

Лекция 8. Поддержка многопоточности в наборе библиотек Qt

Разработка многоплатформенного ПО

12 октября 2016 г.

Применение многопоточности

Основные области применения

- Параллельные алгоритмы.
- Сетевые приложения, асинхронный ввод/вывод.
- Пользовательский интерфейс.

Создание потока (Windows API)

Пример

```
#include <windows.h>

DWORD WINAPI MyThreadProc(
    LPVOID pvData)
{
    // ...
    return 0;
}

int main()
{
```

Пример (окончание)

```
HANDLE hThread = CreateThread(
    NULL,           // атрибуты
    0,             // стек
    MyThreadProc,  // адрес
    NULL,         // параметр
    0,            // флаги
    NULL);        // & номера
// ...
WaitForSingleObject(
    hThread, INFINITE);
CloseHandle(hThread);
} // main()
```

Поддержка потоков в Qt

Уровень	Средства
Нижний	QThread, QThreadPool
Средний	QtConcurrent::run()
Верхний	Qt Concurrent (Map, Filter, Reduce)

Таблица 1: уровни поддержки многопоточности

Технологии многопоточности

Возможность	QThread	Средний	Верхний
Задание приоритета потока	✓		
Запуск цикла обработки сообщений	✓		
Получение обновлений с помощью сигналов	✓		
Управление потоком с помощью сигналов	✓		✓
Слежение за состоянием с помощью QFuture		частично	✓
Встроенные средства паузы/возобновления/отмены			✓

Таблица 2: сравнение технологий многопоточности в Qt

Класс QThread

Методы (открытые слоты)

- **void** start(QThread::Priority nPriority = QThread::InheritPriority);
- **void** terminate();
- **void** quit(); (~ exit(0));

Методы (основные операции)

- **bool** wait(unsigned long ulTime = ULONG_MAX);
- **void** requestInterruption();
- **void** exit(int nReturnCode = 0);

Класс QThread

Методы (открытые слоты)

- **void** start(QThread::Priority nPriority = QThread::InheritPriority);
- **void** terminate();
- **void** quit(); (~ exit(0);)

Методы (основные операции)

- **bool** wait(unsigned long ulTime = ULONG_MAX);
- **void** requestInterruption();
- **void** exit(int nReturnCode = 0);

Класс QThread (продолжение)

Методы (защищённые операции)

- **int** exec();
- **virtual void** run();

Методы (атрибуты)

- **bool** isRunning() **const**;
- **bool** isFinished() **const**;
- **bool** isInterruptionRequested() **const**;
- QThread::Priority priority() **const**;
void setPriority(QThread::Priority nPriority);
- **uint** stackSize() **const**; **void** setStackSize(**uint** uStackSize);

Класс QThread (продолжение)

Методы (защищённые операции)

- **int** exec();
- **virtual void** run();

Методы (атрибуты)

- **bool** isRunning() **const**;
- **bool** isFinished() **const**;
- **bool** isInterruptionRequested() **const**;
- QThread::Priority priority() **const**;
 void setPriority(QThread::Priority nPriority);
- **uint** stackSize() **const**;/**void** setStackSize(**uint** uStackSize);

Класс QThread (окончание)

Методы (сигналы)

- **void** started();
- **void** finished();

Методы (статические функции)

- `QThread *currentThread();`
- `int idealThreadCount();`
- `void sleep(unsigned long ulSecs);` (`msleep()`, `usleep()`)
- `void yieldCurrentThread();`

Класс QThread (окончание)

Методы (сигналы)

- **void** started();
- **void** finished();

Методы (статические функции)

- **QThread** *currentThread();
- **int** idealThreadCount();
- **void** sleep(**unsigned long** ulSecs); (msleep(), usleep())
- **void** yieldCurrentThread();

Пример

Пример (example-14.cpp)

```
#include <QThread>

#include <iostream>

class Thread : public QThread
{
public:
    //
    explicit Thread(char chData);
    //
```

Пример (продолжение)

```
protected:
    //
    virtual void run();
    //
private:
    //
    char m_chData;
}; // class Thread
```

Пример (продолжение)

Пример (продолжение)

```
Thread::Thread(char chData)
    : QThread(0),
      // QThread(QObject * pParent)
      m_chData(chData)
{
    //
}
```

Пример (продолжение)

```
void Thread::run()
{
    forever
    {
        std::cerr << m_chData;
        if (isInterruptionRequested())
            break;
    }
}
```

Пример (окончание)

Пример (продолжение)

```
int main()
{
    Thread threadA('-');
    Thread threadB('*');
    //
    threadA.start();
    threadB.start();
    //
    QThread::sleep(5);
    //
}
```

Пример (окончание)

```
threadA.requestInterruption();
threadB.requestInterruption();
//
threadA.wait();
threadB.wait();
} // main()
```

Потоковая безопасность

Определение

Потоковая безопасность: (*Thread Safety*) — свойство кода, предполагающее его корректное функционирование при одновременном исполнении несколькими потоками.

Основные методы

- Атомарные операции.
- Взаимное исключение.
- Реентрабельность.
- Локальное хранилище потока.

Реентрабельность

Определение

Реентрабельность: (*Reentrancy*) — свойство функции или программы, предполагающее её корректный повторный вызов во время исполнения (например, рекурсивный).

Безопасность вызова из разных потоков в Qt

- Реентрабельные функции — для **разных** данных;
- Потокобезопасные функции — возможно для **общих** данных.

Требования к реентрабельной функции

Реентрабельная функция

- Не должна работать со статическими/глобальными неконстантными данными.
- Не должна возвращать адрес статических/глобальных неконстантных данных.
- Должна работать только с данными, переданными вызывающей стороной.
- Не должна опираться на блокировки глобальных ресурсов.
- Не должна модифицировать своего кода.
- Не должна вызывать нереентрабельных программ или процедур.

Локальное хранилище потока

Определение

Локальное хранилище потока: (*Thread-local storage*, TLS) — набор статических/глобальных переменных, локальных по отношению к использующему их потоку.

Ограничения на классы Qt

Безопасность классов

- Функции `QObject::connect()`, `QCoreApplication::postEvent()` и т. д., классы, относящиеся к потокам (`QMutex` и т. д.) являются **потокобезопасными**.
- Класс `QObject` и большинство классов, не относящихся к пользовательскому интерфейсу, являются **реентрабельными**.
- Объекты, управляемые событиями, (таймер, модуль `Network` и т. д.) могут использоваться **только из одного** потока (или нужно упорядочивать доступ мьютексом и т. п.)
- Классы пользовательского интерфейса (`QWidget` и т. д.), функции `QCoreApplication::exec()` и т. д. **не реентрабельны**, могут использоваться **только из основного** потока.

Ограничения на классы Qt (окончание)

Ограничения QObject

- Объекты QObject и производных классов должны создаваться после QApplication.
- Дочерние для QObject объекты должны создаваться в потоке, в котором был создан его родитель (\Rightarrow родительским объектом не может быть их поток — QThread).
- Необходимо удалить все объекты QObject до удаления потока (QThread), в котором они были созданы (например, локальные объекты QThread::run()).
- Объекты должны удаляться в создавшем их потоке. Иначе можно использовать QObject::deleteLater().

Пример

Пример (worker.h)

```
class Worker : public QObject
{
    Q_OBJECT
    //
public:
    //
    Worker(
        char chData,
        volatile bool &rvbFinish);
    //
signals:
    //
```

Пример (окончание)

```
void resultReady(char chResult);
void finish();
//
public slots:
    //
    void onDoWork();
    //
private:
    //
    volatile bool &m_rvbFinish;
    char m_chData;
};
```

Пример (продолжение)

Пример (worker.cpp)

```
#include "worker.h"

#include <QThread>

Worker::Worker(
    char chData, volatile bool &rvbFinish)
    : m_chData(chData),
      m_rvbFinish(rvbFinish)
{
    //
}
```

Пример (продолжение)

Пример (worker.cpp, окончание)

```
void Worker::onDoWork()
{
    for (int i = 0; i < 1000; ++ i)
    {
        QThread::msleep(50);
        emit resultReady(m_chData);
        if (m_rvbFinish)
            break;
    }
    //
    emit finish();
}
```

Пример

Пример (main-window.h)

```
#include "ui_main-window.h"

#include <QMainWindow>
#include <QThread>

class MainWindow :
    public QMainWindow,
    public Ui::MainWindow
{
    Q_OBJECT
    //
```

Пример (продолжение)

```
public:
    //
    MainWindow(QWidget * pParent = 0);
    ~MainWindow();
    //
private slots:
    //
    void on_m_pActionStart_triggered();
    void onHandleResultReady(
        char chResult);
    void onHandleFinish();
```

Пример (продолжение)

Пример (main-window.cpp, окончание)

```
//  
private:  
//  
volatile bool m_vbFinish;  
QThread m_aWorkThreads[2];  
}; // class MainWindow
```

Пример (продолжение)

Пример (main-window.cpp)

```
#include "main-window.h"
#include "worker.h"

MainWindow::MainWindow(QWidget * pParent)
    : QMainWindow(pParent),
      //
      m_vbFinish(false)
{
    setupUi(this);
    //
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
setCentralWidget(m_pTextEdit);  
setAttribute(Qt::WA_DeleteOnClose);  
connect(  
    m_pActionExit, SIGNAL(triggered()),  
    this, SLOT(close()));  
//  
m_pTextEdit->setWordWrapMode(  
    QTextOption::WrapAnywhere);  
//
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
char achData[] = {'-', '*'};
for (int i = 0; i < 2; ++ i)
{
    Worker *pWorker = new Worker(
        achData[i], m_vbFinish);
    pWorker->moveToThread(&m_aWorkThreads[i]);
    connect(
        &m_aWorkThreads[i], &QThread::finished,
        pWorker, &QObject::deleteLater);
    connect(
        m_pActionStart, &QAction::triggered,
        pWorker, &Worker::onDoWork);
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
connect(
    pWorker, &Worker::resultReady,
    this, &MainWindow::onHandleResultReady);
connect(
    pWorker, &Worker::finish,
    this, &MainWindow::onHandleFinish);
m_aWorkThreads[i].start();
} // for (int i = 0; i < 2; ++ i)
} // MainWindow::MainWindow()
```

Пример (продолжение)

Пример (worker.cpp, продолжение)

```
MainWindow::~MainWindow()
{
    m_vbFinish = true;
    for (int i = 0; i < 2; ++ i)
    {
        m_aWorkThreads[i].quit();
        m_aWorkThreads[i].wait();
    }
}
```

Пример (продолжение)

Пример (worker.cpp, продолжение)

```
void MainWindow::on_m_pActionStart_triggered()
{
    m_pStatusBar->showMessage("Working...");
}

void MainWindow::onHandleResultReady(char chResult)
{
    m_pTextEdit->setPlainText(
        m_pTextEdit->toPlainText() + chResult);
    m_pTextEdit->moveCursor(QTextCursor::End);
}
```

Пример (окончание)

Пример (worker.cpp, окончание)

```
void MainWindow::onHandleFinish()
{
    m_pStatusBar->showMessage("Finished", 5000);
}
```

Пример

Пример (example-16.cpp)

```
#include <QThread>
#include <QMutex>
#include <QMutexLocker>

#include <iostream>

class Thread : public QThread
{
public:
    //
    Thread();
    //
```

Пример (продолжение)

```
public:
    //
    int getData() const;
    void incData();
    void decData();
    //
protected:
    //
    virtual void run();
    //
```

Пример (продолжение)

Пример (продолжение)

```
private:
    //
    int m_nData;
    QMutex m_Mutex;
};    // class Thread

Thread::Thread()
    : m_nData(0)
{
    //
}
```

Пример (продолжение)

```
int Thread::getData() const
{
    return m_nData;
}

void Thread::incData()
{
    QMutexLocker lock(&m_Mutex);
    ++ m_nData;
}
```

Пример (продолжение)

Пример (продолжение)

```
void Thread::decData()
{
    QMutexLocker lock(&m_Mutex);
    -- m_nData;
}
```

Пример (продолжение)

```
void Thread::run()
{
    for (int i = 0; i < 1000; ++ i)
    {
        incData();
        std::cerr << getData() << ' ';
    }
}
```

Пример (окончание)

Пример (example-16.cpp, окончание)

```
int main()
{
    Thread thread;
    thread.start();
    //
    for (int i = 0; i < 1000; ++ i)
    {
        thread.decData();
        std::cerr << thread.getData() << ' ';
    }
    //
    thread.wait();
} // main()
```

Задача о читателях и писателях

Постановка задачи

- Чтение данных возможно **одновременно любым** количеством читателей.
- Запись данных возможна **одновременно только одним** писателем.
- Во время **записи ни один читатель** не имеет доступа к данным.
- Чтение данных **невозможно**, если хотя бы один писатель изъявил о намерении **записи**.

Пример

Пример (example-17.cpp)

```
#include <QThread>
#include <QReadWriteLock>
#include <QReadLocker>
#include <QWriteLocker>
#include <QScopedArrayPointer>

#include <iostream>
#include <set>
```

Пример (продолжение)

```
typedef std::set <int> IntSet;

IntSet g_Bitcoins;
QReadWriteLock g_Lock;

inline bool divides(
    int n1, int n2)
{
    return (n1 % n2 == 0);
}
```

Пример (продолжение)

Пример (продолжение)

```
class Thread : public QThread
{
protected:
    //
    virtual void run();
};    // class Thread
```

Пример (продолжение)

```
void Thread::run()
{
    int n = 2;
    forever
    {
        ++ n;
        bool bFound = true;
        {
            QReadLocker lock(&g_Lock);
```

Пример (продолжение)

Пример (продолжение)

```
IntSet::const_iterator
    i = g_Bitcoins.begin(),
    e = g_Bitcoins.end();
for (; i != e; ++ i)
    if (divides(n, *i))
    {
        bFound = false;
        break;
    }
}
```

Пример (продолжение)

```
if (bFound)
{
    QWriteLocker lock(&g_Lock);
    IntSet::const_iterator
        i = g_Bitcoins.begin(),
        e = g_Bitcoins.end();
    while (i != e)
        if (divides(*i, n))
            g_Bitcoins.erase(i ++);
    else
        ++ i;
}
```

Пример (продолжение)

Пример (example-17.cpp, продолжение)

```
    //  
    g_Bitcoins.insert(n);  
}    // if (bFound)  
//  
if (isInterruptionRequested())  
    break;  
}    // forever  
}    // Thread::run()
```

Пример (продолжение)

Пример (example-17.cpp, продолжение)

```
int main()
{
    g_Bitcoins.insert(2);
    //
    int nThreads = QThread::idealThreadCount();
    if (nThreads < 1)
        nThreads = 2;
    //
    QScopedArrayPointer <Thread> threads(new Thread[nThreads]);
    //
    for (int i = 0; i < nThreads; ++ i)
        threads[i].start();
}
```

Пример (продолжение)

Пример (example-17.cpp, продолжение)

```
//  
QThread::sleep(10);  
//  
for (int i = 0; i < nThreads; ++ i)  
    threads[i].requestInterruption();  
//  
for (int i = 0; i < nThreads; ++ i)  
    threads[i].wait();  
//
```

Пример (окончание)

Пример (example-17.cpp, окончание)

```
IntSet::const_iterator
    i = g_Bitcoins.begin(),
    e = g_Bitcoins.end();
for (; i != e; ++ i)
    std::cout << ' ' << *i;
//
std::cout << std::endl;
}    // main()
```

Кольцевой буфер

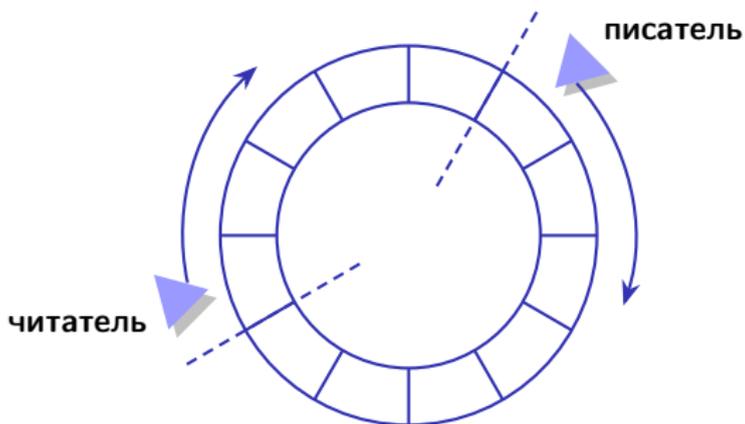


Рис. 1: концепция кольцевого буфера

Пример

Пример (example-18.cpp)

```
#include <QThread>
#include <QSemaphore>

#include <iostream>
#include <cmath>

const int NUM_BLOCKS = 10;
const int BLOCK_SIZE = 8;
const int AMPLITUDE = 100;
const int PERIOD = 30;
const int INTERVAL = 3 * PERIOD;
```

Пример (продолжение)

Пример (example-18.cpp, продолжение)

```
const int BUFFER_SIZE = NUM_BLOCKS * BLOCK_SIZE;

int g_anBuffer[BUFFER_SIZE];
QSemaphore g_FreeSpace(NUM_BLOCKS);
QSemaphore g_UsedSpace(0);

inline bool divides(
    int n1, int n2)
{
    return (n1 % n2 == 0);
}
```

Пример (продолжение)

Пример (example-18.cpp, продолжение)

```
class Thread : public QThread
{
protected:
    //
    virtual void run();
}; // class Thread
```

Пример (продолжение)

Пример (example-18.cpp, продолжение)

```
void Thread::run()
{
    for (int t = 0; t <= INTERVAL; ++ t)
    {
        int i = t % BUFFER_SIZE;
        if (divides(i, BLOCK_SIZE))
            g_FreeSpace.acquire();
        //
    }
}
```

Пример (продолжение)

Пример (example-18.cpp, продолжение)

```
g_anBuffer[i] =  
    t == INTERVAL ?  
    AMPLITUDE + 1 :  
    AMPLITUDE * std::sin(1.0 * t / PERIOD);  
//  
if (t == INTERVAL || divides(i + 1, BLOCK_SIZE))  
    g_UsedSpace.release();  
}    // forever  
}    // Thread::run()
```

Пример (продолжение)

Пример (example-18.cpp, продолжение)

```
int main()
{
    Thread thread;
    thread.start();
    //
    int i = 0, nData;
    do
    {
        if (divides(i, BLOCK_SIZE))
            g_UsedSpace.acquire();
        //
        nData = g_anBuffer[i];
        std::cout << " (" << i << ", " << nData << ")" << std::flush;
```

Пример (продолжение)

Пример (example-18.cpp, продолжение)

```
//  
i = (i + 1) % BUFFER_SIZE;  
//  
if (nData > AMPLITUDE || divides(i, BLOCK_SIZE))  
{  
    g_FreeSpace.release();  
    std::cout << " |" << std::flush;  
}  
}  
while (nData <= AMPLITUDE);
```

Пример (окончание)

Пример (example-18.cpp, окончание)

```
//  
thread.wait();  
//  
std::cout << std::endl;  
} // main()
```

Ложные и украденные пробуждения

Определения

Ложное пробуждение: (*Spurious Wakeup*) — пробуждение, вызванное случайными причинами вместо операции сигнализации.

Украденное пробуждение: (*Stolen Wakeup*) — пробуждение, после которого текущий поток был вытеснен, и сработавший до его продолжения другой поток изменил условие.

Пример

Пример (example-19.cpp)

```
const int MY_MAX_QUEUE = 50;  
const int MY_NUM_PRODUCERS = 6;  
const int MY_NUM_CONSUMERS = 3;  
  
QQueue <int> g_Queue;  
QMutex g_MutexQueue;  
QWaitCondition g_CondNotEmpty;  
QWaitCondition g_CondNotFull;  
volatile bool g_vbRun = true;  
  
// ...
```

Пример (продолжение)

Пример (example-19.cpp, продолжение)

```
void Producer::run()
{
    forever
    {
        QThread::msleep(std::rand() % 100);
        //
        QMutexLocker lock(&g_MutexQueue);
        while (g_Queue.size() == MY_MAX_QUEUE && g_vbRun)
            g_CondNotFull.wait(&g_MutexQueue);
        //
        if (!g_vbRun)
            break;    // lock.unlock()
```

Пример (продолжение)

Пример (example-19.cpp, продолжение)

```
//  
static int s_nItem = 0;  
s_nItem = s_nItem % MY_MAX_QUEUE + 1;  
g_Queue.enqueue(s_nItem);  
std::cout <<  
    "p " << m_nNum << ", " << g_Queue.back() << ") " << std::flush;  
//  
g_CondNotEmpty.wakeOne();  
}    // forever  
}    // Producer::run()
```

Пример (продолжение)

Пример (example-19.cpp, продолжение)

```
// ...  
  
void Consumer::run()  
{  
    forever  
    {  
        QMutexLocker lock(&g_MutexQueue);  
        while (g_Queue.empty() && g_vbRun)  
            g_CondNotEmpty.wait(&g_MutexQueue);  
        //  
        if (!g_vbRun && g_Queue.empty())  
            break;  
    }  
}
```

Пример (продолжение)

Пример (example-19.cpp, продолжение)

```
//  
std::cout <<  
    "(c " << m_nNum << ", " << g_Queue.head() << ") " << std::flush;  
g_Queue.dequeue();  
//  
g_CondNotFull.wakeOne();  
}    // forever  
}    // Consumer::run()
```

Пример (продолжение)

Пример (example-19.cpp, продолжение)

```
int main()
{
    QObject parent;
    QList <QThread *> threads;
    for (int i = 0; i < MY_NUM_PRODUCERS; ++ i)
        threads.append(new Producer(i + 1, &parent));
    //
    for (int i = 0; i < MY_NUM_CONSUMERS; ++ i)
        threads.append(new Consumer(i + 1, &parent));
    //
}
```

Пример (продолжение)

Пример (example-19.cpp, продолжение)

```
std::for_each(
    threads.begin(), threads.end(),
    std::bind2nd(
        std::mem_fun(&QThread::start), QThread::InheritPriority));
//
QThread::sleep(3);
//
g_vbRun = false;
g_CondNotEmpty.wakeAll();
g_CondNotFull.wakeAll();
```

Пример (окончание)

Пример (example-19.cpp, окончание)

```
//  
std::for_each(  
    threads.begin(), threads.end(),  
    std::bind2nd(std::mem_fun(&QThread::wait), ULONG_MAX));  
} // main()
```

Локальное хранилище потока

Пример

```
QThreadStorage <Data> g_Storage;  
  
void setData(int n)  
{  
    if (!g_Storage.hasLocalData())  
        g_Storage.setLocalData(Data());  
    //  
    g_Storage.localData().m_nVal = n;  
}
```

Пример



Рис. 2: приложение с рабочим потоком

Пример

Пример (main-window.h)

```
class MainWindow : public QMainWindow, public Ui::MainWindow
{
    Q_OBJECT
    //
public:
    //
    MainWindow();
    //
public slots:
    //
    void updateProgress(int nPercent);
    //
};
```

Пример (продолжение)

Пример (main-window.h, продолжение)

protected:

```
//  
virtual void closeEvent(QCloseEvent *pEvent);
```

```
//
```

private slots:

```
//  
void on_m_pActionOpen_triggered();  
void on_m_pActionBlur_triggered();  
void on_m_pActionCancel_triggered();  
void onThreadStarted();  
void onThreadFinished();  
void onThreadCanceled();
```

Пример (продолжение)

Пример (main-window.h, окончание)

```
//  
private:  
//  
QImage m_Image;  
WorkerThread *m_pThreadWork;  
bool m_bCanceled;  
}; // class MainWindow
```

Пример (продолжение)

Пример (main-window.cpp)

```
MainWindow::MainWindow()
{
    setupUi(this);
    //
    m_pAreaImage->setBackgroundRole(QPalette::Dark);
    m_pAreaImage->setWidget(m_pLabelImage);
    m_pAreaImage->setWidgetResizable(true);
    setCentralWidget(m_pAreaImage);
    //
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
m_pAreaImage->addAction(m_pActionOpen);  
m_pAreaImage->addAction(m_pActionExit);  
//  
m_pProgressBar->hide();  
m_pButtonCancel->hide();  
statusBar()->addWidget(m_pProgressBar);  
statusBar()->addWidget(m_pButtonCancel);  
//
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
m_pThreadWork = new WorkerThread(this);  
connect(  
    m_pThreadWork, SIGNAL(started()),  
    this, SLOT(onThreadStarted()));  
connect(  
    m_pThreadWork, SIGNAL(finished()),  
    this, SLOT(onThreadFinished()));  
connect(  
    m_pThreadWork, SIGNAL(canceled()),  
    this, SLOT(onThreadCanceled()));  
} // MainWindow::MainWindow()
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
void MainWindow::updateProgress(int nPercent)
{
    m_pProgressBar->setValue(nPercent);
}

void MainWindow::closeEvent(QCloseEvent *pEvent)
{
    pEvent->accept();
    m_pThreadWork->stop();
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
void MainWindow::on_m_pActionOpen_triggered()
{
    QString fileName = QFileDialog::getOpenFileName(this);
    if (!fileName.isEmpty())
        m_pThreadWork->startLoadFile(fileName);
}

void MainWindow::on_m_pActionBlur_triggered()
{
    m_pThreadWork->startBlurImage(m_Image);
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
void MainWindow::on_m_pActionCancel_triggered()
{
    m_pThreadWork->stop();
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
void MainWindow::onThreadStarted()
{
    m_pActionOpen->setEnabled(false);
    m_pActionBlur->setEnabled(false);
    m_pProgressBar->setValue(0);
    m_pProgressBar->show();
    m_pButtonCancel->show();
    m_pActionCancel->setEnabled(true);
    QApplication::setOverrideCursor(Qt::WaitCursor);
    //
    m_bCanceled = false;
}
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
void MainWindow::onThreadFinished()
{
    m_pActionOpen->setEnabled(true);
    m_pActionBlur->setEnabled(true);
    m_pProgressBar->hide();
    m_pButtonCancel->hide();
    m_pActionCancel->setEnabled(false);
    QApplication::restoreOverrideCursor();
    //
```

Пример (продолжение)

Пример (main-window.cpp, продолжение)

```
const QImage *pcImage = m_pThreadWork->getResultImage();
if (pcImage)
{
    m_Image = *pcImage;
    m_pLabelImage->setPixmap(QPixmap::fromImage(m_Image));
}
//
m_pStatusBar->showMessage(
    m_bCanceled ? "Canceled" : "Finished");
} // MainWindow::onThreadFinished()
```

Пример (продолжение)

Пример (main-window.cpp, окончание)

```
void MainWindow::onThreadCanceled()  
{  
    m_bCanceled = true;  
}
```

Пример (продолжение)

Пример (worker-thread.h)

```
class WorkerThread : /* private */ public QThread
{
    Q_OBJECT
    //
public:
    //
    WorkerThread(MainWindow *pWindow);
    ~WorkerThread();
    //
public:
    //
    const QImage *getResultImage() const;
    //
```

Пример (продолжение)

Пример (worker-thread.h, продолжение)

```
public:
    //
    void startLoadFile(const QString &rcFilePath);
    void startBlurImage(const QImage &rcImage);
    void stop();
    //
signals:
    //
    void canceled();
    //
private:
    //
```

Пример (продолжение)

Пример (worker-thread.h, окончание)

```
enum Operation
{
    LoadFile,
    BlurImage
};
//
MainWindow *m_pWindow;
Operation m_nOperation;
QString m_FilePath;
QScopedPointer <QImage> m_ptrInput, m_ptrResult;
//
virtual void run();
}; // class WorkerThread
```

Пример (продолжение)

Пример (worker-thread.cpp)

```
const QImage *WorkerThread::getResultImage() const
{
    assert(isFinished());
    //
    return m_ptrResult.data();
}
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
void WorkerThread::startLoadFile(const QString &rcFilePath)
{
    if (isRunning())
        return;
    //
    m_nOperation = LoadFile;
    m_FilePath = rcFilePath;
    m_ptrInput.reset(new QImage);
    QThread::start();
}
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
void WorkerThread::startBlurImage(const QImage &rcImage)
{
    if (isRunning())
        return;
    //
    m_nOperation = BlurImage;
    m_ptrInput.reset(new QImage(rcImage));
    QThread::start();
}
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
void WorkerThread::stop()
{
    requestInterruption();
}
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
void WorkerThread::run()
{
    switch (m_nOperation)
    {
        case LoadFile:
            //
            if (m_ptrInput->load(m_FilePath))
                m_ptrResult.reset(new QImage(*m_ptrInput));
            //
            break;
            //
    }
}
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
case BlurImage:
    //
    if (m_ptrInput)
    {
        m_ptrResult.reset(new QImage(*m_ptrInput));
        //
        const int cm = m_ptrInput->width();
        const int cn = m_ptrInput->height();
        const int cnTotal = cm * cn;
        const int cnWindow = 5;
        int nCurrent = 0, nPercent, nPrev = 0;
        for (int i = 0; i < cm; ++ i)
            for (int j = 0; j < cn; ++ j)
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
{
    std::vector<int> reds, greens, blues;
    for (int k = i - cnWindow; k < i + cnWindow; ++ k)
        if (k >= 0 && k < cm)
            for (int l = j - cnWindow; l < j + cnWindow; ++ l)
                if (l >= 0 && l < cn)
                    {
                        QColor color = m_ptrInput->pixel(k, l);
                        reds.push_back(color.red());
                        greens.push_back(color.green());
                        blues.push_back(color.blue());
                    }
}
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
//
QColor color(
    std::accumulate(reds.begin(), reds.end(), 0) /
        reds.size(),
    std::accumulate(greens.begin(), greens.end(), 0) /
        greens.size(),
    std::accumulate(blues.begin(), blues.end(), 0) /
        blues.size());
m_ptrResult->setPixel(i, j, color.rgb());
//
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
if (isInterruptionRequested())  
{  
    m_ptrResult.reset();  
    emit canceled();  
    return;  
}  
//
```

Пример (продолжение)

Пример (worker-thread.cpp, продолжение)

```
++ nCurrent;  
nPercent = (100 * nCurrent) / cnTotal;  
if (nPercent > nPrev)  
{  
    nPrev = nPercent;  
    QMetaObject::invokeMethod(  
        m_pWindow,  
        "updateProgress",    // Q_INVOKABLE void myMethod();  
        Qt::QueuedConnection,  
        // Q_RETURN_ARG(QString, retVal)  
        Q_ARG(int, nPercent));
```

Пример (окончание)

Пример (worker-thread.cpp, окончание)

```
        //  
        // emit onPercentChanged(nPercent)  
    }  
} // for (int j = 0; ...)  
} // if (m_ptrInput)  
//  
break;  
} // switch (m_nOperation)  
} // WorkerThread::run()
```

Пример

Пример

```
QFuture <void> future1 = QtConcurrent::run(f, arg1);  
future1.waitForFinished();  
  
Data data;  
QFuture <QString> future2 = QtConcurrent::run(  
    &data, &Data::method, arg1);  
QString result = future2.result();
```

Пример

Пример

```
MainWindow::MainWindow(QWidget *pParent)
{
    // ...
    // QFutureWatcher <int> m_Watcher
    connect(
        &m_Watcher, SIGNAL(finished()), this, SLOT(onFinished()));
    connect(
        &m_Watcher, SIGNAL(started()), this, SLOT(onStarted()));
    // QFuture <int> m_Future;
    m_Future = QtConcurrent::run(f, arg1, arg2);
    m_Watcher.setFuture(m_Future);
}
```