Pemrograman Qt 9
QProcess dan Menjalankan Perintah Linux

Ade Malsasa Akbar

2 Desember 2013

Ringkasan

Bismillahirrahmanirrahim. Kita bisa mengomando Linux melalui GUI. Bagaimana caranya? Ada banyak cara. Lihat dulu contoh semisal Ubuntu Tweak. Aplikasi tersebut menggantikan ratusan perintah Terminal dengan beberapa tombol saja. Jika Anda menekan satu tombol di Ubuntu Tweak, maka itu berarti menjalankan perintah tertentu. Konsep ini (GUI front-end) sangat berguna jika kita ingin membuat aplikasi yang melakukan otomatisasi perintah Terminal yang biasa kita kerjakan. Misalnya kita ingin mengganti alamat sumber repositori Ubuntu. Apa yang kita lakukan? Ubah sources.list secara manual, ketik sendiri alamat-alamat yang banyak itu dari hafalan Anda, lakukan update. Hal yang seperti ini bisa dibuatkan GUI supaya otomatis dengan menyimpan alamat-alamat sumber repositori lalu menambahkan tombol untuk masing-masing repositori. Jadi, cukup satu klik untuk mengganti repositori kita ke server Kambing atau UGM atau yang lain. Ini contoh saja. Sekarang kita akan membuat yang lebih sederhana dari itu. Kita akan memakai QProcess (sebagai ganti *method* system() kemarin) untuk menjalankan perintah Linux lebih canggih lagi di dalam Qt. Mengapa kita beralih ke QProcess? Nanti kita akan tahu, insya Allah.

Daftar Isi

1 Persiapan

1.1 Spesifikasi Sistem

1. Ubuntu 12.04
2. Qt Creator 2.4.1
3. Qt 4.8.0 (32 bit)

1.2 Daftar Kelas

1. QProcess
2. QStringList
3. QByteArray

1.3 Daftar Method

1. start() milik QProcess.
2. waitForFinished() milik QProcess.
3. readAll() milik QProcess.
4. printf() method standar iostream dari C++.

2 Teori

2.1 Arah Tulisan Ini

Saya hanya ingin menunjukkan bagaimana aplikasi Qt bisa dibuat seperti contoh di atas, seperti Ubuntu Tweak yang satu tombolnya menggerakkan beberapa perintah Terminal. Dan saya bilang kali ini lebih canggih karena kita bisa menangkap keluaran dari perintah yang dijalankan. Ini penting untuk mewujudkan aplikasi-aplikasi yang dapat mengeksekusi perintah Terminal, menangkap keluarannya, memroses keluaran tersebut, lalu membuat keluaran baru. Contoh nyatanya adalah aplikasi pengganti sources.list otomatis pada Ringkasan (halaman 1).

3 Pemrograman

3.1 Kode

3.1.1 mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtGui>

class Dialog : public QDialog
{
    Q_OBJECT

public:
    Dialog();
    QVBoxLayout *layout;
    QPushButton *tombola;
    QPushButton *tombolb;
    QPushButton *tombolc;

public slots:
    void perintah_cat();
    void perintah_mkdir();
    void perintah_ls();

private:

};

#endif // MAINWINDOW_H
```

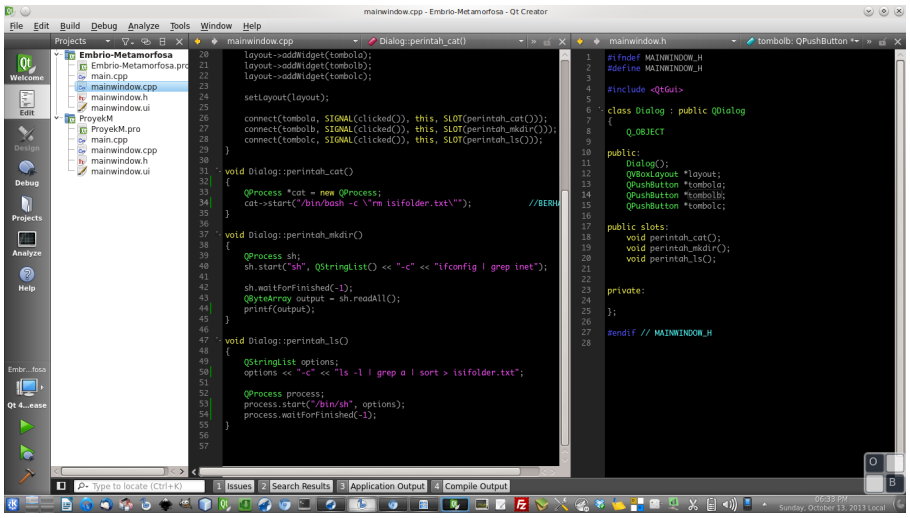
3.1.2 mainwindow.cpp

```
#include <iostream>
#include <QtGui>
#include <mainwindow.h>

Dialog::Dialog()   //kagak usah dikasih void
{
    layout  = new QVBoxLayout;
    tombola = new QPushButton("cat");
    tombolb = new QPushButton("mkdir");
    tombolc = new QPushButton("ls");

    tombola->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding
    );
    tombolb->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding
    );
    tombolc->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding
    );
```

```
15
16      tombola->setMinimumSize(88,55);
17      tombolb->setMinimumSize(88,55);
18      tombolc->setMinimumSize(88,55);
19
20      layout->addWidget(tombola);
21      layout->addWidget(tombolb);
22      layout->addWidget(tombolc);
23
24      setLayout(layout);
25
26      connect(tombola, SIGNAL(clicked()), this, SLOT(perintah_cat()));
27      connect(tombolb, SIGNAL(clicked()), this, SLOT(perintah_mkdir()));
28      connect(tombolc, SIGNAL(clicked()), this, SLOT(perintah_ls()));
29  }
30
31  void Dialog::perintah_cat()
32  {
33      QProcess *cat = new QProcess;
34      cat->start("/bin/bash -c \"rm isifolder.txt\"");
35  }
36
37  void Dialog::perintah_mkdir()
38  {
39      QProcess sh;
40      sh.start("sh", QStringList() << "-c" << "ifconfig | grep inet");
41
42      sh.waitForFinished(-1);
43      QByteArray output = sh.readAll();
44      printf(output);
45  }
46
47  void Dialog::perintah_ls()
48  {
49      QStringList options;
50      options << "-c" << "ls -l | grep a | sort > isifolder.txt";
51
52      QProcess process;
53      process.start("/bin/sh", options);
54      process.waitForFinished(-1);
55  }
```

3.1.3 Qt Creator dan Kode

Panel sebelah kiri berisi mainwindow.cpp dan sebelah kanan mainwindow.h.

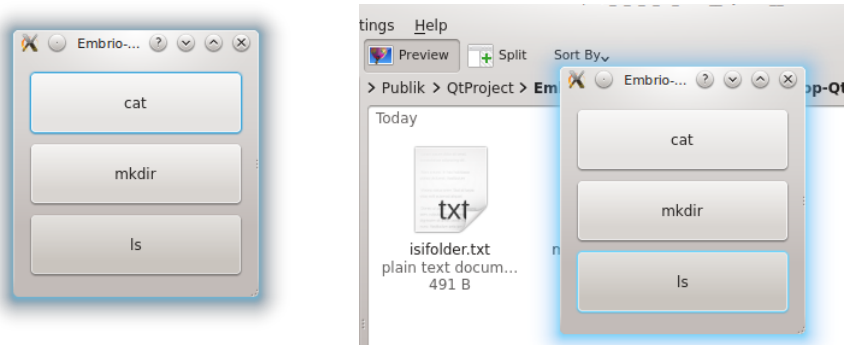


Gambar 1: Qt Creator Menunjukkan Kode-Kode

3.1.4 Hasil Kode

Tombol ls

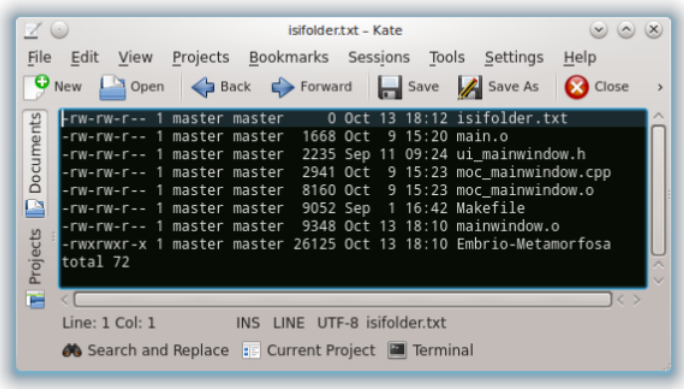
Tampilan program kali ini sama dengan sebelumnya. Bedanya, isi perintah untuk masing-masing tombol saya ubah. Saya akan jelaskan satu per satu hasil perintah mulai dari tombol ketiga.



Gambar 2: Hasil Tombol **ls**

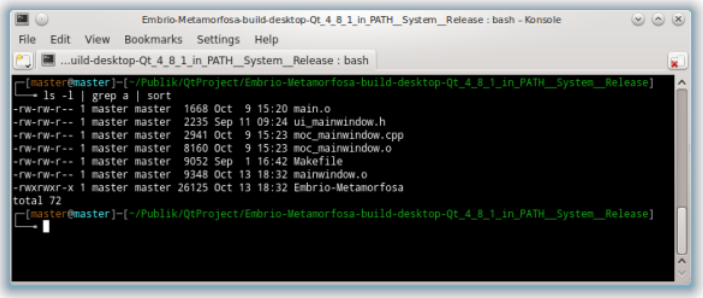
Isi tombol **ls** adalah perintah untuk mendaftar isi folder tempat si program berada, menyaring supaya hanya informasi yang memiliki huruf a saja yang ditampilkan, lalu menuliskannya ke sebuah berkas teks bernama **isifolder.txt**.

Jika tombol **ls** ditekan, maka perintah **ls -l | grep -a > isifolder.txt dijalankan**. Maka jadilah satu berkas teks bernama **isifolder.txt**.



Gambar 3: Hasil *Concatetation* Ditayangkan di Kate

Demikian isi dari berkas isifolder.txt.

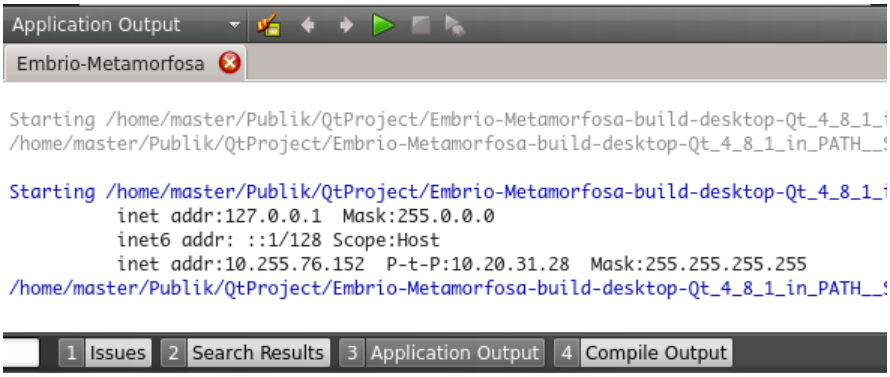


Gambar 4: Hasil *Concatenation* Ditayangkan di Konsole

Demikian keluaran (standard output) dari perintah yang sama tetapi dijalankan dari Terminal. Sama persis dengan isi teks. Ini berarti program valid.

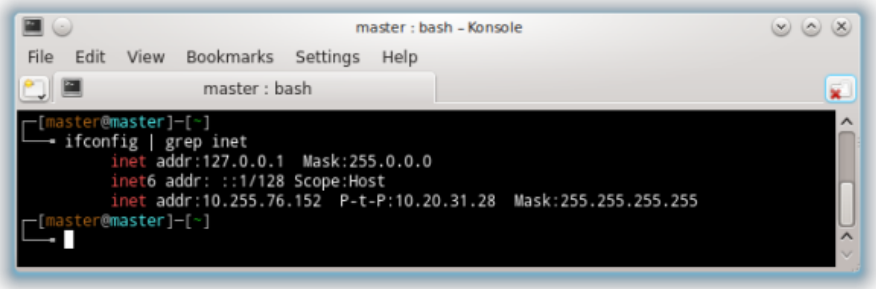
Tombol mkdir

Isi tombol mkdir adalah perintah untuk mengeluarkan informasi jaringan pada baris yang memiliki teks inet saja (membuang semua baris yang lain) dan informasi itu dicetak di dalam Terminal saja. Perintahnya adalah **ifconfig | grep inet**.



Gambar 5: Standard Output dalam Qt Creator

Keluaran (standard output) hanya muncul setelah program ditutup.



Gambar 6: Standard Output dalam Terminal

Sama keluarannya (standard output di Terminal asli dengan Terminal di Qt Creator). Valid.

Tombol cat

Isi tombol cat adalah perintah untuk menghapus berkas **isifolder.txt** yang sudah ada. Karena inilah saya jelaskan dari bawah ke atas.

4 Pembahasan

4.1 mainwindow.h

Sama seperti tulisan sebelumnya. Jika Anda belum mengerti model deklarasi dalam header ini, silakan merujuk ke sini.

4.2 mainwindow.cpp

Pada berkas .cpp proyek ini, ada 3 fungsi buatan yang perlu diperhatikan karena inilah inti program. Kita punya 3 model eksekusi perintah Terminal di Qt, setidaknya dalam program ini. Jika Anda belum mengerti bagaimana membangun elemen-elemen GUI dengan Qt, silakan merujuk ke sini.

4.2.1 Model Eksekusi Perintah Pertama

```
1  QProcess *cat = new QProcess;
2  cat->start("/bin/bash -c \"rm isifolder.txt\"");
```

4.2.2 Model Eksekusi Perintah Kedua

```
1  QProcess sh;
2  sh.start("sh", QStringList() << "-c" << "ifconfig | grep inet");
3
4  sh.waitForFinished(-1);
5  QByteArray output = sh.readAll();
6  printf(output);
```

Inti dari model kedua ini adalah pemanfaatan method start() milik QProcess yang memiliki bentuk umum: start(command, argument);. Perintah kita taruh di command, argumen kita taruh di argument. Sekadar catatan, rm -rf ubuntu.png berarti **rm** itu perintah sedangkan **-rf ubuntu.png** itu argumen. Di sini, perintah yang digunakan adalah **sh** (pasti ada di /bin/ Anda) sedangkan argumen yang digunakan adalah -c dan ifconfig | grep inet. Istimewanya, di sini penggunaannya tidak sesederhana itu. Kita menggunakan kelas tambahan bernama QStringList di dalam argumen start(). QStringList ini adalah kelas yang bisa menampung beberapa string sekaligus. Oleh karena itu, ia dipakai

untuk menampung string yang berisi argumen-argumen perintah. Ia sangat praktis, maka ia sering digunakan. Di sini, QStringList dipakai dengan deklarasi langsung **QStringList() << "-c" << "ifconfig | grep inet"**. Terlihat aneh, terlihat tidak biasa. Namun ini gunanya supaya kita tidak usah membuat satu objek baru.

Teristimewa untuk QByteArray output = sh.readAll(), ini maksudnya memanggil method readAll() milik objek QProcess, yang menghasilkan keluaran dari perintah yang dijalankan oleh QProcess, lalu keluaran itu disimpan pada objek output yang dibuat dari kelas QByteArray. Pendek kata, baris deklarasi ini mengambil standard output dari perintah di dalam objek QProcess. Oleh karena itu, ia akan mengeluarkan output dari perintah **ifconfig | grep inet** ke Terminal di dalam Qt Creator. Apakah ini tidak penting? Tidak, justru ini (*standard output* dan *standard error*) penting sekali untuk membangun aplikasi sebagaimana ditulis di dalam Ringkasan (halaman 1).

4.2.3 Model Eksekusi Perintah Ketiga

```
1  QStringList options;
2  options << "-c" << "ls -l | grep a | sort > isifolder.txt";
3
4  QProcess process;
5  process.start("/bin/sh", options);
6  process.waitForFinished(-1);
```

Model ketiga ini adalah yang paling mudah dipahami. Pertama-tama kita membuat objek QStringList dahulu yang menampung seluruh argumen yang dibutuhkan. Objek ini bernama options. Lalu kita buat objek QProcess yang menampung perintah /bin/sh (bisa diganti /bin/bash atau /bin/zsh jika Anda punya zsh). Objek ini bernama process. Lalu, kita panggil start(/bin/sh, options); untuk process. Maka jadilah program seperti yang saya tampilkan di bagian Hasil Kode (3.1.4) di halaman (6).

Inti dari ketiga jenis eksekusi perintah Terminal dari dalam Qt di atas adalah pemakaian kelas QProcess. Ada beberapa hal penting yang patut diperhatikan:

- Kita bisa menggunakan pipeline (|) dan redirection (>, <, >>, <<) setelah mereka dimasukkan sebagai string sekaligus argumen pada QProcess. Pipeline dan redirection adalah *the ultimate power* di sistem operasi Linux.
- Tidak seperti biasanya, di sini, yang disebut command itu malah /bin/bash (shell) kita bukan langsung pada command yang biasa kita ketik di Terminal. Justru command yang biasa kita pakai itu menjadi argument di sini.

- Adanya method waitForFinished(-1) di sini. Arti nilai -1 ini adalah menunggu sampai objek QProcess selesai mengerjakan tugasnya. **Tanpa method ini, program tidak berjalan semestinya**.
- Seluruh kode yang ada di sini adalah dasar kita untuk membuat aplikasi yang selain mampu mengeksekusi perintah Terminal, juga mampu membaca keluaran dari perintah yang dieksekusi bahkan mampu memrosesnya.

5 Unduh Kode Sumber

Program kali ini bernama Embrio-Metamorfosa juga. Saya menggunakan Git jadi tidak khawatir kode rusak. Silakan unduh dan buka di Qt Creator Anda.

- Tautan: http://otodidak.freeserver.me/tarball/Embrio-Metamorfosa_Edisi_2.tar.gz
- Ukuran: 3 KB

6 Kesimpulan

- Eksekusi perintah Linux dengan Qt bisa dilakukan dengan QProcess selain dengan method **system()**.
- QProcess bisa dimanfaatkan untuk membaca *standard output* dan *standard error* dari segala perintah Linux.
- Ada beberapa model eksekusi perintah Linux di dalam Qt tetapi seluruhnya tidak keluar dari bentuk umum **qprocess.start(command, argument)**.
- Kita beralih ke QProcess karena ia praktis dan memiliki semua perlengkapan untuk memanggil perintah Linux serta membaca dan mengolah standard output-nya.
- Inti dari semua ini adalah pembuatan sebuah proses baru lalu proses itu memanggil perintah Linux.

7 Rujukan

1. http://stackoverflow.com/questions/10701504/command-working-in-terminal-but-not-via-qprocess (untuk model 1 dan 2)
2. http://stackoverflow.com/questions/7597062/qprocess-messes-my-linux-command-up-i-think-how-to-fix (untuk model 3)
3. http://stackoverflow.com/questions/14504201/qprocess-and-shell-destroyed-while-process-is-still-running (untuk -1)

4. http://www.linfo.org/standard_output.html (untuk perkenalan standard output dan standard error)

5. http://en.wikipedia.org/wiki/Front_and_back_ends (untuk perkenalan GUI front-end)

8 Tentang Dokumen Ini

Dokumen ini adalah versi PDF dari posting asli http://malsasa.wordpress.com/2013/10/11/pemrograman-qt-9-qprocess-dan-menjalankan-perintah-linux/. Dokumen ini disusun ulang dengan LaTeX dengan antarmuka Gummi di atas Ubuntu 12.04. Dokumen ini adalah bagian dari usaha kecil memperbanyak panduan Linux dalam Bahasa Indonesia. Dokumen ini tidaklah bebas dari kesalahan yang membuat penulis memohon maaf dan mengajak Anda untuk menulis dokumen yang lebih baik.

Penulis berusaha membuat versi PDF ini semirip mungkin dengan versi HTML. Namun penulis menemukan bahwa kode-kode sumber dalam dokumen ini bisa disalin-tempel (copy-paste) dengan sempurna pada Evince PDF Viewer dan tidak sempurna (membutuhan sedikit penyuntingan tambahan) pada Okular PDF Viewer. Maka penulis menyarankan Anda untuk menggunakan Evince atau menulis sendiri tiap-tiap kode sumber yang ada atau mengunduh langsung versi paket kode sumbernya pada alamat yang tersebut di atas. Penulis memohon maaf atas ketidaknyamanan ini.

Catatan teknis keterbatasan: listings, fontenc, [T1], accsupp, minted (ada bug pada Gummi), Pygments.

9 Tentang Penulis

Penulis adalah warga Forum Ubuntu Indonesia dan aktif di forum-forum Linux lain di Indonesia. Penulis mendukung pendidikan perangkat lunak legal (terutama FOSS) untuk masyarakat. Penulis menyediakan buku-buku panduan Linux untuk pemula maupun ahli untuk diunduh secara gratis[1]. Penulis bisa dihubungi via SMS di nomor 0896 7923 7257.

[1] http://malsasa.wordpress.com/pdf