轻舟机器人 ROS 端 socket 通信实现示例

AI 航团队

一、什么是 socket

socket 的原意是“插座”，在计算机通信领域，socket 被翻译为“套接字”，它是计算机之间进行通信的一种约定或一种方式。通过 socket 这种约定，一台计算机可以接收其他计算机的数据，也可以向其他计算机发送数据。

socket 的典型应用就是 Web 服务器和浏览器：浏览器获取用户输入的 URL，向服务器发起请求，服务器分析接收到的 URL，将对应的网页内容返回给浏览器，浏览器再经过解析和渲染，就将文字、图片、视频等元素呈现给用户。

学习 socket，也就是学习计算机之间如何通信，并编写出实用的程序。

二、套接字有哪些类型？socket 有哪些类型？

1. 流格式套接字（SOCK_STREAM）

流格式套接字（Stream Sockets）也叫“面向连接的套接字”，在代码中使用 SOCK_STREAM 表示。SOCK_STREAM 是一种可靠的、双向的通信数据流，数据可以准确无误地到达另一台计算机，如果损坏或丢失，可以重新发送。

SOCK_STREAM 有以下几个特征：

（1）数据在传输过程中不会消失；

（2）数据是按照顺序传输的；

（3）数据的发送和接收不是同步的（有的教程也称“不存在数据边界”）。

可以将 SOCK_STREAM 比喻成一条传送带，只要传送带本身没有问题（不会断网），就能保证数据不丢失；同时，较晚传送的数据不会先到达，较早传送的数据不会晚到达，这就保证了数据是按照顺序传递的。

为什么流格式套接字可以达到高质量的数据传输呢？这是因为它使用了 TCP 协议（The Transmission Control Protocol，传输控制协议），TCP 协议会控制你的数据按照顺序到达并且没有错误。

流格式套接字的内部有一个缓冲区（也就是字符数组），通过 socket 传输的数据将保存到这个缓冲区。接收端在收到数据后并不一定立即读取，只要数据不超过缓冲区的容量，接收端有可能在缓冲区被填满以后一次性地读取，也可能分成好几次读取。

2. 数据报格式套接字（SOCK_DGRAM）

数据报格式套接字（Datagram Sockets）也叫“无连接的套接字”，在代码中使用 SOCK_DGRAM 表示。

计算机只管传输数据，不作数据校验，如果数据在传输中损坏，或者没有到达另一台计算机，是没有办法补救的。也就是说，数据错了就错了，无法重传。

因为数据报套接字所做的校验工作少，所以在传输效率方面比流格式套接字要高，数据的发送和接收是同步的。

数据报套接字也使用 IP 协议作路由，但是它不使用 TCP 协议，而是使用 UDP 协议（User Datagram Protocol，用户数据报协议）。

通常在视频和音频传输中使用 SOCK_DGRAM 来传输数据，因为首先要保证通信的效率，

尽量减小延迟，而数据的正确性是次要的，即使丢失很小的一部分数据，视频和音频也可以正常解析，最多出现噪点或杂音，不会对通信质量有实质的影响。

三、Linux 下的 socket 演示程序

服务器端代码 server.cpp：

```
01. #include <stdio.h>
02. #include <string.h>
03. #include <stdlib.h>
04. #include <unistd.h>
05. #include <arpa/inet.h>
06. #include <sys/socket.h>
07. #include <netinet/in.h>
08.
09. int main(){
10.     //创建套接字
11.     int serv_sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
12.
13.     //将套接字和IP、端口绑定
14.     struct sockaddr_in serv_addr;
15.     memset(&serv_addr, 0, sizeof(serv_addr));  //每个字节都用0填充
16.     serv_addr.sin_family = AF_INET;  //使用IPv4地址
17.     serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");  //具体的IP地址
18.     serv_addr.sin_port = htons(1234);  //端口
19.     bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
20.
21.     //进入监听状态，等待用户发起请求
22.     listen(serv_sock, 20);
23.
24.     //接收客户端请求
25.     struct sockaddr_in clnt_addr;
26.     socklen_t clnt_addr_size = sizeof(clnt_addr);
27.     int clnt_sock = accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_size);
28.
29.     //向客户端发送数据
30.     char str[] = "http://c.biancheng.net/socket/";
31.     write(clnt_sock, str, sizeof(str));
32.
33.     //关闭套接字
34.     close(clnt_sock);
35.     close(serv_sock);
36.
37.     return 0;
38. }
```

第 11 行通过 socket() 函数创建了一个套接字，参数 AF_INET 表示使用 IPv4 地址，SOCK_STREAM 表示使用面向连接的套接字，IPPROTO_TCP 表示使用 TCP 协议。在 Linux 中，socket 也是一种文件，有文件描述符，可以使用 write() / read() 函数进行 I/O 操作。

第 19 行通过 bind() 函数将套接字 serv_sock 与特定的 IP 地址和端口绑定，IP 地址和端口都保存在 sockaddr_in 结构体中。

socket() 函数确定了套接字的各种属性，bind() 函数让套接字与特定的 IP 地址和端口对

应起来，这样客户端才能连接到该套接字。

第 22 行让套接字处于被动监听状态。所谓被动监听，是指套接字一直处于“睡眠”中，直到客户端发起请求才会被“唤醒”。

第 27 行的 accept() 函数用来接收客户端的请求。程序一旦执行到 accept() 就会被阻塞（暂停运行），直到客户端发起请求。

第 31 行的 write() 函数用来向套接字文件中写入数据，也就是向客户端发送数据。

和普通文件一样，socket 在使用完毕后也要用 close() 关闭。

客户端代码 client.cpp：

```
01. #include <stdio.h>
02. #include <string.h>
03. #include <stdlib.h>
04. #include <unistd.h>
05. #include <arpa/inet.h>
06. #include <sys/socket.h>
07.
08. int main(){
09.     //创建套接字
10.     int sock = socket(AF_INET, SOCK_STREAM, 0);
11.
12.     //向服务器（特定的IP和端口）发起请求
13.     struct sockaddr_in serv_addr;
14.     memset(&serv_addr, 0, sizeof(serv_addr));  //每个字节都用0填充
15.     serv_addr.sin_family = AF_INET;  //使用IPv4地址
16.     serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");  //具体的IP地址
17.     serv_addr.sin_port = htons(1234);  //端口
18.     connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
19.
20.     //读取服务器传回的数据
21.     char buffer[40];
22.     read(sock, buffer, sizeof(buffer)-1);
23.
24.     printf("Message form server: %s\n", buffer);
25.
26.     //关闭套接字
27.     close(sock);
28.
29.     return 0;
30. }
```

第 19 行代码通过 connect() 向服务器发起请求，服务器的 IP 地址和端口号保存在 sockaddr_in 结构体中。直到服务器传回数据后，connect() 才运行结束。

第 23 行代码通过 read() 从套接字文件中读取数据。

更多详细资料，参考链接：http://c.biancheng.net/view/2123.html

四、轻舟机器人 ROS 端 socket 通信实现示例

根据以上对 socket 通信的简单理解，以及教程 365 的通信协议，便可以轻松的实现上位机调度软件和轻舟机器人 ROS 端的 socket 通信。

示例代码如下，只展示了客户端 ROS 部分的收发数据代码：

（1）ROS 端从 MFC 接收数据并解析

```
void app::RecvThreadFromMfc(){
    char recvbuf[9] = {0,};
    char recvInfobuf[10256] = {0,};
    recv_frame_t* pheader = (recv_frame_t*)recvbuf;
    unsigned int frame_len = 0;
    while(1){
        usleep(50000);
        if(clientIsConnect == 1){
            volatile int size = 0;
            volatile int intoret = 0;
            volatile int ret = 0;
            struct timeval tv;
            fd_set readfds;
            FD_ZERO(&readfds);
            FD_SET(clientfd, &readfds);
            tv.tv_sec = 0;
            tv.tv_usec = 50000;
            ret = select(clientfd+1, &readfds, NULL, NULL, &tv);
            if(ret == 0){
                continue;
            }
            else if(ret < 0 && errno != EINTR) {
                clientIsConnect = 0;//I think socket is disconnect
                close(clientfd);
                continue;
            }else if((ret > 0)&&(errno != EINTR)){
                if(FD_ISSET(clientfd,&readfds)){
                    size = recvn(recvbuf, 9);
                    if(size < 0){
                        printf("qingzhou_cloud recv failed....\n");
                        continue;
                    }
                    if(size > 0){
                        heartdisconnectCommand = 0;
                        if(bdebug == 1){
                            printf("recv size is %d\n",size);
                            for(int i = 0;i < size;i++){
                                printf(" %02x",recvbuf[i]);
                            }
                            printf("\n");
                        }
                        if((recvbuf[0] != 0x02)||(recvbuf[1] != 0x20)||(recvbuf[2] != 0x02)||(recvbuf[3] != 0x20)){
                            printf("qingzhou_cloud recv header error....\n");
                            continue;
                        }
                        frame_len = (recvbuf[4] & 0xff) + ((recvbuf[5] & 0xff)<<8) + ((recvbuf[6] & 0xff)<<16) + ((recvbuf[7] & 0xff)<<24);

                        intoret = recvn(recvInfobuf, frame_len-1);
                        if(bdebug == 1){
                            printf("recv len is %d,command is %02x,para[0] is %d\n",pheader->len,pheader->command,recvInfobuf[0]);
                        }
                        if(intoret  < 0){
                            printf("qingzhou_cloud recv Info failed....\n");
                            continue;
                        }else {
                            heartFlag = 0;
                        //    recvflag = 0;
                            if(bdebug == 1){
                                printf("recv info size is %d\n",intoret);
                                for(int i = 0;i < intoret;i++){
                                    printf(" %02x",recvInfobuf[i]);
                                }
                                printf("\n");
                            }
                        }
                        switch(pheader->command){
                        case 0x10:{//start navigation
                            recvflag = 1;
                            qingzhou_cloud::startstopCommand ssCommand;
                            ssCommand.startstopcommand = 0x01;
                            pub_start_stop_command.publish(ssCommand);
                            heartFlag = 0;
                            //printf("recv start navigation command %d\n",recvInfobuf[0]);
                            printf("recv start navigation command %d\n",ssCommand.startstopcommand);
                            break;
                        }
                        case 0x20:{//stop
                            recvflag = 2;
                            qingzhou_cloud::startstopCommand ssCommand;
                            ssCommand.startstopcommand = 0x02;
                            pub_start_stop_command.publish(ssCommand);
                            heartFlag = 0;
                            //printf("Recv stop navigation command %d\n",recvInfobuf[0]);
                            printf("Recv stop navigation command %d\n",ssCommand.startstopcommand);
                            break;
                        }
                        case 0x30:{//first stop point
                            float x = 0,y = 0;
                            memcpy(&x,recvInfobuf,4);
                            memcpy(&y,recvInfobuf+4,4);
                            printf("Recv first stop point x = %.2f,y =  %.2f\n",x,y);
                            recvflag = 7;
                            heartFlag = 0;
                            qingzhou_cloud::stoppoint spoint;
                            spoint.X = x;
                            spoint.Y = y;
                            pub_stop_point.publish(spoint);
                            break;
                        }
```

```
                    }
                    case 0x40:{//second stop point
                        //   printf("Recv second stop point\n");
                        float x = 0,y = 0;
                        memcpy(&x,recvInfobuf,4);
                        memcpy(&y,recvInfobuf+4,4);
                        printf("Recv second stop point x = %.2f,y =  %.2f\n",x,y);
                        recvflag = 3;
                        heartFlag = 0;
                        qingzhou_cloud::stoppoint spoint;
                        spoint.X = x;
                        spoint.Y = y;
                        pub_stop_point.publish(spoint);
                        if(recvInfobuf[0] == 1){
                            //do some work
                        }
                        break;
                    }
                    case 0x50:{//third stop command
                        recvflag = 6;
                        heartFlag = 0;
                        float x = 0,y = 0;
                        memcpy(&x,recvInfobuf,4);
                        memcpy(&y,recvInfobuf+4,4);
                        printf("Recv third stop point x = %.2f,y =  %.2f\n",x,y);

                        qingzhou_cloud::stoppoint spoint;
                        spoint.X = x;
                        spoint.Y = y;
                        pub_stop_point.publish(spoint);
                        break;
                    }
                    case 0x77:{//heart recv
                        heartFlag = 0;
                        break;
                    }
                    default:break;
                    }
                }
            }
        }
    }
}
```

（2）ROS 端发送数据到 MFC

```
void app::dataProcKernelNet(int carID){
    float x = current_location.x;
    float y = current_location.y;
    float theta = current_location.heading;
    float currentAngleSpeed = sOdom.twist.twist.angular.z;
    float currentLinerSpeed = sOdom.twist.twist.linear.x;
    char navstatus = 0;
    char carstatus = carstatusMsg.data;

    struct info{
    int carID1;float x1;float y1;float theta1;float angleSpeed;float linerSpeed;char navstatus1;char carstatus1;
    };

    struct info info1{
        carID,
        x,
        y,
        theta,
        currentAngleSpeed,
        currentLinerSpeed,
        navstatus,
        carstatus,
    };

    int ret = send(clientfd, (char *)&info1, sizeof(info1), 0);
    if(ret < 0)
    printf("qingzhou_cloud-->send to MFC server error %d\n",errno);
}
```

1.同学们在使用过程中，如果发现内容有疏漏或者不严谨的地方，请与我们联系，将会有轻舟积分送上！QQ：270220858

2.内容如有雷同，侵删！

2021 年 4 月