**温室花卉环境信息智能监测系统**

详细设计说明书

Version 1.0

文档编号 TTS\*\*\*

2017-7-23

文档历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 标题 | 内容 | 作者 | 时间 |
| 1.0 | 详细设计说明书 | 详细设计 | 第三组 | 2017-7-23 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

[1. 编写目的 1](#_Toc11154)

[2. 参考文档 1](#_Toc11930)

[3. 名称解释 1](#_Toc8773)

[4. 系统架构综述 1](#_Toc12080)

[4.1. 体系架构 2](#_Toc30238)

[4.2. 逻辑模型 2](#_Toc10228)

[5. 平台约束 3](#_Toc20355)

[6. 静态结构设计 4](#_Toc28369)

[6.1. 类定义 4](#_Toc10085)

[根据需求分析，我们设计出以下数据实体类： 4](#_Toc28894)

[6.1.1. 传感器数据类图 4](#_Toc22209)

[6.1.2. 服务器端类图 4](#_Toc19040)

[6.1.3. 服务器端异常类图： 5](#_Toc31515)

[6.1.4. 客户端类图： 5](#_Toc5074)

[6.1.5. 客户端异常类图： 5](#_Toc8162)

[6.2. 【类和接口定义说明】： 6](#_Toc13380)

[服务器端定义以下类： 6](#_Toc23517)

[SensorData 类：无线传感器通信数据类型 6](#_Toc19337)

[typedef struct SensorData{ 6](#_Toc19215)

[Server 类： 6](#_Toc6930)

[原型：void dataMine(void) throw(ServerException); 6](#_Toc20663)

[原型：Server(LogDao&dao,conststring&ip,shortport)throw(ServerException); 7](#_Toc19051)

[ServerSocket 类： 7](#_Toc10840)

[原型：void acceptClient(void) throw(SocketException); 7](#_Toc3238)

[原型：ServerSocket(const string& ip,short port)throw (SocketException); 8](#_Toc25577)

[ClientThread 类： 8](#_Toc5705)

[原型：void run(void) throw (ThreadException,SendException); 8](#_Toc32414)

[原型：ClientThread(int connfd); 8](#_Toc3955)

[StoreThread 类： 9](#_Toc26347)

[原型：void run(void) throw (ThreadException); 9](#_Toc19534)

[原型：StoreThread(LogDao& dao); 9](#_Toc31664)

[LogBuffer 类： 9](#_Toc9751)

[原型：LogBuffer& operator << (const SensorData& log); 10](#_Toc14254)

[原型：LogBuffer& operator >> (SensorData& log); 10](#_Toc24883)

[原型：LogBuffer(void); 10](#_Toc22452)

[FileDao 类： 10](#_Toc12594)

[原型：void insert(const SensorData& log) throw (DBException); 11](#_Toc27429)

[原型：FileDao(const string& path); 11](#_Toc12723)

[原型：~FileDao(void); 11](#_Toc27081)

[MysqlDao 类： 11](#_Toc9081)

[原型：void insert(const SensorData& log) throw (DBException); 12](#_Toc260)

[原型：void connectDB(void) throw(DBException); 12](#_Toc30527)

[原型：void insertData(const SensorData& log) throw (DBException); 12](#_Toc7146)

[原型：MysqlDao(void); 13](#_Toc15366)

[原型：~MysqlDao(void); 13](#_Toc1224)

[服务器端定义以下异常类： 13](#_Toc29938)

[ServerException 类：exception 13](#_Toc13333)

[原型：ServerException (void):m\_msg("服务器异常！"){}; 13](#_Toc15090)

[原型：ServerException(const string& msg):m\_msg("服务器异常！"); 14](#_Toc16375)

[DBException 类：ServerException 14](#_Toc28871)

[原型：DBException (void):m\_msg("数据库错误！"){}; 14](#_Toc14509)

[原型：DBException(const string& msg):ServerException(msg){}; 14](#_Toc17446)

[SocketException 类：ServerException 14](#_Toc25001)

[原型：SocketException (const string& msg):ServerException(msg){}; 15](#_Toc19722)

[SendException 类：ServerException 15](#_Toc29726)

[原型：SendException (void):m\_msg("发送数据错误！"){}; 15](#_Toc25710)

[原型：SendException (const string& msg):ServerException(msg){}; 15](#_Toc16199)

[ThreadException 类：ServerException 16](#_Toc22028)

[原型：ThreadException (void):m\_msg("线程执行错误！"){}; 16](#_Toc23487)

[原型：ThreadException (const string& msg):ServerException(msg){}; 16](#_Toc27856)

[服务器端定义以下接口： 16](#_Toc4755)

[LogThread 接口: 16](#_Toc6171)

[LogDao 接口: 17](#_Toc24060)

[7. 风险评估 18](#_Toc22760)

[8. 开发计划表 18](#_Toc20913)

[9. 项目总结 19](#_Toc23476)

[10.附录 19](#_Toc11970)

[Makefile 内容： 19](#_Toc14275)

# 1. 编写目的

描述 Sensors1.0 项目的系统结构，数据实体，类的定义，类功能的实现，部署要求，运行环境等内容，该文档用来指导 程序员完成程序代码实现。文档的主要读者包括：项目分析人员，项目设计人员，项目经理，程序员，测试人员，部署人员， 项目技术支持人员等。

# 2. 参考文档

【1】 《Sensors1.0 需求说明书》；文档编号：TTS\*\*\*；版本号 v1.0

【2】 《Sensors1.0 概要设计说明书》文档编号：TTS\*\*\*；版本号 v1.0

# 3. 名称解释

**Sensors：**智慧农业信息监控系统（Sensors System）

**系统架构：**描述系统的核心功能组件和组件间的相互关联。

**数据实体：**描述系统需要采集，处理和保存的数据的结构。

**系统数据：**为远程终端机器实时上传的无线传感器数据。

# 4. 系统架构综述

Sensors1.0 由以下几个功能模块组成：

数据采集模块：该功能模块的主要任务是读取客户端终端设备接收到的传感器数据。

数据解析模块：该模块的主要任务是解析客户端接收到的数据。

数据发送模块：该模块的主要任务是向服务器建立连接请求，连接成功后，将解析后的传感器数据实时发送到服务器。

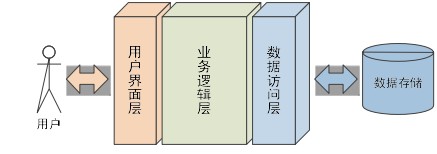
数据接收模块：该模块的主要任务是建立服务器端网络连接，从客户端读取上传的传感器数据。

数据存储模块：该模块的主要任务是将数据接收模块收到的数据保存到指定的数据库表或文件中。

数据统计模块：该模块的主要任务是将数据库或文件中保存的数据汇总统计生成报表或分析生成客户需求数据。

## 4.1. 体系架构

本系统在水平方向上采用三层体系架构，如下图：

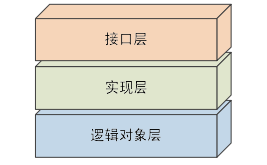


 用户界面层：处理与最终用户的交互，既负责从用户处收集信息，也负责向用户展示结果，结出反馈和提示。

 业务逻辑层：针对用户界面层体现的功能项，以数据访问为基础，实现与业务逻辑相关的算法和流程。

 数据访问层：实现对数据介质的访问，为业务逻辑层提供数据来源，并接受其处理结果。

## 4.2. 逻辑模型



 接口层：定义各功能模块的抽象口，降低各模块间的耦合，提高代码的复用率，降低维护成本，增强软件的可扩展性。

 实现层：对抽象接口的具体实现，本项目服务端用户界面层的接口实现采用控制台方式，而数据访问层的接口实现采用数据库或文件方式。

 逻辑对象层：以逻辑模型的方式以对系统中相关数据加以组织，并构成从用户界面层到业务逻辑层再到数据访问层各层之间的信息载体，本案的逻辑对象包括：客户端和服务端的传感器数据。

# 5. 平台约束

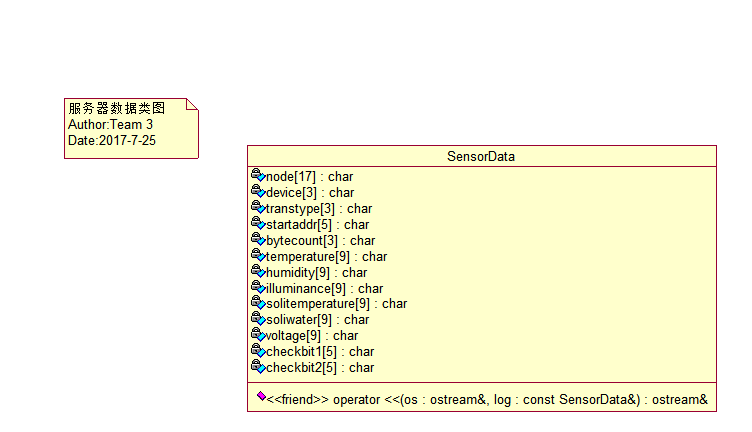
|  |  |
| --- | --- |
| 硬件环境 | Client:32/64 位 Intel x86 及其兼容处理器的个人计算机  Server: CPU：1.2 GHz；内存：512 MB；硬盘：8 GB 剩余空间；显卡：1024x768 以上分辨 率；以太网卡支持(及更高) /HP/IBM x86 x64 |
| 操作系统 | Server:  Ubuntu 10.04 LTS  Linux linux 2.6.32-21-generic #32-Ubuntu i686 GNU/Linux  Client:WindowsXP、Windows 7 及以上 |
| 开发工具 | Server:   C++:  gcc version 4.4.3  Thread model: posix   Mysql: mysql Ver 14.14 Distrib 5.1.73  Client:   Microsoft Visual Studio 10.0 C# 或 Java SE ->jdk1.8 |
| 应用类型 | Server:命令行应用程序； Client:Windows 窗体应用程序 |
| 用户界面 | Server:非全屏模式的控制台字符界面； Client:Windows 窗体界面 |
| 数据存储 | 二进制文件或数据库 |
| 平台中立 | 无要求 |
| 交叉编译 | 无要求 |

# 6. 静态结构设计

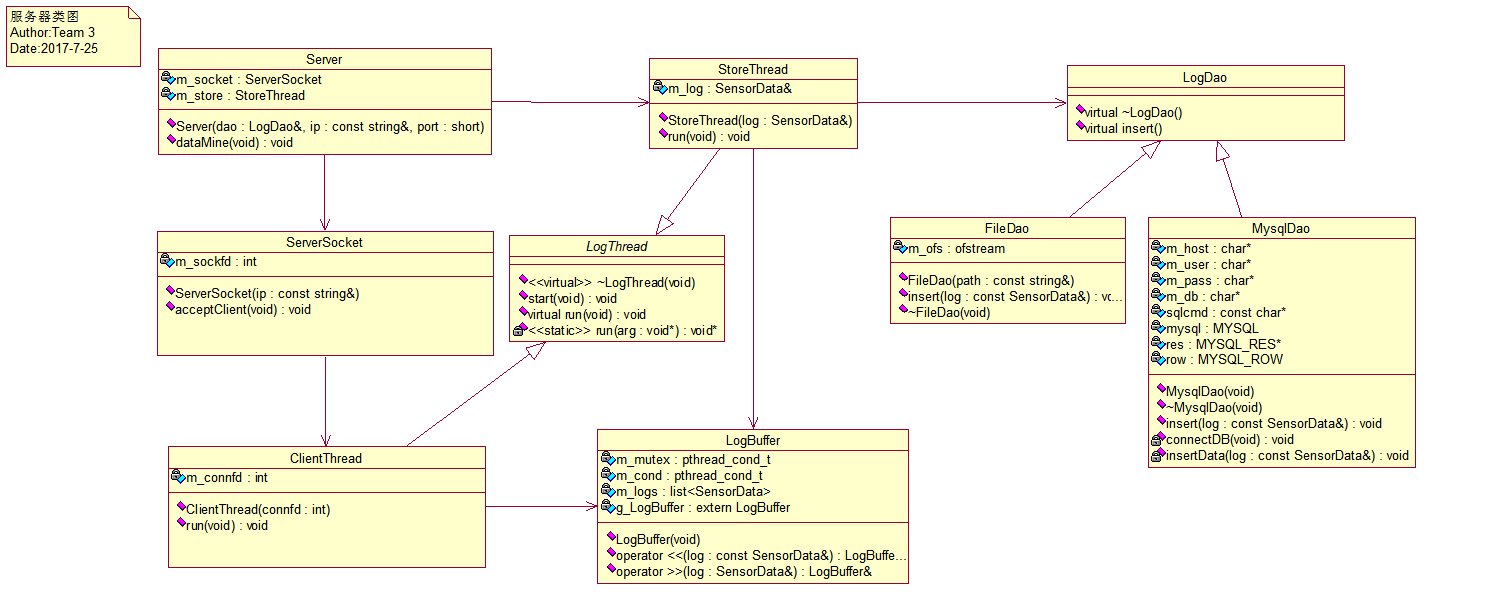
## 6.1. 类定义

## 根据需求分析，我们设计出以下数据实体类：

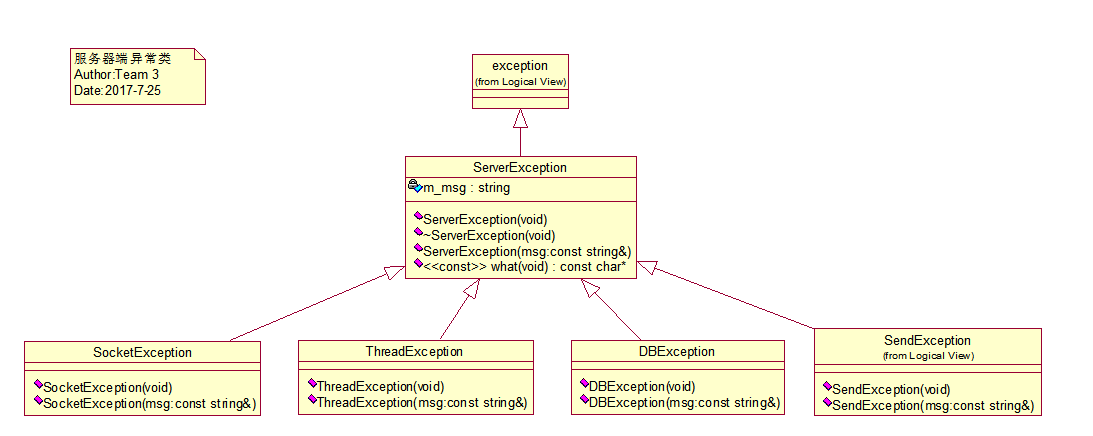
### 6.1.1. 传感器数据类图



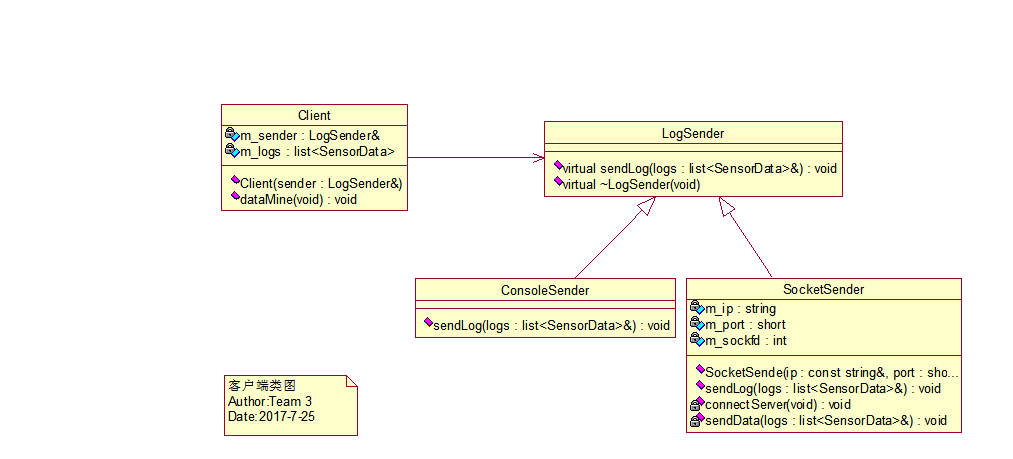
### 6.1.2. 服务器端类图



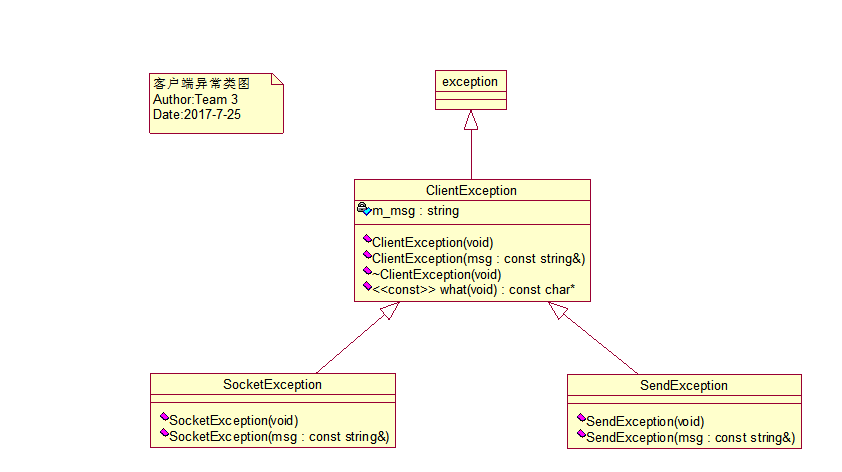
### 6.1.3. 服务器端异常类图：



### 6.1.4. 客户端类图：



### 6.1.5. 客户端异常类图：



## 6.2. 【类和接口定义说明】：

## 服务器端定义以下类：

### SensorData 类：无线传感器通信数据类型

### typedef struct SensorData{

char node[17];//节点地址

char device[3]; //设备地址

char transtype[3];//传输方式 0x60:主动上报

char startaddr[5];//起始地址

char bytecount[3];//字节记数(6 个数据组 \* 每组 2 个数据 \* 1 个数据 2 个字节 =24)

char temperature[9];//温度

char humidity[9];//湿度

char illuminance[9];//光照度

char solitemperature[9];//土壤温度

char soliwater[9];//土壤水份

char voltage[9];//设备电量

char checkbit1[5];//检验 1

char checkbit2[5];//检验 2

//定义输出运算符重载(cout << SensorData << endl;)

friend ostream& operator << (ostream& os,const SensorData& log);

}SensorData;

## Server 类：

**成员变量：**

private:

StoreThread m\_store;//存储线程

ServerSocket m\_socket;//服务器 Socket 套节字

**成员函数：**

### 原型：void dataMine(void) throw(ServerException);

修饰符：public

功能：数据采集服务：启动采集数据的服务 参数：void

异常：出错抛出 ServerException

返回值：void

说明：该函数主要是服务端启动后，启动采集数据的服务

### 原型：Server(LogDao&dao,conststring&ip,shortport)throw(ServerException);

修饰符：public

功能：构造函数重载

参数：

dao：实现 LogDao 接口的类的对象的引用

ip：服务器 IP 地址的引用

port 服务器指定的端口号 1024~65535 范围,本程序是：8888

异常：出错抛出 ServerException

返回值：无

说明：带参的构造函数，用来初始化服务端 Server 类的对象

## ServerSocket 类：

**成员变量：**

private:

int m\_sockfd;//服务器 Socket 套接字

**成员函数：**

### 原型：void acceptClient(void) throw(SocketException);

修饰符：public

功能：响应客户端的请求

参数：void

异常：出错抛出 SocketException

返回值：void

说明：该函数主要功能是用来接收客户端连接请求

### 原型：ServerSocket(const string& ip,short port)throw (SocketException);

修饰符：public

功能：构造函数重载

参数：

ip服务器 IP 地址的引用

port 服务器指定的端口号 1024~65535 范围,本程序是：8888

异常：出错抛出 SocketException

返回值：无

说明：带参的构造函数,用来初始化 ServerSocket 类的对象

### ClientThread 类：

**成员变量：**

private:

int m\_connfd;//客户端套接节

**成员函数：**

### 原型：void run(void) throw (ThreadException,SendException);

修饰符：public

功能：客户线程调用时执行的方法

参数：void

异常：出错抛出 ThreadException, SendException

返回值：void

说明：该函数主要是接收客户端上传的数据，并将数据压入缓冲池(数据链表)

### 原型：ClientThread(int connfd);

修饰符：public

功能：构造函数重载

参数：

connfd 客户端套接节

异常：无

返回值：无

说明：带参的构造函数,用来初始化 ClientThread 类的对象

## StoreThread 类：

**成员变量：**

private:

LogDao& m\_dao;//实现 LogDao 接口的类的对象的引用

**成员函数：**

### 原型：void run(void) throw (ThreadException);

修饰符：public

功能：存储线程调用时执行的方法

参数：void

异常：出错抛出 ThreadException

返回值：void

说明：该函数主要是将缓冲池(数据链表)中的数据取出，并调用实现 LogDao 接口的类的对象的插入数据的方法，将数据保存至数据库或文件中

### 原型：StoreThread(LogDao& dao);

修饰符：public

功能：构造函数重载

参数：

dao 实现 LogDao 接口的类的对象的引用

异常：无

返回值：无

说明：带参的构造函数,用来初始化 StoreThread 类的对象

## LogBuffer 类：

extern LogBuffer g\_LogBuffer; //公共缓冲池对象,主要用途就是为方便其它调用模块.cpp 使用

**成员变量：**

private:

pthread\_mutex\_t m\_mutex;//互斥量

pthread\_cond\_t m\_cond;//条件变更

list <SensorData> m\_logs;//缓冲区链表(客户端上传到服务器的通信数据)

**成员函数：**

### 原型：LogBuffer& operator << (const SensorData& log);

修饰符：public

功能：压入通信数据到缓冲池的链表中

参数：

log 传感器数据类型 SensorData 类的对象的引用

异常：无

返回值：LogBuffer& 数据缓冲池对象的引用

说明：压入通信数据到链表中, LogBuffer 类的输出运算符重载

### 原型：LogBuffer& operator >> (SensorData& log);

修饰符：public

功能：从缓冲池的链表中取得通信数据

参数：

log 传感器数据类型 SensorData 类的对象的引用

异常：无

返回值：LogBuffer& 数据缓冲池对象的引用

说明：从链表中取得通信数据, LogBuffer 类的输入运算符重载

### 原型：LogBuffer(void);

修饰符：public

功能：默认构造函数

参数：void

异常：无

返回值：无

说明：用来初始化 LogBuffer 类的成员变量

## FileDao 类：

**成员变量：**

private:

ofstream m\_ofs;// 输出文件流类的对象

**成员函数：**

### 原型：void insert(const SensorData& log) throw (DBException);

修饰符：public

功能：将传感器数据写入文件

参数：

log 传感器数据类型 SensorData 类的对象的引用

异常：出错抛出 DBException

返回值：void

说明：将缓冲池链表中的数据，写入文件中

### 原型：FileDao(const string& path);

修饰符：public

功能：构造函数重载

参数：

path 文件路径及文件名的引用

异常：无

返回值：无

说明：带参的构造函数，用来初始化 FileDao 类的对象

### 原型：~FileDao(void);

修饰符：public

功能：析构函数

参数：void

异常：无

返回值：无

说明：将输出文件流所关联的磁盘文件关闭，释放资源

## MysqlDao 类：

**成员变量：**

private:

char\* m\_host; //数据库主机 IP 地址

char\* m\_user;// 连接数据库的账号

char\* m\_pass;// 连接数据库的密码

char\* m\_db;// 数据库名称,本项目是：SensorDB

int m\_port;// 数据库主机的连接端口：mysql 3306 \ oracle 1521

const char\* sqlcmd; //sql 语句

MYSQL mysql;//Mysql 的对象

MYSQL\_RES \*res;//Mysql 查询的数据结果集

MYSQL\_ROW row; //Mysql 数据行 rows 对象

**成员函数：**

### 原型：void insert(const SensorData& log) throw (DBException);

修饰符：public

功能：插入函数

参数：

log 传感器数据类型 SensorData 类的对象的引用

异常：出错抛出 DBException

返回值：void

说明：将缓冲池链表中的数据，插入数据库

### 原型：void connectDB(void) throw(DBException);

修饰符：private

功能：连接数据库

参数：void

异常：出错抛出 DBException

返回值：void

说明：连接数据库，本项目是连接 Mysql 数据库

### 原型：void insertData(const SensorData& log) throw (DBException);

修饰符：private

功能：向库中写入数据

参数：

log 传感器数据类型 SensorData 类的对象的引用

异常：出错抛出 DBException

返回值：void

说明：将数据保存至数据库中

### 原型：MysqlDao(void);

修饰符：public

功能：默认构造函数

参数：void

异常：无

返回值：无

说明：默认构造函数

### 原型：~MysqlDao(void);

修饰符：public

功能：析构函数

参数：void

异常：无

返回值：无

说明：析构函数，结束 MySQL 库的使用，包括释放内存等

## 服务器端定义以下异常类：

## ServerException 类：exception

**成员变量：**

private:

string m\_msg;//服务器异常信息

**成员函数：**

### 原型：ServerException (void):m\_msg("服务器异常！"){};

修饰符：public

功能：默认构造

参数：void

异常：无

返回值：无

说明：默认构造函数

### 原型：ServerException(const string& msg):m\_msg("服务器异常！");

修饰符：public

功能：构造重载

参数：

msg 服务器异常消息的引用

异常：无

返回值：无

说明：构造重载

## DBException 类：ServerException

**成员函数：**

### 原型：DBException (void):m\_msg("数据库错误！"){};

修饰符：public

功能：默认构造

参数：void

异常：无

返回值：无

说明：默认构造函数

### 原型：DBException(const string& msg):ServerException(msg){};

修饰符：public

功能：构造重载

参数：

msg 数据库异常消息的引用

异常：无

返回值：无

说明：构造重载

## SocketException 类：ServerException

**成员函数：**

**原型：SocketException (void):m\_msg("Socket 错误！"){};**

修饰符：public

功能：默认构造

参数：void

异常：无

返回值：无

说明：默认构造函数

### 原型：SocketException (const string& msg):ServerException(msg){};

修饰符：public

功能：构造重载

参数：

msg Socket 异常消息的引用

异常：无

返回值：无

说明：构造重载

## SendException 类：ServerException

**成员函数：**

### 原型：SendException (void):m\_msg("发送数据错误！"){};

修饰符：public

功能：默认构造

参数：void

异常：无

返回值：无

说明：默认构造函数

### 原型：SendException (const string& msg):ServerException(msg){};

修饰符：public

功能：构造重载

参数：

msg 发送数据异常消息的引用

异常：无

返回值：无

说明：构造重载

## ThreadException 类：ServerException

**成员函数：**

### 原型：ThreadException (void):m\_msg("线程执行错误！"){};

修饰符：public

功能：默认构造

参数：void

异常：无

返回值：无

说明：默认构造函数

### 原型：ThreadException (const string& msg):ServerException(msg){};

修饰符：public

功能：构造重载

参数：

msg 线程执行异常消息的引用

异常：无

返回值：无

说明：构造重载

## 服务器端定义以下接口：

【c++ 中为抽象类】

## LogThread 接口:

public:

**virtual ~LogThread(void){}; //虚析构函数**

**void start(void) throw (ThreadException); //启动线程的函数**

**virtual void run(void) = 0; //纯虚函数**

private:

**static void\* run(void\* arg);//静态的 run 方法**

## LogDao 接口:

public:

**virtual void insert(const SensorData& log) throw (DBException) = 0; //纯虚函数**

**virtual ~LogDao(void){} //虚析构函数**

# 7. 风险评估

|  |  |  |  |
| --- | --- | --- | --- |
| **风险预测** | **危害程度** | **发生概率** | **规避策略** |
| 环境异常 | 中 | 中 | MIS 协助解决 |
| 人员变动 | 中 | 小 | 项目经理 |
| 需求变更 | 低 | 中 | 产品经理 |
| 不可抗力 | 高 | 小 | -- |

# 8. 开发计划表

|  |  |  |  |
| --- | --- | --- | --- |
| **模块名称** | **功能名称** | **功能说明** | **开发时间** |
| 客户端数据采集模块 | 数据采集 | 该功能模块的主要任务是读取客户端终端设备接收到的传感器数据。 | 2017-7-18 |
| 客户端数据解析模块 | 数据解析 | 该模块的主要任务是解析客户端接收到的数据。 | 2017-7-18 |
| 客户端数据发送模块 | 数据发送 | 该模块的主要任务是向服务器建立连接请求，连接成功后，将解析后的传感器数据实时发送到服务器。 | 2017-7-19 |
| 服务器数据接收模块 | 数据接收 | 该模块的主要任务是建立服务器端网络连接，从客户端读取上传的传感器数据。 | 2017-7-19 |
| 服务器数据存储模块 | 数据存储 | 该模块的主要任务是将数据接收模块收到的数据保存到指定的数据库表或文件中。 | 2017-7-20 |
| 服务器数据统计模块 | 数据统计 | 该模块的主要任务是将数据库或文件中保存的数据汇总统计生成报表或分析生成客户需求数据。 | 2017-7-20 |

# 

# 9. 项目总结

本项目的研发旨在提供一种建立在有限技术基础这上的，适度规模的，相对较接近于真实场景的，具有一定可扩展性的典型案例。以下几个方面共同构成本项目的研发重点：

* 相对完整的项目研发流程：

本安基于采用传统意义上的瀑布式开发流程，从需求开始，再到概要设计，详细设计、编码和测试，整个过程环环相扣，每个步骤都严格依赖于前一个步骤的成果和结论。但在编码阶段，本案又结合了项目的具体特点，引入了现代敏捷开发的和理念，以一种小幅渐近的方式，提高了可交付物的产出效率。

* **多层次的体系和模型设计：**

本案一方面在体系架构上采用了由用户界面、业务逻辑到数据访问共同组成的三层结构。同时又在每一层的内部自上而下

的进行了由接口到实现再到业务逻辑的三层划分。高内聚低耦合乃是本项目开发的宗旨之一。

* **多源文件系统构建：**

本案服务端程序最终由多个.h 和 .cpp 文件组成，通过 makefile 可以对工程的构建方式进行定制，这也是世界中构建 工程的方式之一。一方面通过文件将逻辑层面彼此独立的类和函数组合进行物理层面的划分，另一方面又将函数的声明和 实现分别放到.h 和 .cpp 中。同时保持文件名的相关性。这样既有利于团队的分工合作，又突出了相对稳定的接口定义在 大型项目中的重要性。

* **基本控制台字符界面和基于文件或数据库的存储:**

基本控制台字符界面和基于文件或数据库的存储，虽然难度不大，但处理过程中仍有很多细节需要注意。比如数据文件的损坏、数据库文件的永久持久化，这些涉及到底层的细节都要谨慎对待。

* **借助逻辑对象实现业务逻辑的算法与控制：**

在三层体系架构中，业务逻辑的处理往往最复杂，借助 C++语言的建模能力，结构体现清晰，函数、接口简洁、高效，往往给我们解决复杂问题带来事半功倍的效果。

# 10.附录

## Makefile 内容：

CC = g++ #指定c++的编译器

RM = rm -rf #指定删除指令

CFLAGS = -c -Wall #指定预编译的参数选项 -Wall 编译时显示所有警告信息

Server:main.o fileDao.o mysqlDao.o logBuffer.o logThread.o storeThread.o clientThread.o serverSocket.o server.o

# $(CC) $^ -lpthread -o $@ #由依赖文件生成目标文件

$(CC) $^ -I/usr/include/mysql -L/usr/lib/mysql -lmysqlclient -lpthread -lclntsh -lnsl -o $@

main.o:main.cpp

$(CC) $(CFLAGS) $^

fileDao.o:fileDao.cpp

$(CC) $(CFLAGS) $^

mysqlDao.o:mysqlDao.cpp

$(CC) $(CFLAGS) $^

logBuffer.o:logBuffer.cpp

$(CC) $(CFLAGS) $^

logThread.o:logThread.cpp

$(CC) $(CFLAGS) $^

storeThread.o:storeThread.cpp

$(CC) $(CFLAGS) $^

clientThread.o:clientThread.cpp

$(CC) $(CFLAGS) $^

serverSocket.o:serverSocket.cpp

$(CC) $(CFLAGS) $^

server.o:server.cpp

$(CC) $(CFLAGS) $^

clean:

$(RM) \*.o \*.\*.gch .\*.\*.swp