77 QThread

77.1 QThread类的工作原理

在Qt中， QThread类以系统无关的方式，封装了有关线程管理的底层细节，在逻辑层面将线程对象化。一个QThread类及其子类的对象就是一个线程。定义自己的线程，只需继承QThread类，并覆盖其中名为run的虚函数。启动线程，只需通过该子类的实例化对象，调用其继承自基类的start方法。这时run函数在子类中的覆盖版本将被子线程执行。

Qt所做的工作：

```
class QThread
{
    ...
public:
    ...
    void start(void) // 启动线程
    {
        ...
        pthread_create(&thread, &attr, start_routine, this); // 创建子线程
        ...
    }
    ...
protected:
    ...
    virtual void run(void) // 被线程过程函数调用的虚函数
    {
        ...
        exec(); // 陷入事件循环
        ...
    }
    ...
private:
    ...
    static void* start_routine(void* arg) // 线程过程函数
    {
        ...
        ((QThread*)arg)->run(); // 调用虚函数
        ...
    }
    ...
    pthread_t thread;    // 线程标识
    pthread_attr_t attr; // 线程属性
    ...
};
```

子线程开发者所做的工作：

```
class WorkThread: public QThread
{
    ...
protected:
    ...
    void run(void) // 覆盖基类中的虚函数
    {
        ... // 在子线程中完成的任务
    }
    ...
};
```

子线程使用者所做的工作：

```
WorkThread work(...);
work.start();
```

77.2 QThread类的成员函数

77.2.1 公有函数

```
bool     QThread::isRunning();                                  // 线程是否正在运行
bool     QThread::isFinished();                                 // 线程是否已经终止
Priority QThread::priority();                                   // 获取线程的优先级
void     QThread::setPrioriy(QThread::Priority priority); // 设置线程的优先级
void     QThread::exit(int returnCode = 0);                     // 以returnCode为退出码退出事件循环，退出码为0表示成功，非0表示失败
bool     QThread::wait(unsigned long time);                     // 等待线程终止，超过time毫秒提前返回
```

77.2.2 公有槽函数

```
void QThread::start(QThread::Priority priority = InheritPriority); // 以priority优先级启动线程
void QThread::quit();                                                   // 以0为退出码退出事件循环，等价于exit(0)
void QThread::terminate();                                              // 终止线程，但线程不一定立即终止，应使用wait等待其终止
```

77.2.3 信号函数

```
void QThread::started();  // 线程刚刚启动，执行run函数之前，发射此信号
void QThread::finished(); // 线程即将终止，发射此信号
```

77.2.4 静态公有函数

```
void QThread::sleep(unsigned long secs);   // 当前线程睡眠secs秒
void QThread::msleep(unsigned long msecs); // 当前线程睡眠msecs毫秒
void QThread::usleep(unsigned long usecs); // 当前线程睡眠usecs微秒
int  QThread::idealThreadCount();          // 获取系统可运行线程数的理想值
```

77.2.5 保护函数

```
virtual void QThread::run();  // 在所创建的线程中被执行
        int  QThread::exec(); // 陷入事件循环，直到exit或quit被调用
```