1. 补充下多线程编程的知识 《应用篇》线程基本概念。以及并行计算多线程编程。

8-27 界面线程与工作线程

控制台程序卡死：当主循环被卡死时的现象。

原因是某个处理时间太长。

解决：把耗时任务放到一个线程(工作线程)中进行处理，保持命令行界面处于活动状态。

PS：Windows下的休眠。

Winsock2.h

Windows.h

::Sleep(ms);

GUI情况下，处理事件的主循环，称为事件循环。这个用于处理各种界面的线程，称为“界面线程”

int mian()

{

while()

{

event = GetInput();

if(鼠标事件)

…

if(键盘事件)

…

}

}

所有的用户操作：移动鼠标、窗口，点击按钮，输入文字。。。全部都归结为事件，在事件循环中一一串行处理。

注意：所有的事件处理函数（slots），都应该迅速返回，否则主界面会卡死。

例如，某鼠标点击处理函数执行耗时任务，未及时快速返回。在其处理期间界面是卡死状态。（因此函数处于事件循环之中。）

解决方式：将耗时任务放入工作线程中进行处理，避免界面卡死。

08.2 定时器

1.头文件中写好

virtual void timerEvent(QTimerEvent \* event)

1. 源文件cpp 实现

timerEvent(QTimerEvent \* event)

{

if(event->timerId() == m\_timerId)

{

//具体处理

}

}

1. int m\_timerID = startTimer(5000);
2. killTimer销毁定时器

定时器不是精确地、因为是放在事件循环中处理的。差不多而已。

定时器不是线程、它和鼠标键盘一样，进入事件循环进行处理。因此，定时器的 处理函数需要尽快完成，不然也会把界面卡死。

8.3 使用工作线程

在Qt里，线程相关的类

QThread线程

QMutex 互斥锁

QSemaphore 信号量

* 创建线程

派生一个线程类

class MyThread : public QTread

{

Q\_OBJECT

public:

MyThread(QObject \*parent);

~MyThread();

void run(); //线程入口函数

};

* 启动线程
* MyThread \* m\_thread = new MyThread();

m\_Thread->start();

* 回收线程

m\_Thread->wait();

* 另可添加一些成员函数用户与线程进行交互

如获取工作状态、获取处理进度的方法

int GetStatus(); // 1已完成 0 正在进行 -1 发生错误，已经终止。

进度： int GetProgress(); //0 – 100

* 创建一个工作线程

“工作线程”：是一个普通的线程，区别于“界面线程”。界面线程是一个不普通的线程。

当一个事件处理耗时较长时，创建一个线程来完成，以免界面卡死。

步骤：

1. 创建并启动一个工作线程
2. 显示进度条或者等待对话框，对用户进行必要的交互显示。
3. 如果交互需要：启动一个定时器，定时查询工作线程的工作进度和状态，在其工作完成之后，填满进度条，结束等待。可以让线程自己去通知相关控件么？及时反过来做。

8.4 实战抽奖器

写应用程序，就如同堆积木，重点是学习造积木、堆积木的思路（代码规模并不重要）

学习积累**原材料**：语法、线程、SOCKET、SQL，加密解密，标准C库，STL库。。。

构建积累**积木**：类、函数

堆积木：利用积木构建应用程序：将各个模块联合起来。

语法篇 STL Qt的QlistWidget，定时器，XML读写：tinyXML 或者QXML

抽奖器 业务简介：

主界面显示所有被抽奖电话号码，

点击抽奖，弹出抽奖界面，开始抽奖->被抽号码滚动显示。停止抽奖，停在被抽中的号码时，并将抽奖号放置到右侧列表栏中。

退出抽奖界面，将抽奖结果保存至result.txt文件中

* 单元测试

**单元** ：一个函数、类、模块。。。

在不同的场景中有不同的含义，看你怎么划分了。

一个Widget也可以当做一个单元来做独立的测试。

可以将单元对外的依赖接口，通过程序等方式做 “隔离”式的测试。

* 窗口的隐藏与显示

this->hide(); //隐藏

this->showNormal(); //正常显示

this->activateWindow(); //处于活动状态

* 同一按钮的不同状态转换

状态机状态转换。State Machine

可以用一个变量来记录描述其状态。

* 响应窗口关闭事件

void DlgSelect::closeEvent(QCloseEvent \* event)

{

//如果失败，则不关闭对话框

if(saveResults() < 0 )

{

event->ignore();//不关闭程序

return ;

}

QDialog::closeEvent(event);

}

* View 和 item 之辨

<http://blog.csdn.net/leo115/article/details/7532677> 参考

### [Qt Mode/View 学习笔记 —— 概述和Model-博客-云栖社区-阿里云](https://www.baidu.com/link?url=mv5kQzjgMj78oIM7d9aoAgd2AkLSr4EqDpYugXm5NZb8_lecISxakte7bZqDfrb8&wd=&eqid=8edac09500064e170000000359944145)