DMS数据采集系统

一、项目背景介绍

01_dms.pdf

二、项目开发流程

1. 需求分析

按照软件工程的要求，复述用户的需求，得到用户的认可

面向业务人员，用户化的需求文档
面向设计人员，专业化的需求文档

用例、事件流、异常流、前置条件、后置条件

分析阶段不考虑和具体设计有关的内容

简单的演示程序(demo)在招标中的作用至关重要

2. 概要设计

面向过程的主框架：事件流
面向对象的主框架：对象的过程

面向对象设计的首要任务就是找对象

描述事件流：依据事件的描述、依据数据的描述、两者并重的描述

读取->文件，形成->记录
事件->数据，事件->数据

主谓宾要齐全

理论：根据需求事件流中的名词和动词找对象

实践：
根据需求事件流中的事件找对象，事件由对象触发，责任分配，关注行为，接口驱动
根据需求事件流中的数据找对象，关注属性，模型驱动
根据需求异常流中的事件找对象，关注异常，异常驱动
根据这三者结合找备选对象

通过时序图验证对象能否实现事件流

3. 详细设计

1) 根据对象抽象类，形成类图
2) 对类分配责任，即主要成员函数

3) 通过时序图验证类的成员函数能否实现事件流
4) 对类设计成员变量，成员变量来自前置条件、后置条件、方法中的临时变量
5) 对类分析设计成员函数的返回值、参数、函数名、访控属性
6) 设计成员函数的过程：流程(活动图/状态图)、异常
7) 其它细节：构造函数、析构函数、拷贝构造函数、操作符重载
8) 通过继承和多态引入抽象
9) 套用设计模式

4. 编写代码/设计用例

5. 测试验证/修改错误

6. 产品发布/工程实施

三、数据采集客户机
---

1. 用例描述

数据采集客户机备份并读取登录日志文件，形成匹配日志记录，
通过网络发送给数据采集服务器

2. 参与者

系统管理员、数据采集客户机、日志读取器、日志发送器

3. 基本事件流

1) 备份系统日志，产生备份文件，以日期时间为后缀，如：
   wtmpx.20101101102745
2) 读取备份文件，产生登入登出日志记录：
   登录名、登录IP、登录进程ID、登入登出时间
3) 匹配登入登出日志记录生成匹配日志记录：
   登录名、登录IP、登录进程ID、登入时间、登出时间、登录时段
4) 根据服务器的IP地址和端口号向其发出连接请求，产生TCP连接
5) 将匹配日志记录发送到服务器

4. 可选事件流

无

5. 异常事件流

1) 备份系统日志失败，终止采集
2) 读取备份文件失败，终止采集
3) 匹配日志记录，有登出无登入的记录(计费系统运行之前登入)，直接丢弃，
   有登入无登出的记录(采集时仍在线)，存入登入文件，
   下次采集时再读出作为登入日志记录参与匹配

4) 连接服务器失败，将全部匹配日志记录存入发送失败文件，下次采集时重发
5) 发送过程中失败，将剩余匹配日志记录存入发送失败文件，下次采集时重发

6. 前置条件

1) 日志文件的路径：/var/adm/wtmpx
2) 登入文件的路径：<可执行程序所在目录>/logins.dat
3) 发送失败文件的路径：<可执行程序所在目录>/fail.dat
4) 服务器的IP地址和端口号

7. 后置条件

匹配日志记录集

8. 特殊规则

无

9. 用例图

02_client_usecase.pdf

10. 类图

03_client_data.pdf
04_client_exception.pdf
05_client_class.pdf

11. 时序图

06_client_sequence.pdf

12. 活动图/状态图

07_client_activity.pdf
08_client_state.pdf

13. 编写代码

client/

四、DMS数据采集服务器
---

1. 用例描述

数据采集服务器通过网络接收数据采集客户机上传的匹配日志记录，
保存到数据库中

2. 参与者

系统管理员、数据采集服务器、服务器套接字、客户线程、日志队列、存储线程、数据访问对象、数据库

3. 基本事件流

1) 接收匹配日志记录

A. 建立服务器侦听套接字
B. 等待并接受来自客户机的连接请求
C. 为该客户机创建客户线程，接收其上传的匹配日志记录
D. 将接收到的匹配日志记录压入日志队列

2) 保存匹配日志记录

A. 创建存储线程
B. 创建数据访问对象，建立与数据库的连接
C. 监视日志队列，从中弹出匹配日志记录
D. 将匹配日志记录通过数据访问对象保存到数据库中
E. 销毁数据访问对象，关闭与数据库的连接

多个客户线程将从客户机接收到的匹配日志记录保存到日志队列中缓冲，
单独的存储线程从日志队列中提取匹配日志记录保存到数据库中。这是
典型的生产者——消费者模型，可用条件变量进行同步

条件变量与生产者——消费者模型

生产者：产生数据的线程
消费者：使用数据的线程

通过缓冲区隔离生产者和消费者，与二者直连相比，
避免相互等待，提高运行效率

生产快于消费，缓冲区满，撑死
消费快于生产，缓冲区空，饿死

条件变量可以让调用线程在满足特定条件的情况下暂停

```
int pthread_cond_init (pthread_cond_t* cond,
    const pthread_condattr_t* attr);
```

亦可

```
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;

// 使调用线程睡入条件变量cond，同时释放互斥锁mutex
int pthread_cond_wait (pthread_cond_t* cond,
    pthread_mutex_t* mutex);

int pthread_cond_timedwait (pthread_cond_t* cond,
```

```
    pthread_mutex_t* mutex,
    const struct timespec* abstime);

struct timespec {
    time_t tv_sec;  // Seconds
    long   tv_nsec; // Nanoseconds [0 - 999999999]
};

// 从条件变量cond中唤出一个线程，
// 如果该线程能够获得原先的互斥锁，
// 则从pthread_cond_wait/pthread_cond_timedwait函数中返回，
// 否则继续阻塞于等待加锁
int pthread_cond_signal (pthread_cond_t* cond);

// 从条件变量cond中唤出所有线程，
// 如果其中的某个线程能够获得原先的互斥锁，
// 则从pthread_cond_wait/pthread_cond_timedwait函数中返回，
// 其余线程继续阻塞于等待加锁
int pthread_cond_broadcast (pthread_cond_t* cond);

int pthread_cond_destroy (pthread_cond_t* cond);
```

范例：cond.c

注意：一个线程被从条件变量中唤出以后，
在其重新获得先前被释放的互斥锁之前，
其它线程有可能执行锁区间的代码。
因此从pthread_cond_wait/pthread_cond_timedwait函数返回以后，
有必要对导致该线程睡入条件变量的条件再做一次判断。

范例：bc.c (if->while)

---

4. 可选事件流

无

5. 异常事件流

1) 访问数据库失败，终止采集
2) 网络通信失败，终止采集
3) 创建线程失败，终止采集

6. 前置条件

1) 用于套接字绑定的IP地址和端口号
2) 数据库的用户名和口令

7. 后置条件

数据库中记录

8. 特殊规则

无

9. 用例图

09_server_usecase.pdf

10. 类图

10_server_data.pdf
11_server_exception.pdf

线程封装

```
class Thread {
public:
    void start (void) {
        pthread_t tid;
        pthread_create (&tid, NULL, run, this);
    }

private:
    static void* run (void* arg) {
        static_cast<Thread*> (arg)->run ();
        return NULL;
    }
    virtual void run (void) = 0;
};

class MyThread : public Thread {
private:
    void run (void) { ... }
};

MyThread thread;
thread.start ();
```

12_server_class.pdf

11. 时序图

13_server_sequence.pdf

12. 编写代码

server/

五、定时自动运行数据采集客户机

启动服务器、复位客户机
编写客户机启动脚本client.sh
执行crontab -e命令
选3
进入vi编辑环境
插入* * * * * /home/tarena/dms/client/client.sh
底行:wq存退
等1分钟，观察服务器的反应

crontab定时任务
A B C D E <命令>
每隔一段时间执行一次<命令>
A：分钟，0-59
B：小时，0-23
C：天，1-31
D：月，1-12
E：星期，0-6，0星期天
*：通配符，任意

例如
* * * * *
任意月任意天任意小时任意分钟，无论星期几执行
15 * * * 1-5
任意月任意天任意小时第15分钟，周一到周五执行
0,30 1,3,5 * * 1-5
任意月任意天1点3点5点的第0分或30分，周一到周五执行

六、数据采集客户机增加GUI
---

qmake -project
qmake
make

七、数据采集服务器增加Pro*C
---

mv oracledao.cpp oracledao.pc
vi oracledao.pc
加入Pro*C代码，上传至Oracle服务器，执行
proc oracledao.pc oname=oracledao.cpp parse=none code=cpp
得到
oracledao.cpp
上传数据采集服务器的其它源代码，执行make
注意增加库-lsocket -lnsl -lclntsh

八、建表并测试

启动数据库，登录oracle用户，执行
sqlplus / as sysdba
startup

删序列、删表，在sqlplus中执行
```
drop sequence dmslog_id;
truncate table dmslog;
drop table dmslog purge;
```

建序列、建表，在sqlplus中执行
```
create sequence dmslog_id;
create table dmslog (
    id         number primary key,
    logname    varchar2(32),
    logip      varchar2(32),
    pid        number,
    logintime  date,
    logouttime date,
    durations  number);
```

启动服务器、复位并启动客户机、查看dmslog表
```
select * from dmslog;
```

查看Oracle的错误信息
```
oerr ora 1034
```

九、项目总结
---

1. 核心技术

1) 基于UML的面向对象设计技术
2) C/C++语言编程技术
3) Shell脚本语言编程技术
4) Linux/Unix操作系统编程技术
5) 基于Pro*C的Oracle数据库访问技术
6) 基于PL/SQL的Oracle数据库编程技术
7) 基于Socket的网络编程技术
8) POSIX多线程及线程同步技术
9) 基于Qt的图形界面编程技术
10) STL标准模板库技术

2. 技术难点

3. 团队协作

4. 经验教训

5. 展望未来