How to execute complex linux commands in Qt? [duplicate]

This question already has an answer here:
Piping (or command chaining) with QProcess *3 answers*

I want to restart the computer by running a command in linux using `QProcess` . I have hard-coded my root password in my application.

When i run the following in a terminal it works perfect:

```
echo myPass | sudo -S shutdown -r now
```

When i put the command in a shell script and call it via `QProcess` it is also successful :

```
QProcess process;
process.startDetached("/bin/sh", QStringList()<< "myScript.sh");
```

But i can not run it by directly passing to `QProcess` :

```
process.startDetached("echo myPass | sudo -S shutdown -r now ");
```

It will just print `myPass | sudo -S shutdown -r now`

How is it possible to run such relatively complex commands directly using `QProcess` . (Not putting in a shell script).

c++ linux qt shell qprocess

edited Oct 4 '14 at 10:33

asked Apr 27 '14 at 11:21


Nejat
14.5k 3 20 42

marked as duplicate by lpapp Jun 21 '14 at 6:19

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

3 Answers

The key methods that exist for this purpose established in `QProcess` :

> void QProcess::setProcessChannelMode(ProcessChannelMode mode)

and

> void QProcess::setStandardOutputProcess(QProcess * destination)

Therefore, the following code snippet would be the equivalence of `command1 | command2`
without limiting yourself to one interpreter or another:

```
QProcess process1
QProcess process2;

process1.setStandardOutputProcess(&process2);

process1.start("echo myPass");
process2.start("sudo -S shutdown -r now");
process2.setProcessChannelMode(QProcess::ForwardedChannels);

// Wait for it to start
if(!process1.waitForStarted())
    return 0;

bool retval = false;
```

```
QByteArray buffer;
// To be fair: you only need to wait here for a bit with shutdown,
// but I will still leave the rest here for a generic solution
while ((retval = process2.waitForFinished()));
    buffer.append(process2.readAll());

if (!retval) {
    qDebug() << "Process 2 error:" << process2.errorString();
    return 1;
}
```

You could drop the `sudo -S` part because you could run this small program as root, as well as setting up the rights. You could even set setuid or setcap for the shutdown program.

What we usually do when building commercial Linux systems is to have a minimal application that can get setuid or setcap for the activity it is trying to do, and then we call that explicitly with `system(3)` or `QProcess` on Linux. Basically,

I would write that small application to avoid giving full root access to the whole application, so to restrict the access right against malicious use as follows:

```
sudo chmod u+s /path/to/my/application
```

edited Apr 27 '14 at 11:43

answered Apr 27 '14 at 11:32

lpapp
1

This works well and i like it because it is more general and more in a Qt way. I don't know what is this for: process2.setProcessChannelMode(QProcess::ForwardedChannels); – Nejat Apr 27 '14 at 12:06

@Nejat: yeah, it may work without that, but last time I had issues when not having that. The purpose of it is to redirect the child process' output to the main process, so, yeah, that may be unnecessary for your case. – lpapp Apr 27 '14 at 12:08

You must put your command in a shell script and execute `sh` or `bash` with QProcess with your shell script as argument, because your command contains `|`, which *must* be interpreted by `sh` or `bash`.

However, it's just my opinion, but: I don't think it is a good solution to do what you are doing, i.e. include your root password in an executable.

answered Apr 27 '14 at 11:27

AntiClimacus
814 1 10

This is not recommended as it restricts the availability to the given interpreter. – lpapp Apr 27 '14 at 11:34

I have already tried that successfully if you read the question. – Nejat Apr 27 '14 at 11:40

@LaszloPapp: Do you know a Linux system without /bin/sh ? The question was about Linux, not all operating systems. If the command was more complex, with several pipes / redirectors, what do you do ? 10 QProcess ? It's also more flexible to dedicate the command in a shell script: if you would like to change the command, you don't need to compile again your executable. – AntiClimacus Apr 27 '14 at 11:42

The OP explicitly tried this and asked: `How is it possible to run such relatively complex commands directly using QProcess. (Not putting in a shell script).` -> You are not attempting to answer the question. Besides, you are also confused: interpreter is *not* sh, but *bash*. And, yes, the whole embedded industry is based on busybox or toybox without bash. Many people also use zsh instead of bash on desktop, et cetera. The OP wanted to avoid it and find a cross-solution. I still think you are just reiterating what the OP wrote, so it is not an attempt to answer the question. – lpapp Apr 27 '14 at 11:45

I'm not confused: sh is not bash, and is not always a symlink to bash, sh is an entire and independent shell (en.wikipedia.org/wiki/Bourne_shell). And even there is no bash and no sh, e.g. if you use zsh, dash, or other shells, then, in this case, sh points to the right interpreter. – AntiClimacus Apr 27 '14 at 11:51

First, you could configure `sudo` to avoid asking you the password. For instance by being member of the `sudo` group and having the line

```
%sudo  ALL=NOPASSWD:  ALL
```

in your `/etc/sudoers` file. Of course not asking the password lowers the security of your system.

To answer your question about Qt, remember that bash(1), like all Posix shells, hence `/bin/sh`, accept the `-c` argument with a string (actually system(3) is forking a `/bin/sh -c`). So just execute

```
process.startDetached("/bin/sh", QStringList()<< "-c"
                      << "echo myPass | sudo -S shutdown -r now");
```

As AntiClimacus answered, puting your root password inside an executable is a bad idea.

answered Apr 27 '14 at 11:29

Basile Starynkevitch
90.8k 5 50 122

It is a suboptimal idea, because Qt has dedicated API for this which does not limit you to an interpreter. – lpapp Apr 27 '14 at 11:35

Also, it seems the OP already tried this, and also asked `How is it possible to run such relatively complex commands directly using QProcess. (Not putting in a shell script).` – lpapp Apr 27 '14 at 11:42