget



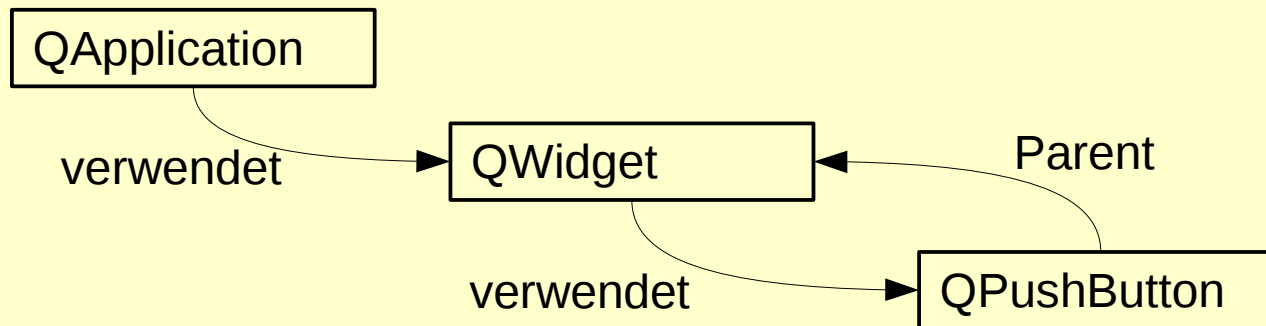


# Einführung in Qt

- QWidget w erzeugt ein den Button umgebendes Window
- w wird ist MainWidget
- QPushButton erhält bei der Erzeugung w als Parentwindow
- QPushButton quit ist das ChildWindow von w
- move ist eine überschriebene Funktion von QWidget in QPushButton und bestimmt die Position des Buttons, resize bestimmt die Größe des Buttons

# Einführung in Qt

Es ergibt sich nun eine Objekthierarchie



# Einführung in Qt

- Einteilung in mehrere Quellfiles
  - Main.cpp
  - Headerfile mit Klassenvereinbarung
  - Implementationsfile dazu
- Mit Vererbung baut man sich eigene Widgets

# Einführung in Qt

## Beispiel 4

```
// Qt4Main.cpp
#include <QApplication>
#include <QWidget>
#include <QPushButton>
#include "qt4.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );

    MyWidget w;
    w.setGeometry( 100, 100, 200, 120 );

    w.show();
    return a.exec();
}
```

MyWidget:  
Durch Vererbung  
spezialisiertes  
Widget

# Einführung in Qt

## Beispiel 4

```
// qt4.h
#ifndef __QT4__H__
#define __QT4__H__

class MyWidget : public QWidget
{
    public: MyWidget( QWidget *parent=0);
};

#endif
```

# Einführung in Qt

## Beispiel 4

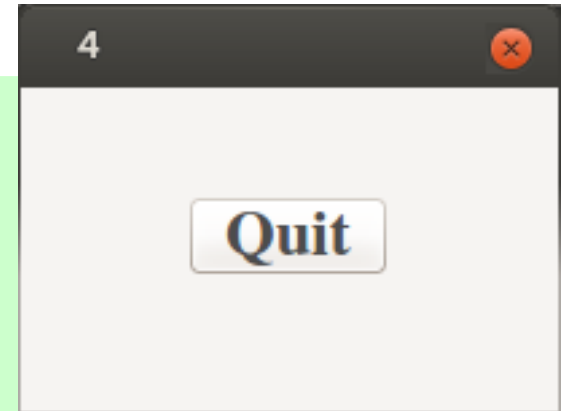
```
// qt4.cpp
```

```
#include <QWidget>
#include <QPushButton>
#include <QCoreApplication>
#include "qt4.h"
```

```
MyWidget::MyWidget( QWidget *parent): QWidget( parent )
{
    setMinimumSize( 200, 120 );
    setMaximumSize( 400, 240 );

    QPushButton *quit = new QPushButton( "Quit", this);
    quit->setGeometry( 62, 40, 75, 30 );
    quit->setFont( QFont( "Times", 18, QFont::Bold ) );

    connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );
}
```



# Einführung in Qt

- In Beispiel 5 werden zwei Widgets miteinander verknüpft.
- Über Signals und Slots tauschen Widgets Informationen aus.
- Der Main-Modul bleibt dabei unverändert.
- Unser Widget soll drei Widgets beinhalten:
  - den Quit PushButton (QPushButton)
  - einen Scrollbar (QScrollBar)
  - Ein LCD -Zahlendisplay (QLCDNumber)

# Einführung in Qt

## Beispiel 5

```
#include <QApplication>
#include <QWidget>
#include <QPushButton>
#include <QScrollBar>
#include <QLCDNumber>
#include <QCoreApplication>
#include "qt5.h"

int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    MyWidget w;
    w.setGeometry( 100, 100, 200, 200 );

    w.show();
    return a.exec();
}
```



# Einführung in Qt

## Beispiel 5

```
#ifndef __QT5__H__
#define __QT5__H__

class MyWidget : public QWidget
{
public:
    MyWidget( QWidget *parent=0);
protected:
    void resizeEvent( QResizeEvent * );
private:
    QPushButton *quit;
    QScrollBar *sBar;
    QLCDNumber *lcd;
};
#endif
```

# Einführung in Qt

## Beispiel 5

```
#include <QApplication>
#include <QWidget>
#include <QPushButton>
#include <QScrollBar>
#include <QLCDNumber>
#include <QCoreApplication>
#include "qt5.h"
```

```
void MyWidget::resizeEvent( QResizeEvent * )
{
    sBar->setGeometry( 10, height()-10-16,width()-20, 16 );
    lcd->resize( sBar->width(), sBar->y() - lcd->y() - 5 );
}
```



this->width()

# Einführung in Qt

## Beispiel 5

```
MyWidget::MyWidget( QWidget *parent)
    : QWidget( parent )
{
    setMinimumSize( 200, 200 );
    quit = new QPushButton( "Quit", this);
    quit->setGeometry( 10, 10, 75, 30 );
    quit->setFont( QFont( "Times", 18, QFont::Bold ) );
    connect( quit, SIGNAL(clicked()), qApp, SLOT(quit()) );
    lcd = new QLCDNumber( 3, this);
    lcd->move( 10, quit->y() + quit->height() + 10 );
    sBar = new QScrollBar( Qt::Horizontal, // orientation
                          this);          // parent, name

    sBar->setMinimum(0);
    sBar->setMaximum(120);
    connect(sBar, SIGNAL(valueChanged(int)),
           lcd ,SLOT(display(int)) );
}
```



# Einführung in Qt

- Die Schritte im Einzelnen:
  - Erzeugen der Widgets
  - Eigenschaften einstellen
  - Verbinden via Signals und Slots
- Dokumentation zu den einzelnen Funktionen dazu ist zu finden unter

**`file:///usr/share/qt4/doc/html/classes.html`**

# Einführung in Qt

- Eventhandling
- Eventhandling erfolgt nach zwei Strategien:
- Überschriebene Eventfunktionen – wenn das Ereignis eintritt, wird die, ggf überschriebene Funktion ausgeführt (resizeEvent)
- Signals/Slots – hier wird ein Observerpattern implementiert. Das Eintreten eines Ereignisses (Signal) wird einem Objekt mitgeteilt in dem die entsprechende Funktion (Slot) aufgerufen wird.

# Einführung in Qt

## Beispiel 6 Fileviewer

```
#include <QObject>
#include <QApplication>
#include <QWidget>
#include <QPushButton>
#include <QCoreApplication>
#include <QMenuBar>
#include <QMenu>
#include <QFileDialog>
#include <QTextEdit>
#include <QString>
#include <QResizeEvent>
#include <QString>
#include "fv.h"
```

Das Main File

```
int main( int argc, char **argv )
{
    QApplication a( argc, argv );
    myWidget w(argv[1]);
    w.show();
    return a.exec();
}
```

# Einführung in Qt

## Beispiel 6 Fileviewer

Hier includes einfügen!

```
#ifndef _H_FV
#define _H_FV
class myWidget : public QWidget
{
    Q_OBJECT

public:
    myWidget(const char* file, QWidget *parent=0);
    virtual ~myWidget(){}
protected:
    void resizeEvent( QResizeEvent * );
private:
    QMenuBar * MenuBar;
    QTextEdit* MLE; // MultiLineEdit

public slots:
    void open ();
};
#endif
```

```
#include <QObject>
#include <QApplication>
#include <QWidget>
#include <QPushButton>
#include <QCoreApplication>
#include <QMenuBar>
#include <QMenu>
#include <QFileDialog>
#include <QTextEdit>
#include <QString>
#include <QResizeEvent>
```

# Einführung in Qt

## Beispiel 6 Fileviewer

```
myWidget::myWidget(const char* file, QWidget *parent)
    :QWidget(parent)
{
    setMinimumSize(200,150);

    MenuBar=new QMenuBar();
    QMenu *File=MenuBar->addMenu("File");
    File->addAction("&Open",this,SLOT(open()));
    File->addAction("&Quit",qApp,SLOT(quit()));

    MLE=new QTextEdit();
    MLE->setMinimumSize(180,100);
    MLE->setFont(QFont("Helvetica", 12, QFont::Normal ));
    MLE->setReadOnly ( FALSE );

    QVBoxLayout *vbox=new QVBoxLayout;
    setLayout(vbox);
    vbox->addWidget(MenuBar);
    vbox->addWidget(MLE);
}
```

```
#include <QObject>
#include <QApplication>
#include <QWidget>
#include <QPushButton>
#include <QCoreApplication>
#include <QMenuBar>
#include <QMenu>
#include <QFileDialog>
#include <QTextEdit>
#include <QString>
#include <QVBoxLayout>
#include <QResizeEvent>
#include "fv.h"
#include <iostream>
#include <fstream>
using namespace std;
```



# Einführung in Qt

- Hier erfolgt die Verknüpfung der Widgets anders.
- QVBoxLayout erhält die von ihm verwalteten Widgets über

```
this->setLayout ( vbox ) ;
```

```
vbox->addWidget ( MenuBar ) ;
```

```
vbox->addWidget ( MLE ) ;
```

# Einführung in Qt

## Beispiel 6 Fileviewer

```
/* Fortsetzung des Quelltextes */

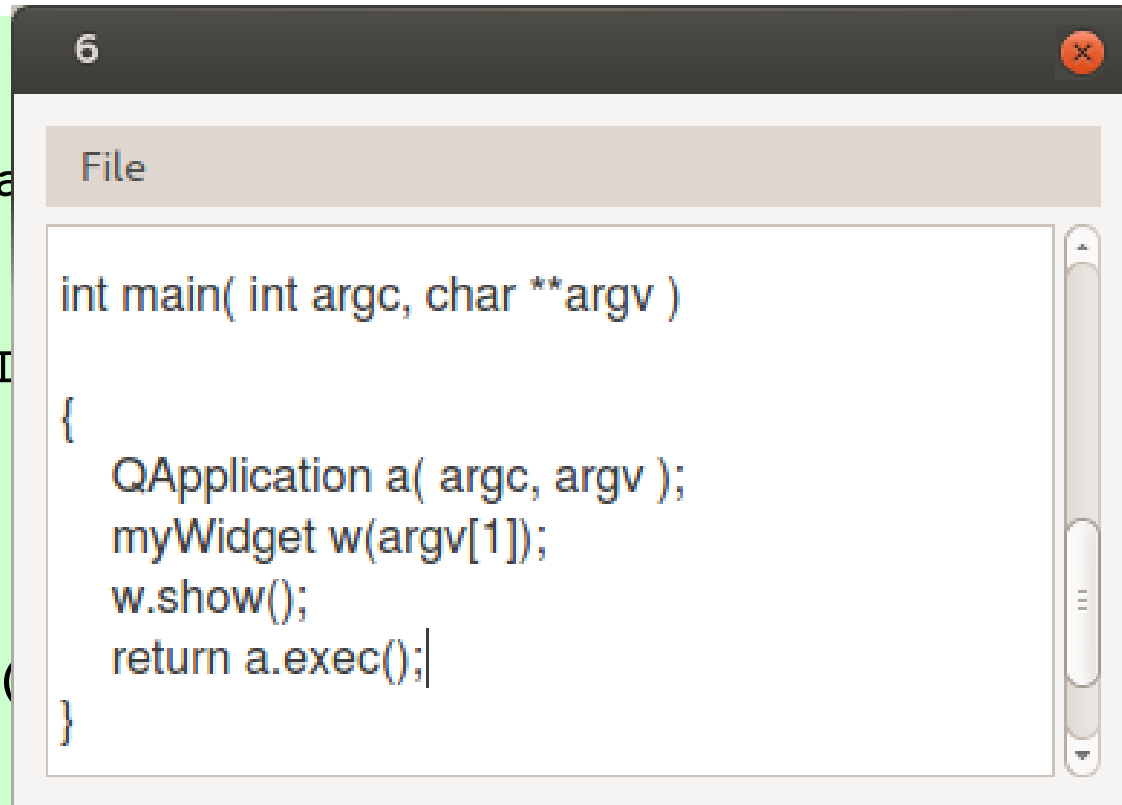
ifstream is(file );
if (is)
{
    char vBuf[1024];
    MLE->clear();
    while(NULL!=is.getline(vBuf,1024)) MLE->append(vBuf);
}
else MLE->append("NIXda");
}

void myWidget::resizeEvent( QResizeEvent * E)
{
    // width und height sind Member von QWidget
    MLE->setGeometry(5,45,width()-10,height()-50);
}
```

# Einführung in Qt

## Beispiel 6 Fileviewer

```
void myWidget::open()
{
    QString FileName=QFileDialog
    if ( !FileName.isEmpty()
    {
        ifstream is(FileName.toI
        if (is)
        {
            char vBuf[1024];
            MLE->clear();
            while(NULL!=is.getline(
        }
    }
    else
    {
        MLE->append("Datei existiert nicht");
    }
    show();
}
```



# Einführung in Qt

- Größere Projekte werden sinnvollerweise mit einer unterstützenden IDE realisiert
- QtCreator bietet sich hier an
- Gute Tutorials dazu unter:
  - [https://de.wikibooks.org/wiki/Qt\\_f%C3%BCr\\_C%2B%2B-Anf%C3%A4nger](https://de.wikibooks.org/wiki/Qt_f%C3%BCr_C%2B%2B-Anf%C3%A4nger)
  - <http://www.eyeshadowpictures.de/main/qt4-02-tabelle.php>