Linux Programozás

Grafikus programok

XWindow

X Window (röviden: X)

- Unix/Linux rendszerek grafikus felülete
- MIT projekt
- Úgy alakították ki, hogy illeszkedjen a Unix multiuser, multitask struktúrájába.
- Történelme:
  - 1984: Athena projekt
  - 1988: X konzorcium
- Implementációk: XFree86, X.Org

Az X felépítése

- Kliens-szerver architektúra
- Szerver
  - Kezeli a képernyőt és a bemeneti eszközöket.
  - Elfedi a HW-t.
- Kliens
  - Lényegében az alkalmazások.
  - Üzenetekkel utasítják a szervert.
  - A HW-hez nem férnek hozzá.
- A kliens-szerver kommunikáció hálózaton keresztül is mehet.
- A kommunikációt az Xlib könyvtár tartalmazza.

Az X felépítése

- Ablakkezelő (Window Manager)
  - „Az ablak kerete”
  - Az ablakok kezelését (kiválasztás, mozgatás, méretezés) teszi lehetővé.
  - Kliens program – akár menet közben is lecserélhető

X kliens alkalmazás fejlesztése

- Xlib használatával
  - Alap szintű – szinte csak a rajzolási primitíveket tartalmazza.
  - A vezérlőelemeket nekünk kell leimplementálni.
- Widget könyvtárak
  - Xlib-re épülnek.
  - Számos vezérlő elemet implementálnak.
  - A program megjelenése a widget könyvtártól függ.
- Widget könyvtárak történelme
  - Motif
  - Gtk
  - Qt
  - LessTif

Szabad választás problematikája

- Ablakkezelő
  - Felhasználó választja
  - Befolyásolja a klienseken kívüli felület és az ablakkeretek működését.
- Widget könyvtár
  - Fejlesztő választja
  - A kliens program felületének megjelenését és működését befolyásolja.
- Különböző programok – különböző widget könyvtárak
  - Eltérően megjelenő és működő programok.
  - Minden használt widget könyvtár betöltődik a memóriába.

Asztali környezet (Desktop Environment)

- Az egységes felület érdekében:
  - Néhány ablakkezelő támogatása – egységes működés
  - Widget könyvtár
  - Launcher
  - Központi konfiguráció és szolgáltatások
  - Módszertan
- Példák:
  - CDE (Common Desktop Environment) – Motif
  - KDE (K Desktop Environment) – Qt
  - GNOME – Gtk

Qt

Qt

- Cross-platform C++ osztálykönyvtár (kb. 500 osztály, 4000 metódus)
  - Eleinte csak UI
  - Később kiegészítettek egy általános alkalmazás keretrendszerré
- Tartalmaz néhány nyelvi kiegészítést, melyek túlmutatnak standard C++-on
- Saját deklaratív UI leíró nyelv: Qt Quick (QML)
- Saját fejlesztőeszközök, IDE (is)
- Open-source (GPL + több fajta licenc)

Qt szolgáltatások 1/3

- UI
  - Széles widget választék
  - UI editor
  - Adott platform look-and-feel
- 2D grafika
  - Saját objektum-orientált világmodell, nézetekkel (kb. mini játék-engine)
  - Nagyszámú elem jeleníthető meg
- 3D grafika

Qt szolgáltatások 2/3

- Szálkezelés, eseményvezérelt programozás
  - Eseménykezelő ciklus
  - Signals and slots
- Kommunikáció
  - HTTP, FTP, TCP/UDP socketek
- XML
- WebKit
  - Az Apple és a Google által is használt böngészőmotor

Qt szolgáltatások 3/3

- Adatbázis motor
  - Összes főbb adatbázis támogatása
- QtScript
  - Saját ECMAScript (~JavaScript, ActionScript) alapú script motor
  - Objektumok futási időben módosíthatok scripteken keresztül (property-k, illetve signals and slots is)
- Mobilspecifikus funkciók: Qt Mobility

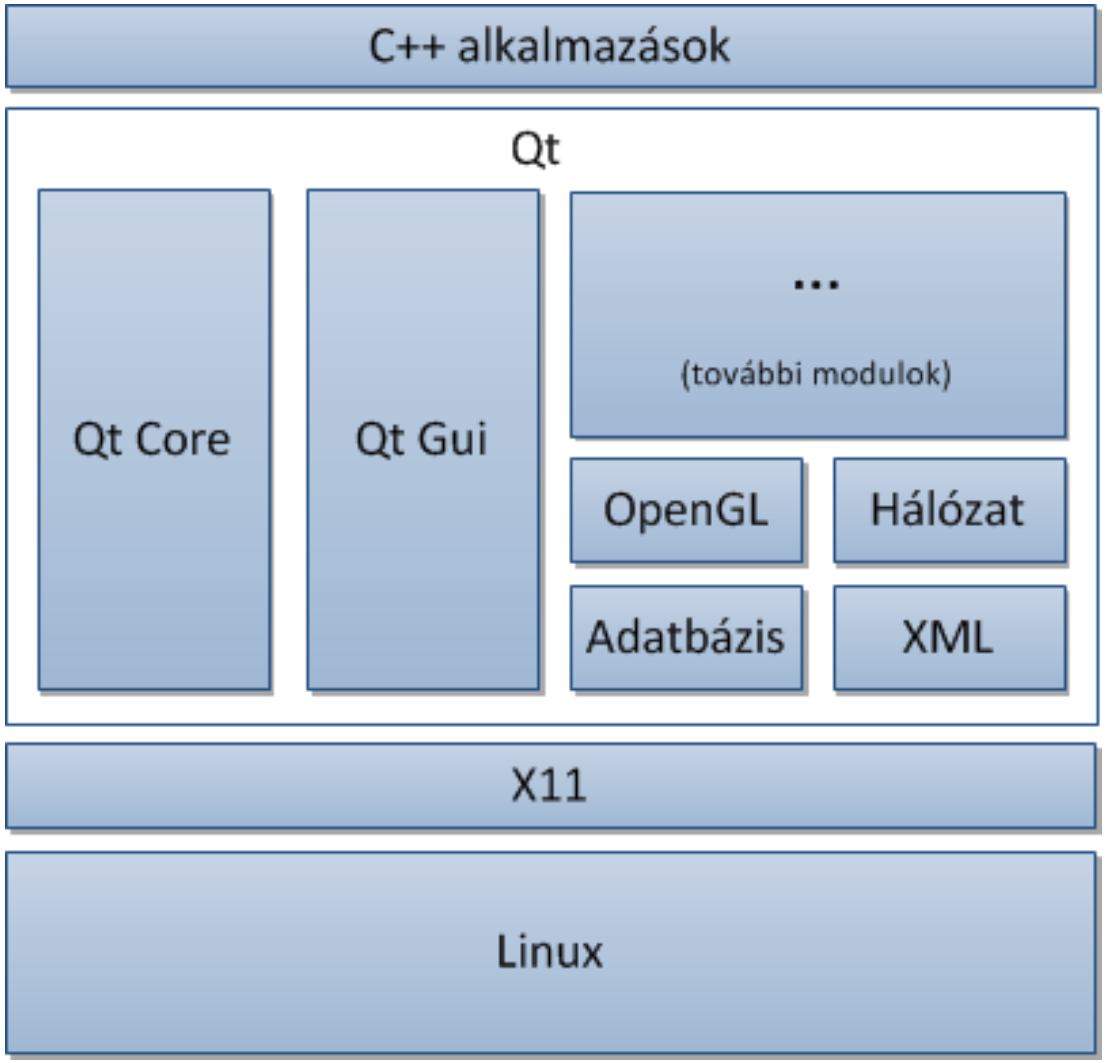
Qt történelem

- 1994: megjelenik az első Qt
  - Két norvég programozó munkája, céljuk: „megalkotni a világ legjobb C++ GUI keretrendszerét”
  - „Q” (tetszett a betű), „t” (toolkit)
  - A fejlesztő cég: Quasar Technologies → Troll Tech → Trolltech → Qt Software
- 1997: Qt 1.2, felhasználják KDE megírásához
  - Qt a Linux GUI fejlesztés egyik de facto standardja
- 1999-2005: Qt folyamatosan bővül, több támogatott platform, funkciók bővítése
- 2008: Nokia felvásárolja a Trolltech-et
- 2012: A Digia megvásárolja a Qt-t a Nokia-tól

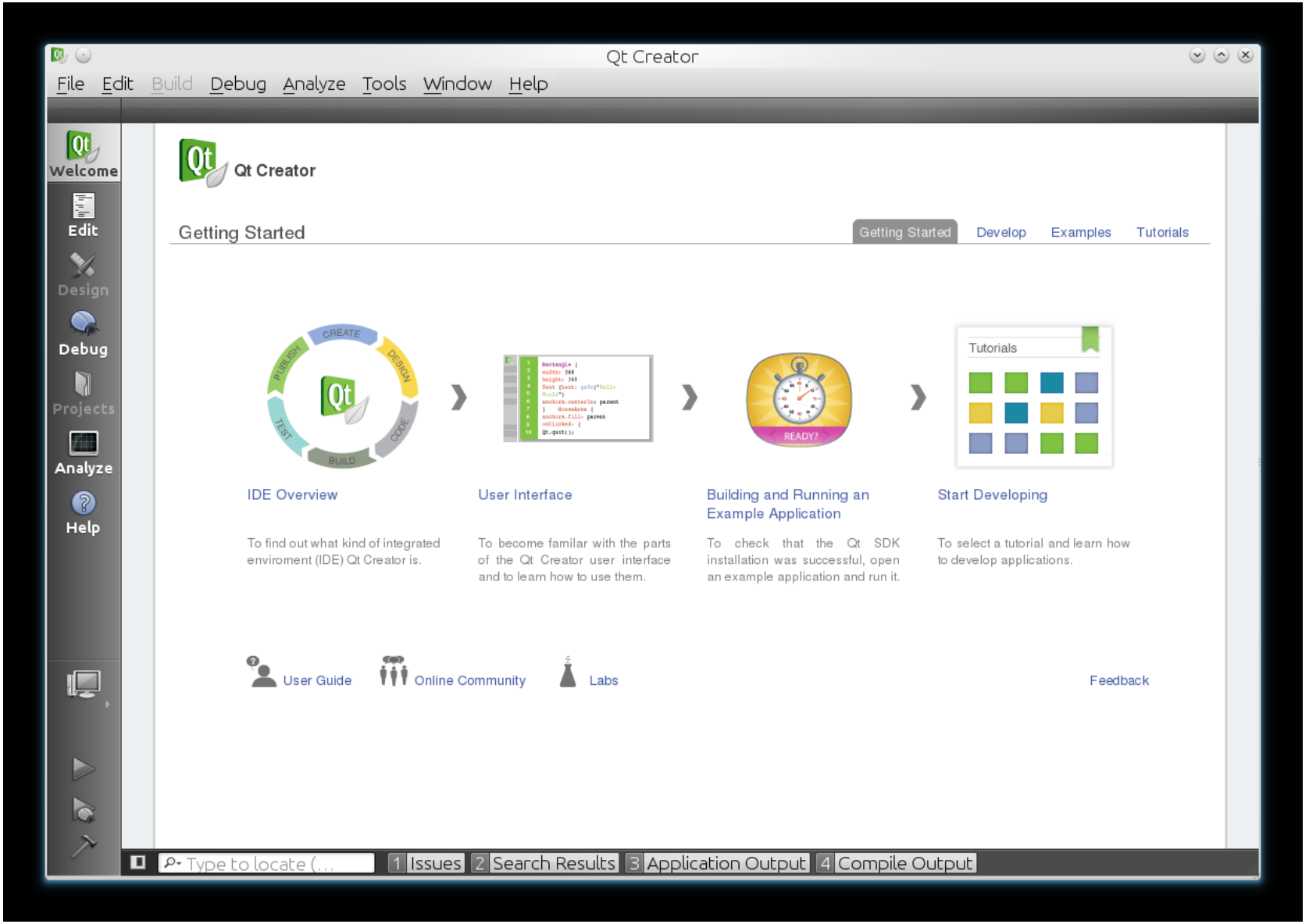
Támogatott platformok

- Támogatott platformok:
  - Linux
  - Windows
  - Mac OS X
  - Solaris
  - Beágyazott Linux
  - Windows CE (NEM Windows Phone)
- Minden platformra ugyanaz a C++ kód fordul le
- Qt funkcióit egyes platformokon más programozási nyelvekből is el lehet érni (language binding), pl. Java, C#, Python

Qt architektúra



IDE: Qt Creator



Hello World

```
#include <QApplication>
#include <QLabel>

int main(int argc, char* argv[])
{
  QApplication app(argc,argv);
  QLabel label("Hello World!");
  label.show();
  return app.exec();
}
```

HelloWorld – Osztályok

- QApplication
  - GUI alkalmazásonként egy példány
  - Futtatja az alkalmazás eseménykezelő ciklusát
  - Menedzseli az alkalmazás-szintű erőforrásokat (default betűkészlet, egérkurzor, stb.)
- QLabel
  - Egy alap megjelenítő widget (QWidget-ből származik)

Projektleíró, fordítás

- Platformfüggetlen projektleíró fájl (.pro)
  - Összefogja a projekthez tartozó fájlokat
  - Konkrét, platform specifikus make fájlok ez alapján generálódnak (qmake)
- Fordítás az adott platformra tipikusan gcc-vel (GNU make hívja)
  - Parancssorból vagy Qt Designer-el

Platform független projektleíró

- A projekt adatait (forrásfájlok, elérési utak, library-k) a projektleíró .pro fájl tartalmazza

```
TEMPLATE = app
TARGET =
DEPENDPATH += .
INCLUDEPATH += .

# Input
SOURCES += qhello.cpp
```

Projektfájlok létrehozása

- Platform független projektleíró (.pro)
  - Lehetőség van manuálisan megírni
  - Automatikus generálás: qmake -project
    - Végignézi a könyvtárban található forrásfájlokat (.h, .cpp, .ui) és létrehozza .pro fájlt
- Platformspecifikus állományok: qmake
  - Legenerálja a szükséges állományokat
    - Linux esetén Makefile-ok

Projekt fordítása

- A qmake-el legeneráltuk a Makefile-okat
- make

Qt object model

- Funkciók
  - **Signals and slots**
  - **Properties**
  - **Események és esemény szűrők**
  - **Objektum hierarchia (fa-alapú)**
  - Védett pointerek
  - Futás idejű típusazonosítás (RTTI)
  - Internalizáció, nyelvfüggő sztringkezelés
- Legtöbb funkció QObject-re épül

QObject

- Qt objektumok ősosztálya, sok extrával
- Nem adható át érték szerint
  - Egyedi objektumokat reprezentálnak (nem „értékeket”)
  - Nevet adhatunk a példányoknak: QObject::objectName()
  - Copy-constructor nem publikus
  - Másolás helyett „klónozás”: clone()

QObject hierarchia

- Fa-alapú hierarchia:
  - Minden objektumhoz maximum egy szülő
  - Tetszőleges számú gyereke lehet
- Egy QObject gyerekei törlődnek a szülő törlésekor
- Gyerek törlésekor automatikusan kikerül a szülő gyerekei közül

QObject lekérdezések

- Gyerek objektumok megtalálása

```
QList<T> parentObj.findChildren<T> (const
QString& name) const;
```

  - T típusszűrő, ha name üres sztring, akkor visszatér az összes T típusú gyerekkel

- Példa, az összes „Product” típusú gyerek lekérdezése

```
QList<Product*> childlist =
parent.findChildren<Product*>();


foreach (Product* current, childlist) {
  qDebug() << current->toString();
}
```

Óvatosan!

- Stack-en létrehozott objektumokkal csak óvatosan

```
{
  QObject* parent = new QObject();
  QObject child(parent);
  delete parent; // child is törlődik
} // child-ot újra törli a runtime a
stack visszagörgetésekor, FATAL ERROR
```

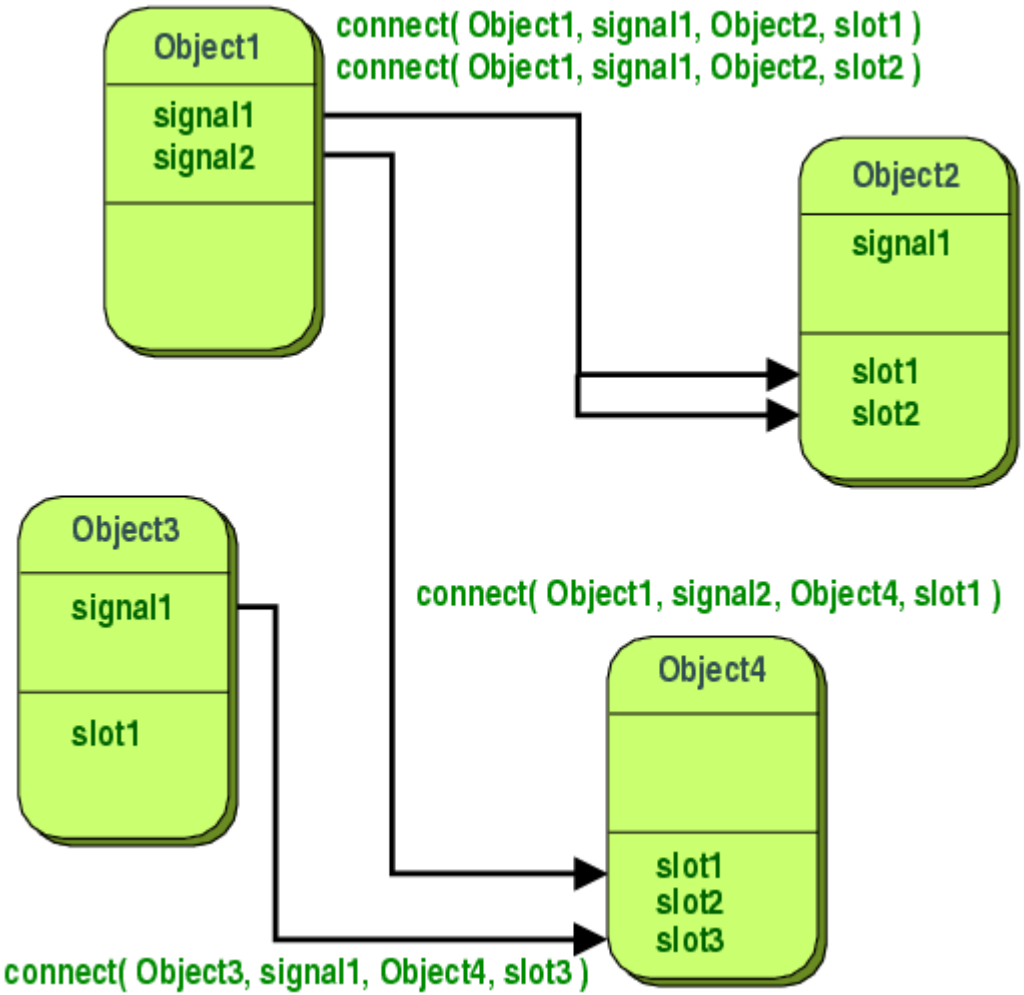
Observer design pattern

- Definíció
  - Objektumok az állapotuk megváltozásáról szeretnék értesíteni egymást anélkül, hogy az függjön a konkrét osztálytípustól
- Lehetséges megvalósítások:
  - Standard C++: interfész öröklés
  - C: callback (függvény mutató)
  - .NET: delegate
  - Qt: Signals and slots

Signal – slot mechanizmus

- Qt által használt speciális mechanizmus az eseménykibocsátó és -feldolgozó objektumok összekapcsolásához
- Signal: egy eseményforrás (üzenet, jelzés)
  - Egy adott osztályhoz/objektumhoz tartozik
  - Egy osztályhoz több signal is tartozhat
  - „Ki lehet bocsátani” (ez jelenti az esemény bekövetkeztét)
- Slot: egy eseményfeldolgozó
  - Egy objektum metódusa, melyet meg szeretnénk hívni adott esemény bekövetkeztekor
- A signal-ok összekapcsolhatók a slot-okkal
  - A „slot” meghívódik a „signal” kibocsátásakor

Signal – slot mechanizmus



Signal

```
#include <QObject>
#include <QString>
class TextContent : public QObject
{
    Q_OBJECT // kötelező Q_OBJECT makró
public:
    void setText(const char * newText);
signals:
    void textChanged(const char * newText);
private:
    QString * text;
};

void TextContent::setText(const char *newText){
    text = new QString(newText);
    emit textChanged(text->data());
}
```

Slot

```
#include <QObject>
class ContentWatcher : public QObject
{
    Q_OBJECT
public slots:
    void contentChanged(const char * newText);
};

void ContentWatcher::contentChanged(const char *
newText){
    std::cout << "Content has changed: " << newText;
}
```

Összekapcsolás

```
#include <QObject>
bool QObject::connect(const QObject * sender,
const char * signal, const QObject *
receiver, const char *method,
Qt::ConnectionType type =
Qt::AutoConnection )
```

- sender: Forrás objektum
- signal: Signál
- receiver: Cél objektum
- method: Slot függvény
- type: A kapcsolat típusa

Kapcsolat típusok

Típus	Leírás
Qt::AutoConnection	Ez az alapértelmezett típus. Ilyenkor a rendszer automatikusan választ aszerint, hogy a szignált publikáló és a szlotot definiáló objektumok egy szálban vannak-e.
Qt::DirectConnection	A szlotot azonnal meghívjuk, amikor a szignált kibocsátjuk.
Qt::QueuedConnection	A szignált akkor hívja meg a keretrendszer, amikor a szlotfüggvényt tartalmazó objektum szálának az eseménykezelő ciklusa feldolgozza a szignál hatására küldött üzenetet.
Qt::BlockingQueuedConnection	Ugyanúgy működik, mint a QueuedConnection, csak a szignál kibocsátásakor a küldő szál blokkolódik, amíg a szlot függvény le nem fut. (Csak különböző szálak esetén használható.)
Qt::UniqueConnection	Ugyanúgy működik, mint az AutoConnection, de csak akkor történik meg az összekapcsolás, ha az korábban még nem létezett.

Slot átmeneti objektumban

- Megsemmisítés előtt meg kell szüntetni az összekötést.

```
bool QObject::disconnect(const
QObject * sender, const char *
signal, const QObject * receiver,
const char * method )
```

- Használhatunk NULL értéket, mert egy szűrőt adunk meg.

Meta Object Compiler (MOC)

- Qt által használt C++ kiegészítések megvalósítását végzi
  - pl. Signals and slots, metaobjektum, properties
- Forrásfájlokat dolgozza fel (szövegfeldolgozó)
  - Végignézi a .h fájlokat és ha olyan osztályt talál, melyben Q_OBJECT direktíva szerepel, abból standard C++ forrást generál
  - osztály.h -> moc_osztály.cpp