QtCon BR

INICIANDO E MANIPULANDO APLICAÇÕES NO LINUX COM QT

Cleiton - Bueno
cleiton.bueno@b2open.com

OLA!

Eu sou Cleiton Bueno

Engenheiro, ciclista*, FOSS, e tudo isso junto

1.

Processo

“

Basicamente um processo é a execução de uma instancia de um programa do computador

“

A partir deste momento se tem PID, PPID, IPC e diversos recursos de um processo ou para se criar um... (fork, exec, system, ...)

2.

QProcess

Let’s start with the first set of slides

A classe QProcess é usada para iniciar programas externos e se comunicar com eles.

Pode incluir **parâmetros**, **workdir** e até configurar **variáveis ambiente** antes de executar o programa

Um PID é criado e poderá ser utilizado para monitorar estados, comunicar com a aplicação, encerrar a aplicação e matar o processo :(

Executando Aplicações

```
QString cmd = "/usr/local/bin/updateFW";
QStringList parameters;
arguments << "-usb" << "-force";

QProcess *myProc = new QProcess(parent);
myProc->start(cmd, parameters);
```

Executando Aplicações

```
QString cmd = "/usr/local/bin/updateFW";
QStringList parameters;
arguments << "-usb" << "-force";

QProcess *myProc = new QProcess(parent);
myProc->start(cmd, parameters);


qint64 myPid = myProc->processId();
```

Executando Aplicações

```
QProcess *myProc = new QProcess(parent);
myProc->start(cmd, parameters);
```

```
$ ps -eo "%p %P %y %x %c"
  PID  PPID TTY          TIME COMMAND
    1     0 ?        00:00:01 systemd
 ...
 3565     1 ?        00:00:12 mate-terminal
 ...
 4021  3565 pts/4    00:00:00 bash
 4105  4021 pts/4    00:00:01 myProc
 ...
 4304  4105 pts/4    00:00:00 app1.sh
 4306  4304 pts/4    00:00:00 sleep
```

```
QProcess *myProc = new QProcess(parent);
myProc->startDetached(cmd, parameters);
```

```
$ ps -eo "%p %P %y %x %c"
  PID  PPID TTY          TIME COMMAND
    1     0 ?        00:00:01 systemd

 3565     1 ?        00:00:14 mate-terminal
 ...
 4021  3565 pts/4    00:00:00 bash
 ...
 4492  4021 pts/4    00:00:00 myProc
 4497     1 ?        00:00:00 app1.sh
 4504  4497 ?        00:00:00 sleep
```

Executando Aplicações

```
QProcess *myProc = new QProcess(parent);
myProc->start(cmd, parameters);
```

```
$ ps -eo "%p %P %y %x %c"
  PID  PPID TTY          TIME COMMAND
    1     0 ?        00:00:01 systemd
 ...
 3565     1 ?        00:00:12 mate-terminal
 ...
 4021  3565 pts/4    00:00:00 bash
 4105  4021 pts/4    00:00:01 myProc
 ...
 4304  4105 pts/4    00:00:00 app1.sh
 4306  4304 pts/4    00:00:00 sleep
```

```
QProcess *myProc = new QProcess(parent);
myProc->startDetached(cmd, parameters,
                      ".", &m_pidDetached);
```

```
$ ps -eo "%p %P %y %x %c"
  PID  PPID TTY          TIME COMMAND
    1     0 ?        00:00:01 systemd
 ...
 3565     1 ?        00:00:14 mate-terminal
 ...
 4021  3565 pts/4    00:00:00 bash
 ...
 4492  4021 pts/4    00:00:00 myProc
 4497     1 ?        00:00:00 app1.sh
 4504  4497 ?        00:00:00 sleep
```

Executando Aplicações

```
QProcess *myProc = new QProcess(parent);
myProc->start(cmd, parameters);
myProc->waitForStarted();
```

Com **waitForStarted()**

Saida

```
Process Started
-*- PROGRAMA 1 -*-

   Nome:  /tmp/app1.sh
   PID:   7972
1

2
```

Sem **waitForStarted()**

Saida

```
-*- PROGRAMA 1 -*-

   Nome:  /tmp/app1.sh
   PID:   7931
1

Process Started
2
```

Executando Aplicações

enum **ProcessState** { NotRunning, Starting, Running }

```
switch (m_process->state()) {
case QProcess::Running:
    m_process->terminate();
    break;
case QProcess::Starting:
    m_process->terminate();
    break;
case QProcess::NotRunning:
    qDebug() << "Process not running!" << endl;
    break;
}
```

Executando Aplicações

enum **ExitStatus** { NormalExit, CrashExit }

```
connect(m_process, SIGNAL(finished(int)), this, SLOT(pFinished(int)));
```

```
void MyProc::pFinished(int signal)
{
    switch (signal) {
    case QProcess::NormalExit:
        setOut(QString("Process Finished: Success"));
        break;
    case QProcess::CrashExit:
        setOut(QString("Process Finished: Error"));
        break;
    default:
        setOut(QString("Process Finished: Undefined"));
        break;
    }

    emit finishedChanged();
}
```

Encerrando Aplicações

```
// Encerra o processo imediatamente, enviando um
// SIGKILl
m_process->kill();
```

```
// Tentar terminar a execução do processo, enviando um
// SIGTERM
m_process->terminate();
```

Sinais

```
void    errorOccurred(QProcess::ProcessError error)
void    finished(int exitCode, QProcess::ExitStatus exitStatus)
void    readyReadStandardError()
void    readyReadStandardOutput()
void    started()
void    stateChanged(QProcess::ProcessState newState)


// QIODevice
void    readyRead()
```

Qt

2.1

LABORATÓRIO

QtCon BR

OBRIGADO!

Duvidas?

Pode me encontrar em @cleitonrbueno &
cleiton.bueno@b2open.com

Qt