监控系统

监控系统，包括一个监控器管理器，该管理器负责添加删除监控器。并且提供启动监控和停止监控两个方法，可以同时启动或停止监控器列表中的所有监控。

监控器包括监控对象、轮询和报警器三个部分。监控器包括开始监控、停止监控，以及监控三个方法。监控方法用来判断监控对象是否满足监控条件，在满足监控条件达到指定次数的时候，开始报警。

监控对象是一个接口，包括读取配置的方法（readConfig方法），以及是否满足触发条件的方法（isSatisfied方法）供外部调用。监控对象先读取配置文件，然后获取监控对象的初始状态。在isSatisfied方法中，读取监控对象的当前状态，并且和初始状态比较，以判断是否满足触发条件。比较之后，将当前状态作为初始状态，等待下一次轮询。

轮询需要指定轮询周期，并在轮询时调用监控对象 isSatisfied 方法来判断是否满足监控条件。轮询通过 SetTimer 的方式实现。

报警器是一个接口，提供报警方法。目前可以采用指令集的方式来报警。指令集包含了在满足触发条件情况下的所要执行的一系列指令，还可以设置在每条指令执行失败时需要执行的备用指令等内容。（目前先按照一条指令并且没有备用指令来做。）

为实现一个监控器，只需要给监控器设置一个继承自监控对象接口的监控对象，和一个继承自报警器接口的报警器。在需要监控时调用开始监控方法，完成时调用停止监控方法即可。

类设计（仅供参考）

监控器接口

.h 文件

class CMonitor

// Methods

Public:

void CMonitor(); // 构造函数

Protected:

OnTimer();

// Properties

Public:

IMonitedObject \*MonitedObject;

IAlertor \*Alertor;

Int Interval;

.cpp文件

void CMonitor()

{

MonitedObject=NULL;

Alertor=NULL;

}

class IMonitedObject

public:

BOOL readConfig(string configXmlRoot);

BOOL isSatisfied();

protected:

IStatus \*getStatus();

class IStatus

public:

int compare(IStatus \*status);

class IAlertor

public:

BOOL readConfig(string configXmlRoot);

BOOL alert();

配置文件样例：

<?xml version="1.0"?>

<MonitorSystem version="1.0">

<!-- 每 10 分钟轮询一次 -->

<!-- 条件满足 1 次则报警 -->

<Monitor interval="10" times="1">

<!-- 监控文件的大小 -->

<MonitedObject type="File" status="Size">

<!-- 文件所在路径 -->

<Path>D:\cjb\DataUp\Logs\log%year%%month%%day%.log"</Path>

<!-- 所满足的条件是文件大小不改变 -->

<Condition>NotChanged</Condition>

</MonitedObject>

<Alertor>

<Commands>

<Command>taskkill /im notepad.exe</Command>

</Commands>

</Alertor>

</Monitor>

</MonitorSystem>