互联网专车系统课程设计

1.应用背景介绍和功能描述

1.1 应用背景介绍

互联网专车系统是由约车客户端和车载终端以及后台中心调度服务器构成的一套在线约车的调度系统。

约车客户端是安装在具有 GPS（global positioning system 全球定位系统）定位功能的移动终端（手机，pad 等）上的一款 APP。可以实现约车客户的叫车，约车，付费等功能。

车载终端是安装在专车上的一款集导航，调度，娱乐，远程刷卡，电话等为一体的智能移动终端设备。有了 GPS 车载终端（专车司机端）和调度中心后台（服务端），可以最优化驾驶员和乘客的出车和求车的资源配置。

1.2 功能描述

约车客户端的功能主要有：

1、app 需要有客户的注册，登陆，退出等功能。

2、实现客户端的 GPS 定位并上报服务器;

3、打车、约车功能;

4、显示司机信息如证件号、车牌号、照片等;

5、显示司机专车与客户的距离信息（可实时更新）;

6、显示并查询行程信息，如历史约车记录、付费情况等;

7、与驾驶员的通信功能（如文字聊天，语音聊天等）

车载系统提供的功能有：

1、驾驶员登录或签退功能：终端必须在登录的情况下，才能实现以下的功能。当驾驶员上班时登录，下班或交接班时签退。

2、导航功能：采用最先进的凯立德或高德电子地图导航软件，为驾驶员提供语音导航服务。

3、接单功能：通过 3G /4G 移动通信实现与中心调度服务器的链接，接收服务器上委派的约车任务;

4、关单功能：任务完成后，由驾驶员实现关单，交给服务器统一处理计费;

5、与约车客户的通信功能（如文字聊天，语音聊天等）;

6、定时上报专车 GPS 位置信息;

中心调度服务器功能有：

1、管理约车客户和驾驶员的登录、注册、退出等功能;

2、管理和维护驾驶员在线状态信息（如 GPS 定位，心跳包等）;

3、接收约车客户的下单任务;

4、分发下单任务给指定驾驶员（须计算乘客和驾驶员距离）;

5、计费结算;

本项目为模拟专车系统调度应用，模拟车载终端和调度中心分别为客户端和服务器端。

注意：本项目有些应用无法模拟的地方均作了简化。另外，文中提到的通信格式（协议）通俗讲是一种封装，就像收发信件需要信封一样。每条协议都是有加 header 头的，一是考虑安全，二是考虑你这条协议是干什么的，这样 socket 通信中的网络数据互不干扰，各自处理。客户端和服务器端模型均采用 linux + epoll IO 复用实现。

2.通讯格式和消息类型

2.1 通讯格式

事件类型 TYPE	1
	2
数据长度[注 1]	3
	4
消息体（body）	5
	…
	n

注 1：长度= 消息内容长度 低位在前 （小端字节）

2.2 车载终端发起消息类型

客户端本地原本处理 1 秒的定位信息:本意为客户端每秒检查 GPS 串口和车辆参数，现在简单模拟为客户端线程每秒读取文本数据，接着解析 GPS 数据显示年、月、日、星期、方位，速度等数据。

另外客户端有了定位和车辆参数信息，每隔一段时间会向调度中心发送心跳包汇报情况。

FLAG	消息类型	附
1000	上报ＧＰＳ信息	车载终端程序启动后，首先开启，定时上报
1001	获取任务	受限操作，必须驾驶员登录后才能操作 定时（例如每个 5 秒）从服务器获取约车任务队列；
1002	驾驶员注册	简单模拟，检查数据库信息，建立链表遍历成功后，写数据到数据库
1003	驾驶员登录签到	同注册，遍历比较，登陆成功后，上报 IP 和 port

1004	驾驶员登出签退	驾驶员状态为登出
1005	上传驾驶员相片信息	受限操作，必须驾驶员登录后才能操作
1006	保留	
1007	心跳包	客户端每 5 秒发一次心跳包，服务器子线程每 5 秒扫描一次在线用户链表，超过 60 次（5 分钟），可判断客户端已死亡，并从链表中删除节点，服务器主线程收到客户端心跳包后清零此客户端的 timers
1008	接受任务	上传接收任务的编号等信息，服务器收到后会把该任务置为已运行状态
1009	关单	驾驶员护送乘客到达目的地后，点击结算，此任务将被服务器关闭，并实现计费；

2.3 乘客终端发起消息类型

客户端本地原本处理 1 秒的定位信息:本意为客户端每秒检查 GPS 串口，现在简单模拟为客户端线程每秒读取文本数据，接着解析 GPS 数据显示年、月、日、星期、方位，速度等数据。

FLAG	消息类型	附
2000	乘客注册	简单模拟，检查数据库信息，建立链表遍历成功后，写数据到数据库
2001	乘客登录签到	同注册，遍历比较，登陆成功后，上报 IP 和 port
2002	驾驶员登出签退	驾驶员状态为登出
2003	下单	发送约车请求

2004	请求下载驾驶员相片	受限操作，必须乘客下单后才能操作
2005	请求驾驶员相片信息	受限操作，必须乘客下单后才能操作
2006	保留	
2007	心跳包	客户端每 5 秒发一次心跳包，服务器子线程每 5 秒扫描一次在线用户链表，超过 60 次（5 分钟），可判断客户端已死亡，并从链表中删除节点，服务器主线程收到客户端心跳包后清零此客户端的 timers
2008	查询历史约车记录	

3. 数据结构定义

3.1 在线乘客信息

3.2 在线驾驶员信息

3.4 约车任务信息

3.6 约车任务文件存储格式

3.7

3.8 一秒的定位信息（模拟来自美国全球卫星 GPS 定位数据）

客户端处理（模拟线程每秒解析数据，处理成北京时间，显示年月日时分秒星期)

参考数据：

$GPRMC,100119.999,A,2236.8226,N,11403.7299,E,0.62,120.87,220506,,**（详解见下文）

3.8.1 GPS 解析知识

提取定位数据

GPS 接收机只要处于工作状态就会源源不断地接收 GPS 导航定位信息。把数据放入缓存发送到车载终端进程处理，在没有进一步处理之前缓存中是一长串字节流，这些信息在没有经过分类提取之前是无法加以利用的。因此，必须通过程序将各个字段的信息从缓存字节流中提取出来，将其转化成有实际意义的。同其他通讯协议类似，对 GPS 进行信息提取必须首先明确其帧结构，然后才能根据其结构完成对各定位信息的提取。在本文中，其接受的数据主要由帧头、帧尾和帧内数据组成，根据数据帧的不同，帧头也不相同，主要有"$GPGGA"、"$GPGSA"、"$GPGSV"以及"$GPRMC"等。这些帧头标识了后续帧内数据的组成结构，各帧均以回车符和换行符（0X0D、0X0A）作为帧尾标识一帧的结束。对于通常的情况，我们所关心的定位数据如经纬度、速度、时间等均可以从"$GPRMC"帧中获取得到，该帧的结构及各字段释义如下，

数据丰富的最典型情况，均为 ASCII 字符，可以通过 buf-0x30 转换为数字。

$GPRMC,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,<10>,<11>*hh

<1> 当前位置的格林尼治 UTC 时间，格式为 hhmmss.sss

<2> 状态， A 为有效位置， V 为非有效接收警告，即当前天线视野上方的卫星个数少于 3 颗。A 为车辆已经定位，V 为没有定位

<3> 纬度， 格式为 ddmm.mmmm 格式不定长，例如：3111.4364

<4> 标明南北半球， N 为北半球、S 为南半球

<5> 经度，格式为 dddmm.mmmm 格式不定长，例如：12125.1027

<6> 标明东西半球，E 为东半球、W 为西半球

<7> 地面上的速度，范围为 0.0 到 999.9

<8> 方位角，范围为 000.0 到 359.9 度

<9> 日期， 格式为 ddmmyy 需要通过+2000 转换

<10> 地磁变化，从 000.0 到 180.0 度(不考虑)

<11> 地磁变化方向，为 E 或 W（不考虑）

数据缺失的最典型情况将是：

$GPRMC,<1>,<2>,,,,,,,,,*hh

<1> 当前位置的格林尼治时间，格式为 hhmmss.sss

<2> 状态， A 为有效位置， V 为非有效接收警告，即当前天线视野上方的卫星个数少于 3 颗。

数据将会出项缺失，时间回复到出厂时间，定位情况为 V，（不定位警告）。

至于其他几种帧格式，除了特殊用途外，平时并不常用，虽然接收机也在源源不断地向主机发送各种数据帧，但在处理时一般先通过对帧头的判断而只对"$GPRMC"帧进行数据的提取处理。如果情况特殊，需要从其他帧获取数据，处理方法与之也是完全类似的。由于帧内各数据段由逗号分割，因此在处理缓存数据时一般是通过搜寻 ASCII 码"$"来判断是否是帧头，在对帧头的类别进行识别后再通过对所经历逗号个数的计数来判断出当前正在处理的是哪一种定位导航参数，并作出相应的处理。

例如某数据：

$GPRMC,100119.999,A,2236.8226,N,11403.7299,E,0.62,120.87,220506,,**

’100119.999 --时分秒（格林威治时间）

’2236.8226,N --北纬坐标（2236.8226）

2236.8226,S ----南纬坐标

’11403.7299,E --东经坐标（11403.7299）

11403.7299,W ----西经坐标

’0.62 --gps 的移动速度

’120.87 --地面的方位角

’220506 --日月年 06+2000=2006 年

```
//参考 GPS 数据结构，存储 ascii 字符串
typedef  struct  _GPSData
{
char  date[15]  ;  //Gps 数据日期
char  time[15]  ;    //Gps 数据时间
char  latitude_type;  //纬度类型，北纬，南纬
char  latitude[15]  ;  //纬度值
char  longitude_type;  //经度类型，东经，西经
char  longitude[15]  ;//经度值
char  speed[6];//速度
char  cog[10];//方位角
char  IsValid;//是否定位标志
```

}GPSData;

到此为止，已将时间和经纬度信息提取到 GPS 结构数组 GPSData 中的各个变量中去，后续的处理可根据该结构中存储的数据作出相应的处理。

比如：1、请继续判断，是否定位；

2、根据方位角判断目前车辆定位在是正东，正西，正南，正北，东南，东北，西南，西北；

3、请对 GPS 格林尼治时间（世界时间）进行+8 转化为北京时间，提示：请进行平年闰年月份等方面考虑。闰年一年 366 天，平年 365 天，闰年在 2 月份有 29 天，平年 2 月份 28 天。闰年特点：1、能被 4 整除并且不能被 100 整除。2、或者是能被 400 直接整除。

4、显示出星期几，提示：可以以 2000 年 1 月 1 日星期六为参考起点，计算总天数。

3.9 业务处理信息（**模拟调度信息发送**）（略）

4. Sock 编程

1、基于 tcp 协议的客户端服务端通讯。

2、客户端和服务端均使用 epoll I/O 复用机制，均关注 socket 文件描述符，支持多客户端连接。

3、客户端和服务端进行数据交互，均采用协议的方式处理。

4、服务端使用链表记录（单链表即可）当前客户端的会话连接，并动态维护会话。

5. 配置文件

本文使用配置文件(.ini.dat.cfg.txt 等)作为数据库，存放用户名和密码信息，存放参数信息。

6. 用户认证

服务端和客户端需要用户名密码登陆机制。客户端输入账号密码后，服务器从配置文件中读取已认证的合法用户信息，读入链表中进行比较。

7. 心跳机制

客户端与服务端之间使用心跳机制。心跳机制：客户端定时向服务端发送一个数据包（心跳包），证明自己活着，也可以汇报状况。服务器超过一定的时间没有收到服务

端的心跳包则说明客户端出现问题（可能出现了客户端死机现象），做出相应的处理（一般处理记录状态，并且断开连接）。

8.多线程处理

客户端和服务端可能均可能有多线程处理。

9.项目要求

1、采用 C 语言完成代码的编写。

2、分别给客户端和服务端编写 makefile 管理整个项目

3、编写项目设计书。

4、以模块化编写项目代码，按照不同模块组织.h/.c 文件

5、规范代码格式并添加注释，善用小函数，函数不宜过长。

6、单元测试，每个函数都必须测试好，尽量减小 bug。

7、项目中遇到的问题和解决方法，项目总结写心得体会。