Curso Qt com Linux Embarcado - Extras

Data: 29/10/2018 **Revisão:** 1.0

1. Configurando evdev-touchscreen e evdev-keyboard no Linux e Qt:

```
# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      stmpe-ts
/dev/input/event1:      gpio-keys
Select the device event number [0-1]:

QT_QPA_EVDEV_KEYBOARD_PARAMETERS=/dev/input/event1
QT_QPA_EVDEV_TOUCHSCREEN_PARAMETERS=/dev/input/event0
```

2. Habilitar v-sync Qt:

```
QT_QPA_EGLFS_FORCEVSYNC=1
```

3. Obter informações adicionais sobre EGLFS e Inputs(Mouse, TouchScreen, Teclado):

```
QT_LOGGING_RULES=qt.qpa.*=true
```

Alguns itens específicos para menos verbose geral:
qt.qpa.gl : GL, usar QT_QPA_EGLFS_DEBUG é uma saída formada e não tão aglomerada
qt.qpa.input : Input Handlers
qt.multimedia.* : Qt Multimedia

4. Irá imprimir cada plugin C++ carregado ou tentou:

```
QT_DEBUG_PLUGINS=1
```

Saída particial:

```
Found metadata in lib /usr/lib/qt/plugins/egldeviceintegrations/libqeglfs-viv-integration.so,
metadata=
{
   "MetaData": { "Keys": ["eglfs_viv"]},
   "className": "QEglFSVivIntegrationPlugin",
   "debug": false,
   "version": 330497
}
Got keys from plugin meta data ("eglfs_viv")


Found metadata in lib /usr/lib/qt/plugins/imageformats/libqsvg.so
Got keys from plugin meta data ("svg", "svgz")

Found metadata in lib /usr/lib/qt/plugins/imageformats/libqtiff.so
Got keys from plugin meta data ("tiff", "tif")

loaded library "/usr/lib/qt/plugins/imageformats/libqicns.so"
loaded library "/usr/lib/qt/plugins/imageformats/libqico.so"
loaded library "/usr/lib/qt/plugins/imageformats/libqsvg.so"
loaded library "/usr/lib/qt/plugins/imageformats/libqtga.so"
loaded library "/usr/lib/qt/plugins/imageformats/libqtiff.so"
loaded library "/usr/lib/qt/plugins/imageformats/libqwbmp.so"
loaded library "/usr/lib/qt/plugins/imageformats/libqwebp.so"


QFactoryLoader::QFactoryLoader() looking at "/usr/lib/qt/plugins/generic/libqtslibplugin.so"
Found metadata in lib /usr/lib/qt/plugins/generic/libqtslibplugin.so, metadata=
{
   "IID": "org.qt-project.Qt.QGenericPluginFactoryInterface",
   "MetaData": {"Keys": ["Tslib", "TslibRaw”]},
   "className": "QTsLibPlugin",
   "debug": false,
   "version": 330497
}
Got keys from plugin meta data ("tslib", "tslibraw")

Found metadata in lib /usr/qml/QtGraphicalEffects/libqtgraphicaleffectsplugin.so, metadata=
{
   "IID": "org.qt-project.Qt.QQmlExtensionInterface/1.0",
   "MetaData": {
```

```
    },
    "className": "QtGraphicalEffectsPlugin",
    "debug": false,
    "uri": [
        "QtGraphicalEffects"
    "IID": "org.qt-project.qt.qpa.egl.QEglFSDeviceIntegrationFactoryInterface.5.5",
    ],
    "version": 330497
}
…
…
…
```

5. Configurando High DPI:

Por padrão no main.cpp já é adicionado QcoreApplication::setAttribute(Qt::AA_EnableHighDpiScaling), porém, é possível exportar uma variavel para tal caso não tenha adicionado em código:

```
QT_AUTO_SCREEN_SCALE_FACTOR ="1"
```

6. Imprimir variáveis ambiente do terminal lançado a aplicação:

Adicione o seguinte código em sua aplicação:

```
#include <QProcessEnvironment>

...
QProcessEnvironment procEnv;
qDebug() << "Variables: " << procEnv.systemEnvironment().toStringList();
...
```

Saída:

```
Variables:  ("EDITOR=/bin/vi", "FB_MULTI_BUFFER=2", "HOME=/root", "LOGNAME=root",
"MAIL=/var/mail/root", "PAGER=/bin/more", "PATH=/bin:/sbin:/usr/bin:/usr/sbin", "PS1=# ",
"PWD=/root", "SHELL=/bin/sh", "SHLVL=1", "SSH_CLIENT=192.168.0.1 59562 22",
"SSH_CONNECTION=192.168.0.1 59562 192.168.0.2 22", "USER=root")
```

7. Para saber os QPA suportados, passe um QPA invalido:

```
# /opt/qtcon2018/bin/b2weather -platform abc
qt.qpa.plugin: Could not find the Qt platform plugin "abc" in ""
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.

Available platform plugins are: eglfs, minimal, minimalegl, offscreen, vnc.

Aborted
```

8. Obter o PID do processo:

```
qDebug() << "PID: " << QCoreApplication::applicationPid();
```

9. Obtendo timestamp:

```
#include <QDateTime>

...
qDebug() << "Timestamp(ms): " << QDateTime::currentMSecsSinceEpoch();
qDebug() << "Timestamp(s) : " << QDateTime::currentSecsSinceEpoch();
...
```

10. Converter int para string:

```
int number1{42};
qDebug() << "Numero: " << QString::number(number1, 10);
qDebug() << "Hex    : " << QString::number(number1, 16);
```

Saída:

```
Numero: "42"
Hex     : "2a"
```

11. Gerando números randômicos, Qt >= 5.10:

```
#include <QRandomGenerator>

for(int i=0; i<10; i++) {
  qDebug() << "Random    : " << QRandomGenerator::global()->generate();
  qDebug() << "Bounded   : " << QRandomGenerator::global()->bounded(42);
}
```

Saída:

```
Random : 3350992337

Bounded: 6
Random : 1172277482
Bounded: 24
Random : 3179183866
Bounded: 8
Random : 3946650371
Bounded: 25
Random : 1637693796
Bounded: 33
Random : 1823602369
Bounded: 24
Random : 1030572474
Bounded: 41
Random : 4085660319
Bounded: 6
Random : 726771504
Bounded: 0
Random : 2194147590
Bounded: 4
```