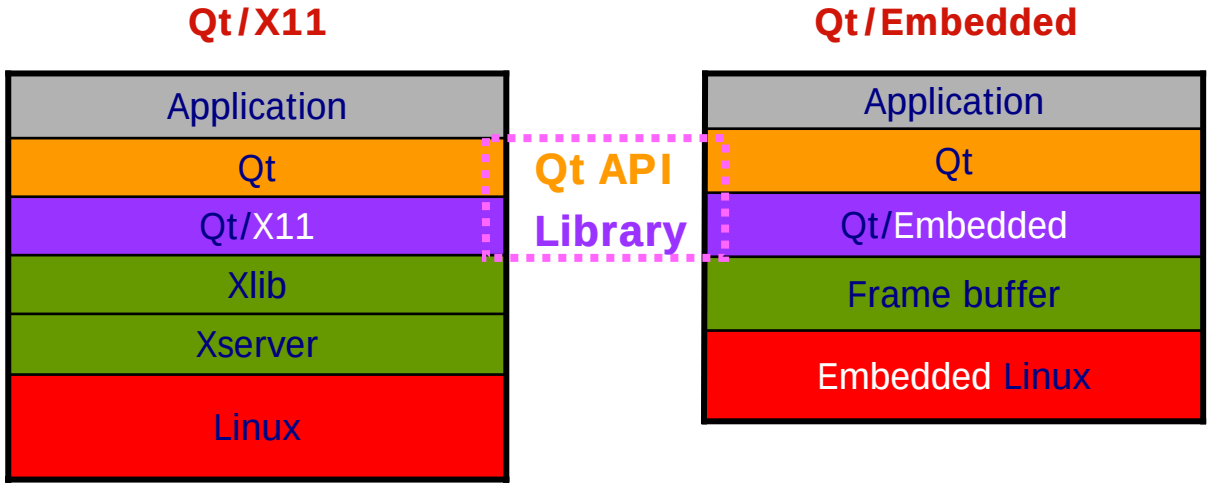
Embedded Systems

Prof. Myung-Eui Lee (F-102)
melee@kut.ac.kr
한국기술교육대학교
korea University of Technology and Education

GUI Applications

- **Embedded target system GUI :**
  - Nano-X, Qt/embedded, MiniGUI, MatchBox, PicoGUI, FLNX, DirectFB, GTK+/FB, TinyX
- **Qt/Embedded : Frame buffer**
  - Designed for embedded device with limited resources
  - Provide full GUI functionality without X11
  - Reduce memory and CPU demand
  - Can be scaled down for dedicated system
  - Fully compatible with Qt/Desktop



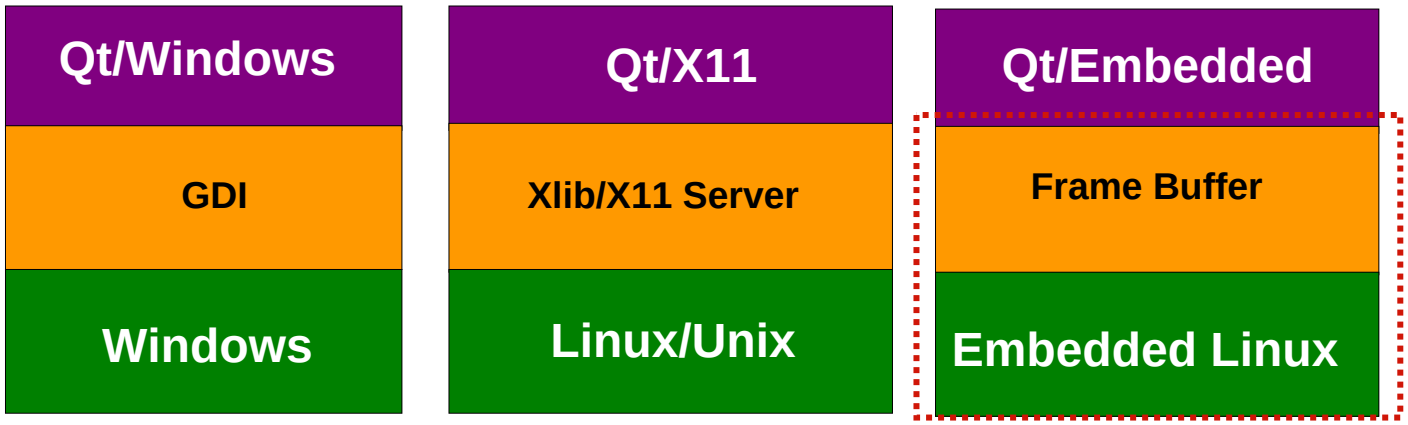
Qt

- **Qt**
  - C++ toolkits for application development
  - Target all major OS with a single application source code
  - Provide platform-independent API
  - Native C APIs are encapsulated with object-oriented C++ classes
- **Qt Desktop : tools for GUI application development**
  - **Qt/Windows** : MS Windows
  - **Qt/X11** : Linux, Solaris, HP-UX, Irix, AIX
  - **Qt/Mac** : Apple Mac OS X
- **Qt Embedded :**
  - **Qt/Embedded** : tools for GUI application development
  - **Qtopia** : window environment and application suite designed for Embedded systems based on Qt/Embedded

Qt

Single Source Qt Application

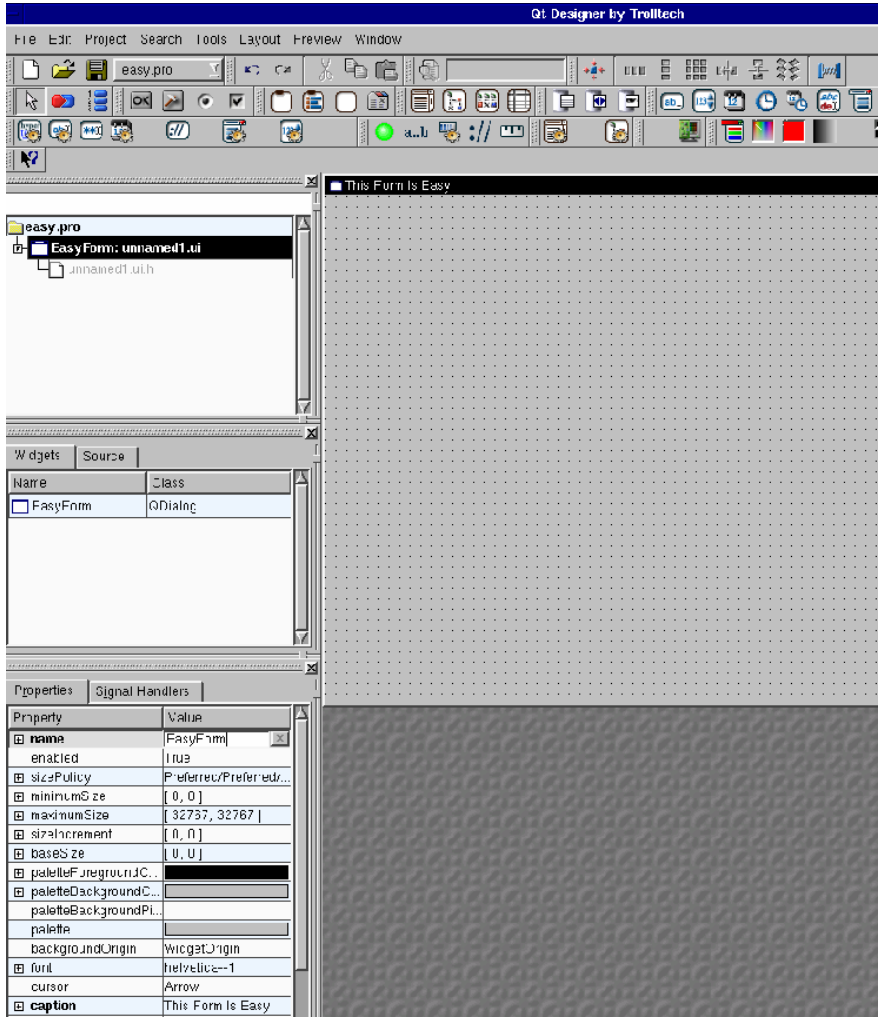
Qt API

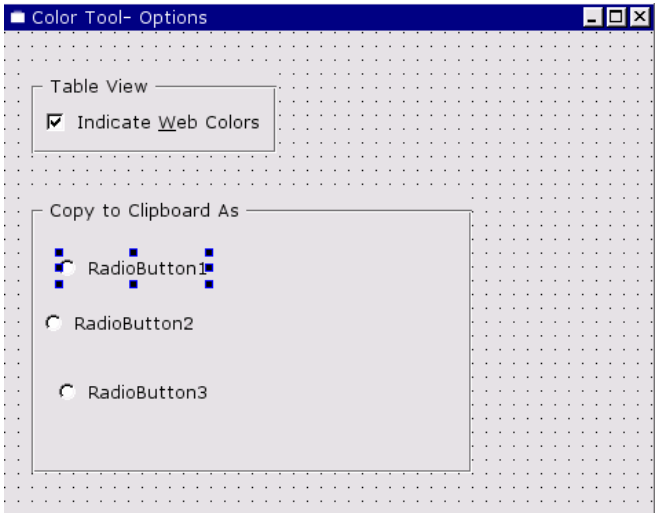


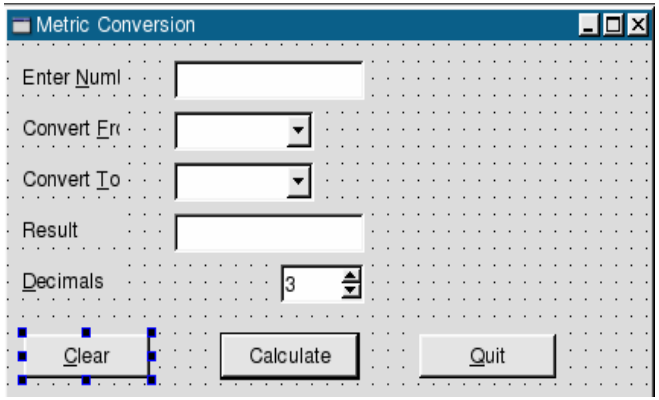
Qt

- **Qt Designer**
  - A visual form designer for Qt provided in Qt/x11
  - Simple implementation and design of user interfaces
  - Allow to add actions and widgets (such as button or tool bar) on the form visually
- **Steps in creating a form with Qt Designer and uic**
  - 1. Start the designer in Qt/X11
  - 2. Add the necessary widgets in a form using the designer
  - 3. It will create a .ui file
  - 4. Run uic to convert the .ui file to a .h file and .cpp file (implementation files)
  - 5. Create main.cpp to generate Qt applications (QApplication object) and start the event loop

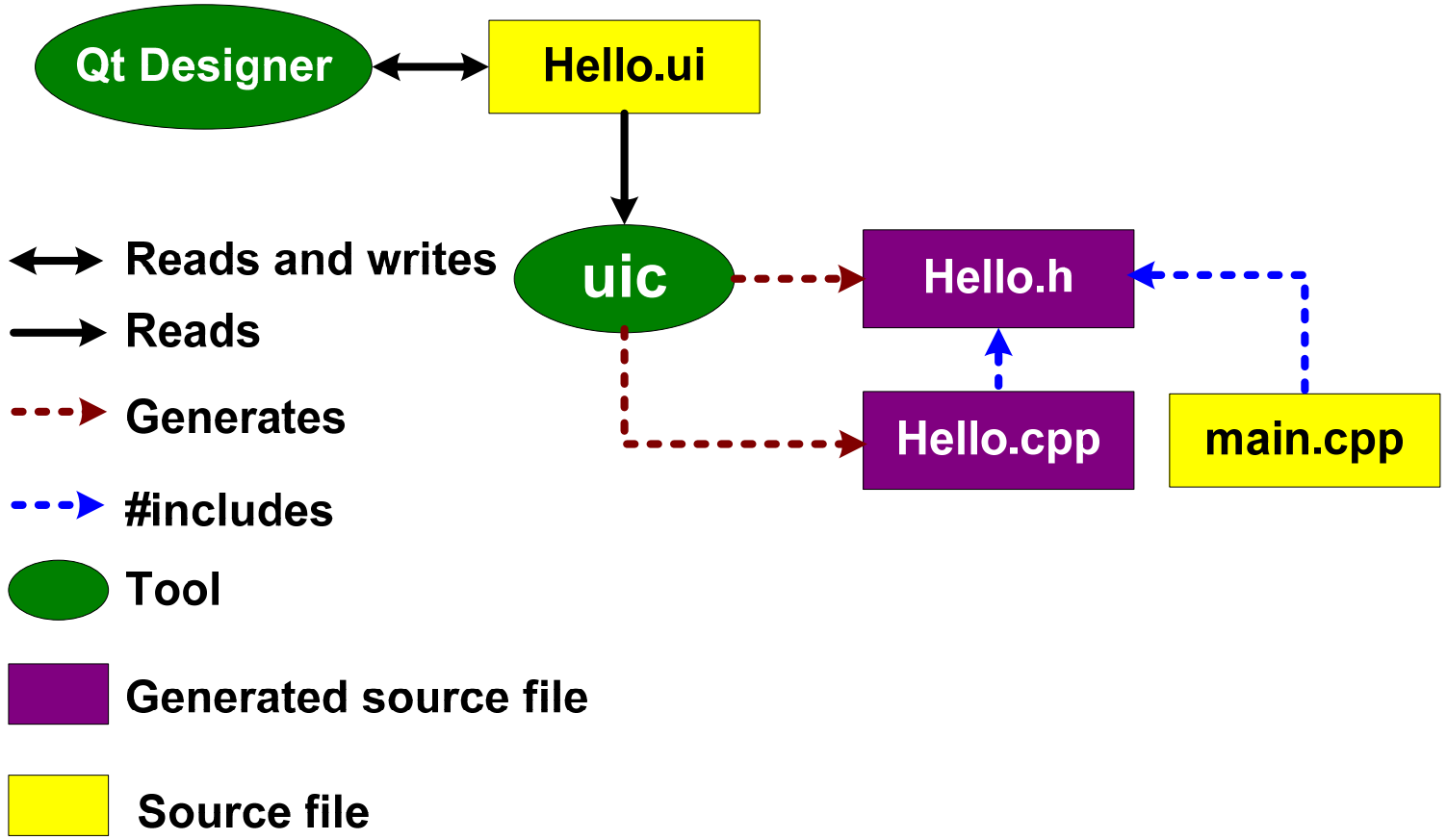
- **Qt Designer**


Qt Designer by Trolltech
File Edit Project Search Tools Layout Preview Window
easy.pro
This Form Is Easy
easy.pro
EasyForm: unnamed1.ui
Widgets
Source
Name
Class
EasyForm
QDialog
Properties
Signal Handlers
Property
Value
name
EasyForm
enabled
true
sizePolicy
minimumSize
maximumSize
sizeIncrement
baseSize
palette
backgroundOrigin
font
helvetica--1
cursor
Arrow
caption
This Form Is Easy


Color Tool- Options
Table View
Indicate Web Colors
Copy to Clipboard As
RadioButton1
RadioButton2
RadioButton3


Metric Conversion
Enter Numl
Convert Fr
Convert To
Result
Decimals
3
Clear
Calculate
Quit

- **Qt files** /root/qt/qte-3.3.3/doc/html/index.html



Qt

- **Qt's Utilities (tools)**
  - **qvfb : run and test embedded applications**
    - tools/qvfb (source) : ***qvfb –skin pda.skin***
  - **uic : the User Interface Compiler**
    - used to convert Qt designer files (.ui) to .h and .cpp files
      uic -o hello.h hello.ui
      uic -i hello.h -o hello.cpp hello.ui
    - /bin - tools/designer (source)
  - **moc : the Meta Object Compiler**
    - required for the signal/slot mechanism
  - **qmake : .pro & Makefile**
    - 2 mode : creating project files (.pro) and makefiles
    - generating .pro from the main.cpp — ***progen***
      qmake –project –o hello.pro main.cpp
    - generating Makefile from the project file (.pro)
      qmake –o Makefile hello.pro
    - run ***Makefile***

Qt

- **Qt = Component Programming**
  - » **Structure the application code in independent, reusable components**
- **Signal and slot mechanisms**
  - » **Provides inter-object communication.**
    - powerful inter-object communication mechanism
  - » **Fast, type-safe, flexible, fully object-oriented and implemented in C++**
  - » **Easy to understand and use**
  - » **Qt widgets emit signals when events occur.**
    - A button will emit a 'clicked' signal when it is clicked. The programmer can choose to connect to a signal by creating a function (called a slot) and calling the connect() function to relate the signal to the slot.
    - If a Quit button's clicked() signal is connected to the application's quit() slot, a user's click on Quit makes the application terminate.

connect(button, SIGNAL(clicked()), qApp, SLOT(quit()));

Qt/X11

- **Qt/X11 Libraries Install**
  - » **microcom homepage** or
    - ftp://ftp.trolltech.com/qt/source/qt-x11-free-3.3.3.tar.bz2
  - » **/root/qt# tar xjvf qt-x11-free-3.3.3.tar.bz2**
  - » **/root/qt# mv qt-x11-free-3.3.3 qtx-3.3.3**

QT/X11 Environment Var.

- » **cd /root/qt/qtx-3.3.3**
- » **/root/qt/qtx-3.3.3# vi ~/.bash_profile**
- » **Add Environment Var.**

```
export QTDIR=/root/qt/qtx-3.3.3
export PATH=$QTDIR/bin:$PATH
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QTDIR/lib
```

- » **/root/qt/qtx-3.3.3# source ~/.bash_profile**
  - **ctl+alt+BS,** and **log-in**

Qt/X11 Configuration & make

- /root/qt/qtx-3.3.3# vi Makefile
  - all: symlinks src-qmake src-moc sub-src : **remove tutorial examples**
- /root/qt/qtx-3.3.3# ./configure –no-xft anti-aliased font
- /root/qt/qtx-3.3.3# make
- /root/qt/qtx-3.3.3# ls –al lib
  - libqt.so, libqt.so.3, libqt.so.3.3, libqt.so.3.3.3

“hello” Compile & Run

- cd /qtx-3.3.3/examples/hello qmake –o Makefile hello.pro
- make
- ./hello or ./hello test

“hello” source

This example brings up the words "Hello, World" moving up and down, and in different colors.

- Header file: hello.h
  Main: main.cpp
  Implementation: hello.cpp

- **main.cpp**

```
int main( int argc, char **argv )
{
   QApplication a(argc,argv); generate Qt application with these arguments

   QString s;
   for ( int i=1; i<argc; i++ ) {
                s += argv[i];
                if ( i<argc-1 )
                   s += " ";
   }
   if ( s.isEmpty() )
                s = "Hello, World";
   Hello h( s );
#ifndef QT_NO_WIDGET_TOPEXTRA    // for Qt/Embedded minimal build
   h.setCaption( "Qt says hello" );
#endif
   QObject::connect( &h, SIGNAL(clicked()), &a, SLOT(quit()) );
   h.setFont( QFont("times",32,QFont::Bold) );                    // default font
   h.setBackgroundColor( Qt::white );                             // default bg color
   a.setMainWidget( &h );
   h.show(); set our main widget to be visible. always call show() in order to make your widget visible
   return a.exec(); finally pass the control to Qt.  exec() keeps running till the application is alive and returns when the
application exits
}
```

Qt/Embedded

- **Qt/Embedded Libraries Install**

  Qt/Embedded Download & Extract

  - **microcom homepage** or
    - ftp://ftp.trolltech.com/qt/source/qt-embedded-free-3.3.3.tar.bz2
  - **/root/qt# tar xjvf qt-embedded-free-3.3.3.tar.bz2**
  - **/root/qt# mv qt-embedded-free-3.3.3 qte-3.3.3**

  QT/Embedded Environment Var.

  - **cd /root/qt/qte-3.3.3**
  - **/root/qt/qte-3.3.3# vi ~/.bash_profile**
  - **Add Environment Var.**

    export QTDIR=/root/qt/qte-3.3.3
    export PATH=$QTDIR/bin:$PATH
    export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QTDIR/lib

  - **/root/qt/qte-3.3.3# source ~/.bash_profile** ctl+alt+BS, log-in

Touch Screen Device Driver

- cd /root/qt/qte-3.3.3/src/embedded
- vi qmouselinuxtp_qws.cpp
- Change Touch Screen Device Driver
  - line number : 155, 157, 159

```
 if ((mouseFD = open( "/dev/ts", O_RDONLY | O_NDELAY)) < 0) {
# else
   if ((mouseFD = open( "/dev/ts", O_RDONLY | O_NDELAY)) < 0) {
# endif
            qWarning( "Cannot open /dev/ts (%s)", strerror(errno));
```

Qt/Embedded

Qt/Embedded Configuration

- /root/qt/qte-3.3.3# ./configure –embedded ipaq –shared –depths 16 –no-cups –qt-libjpeg –qt-mouse-linuxtp
  - ./configure –help
    - embedded arch - enable the embedded build
    - shared - create and use a shared Qt library (libqt.so).
    - depths - bit-per-pixel depths, from: 4, 8, 16, 24, and 32.
    - no-cups - do not compile CUPS (Common Unix Printer System).
    - qt-libjpeg - use the libjpeg bundled with Qt.
    - qt-mouse- enable a mouse <driver> in the Qt Library.
  - PLATFORMS = arch : *vi /root/qt/qtx-3.3.3/PLATFORMS*
    - generic - Use g++ of host machine.
    - x86 - Use g++ of host machine, add x86 options.
    - arm - Use arm-linux-g++ as compiler.
    - mips - Use mipsel-linux-g++ as compiler.
    - ipaq - Use arm config, add iPAQ options (touch panel).
    - sharp - Use arm config, add SHARP Zaurus options.

Qt/Embedded

C++ Library for Qt/Embedded Compile

- **Download library from microcom**
  - /mnt/share/qt/qt_libstdc++-3libc6.1.2.2.10.0.so
- **cp qt_libstdc++-3-libc6.1.2.2.10.0.so /root/qt/qte-3.3.3/lib**
- **cd /root/qt/qte-3.3.3/lib**
- **ln –s qt_libstdc++-3-libc6.1.2.2.10.0.so /opt/iwmmxt-1.0.0/lib/libstdc++.so**
- **cp qt_libstdc++-3-libc6.1.2.2.10.0.so /opt/iwmmxt-1.0.0/lib/libstdc++.so**

UIC Utility Copy

- **cd /root/qt/qtx-3.3.3/bin** : tools/designer/uic
- **cp uic /root/qt/qte-3.3.3/bin**
  - ***uic*** for Linux (Qt/X11) : currently running under Linux

Qt/Embedded

Compile Qt/Embedded Library

- **cd /root/qt/qte-3.3.3**
- **vi Makefile**
  - all: symlinks src-qmake src-moc sub-src ... : delete **tutorial examples**
- **make**

List Library

- **cd /qte-3.3.3/lib**
- **ls –al**
  - **libqte.so, libqte.so.3, libqte.so.3.3, libqte.so.3.3.3**

Qt/Embedded Libraries Install is finished !

Qt/Embedded

- **"hello" Compile**
  - cd /qte-3.3.3/examples/hello
  - make
  - file hello
- **NFS Setting** : Host
  - vi /etc/exports
    - /mnt/nfs * (rw, no_root_squash)
  - service nfs restart
- **Qt/Embedded Copy** : Host
  - mkdir /mnt/nfs, and cd /root/qt
  - cp –rf qte-3.3.3/ /mnt/nfs/
  - ls /mnt/nfs
    - qte-3.3.3 (dir)

192.168.1.100 : Host IP
192.168.1.128 : Target IP

Qt/Embedded

- **Target IP Setting** : target
  - ifconfig eth0 192.168.1.128
  - ping 192.168.1.100 : host ip
- **Qt/Embedded Mounting** : target
  - mkdir /mnt/nfs
  - mount –t nfs 192.168.1.100:/mnt/nfs /mnt/nfs
  - df or (cd /mnt/nfs and ls –al)
- **Target Environment Var.**
  - vi ~/.profile

```
export QTDIR=/mnt/nfs/qte-3.3.3
export LD_LIBRARY_PATH=/mnt/nfs/qte-3.3.3/lib:$LD_LIBRARY_PATH
export QWS_MOUSE_PROTO=linuxtp:/dev/ts
```

Qt/Embedded

- **"hello" run**
  - **ps** : matchbox-desktop App.
  - **kill -9 1031**
  - **cd /mnt/nfs/qte-3.3.3/examples/hello**
  - **./hello test –qws**
    - qws - Q Windows System
    - Qt built-in window system (frame buffer) is used
      - → Using Q Windows System Server
      - → Not using X Windows System Server

- **etc.**
  - **ftp://ftp.trolltech.com/ - all files**
  - **doc/html/index.html – all helps**

Qt/Embedded

- **.bash_profile**
  - /root/runX_hybus.sh
- **runX_hybus.sh**
  - Xfbdev & : X server using frame buffer /usr/X11R6/bin
  - matchbox-desktop …….. & : /usr/bin
  - matchbox-panel --orientation south &
  - matchbox-window-manager &

Applications running (nor flash)

- **Target Environment Var.**
  - **vi ~/.profile**

    export QTDIR=/???/qte-3.3.3
    export LD_LIBRARY_PATH=/???/qte-3.3.3/lib:$LD_LIBRARY_PATH
    export QWS_MOUSE_PROTO=linuxtp:/dev/ts

  - **vi ~/.bash_profile**
    - delete /root/runX_hybus.sh : do not need Xfbdev
      - → Qt use the built-in Q Windows System (with –qws option)
    - /???/qte-3.3.3/examples/hello/hello test &
- **File system**
  - **cd /pxa270/filesystem/rootfs**
  - **rm –rf rootfs.img**
  - **./mkjffs2 : script → mkfs.jffs2**
  - **cp rootfs.img /tftpboot**
  - **tftp rootfs.img root**

etc.

- **New Logo**
  - kernel/linux-2.6.11-h270-tku_v1.1/drivers/video/logo
    - logo_linux_clut224.ppm
- **Conversion to ppm file (Three ways)**
  - 1. pngtopnm logo.png | pnmtoplainpnm
    > logo_linux_clut224.ppm
  - 2. pngtopnm logo.png | ppmquant -fs 224 | pnmtoplainpnm
    > logo_linux_clut224.ppm
  - 3. convert .bmp (.png) to .ppm using gimp utility
    - gimp is automatically installed when you type “gimp” first time
  - color : less than 224
  - size : less than 640 x 480
  - cp logo_linux_clut224.ppm kernel/linux-2.6.11-h270-tku_v1.1/drivers/video/logo

etc.

- **Kernel compile**
  - make zImage
  - cp arch/arm/boot/zImage /tftpboot
  - service xinetd restart
  - tftp zImage kernel (at target)
  - flash kernel