THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Lecture Qt002 Command Line Qt 5.x

Instructor: David J. Coe

CPE 353 – Software Design and Engineering

Department of Electrical and Computer Engineering

Outline

- Hands-On Exercise: ***Hello World*** via Command Line
  - Preparation
  - Code Entry
  - Generate Project File
  - Generate Makefile
  - Compile and Execute
  - Edit Program
  - Recompile and Execute
- Summary - Qt Command Line Compilation
- Qt ***Hello World***
- Key Points

Hands-On Exercise: Command Line Qt Preparation

- Open Linux terminal window
- Enter **cd** to return to your home directory
- Enter **mkdir cpe353** to create a subdirectory
- Enter **cd cpe353** to make **cpe353** your working directory
- Enter **mkdir qthw** to make a **qthw** subdirectory
- Enter **cd qthw** to make **qthw** your working directory
- Enter **gedit qthw.cpp** to create a file named **qthw.cpp**

Hands-On Exercise: Command Line Qt Code Entry

- Use gedit to add the text below to **qthw.cpp**
- Be sure to save your changes before exiting **gedit**

```
// Qt Hello World program - qthw.cpp
#include <QApplication>
#include <QLabel>
int main(int argc, char* argv[])
{
  QApplication myApp(argc,argv);
  QLabel label(“Hello World”);
  label.show();
  return myApp.exec();
}
```

Hands-On Exercise: Command Line Qt Generate Project File

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

- Enter **qmake -project**
  - You have just generated a ***platform independent description*** of the project's components
  - This description is the basis for building your application for any target (Linux, Mac OS X, Windows, Android, iOS, etc.)
- Enter **ls** to view a list of files in your working directory
- To view this project file which has a **.pro** file extension, enter **cat qthw.pro** (or **cat *.pro**)

Hands-On Exercise: Command Line Qt
Verify Project File

- Contents of **qthw.pro** file should include QT line

```
# Contents of qthw.pro
QT = core gui widgets
SOURCES = qthw.cpp
HEADERS +=
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Hands-On Exercise: Command Line Qt Generate Makefile

- Enter **qmake**
  - You have just generated a ***platform specific Makefile*** for the current target environment
- Enter **ls** to view a list of files in your working directory
- To view the contents of **Makefile**,

  enter **cat Makefile** or

  more Makefile
- Note that **Makefile** includes paths to specific libraries located on your machine

Hands-On Exercise: Command Line Qt Compile and Execute

- Enter **make**
  - You have just compiled your Qt program for the current target environment
- Enter **ls** to view a list of files in your working directory
- Assuming that your subdirectory is named **qthw**, you may execute your program by entering

  ./qthw

Hands-On Exercise: Command Line Qt Edit Program

- Use **gedit** to modify the text as shown below
- Be sure to save your changes before exiting **gedit**

```
// Qt Hello World program - qthw.cpp
#include <QApplication>
#include <QLabel>
int main(int argc, char* argv[])
{
  QApplication myApp(argc,argv);
  QLabel label("Qt Hello World");
  label.show();
  return myApp.exec();
}
```

Hands-On Exercise: Command Line Qt Recompile and Execute

- Enter **make**
  - You have just ***recompiled*** and ***relinked*** your modified program for the current target environment
- Assuming that your subdirectory is named **qthw**, you may execute your program by entering

```
./qthw
```

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Summary - Qt Command Line Compilation

- Editing and Compiling a Qt Program from the Command Line
  - Use **qmake -project** to create a platform-independent description of product to be built (**.pro** file)
  - Use **qmake** to generate a platform-specific **Makefile** from the generic project file
  - Use **make** to build your application using the platform-specific makefile
    - If you are using Microsoft compiler, type **nmake**
  - At this point, if you edit existing source files, you can use **make** to rebuild your product
  - If you add new files to your product, you will need to regenerate the **.pro** project file and **Makefile** by repeating the entire procedure

Qt Hello World

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

```
// Qt Hello World program - qthw.cpp
#include <QApplication>
#include <QLabel>
int main(int argc, char* argv[])
{
  QApplication myApp(argc,argv);
  QLabel label("Qt Hello World");
  label.show();
  return myApp.exec();
}
```

- QLabel is a **widget** (a visual element in a user interface), i.e. "window gadget"
  - Any widget may be an application window in Qt
- Widgets hidden by default, **show()** makes them visible
- Widgets may contain other widgets
- QApplication provides the event loop, which keeps window open and waiting for events caused by user actions such as button clicking
  - The **exec()** call initiates the event loop
  - Once started, event loop forwards events to appropriate objects
- Event loop terminated by closing label window

Key Points

- It is possible to develop Qt programs via the Linux Command Line
  - On a Linux system, the steps described above will place all source code and object files within the same working directory
  - In some situations, such as when using **gcov** for test coverage analysis, it is easiest when all files are in the same directory
- In most cases, you will want to develop your programs using the **Qt Creator** integrated development environment