Chassis Simulator

背景介绍

这是一份模拟底盘和GPS模块的一份源码，该工程的目的是利用纯运动估计(wheel odometry)来模拟车底盘的实际运动情况, 并实时反馈车的姿态信息和GPS坐标。在不上车的情况下验证 `planner` 层和 `chassis` 层的程序和功能。

目录结构

```
目录结构描述
├── README.md                          // 说明文件
├── Reference.md                       // 源码参考和借鉴的一些链接
├── src                                // 源码.c, .h, 和 Makefile的文件夹
│   ├── canpacket.c                    // 串口, TCP-IP, SocketCan读写can包
│   ├── canpacket.h                    // can包相关结构体的定义
│   ├── coordinate.c                   // UTM坐标和WGS84坐标相互转换
│   ├── coordinate.h                   // 幅度和度的相互转换
│   ├── initialsrv.c                   // TCP-IP, SocketCan, 串口初始化设置
│   ├── initialsrv.h                   // 初始化接口
│   ├── Makefile                       // 执行编译的脚本文件
│   ├── nmeagps.c                      // NMEA标准的GPS命令的解编码, GGA,RMC,HDT
│   ├── nmeagps.h                      // GGA,RMC,HDT命令相关的结构体定义
│   ├── pthreadutil.c                  // 线程和定时器相关接口的封装
│   ├── pthreadutil.h                  // 条件变量, 互斥量, 定时器的结构体
│   ├── threadchassiscan.c             // 获取chassis发过来的姿态
│   ├── threadchassiscan.h             // 读can包的线程
│   ├── threadchassisgps.c             // 初始化初始经纬度坐标, 实时坐标换算
│   ├── threadchassisgps.h             // GPS坐标换算的线程
│   ├── threadepollsrv.c               // 主线程, 创建三子线程, epoll IO复用机制
│   ├── threadvehicle.c                // 实时调整车的姿态,利用惯性环节,定时器初始
化
│   ├── threadvehicle.h                // 车辆姿态模拟线程
│   ├── threadvehicle.c                // 实时调整车的姿态,利用惯性环节,定时器初始
化
│   └── utilconfig.h                   // 配置工具选项; 时间戳, 延时, 文件描述符函
数
└── README.pdf                         // 工程说明文件的pdf格式
```

整个工程结构简介

整个工程利用了epoll IO复用机制来管理TCP-IP的建连, 管理定时器来发送数据(`create_timerfd` 函数把定时间抽象为文件描述符)。

该工程兼容速度、方向盘、刹车三报文格式和 `gacu` 报文格式。

整个工程利用了4个线程来进行并发管理读写, 计算车的姿态(前轮转角, 车速)和GPS坐标。

1. `threadepollsrv` 线程: 主线程。
   - 输入参数的解析;
   - `thread_chassis_gps` , `thread_chassis_can` , `thread_vehicle` 三个子线程的创建;
   - `epoll` IO复用机制的管理;
   - 兼容 `TCP-IP` , `SocketCan` 和 `Serial` 三种不同的通信协议;
   - 对于 `thread_chassis_can` , `thread_vehicle` 这两个线程, 如果使用的是 `TCP-IP` 端口, 由 `CAN` 端口的 `TCP-IP` 建连时创建, 如果是使用 `SocketCan` 和 `Serial` 这两种端口, 由 `gps` 端口的 `TCP-IP` 建连时穿件。
2. `thread_chassis_gps` 线程: 实时计算 `gps` 坐标的线程。
   - 1ms刷新一次;
   - 根据车辆的不同, 航向角初始值应该有所不同, 航向角本身以正北为参考方向, 有可能公司的车配置的参考方向并不是正北, 比如 `car9` 的参考方向应该是正西, 所以航向角初始化应该是 `90°` ;
   - 系统经纬度使用的坐标系是**谷歌地球**的经纬度, 其他的坐标系应该有误差;
   - `thread_chassis_gps` 线程利用 `./launch_uos.sh` 启动的时候会接收一个 `SIGPIPE` 信号(值为：13, 表示管道破裂: 写一个没有读端口的管道 — 对一个对端已经关闭的socket调用两次write, 意味着对端会把套接字关闭), 采取的解决方案是直接忽略了该信号。
3. `thread_chassis_can` 线程: 读 `can` 包, 并将其转换为车的姿态的线程。
   - 针对不同 `can id` 分开解析报文数据, 并把报文中的8位数据转换为车的实际需要的姿态信息(车速, 前轮转角, 刹车)
   - 为了兼容更多的 `can` 包, 处理上做得有点复杂: 收到的10字节的 `can` 包 –> 按不同的 `can id` 区分出 `can` 包 –> 解析 `can` 包每位对应的含义 –> 再将每位的含义转换为车的姿态(`double` 类型); 发送编码车姿态的过程与之相反;
   - 全局变量 `enable_gacu` 如果为1, 则使用的是 `gacu` 报文格式进行通信, 如果为0, 则使用的是速度、方向和刹车分开的 `can` 包进行通信, 其他值是没有被定义。
4. `thread_vehicle` 线程: 模拟车身的线程, 并通过简单的 `Bicycle Model` 计算 `Wheel Odometry` 的 `x, y` 坐标。
   - 车速、前轮转角、刹车的调整分别对应一个惯性环节, 车速和刹车的tao值应该大一点, 前轮转角应该小一些;
   - `Wheel Odometry` (里程计)的计算, 依赖车的航向角, 前轮转角, 车速三个参数, 而且车姿态初始化时应该把前轮转角设置为0, 这样可以方位角和航向角相等, 方便计算。
   - 计算车下一个时刻的位置的大致过程为: 先将 `WGS84` 格式的经纬度转换为 `UTM` 坐标系 –> 再利用 `Wheel Odometry` 计算公式在 `UTM` 坐标系中计算位移量、速度增量、航向角增量 –> 加上计算出的位移增量, 并将 `UTM` 坐标转换为 `WGS84` 格式的经纬度

编译说明

进入 src 子目录下, 直接 make 便可以编译该工程, 但注意的是该工程依赖 -lpthread 和 -lm 库。

编译过程为: 直接 make

```
> make
```

会得到以下提示:

```
cc -Wall -g -c threadepollsrv.c -o threadepollsrv.o
cc -Wall -g -c canpacket.c -o canpacket.o
cc -Wall -g -c nmeagps.c -o nmeagps.o
cc -Wall -g -c pthreadutil.c -o pthreadutil.o
cc -Wall -g -c coordinate.c -o coordinate.o
cc -Wall -g -c threadchassiscan.c -o threadchassiscan.o
cc -Wall -g -c threadchassisgps.c -o threadchassisgps.o
cc -Wall -g -c threadvehicle.c -o threadvehicle.o
threadvehicle.c: In function 'thread_vehicle':
threadvehicle.c:341:6: warning: unused variable 'conn_gps' [-Wunused-variable]
  int conn_gps = *((int*) arg);
      ^
cc -Wall -g -c initialsrv.c -o initialsrv.o
cc -Wall -g threadepollsrv.o canpacket.o nmeagps.o pthreadutil.o coordinate.o
threadchassiscan.o threadchassisgps.o threadvehicle.o initialsrv.o
/usr/lib/aarch64-linux-gnu/libpthread.so /usr/lib/aarch64-linux-gnu/libm.so -o
threadepollsrv
```

编译会生成 threadepollsrv 这个可执行文件。

使用说明

该工程通过输入命令行参数来支持 serial， TCP-IP， SocketCan 和 gacu 报文格式, 速度、方向、刹车分开的三报文格式。

1. 使用 serial 端口和三报文格式启动该工程, 具体命令行参数输入如下:

   ```
   > sudo ./threadepollsrv -s 192.168.100.246 5017
   ```

   - 因为串口默认端口是 /dev/ttyUSB0， 而且需要超级用户才能访问, 需要加 sudo；
   - -s 参数选项表示使用 serial 端口通信, 默认使用三报文格式通过 can 包发送车的姿态;
   - 192.168.100.246 是服务器端的 ip 地址，5017 是 gps 监听对应端口号;
   - 利用 ./launch_uos.sh 启动对端程序建连成功后会有如下提示:

```
use serial to link!                                    //说明使用的serial端口
进行通信
ip addr:192.168.100.246 gps port number:5017     //gps的ip地址和端口号
listencanfd:4 listengpsfd:5        //串口的文件描述符, gps的监听文件描述
符
enable_gacu = 0                                        //表示使用的是三报文格式
ip = 192.168.100.100 port = 55482                  //gps建连成功, 对端ip地
址和端口号
listengpsfd = 5, conngpsfd = 6                    //gps的监听文件描述符, 连
接文件描述符
listencanfd = 4
gpspacketfd: 7                                         //发送gps数据的定时器的
文件描述符
thread_vehicle enable_gacu = 0
packet302fd: 8                                         //发送302can包的定时器
文件描述符
packet704fd: 9                                         //发送704can包的定时器
文件描述符
packet132fd: 10                                        //发送132can包的定时器
文件描述符
recv a sig = 13                    //thread_chassis_gps线程收到的SIGPIPE
信号
gps write error! ID:139882780575488 cancel       //取消
thread_chassis_gps线程
ID:139882763790080 cancel                          //取消
thread_chassis_can线程
ID:139882772182784 cancel                          //取消thread_vehicle线
程
ip = 192.168.100.100 port = 55484                  //gps客户端和服务器重新
建连
listengpsfd = 5, conngpsfd = 7
listencanfd = 4
gpspacketfd: 8                                         //定时器的文件描述符
thread_vehicle enable_gacu = 0
packet302fd: 9
packet704fd: 10
packet132fd: 11
```

2. 使用 serial 端口和 gacu 格式启动该工程, 具体命令行参数输入如下:

```
> sudo ./threadepollsrv -sg 192.168.100.246 5017
```

- 使用 gacu 报文格式只需在命令选项后添加一个 g 即可, 其他详细信息同上;
- g 为命令参数 -c 的选项参数, 代表是否使用 gacu 报文格式;
- 注意:使用 gacu 报文格式，一定要在 uos_common.json 中加上 "enable_gacu": 1 ;
- 利用 ./launch_uos.sh 启动对端程序建连成功后会有如下提示:

```
use serial to link!
usage gacu can packets!
ip addr:192.168.100.246 gps port number:5017
listencanfd:4 listengpsfd:5
enable_gacu = 1                    //enable_gacu为全局变量，1表示使用
gacu报文格式
ip = 192.168.100.100 port = 55494
listengpsfd = 5, conngpsfd = 6
listencanfd = 4
gpspacketfd: 7
thread_vehicle enable_gacu = 1
packet5A2fd: 8                     //发送5A2报文的定时器
recv a sig = 13
gps write error! ID:140594141251328 cancel
ID:140594124465920 cancel
ID:140594132858624 cancel
ip = 192.168.100.100 port = 55496
listengpsfd = 5, conngpsfd = 7
listencanfd = 4
gpspacketfd: 8
thread_vehicle enable_gacu = 1
packet5A2fd: 9
```

3. 使用 TCP-IP 和三报文格式启动该工程，具体命令行参数输入如下:

```
> ./threadepollsrv -t 192.168.100.246 5017 5016
```

- 使用 -t 表示使用 TCP-IP 接口发送 can 包;
- 192.168.100.246 是服务器端的 ip 地址，5017 是 gps 对应的监听端口号，5016 是 can 包对应的监听端口号;
- uos_common.json 中要把端口号和 ip 地址配置一致;
- 利用 ./launch_uos.sh 启动对端程序建连成功后会有如下提示:

```
use tcp ip to link!                                  //使用tcp-ip协议通信
ip addr:192.168.100.246 can port number:5016         //发送can包的监听端口号
ip addr:192.168.100.246 gps port number:5017         //发送gps数据的监听端口
号
listencanfd:4 listengpsfd:5                          //监听端口号对应的文件描
述符
enable_gacu = 0                                      //表示不是用gacu格式发送
can包
ip = 192.168.100.100 port = 55500                    //gps连接的ip地址和端口
号
listengpsfd = 5, conngpsfd = 6                       //gps监听和连接的文件描
述符
gpspacketfd: 7                                       //发送gps数据对应的定时
器文件描述符
recv a sig = 13                                      //收到一个管道破裂的信
号，gps会重联
gps write error! ID:140558644332288 cancel           //取消
thread_chassis_gps线程
ip = 192.168.100.100 port = 55502                    //新连接的端口号
listengpsfd = 5, conngpsfd = 7
gpspacketfd: 8
ip = 192.168.100.100 port = 58848                    //can连接的ip地址和端口
号
thread_vehicle enable_gacu = 0                       //使用三数据包格式接受，
发送车的姿态
packet302fd: 10                                      //发送302报文的定时器文
件描述符
packet704fd: 11                                      //发送704报文的定时器文
件描述符
packet132fd: 12                                      //发送132报文的定时器文
件描述符
EHB packet error for steer control                   //tcp-ip通信时，不能收
到131报文
EHB packet error for steer control                   //此错误可以忽略
EHB packet error for steer control
```

4. 使用 TCP-IP 和 gacu 报文格式启动该工程，具体命令行参数输入如下:

```
> ./threadepollsrv -tg 192.168.100.246 5017 5016
```

- 使用 -tg 表示使用 TCP-IP 接口发送 can 包, 并使用 gacu 格式发送 can 包;
- 192.168.100.246 是服务器端的 ip 地址，5017 是 gps 对应的监听端口号，5016 是 can 包对应的监听端口号;
- uos_common.json 中要把端口号和 ip 地址配置一致;
- uos_common.json 中要把 enable_gacu 设置为1;
- 利用 ./launch_uos.sh 启动对端程序建连成功后会有如下提示:

```
use tcp ip to link!
usage gacu can packets!
ip addr:192.168.100.246 can port number:5016
ip addr:192.168.100.246 gps port number:5017
listencanfd:4 listengpsfd:5
enable_gacu = 1                                    //使能`gacu`格式发送
`can`包
ip = 192.168.100.100 port = 55508
listengpsfd = 5, conngpsfd = 6
gpspacketfd: 7
recv a sig = 13
gps write error! ID:140245759547136 cancel
ip = 192.168.100.100 port = 55510
listengpsfd = 5, conngpsfd = 7
gpspacketfd: 8
ip = 192.168.100.100 port = 58856
thread_vehicle enable_gacu = 1
packet5A2fd: 10
```

5. 使用 SocketCan 和三报文格式启动该工程，具体命令行参数输入如下:

```
> ./threadepollsrv -c 192.168.100.100 5017
```

- 使用 -c 表示使用 SocketCan 接口发送 can 包;
- 默认是使用 can1 端口, 也可以使用 ./threadepollsrv -c 192.168.100.100 5017 can0 使用 can0 端口;
- uos_common.json 中要把端口号和 ip 地址配置一致;
- 注意: 如果使用 can 回环(can1 发包，can1 也收包)–需要挂一个 can 设备才能成功;
- SocketCan 仅在同一台 TX2 上测试过;
- 利用 ./launch_uos.sh 启动对端程序建连成功后会有如下提示:

```
use raw can to link!                                    //使用socket-can连接
interface = can1, family = 29, type = 3, proto = 1  //can 端口号和配置信息
ip addr:192.168.100.100 gps port number:5017        //gps的ip地址和监听端口号
listencanfd:4 listengpsfd:5                             //对应的文件描述符
enable_gacu = 0                                         //使用三包格式发送和接受车的姿态
ip = 192.168.100.100 port = 60170                   //对端ip地址和连接端口号
listengpsfd = 5, conngpsfd = 6
listencanfd = 4
thread_vehicle enable_gacu = 0
packet302fd: 7                                      //发送数据的四个定时器的文件描述符
packet704fd: 8
packet132fd: 9
gpspacketfd: 10
```

6. 使用 SocketCan 和 gacu 报文格式启动该工程，具体命令行参数输入如下:

```
> ./threadepollsrv -cg 192.168.100.100 5017
```

- 使用 -cg 表示使用 SocketCan 接口发送 can 包, 并使用 gacu 格式发送 can 包;
- 192.168.100.100 是服务器端的 ip 地址，5017 是 gps 对应的监听端口号;
- 默认是使用 can1 端口, 也可以使用 ./threadepollsrv -cg 192.168.100.100 5017 can0 使用 can0 端口;
- uos_common.json 中要把端口号和 ip 地址配置一致;
- uos_common.json 中要把 enable_gacu 设置为1;
- 利用 ./launch_uos.sh 启动对端程序建连成功后会有如下提示:

```
use raw can to link!
usage gacu can packets!
interface = can1, family = 29, type = 3, proto = 1
ip addr:192.168.100.100 gps port number:5017
listencanfd:4 listengpsfd:5
enable_gacu = 1                                         //使能`gacu`格式的can包
ip = 192.168.100.100 port = 60188
listengpsfd = 5, conngpsfd = 6
listencanfd = 4
gpspacketfd: 7
thread_vehicle enable_gacu = 1
packet5A2fd: 8
```

7. 特别说明

- 使用 gacu 格式时,不用管 ./launch_uos.sh 报的 GACU Write CAN 错误;

- 在选择对应命令行参数启动该工程时, 一定要在 `uos_common.json` 中配置对应的参数; 否则程序无法正确运行;
- 该程序默认是 `epoll` 机制管理，所以对端死掉后不会自动关闭程序, 需要 `ctrl+c` 来关闭程序;

作者


```
@Author: 蒋龙
@E-Mail: long.jiang@uisee.com
@QQ/Wechat: 1417700745
@Telephone: 17781689052
```