

liuwinner


2019-02-14 17:59:55

4632

收藏 35

版本

分类专栏：[Linux](#) [QT](#)

 Linux

同时被 2 个专栏收录

附上Linux下周立功can配置文档：

1.cpp代码

```
1 Can_Control::Can_Control()
2 {
3
4
5 }
6
7 int Can_Control::can_send()
8 {
9     int stSocket_L0, stSend_L0;
10    struct sockaddr_can addr; //can总线的地址 同socket编程里面的 sockaddr结构体 用来设置can外设的信息
11    struct ifreq ifr; //接口请求结构体
12
13    struct can_frame frame[2] = {{0}}; //要发送的buffer
14    /* 创建socket套接字
15     * PF_CAN 为地址 同网络编程中的AF_INET 即ipv4协议
16     * SOCK_RAW使用的协议类型 SOCK_RAW表示原始套接字 报文头由自己创建
17     * CAN_RAW为使用的具体协议 为can总线协议
18     */
19    stSocket_L0 = socket(PF_CAN, SOCK_RAW, CAN_RAW); //创建套接字
20    strcpy(ifr.ifr_name, "can0");
21    ioctl(stSocket_L0, SIOCGIFINDEX, &ifr); //指定 can0 设备
22    addr.can_family = AF_CAN; //协议类型
23    addr.can_ifindex = ifr.ifr_ifindex; //can总线外设的具体索引 类似 ip地址
24    bind(stSocket_L0, (struct sockaddr *)&addr, sizeof(addr)); //将套接字和canbus外设进行绑定,即套接字与 can0 绑定
25    //禁用过滤规则,本进程不接收报文,只负责发送
26    setsockopt(stSocket_L0, SOL_CAN_RAW, CAN_RAW_FILTER, NULL, 0);
27    //生成两个报文
28    frame[0].can_id = 0x11;
29    frame[0].can_dlc = 1;
30    frame[0].data[0] = '1';
31    frame[1].can_id = 0x22;
32    frame[1].can_dlc = 1;
33    frame[1].data[0] = '2';
34    //循环发送两个报文
35    while(1)
36    {
37        stSend_L0 = write(stSocket_L0, &frame[0], sizeof(frame[0])); //发送 frame[0]
38        if(stSend_L0 != sizeof(frame[0]))
39        {
40            printf("Send Error frame[0]\n");
41            break; //发送错误,退出
42        }
43        sleep(1);
44        stSend_L0 = write(stSocket_L0, &frame[1], sizeof(frame[1])); //发送 frame[1]
45        if(stSend_L0 != sizeof(frame[0]))
46        {
47            printf("Send Error frame[1]\n");
48            break;
49        }
50        sleep(1);
51    }
52    close(stSocket_L0); //关闭套接字
53    return 0;
54 }
55
56 int Can_Control::can_recv()
57 {
58
59     int stSocket_L0, stRecv_L0;
60     struct sockaddr_can addr;
61     struct ifreq ifr;
62     struct can_frame frame;
63     struct can_filter rfilter[1];
64     stSocket_L0 = socket(PF_CAN, SOCK_RAW, CAN_RAW); //创建套接字
65     strcpy(ifr.ifr_name, "can0");
66     ioctl(stSocket_L0, SIOCGIFINDEX, &ifr); //指定 can0 设备
67     addr.can_family = AF_CAN;
68     addr.can_ifindex = ifr.ifr_ifindex;
69     bind(stSocket_L0, (struct sockaddr *)&addr, sizeof(addr)); //将套接字与 can0 绑定
70     //定义接收规则,只接收表示符等于 0x11 的报文
71     rfilter[0].can_id = 0x11;
72     rfilter[0].can_mask = CAN_SFF_MASK;
73     //设置过滤规则
74     setsockopt(stSocket_L0, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));
75     while(1)
76     {
77         stRecv_L0 = read(stSocket_L0, &frame, sizeof(frame)); //接收报文
78         //显示报文
79         if(stRecv_L0 > 0)
80         {
81             printf("Idata[0]=0x%x\n", frame.data[0]);
82         }
83     }
84     close(stSocket_L0);
85     return 0;
86 }
87
88
89
90
91 int Can_Control::socket_connect(const char*m_port)
92 {
93     int stSocket_L0 = -1;
94     int ret = 0;
95     struct sockaddr_can canbus_addr_L0; //can总线的地址 同socket编程里面的 sockaddr结构体 用来设置can外设的信息
96     struct ifreq ifreq_L0; //接口请求结构体
97     stSocket_L0 = socket(PF_CAN, SOCK_RAW, CAN_RAW); //创建套接字
98     if(stSocket_L0<0)
99     {
100         QMessageBox::information(NULL, QString::fromLocal8Bit("错误"), QString::fromLocal8Bit("打开CAN设备失败"));
101         return -1;
102     }
103     strcpy(ifreq_L0.ifr_name, "can0"); //对CAN接口进行初始化,如设置CAN接口名,即当我们用ifconfig命令时显示的名字
104     ret=ioctl(stSocket_L0, SIOCGIFINDEX, &ifreq_L0); //指定 can设备
105     if(ret<0)
106     {
107         QMessageBox::information(NULL, QString::fromUtf8("错误"), QString::fromUtf8("匹配CAN设备错误"));
108         return -1;
109     }
110     /* 设置CAN协议 */
111     canbus_addr_L0.can_family = AF_CAN; //协议类型
112     canbus_addr_L0.can_ifindex = ifreq_L0.ifr_ifindex; //can总线外设的具体索引 类似 ip地址
113     ret=bind(stSocket_L0, (struct sockaddr *)&canbus_addr_L0, sizeof(canbus_addr_L0)); //将套接字和canbus外设进行绑定
114     if(ret<0)
115     {
116         close_socket(stSocket_L0); //关闭套接字
117         QMessageBox::information(NULL, QString::fromUtf8("错误"), QString::fromUtf8("绑定套接字错误"));
118     }
```

分类专栏

0 订阅

14 篇文章

2 篇文章

订阅专栏

```

119         return -1;
120     }
121 }
122 return stSocket_L0;
123
124
125 }
126
127
128 void Can_Control::close_socket(const int sockfd)
129 {
130     if (sockfd != -1)
131     {
132         close(sockfd);
133     }
134 }
135
136 void Can_Control::set_can_filter()//设置滤波
137 {
138
139     const int n = 1;
140     struct can_filter rfilter[n];
141
142     // 过滤规则：当<received_can_id> & mask == can_id & mask时，接收报文，否则过滤掉。
143     // 可以同时添加多条过滤规则
144
145     // 在用来发送CAN的CAN_RAW套接字上不接收任何CAN帧
146     rfilter[0].can_id = 0x00000000;
147     rfilter[0].can_mask = CAN_EFF_MASK;
148     (void)setsockopt(send_socket_fd, SOL_CAN_RAW, CAN_RAW_FILTER, rfilter, n * sizeof(struct can_filter));
149
150     // 在用来接收CAN的CAN_RAW套接字上启用接收过滤规则
151     (void)setsockopt(recv_socket_fd, SOL_CAN_RAW, CAN_RAW_FILTER, NULL, 0);
152 }
153
154
155
156
157 int Can_Control::send_frame( int sockfd, const void* data, const int count)
158 {
159     int ret = write(sockfd, (const char*)data, count);
160     if (ret != count)
161     {
162         QMessageBox::information(NULL, QString::fromUtf8("错误"), QString::fromUtf8("发送错误"));
163         return 1;
164     }
165     return 0;
166 }
167
168
169 int Can_Control::recv_frame( int sockfd, byte* buf, const int count, const int timeout_ms)
170 {
171     struct timeval tv_timeout;
172     tv_timeout.tv_sec = timeout_ms / 1000;
173     tv_timeout.tv_usec = (timeout_ms % 1000) * 1000;
174     fd_set fs_read;
175
176     FD_ZERO(&fs_read);
177     FD_SET(sockfd, &fs_read); //如果fd == -1, FD_SET将在此阻塞
178
179     int ret = select((int)sockfd + 1, &fs_read, NULL, NULL, &tv_timeout);
180     if (ret == 0) // recv 超时
181     {
182         return 0;
183     }
184     if (ret < 0) // select 错误
185     {
186         return ret;
187     }
188
189     ret = read(sockfd, (char*)buf, count);
190
191     //ret = recv(sockfd, (char*)buf, count, 0);
192
193
194     if (ret <= 0)
195     {
196         return -1;
197     }
198     return ret;
199 }
200

```

用Can_Control类管理can的通信。

can_send () 和can_recv () 函数是示例函数，调用可以直接接收发送，用意在于帮助大家了解基于Socket的can通信的流程。

后面对各模块的配置函数，如链接函数负责开启can通信并进行相应配置，本例还有许多可优化的地方，欢迎大家积极参与并分享。

下面是头文件：

```

1  #ifndef CAN_CONTROL_H
2  #define CAN_CONTROL_H
3
4  #include <QString>
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8  #include <unistd.h>
9  #include <net/if.h>
10 #include <sys/ioctl.h>
11 #include <sys/socket.h>
12 #include <linux/can.h>
13 #include <linux/can/raw.h>
14 #include <QDebug>
15
16 #define printf qDebug
17 typedef __u8    byte;
18 static int  send_socket_fd=-1 ;//接受这套接字
19 static int  recv_socket_fd=-1 ;
20 class Can_Control
21 {
22 public:
23     Can_Control();
24     int can_send();
25     int can_recv();
26     int socket_connect(const char*m_port);
27     void close_socket(const int sockfd);
28     void set_can_filter();//设置滤波
29     int send_frame(int sockfd, const void* data, const int count);
30     int recv_frame( int sockfd, byte* buf, const int count, const int timeout_ms);
31     //struct canfd_frame fs_nframe; //要发送的buffer
32
33     //struct canfd_frame Rr_nframe; //要接收的buffer
34     //string m_candevic; //can设备号，默认can0
35     int stSocket;
36
37
38
39 };
40
41 #endif // CAN_CONTROL_H

```

用法实例：

```
1 void MainWindow::on_toolButton_4_clicked()
2 {
3     // ErrorMessageDlg *errorDlg=new ErrorMessageDlg();
4     // errorDlg->show();
5     struct can_frame frame[2] = {{0}}; //要发送的buffer
6
7     frame[0].can_id = 0x11;
8     frame[0]. can_dlc = 8;
9     frame[0].data[0] = 0x03;
10    frame[0].data[1] = 0x1a;
11    frame[1].can_id = 0x22;
12    frame[1]. can_dlc = 1;
13    frame[1].data[0] = 0x22;
14    //const byte* data, const int count
15    int test;
16    // __u8 send_frame_data[8] = {0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
17
18    // memcpy(frame[0].data, send_frame_data, 8);
19    test=can0.socket_connect("can0");
20    can0.set_can_filter();
21    can0.send_frame(test,&frame[0],sizeof(frame[0]));//
22 }
23
24 void MainWindow::on_toolButton_5_clicked()
25 {
26     //Operation_records_Dlg* recordDlg=new Operation_records_Dlg();
27     // recordDlg->show();
28     struct can_frame recvframe;
29
30     byte *prevframe = (byte *)&recvframe;
31
32     const int can_frame_len = sizeof(can_frame);
33     int test;
34     int ret;
35     //struct can_frame frame[50];
36     test=can0.socket_connect("can0");
37     memset(prevframe, 0x00, can_frame_len);
38     ret=can0.recv_frame(test,prevframe,can_frame_len,5*1000);
39     if(ret>0)
40     {
41         QString buffer;
42
43         QByteArray temp((const char*)(recvframe.data),8);
44
45         buffer = temp.toHex();
46         QMessageBox::information(NULL, QString::fromUtf8("错误"), buffer);
47     }
48 }
```

使用按钮函数触发发送没有毛病，但是在接收时，需将cantest的发送定为持续的，500ms发一次，发送50次。

好像只能一次发送一包数据，如果有谁知道如何实现一次发送多包数据的还望告知，感谢！！

Q CAN、串口、网络测试程序

使用Qt进行开发的多线程实例，包含CAN总线通讯，串口通讯，网络UDP通讯

03-21

linux下socket can 编程详解

1、can总线介绍 2、CAN工作原理 3、CAN总线工作特点 4 can总线协议 5、CAN总线报文结构 6、总线配置

09-08



优质评论可以帮助作者获得更高权重

评论



river_of_gold: 请问SocketCan接收数据时会不会发出类似于TCPSocket的readyRead信号,我想通过这种方式实现数据实时接收 1年前 回复 ···

2



/哈哈: 请问用Qt写canopen通信协议的话,需要移植canfestival源码吗 1年前 回复 ···

1

登录 查看 10 条热评

QT 环境下开发socketCan接口程序 预腾的博客

7-13

#include <linux/can/raw.h> #define PF_CAN 29 定义相关变量 struct sockadd_r can addr; struct lfreq lfr; struct can_frame frame; int canfd; 接口初始化 C...

Linux系统QT编写简单C++的C/S (socket连接) .cxd131452...

8-12

7万 在Linux系统QT编写简单C++的C/S (socket连接) Client.cpp #include<stdio.h> #include<iostream.h> #include<string.h> #include<errno.h> #include<sys/t...

基于Qt的Can通信代码

总排名

访问

等级

05-26

在基于Qt的平台上实现同CAN总线的通信，使用Qt加载动态库来实现CAN通信。周立功为CAN通信提供了动态库：官方提供了很多相关动态库和lib等。其...

基于linux qt fd socket,Qt Socket简单通信

获赞

评论

weixin_1620的博客

09

时间：2013年03月12日目录最近要用到Qt的Socket部分，网上关于这部分资料都比较复杂，我在这总结一下，把Socket的主要部分提取出来，实现...

Linux下的socket编程-基于Qt的客户端 步步天明的专栏

7-17

我们在学习Linux下的socket编程-基于Qt的客户端2011年3月20日 由 edsionte留言 » 上文中对面向连接的C/S模型中的服务器进行了简单描述.本文将说明如何编写一...

qt中socket通信流程图_linux网络通信(TCP/IP)Rice lin...

7-31

流式socket(SOCK_STREAM)流式直接字提供可靠的、面向连接的通信流.它使用TCP协议.从而保证了数据传输的正确性和顺序性。数据报socket(SOCK...

Linux内核Socket CAN中文文档

yuanlulu的博客

25+

自己在年假中空闲之余翻译的内核中Socket CAN的文档，原文地址在：http://lxr.linux.no/linux+v2.6.34/Documentation/networking/can.txt 但是这篇文档...

基于QT开发CAN总线上位机

10-14

QT开发的CAN总线上位机，应该有一些能参考的东西，不贵

Linux内核Socket CAN中文文档_预腾的博客

7-16

socketcan子系统是在Linux下CAN协议(Controller Area Network)实现的一种实现方法。CAN是一种在世界范围内广泛用于自动控制、嵌入式设备和汽车...

QT - 创建TCP Socket通信_bailang_zhizun的博客_qt socket

9-7

最近在学习QT,了解到QT可以进行Socket网络通信.进行学习,并建立一个简单的聊天DEMO.为了测试是否能与VS2012下的程序进行通信.在VS2012下...

qt5 显示调用CAN卡第三方库并调用函数

intelligence123的博客

7782

Qt调用第三方库一般有两种方式，显示和隐式。这里主要介绍显示调用CAN卡第三方库，还介绍一般隐式调用方法。

qt can通信_使用信号槽通信

weixin_39768695的博客

697

我们都知通信信号槽.但很多同学不知道怎么用它在Qt里实现类与类通信.首先要搞清楚:Qt类与类之间通信有2种情况:(1)两个类之间没有父子关系;(2)两个类...

Linux环境下基于Socket的网络通信

03-04

随着Linux 操作系统的不断推广，Linux 环境下的Socket 开发和研究已成为人们关注的热点。Socket 既适用于同一台计算机上的进程间通信，也适用于网...

基于QT CAN通信 网络通信的虚拟仪表

01-24

通过CAN通信、网络通信控制虚拟仪表，在linux平台运行

基于Socket的局域网网络通信

05-04

基于Socket的局域网网络通信 基于Socket的局域网网络通信 基于Socket的局域网网络通信 基于Socket的局域网网络通信 基于Socket的局域网网络通信 基于Socket...

Linux下的网络通信程序Socket c 源码

05-07

Linux下的网络通信程序.请把SERVER跟CLIENT分别放在不同的两台机上运行测试。

CAN网络通信

JinZTa123的博客

322

汽车电子C A N网络通信的TP应用

Linux+QT+SocketCAN：使用信号槽机制实现数据收发

花洛兮...

1015

Linux+QT+SocketCAN：使用信号槽机制实现数据收发 最近在考虑采用面向对象的方式重新搭建机器人的主控制程序框架，虽然之前的框架也是有这样的思想...

基于Socket的局域网网络通信软件开发

03-09

采用Socket套接字通信，使用MFC编程技术，完成局域网内多机通信的功能。将本地机和目标机，操作界面统一起来，集成到一个界面中。这种实现方案不...

android通过jni的形式open video显示视频，C语言

12-08

android通过jni的形式open video显示视频，C语言，需要 给/dev/video0 加权限， chmod 777 -R /dev/video0。不通过camera类 流程，直接用C语言调...

Q-CAN数据传输-模板

DreamLife

1万+

这个模板是在上次的基础上添加CAN模块来写的，上图 主界面，有三大部分组成，最中间的是按钮，左边的设备信息和右边的CAN数据信息 设置...



liuwinner

码龄3年 暂无认证

29

原创

1348

积分



私信

关注

搜博主文章



热门文章

C#使用CSkin界面库开发精美界面 18680

Linux自启动脚本及解决rc.local不执行脚本问题 9109

出现CString, BYTE, DWORD等未定义的标志符的错误 8335

Linux周立功CAN驱动安装指导 7299

解决win10系统点击飞行模式后找不到WiFi连接问题 6871

最新评论

Linux周立功CAN驱动安装指导

holly_huang: modprobe: FATAL: Module sjat1000 not found in directory /lib/module@...

出现CString, BYTE, DWORD等未定义... 听声是你: 有用，感谢

linux下QT基于socket的can网络通信 哈哈哈哈哈哈哈哈哈哈哈哈: 使用QProcess类

破解linux的QT Creator黑屏后闪退问题方法 走上不归路: 666 可以使用了

C#使用CSkin界面库开发精美界面 zwb_578209160: Mark，感谢博主分享

您愿意向朋友推荐“博客详情页”吗？



强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

使用simulink搭建CRC8校验算法



聊一聊汽车制动系统
Simulink连线脚本

2020年 4篇
2019年 27篇
2018年 7篇

配置USBCAN通信

周立功为CAN通信提供了动态库：官方提供了很多相关动态库和lib等，如图，其中kerneldlls里还有很多动态库，还有一个配置文件。其实这么多的文件，...

weixin_30576859的博客 3153

©2020 CSDN 皮肤主题: 大白 设计师: CSDN官方博客 返回首页

liuwinner 关注

5 10 35 专栏目录