socket编程

参考： https://blog.csdn.net/liranke/article/details/6079393/

这个代码在linux系统下可直接使用，同样也适于xilinx zynq petalinx下使用。

1. 服务端代码

```
#include <stdio.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <string.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <arpa/inet.h>

#define MAX_READ_LINE 1024

int main(void) {
    int recv_len = -1;
    int conn_fd = -1;
    int ret = -1;

    //ip端口，必须和client的一样，才能正常建立链接
    int server_ip_port = 996;

    //用于存储接收到的数据
    char buff[MAX_READ_LINE];

    //初始化sockaddr_in结构体
    struct sockaddr_in t_sockaddr;
    memset(&t_sockaddr, 0, sizeof(t_sockaddr));
    t_sockaddr.sin_family = AF_INET;
    t_sockaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    t_sockaddr.sin_port = htons(server_ip_port);

    //创建server端的socket套接字
    int listen_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_fd < 0) {
        fprintf(stderr, "socket error %s errno: %d\n", strerror(errno),
errno);
    }

    //绑定
    ret = bind(listen_fd,(struct sockaddr *)
&t_sockaddr,sizeof(t_sockaddr));
```

```
    if (ret < 0) {
        fprintf(stderr, "bind socket error %s errno: %d\n", strerror(errno),
errno);
    }

    //监听
    ret = listen(listen_fd, 1024);
    if (ret < 0) {
        fprintf(stderr, "listen error %s errno: %d\n", strerror(errno),
errno);
    }

    //接收来自客户端的链接请求
    for(;;) {
        conn_fd = accept(listen_fd, (struct sockaddr*)NULL, NULL);
        if(conn_fd < 0) {
            fprintf(stderr, "accpet socket error: %s errno :%d\n",
strerror(errno), errno);
            continue;
        }

        //读取数据到buff中
        recv_len = recv(conn_fd, buff, MAX_READ_LINE, 0);
        if (recv_len < 0) {
            fprintf(stderr, "recv error %s errno: %d\n", strerror(errno),
errno);
            continue;
        }

        buff[recv_len] = '\0';
        //将接收到的数据标准输出
        fprintf(stdout, "recv msg from client: %s\n", buff);
        close(conn_fd);
        conn_fd = -1;
    }

    //关闭套接字
    close(listen_fd);
    listen_fd = -1;

    return 0;
}
```

2. 客户端代码

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```c
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(void) {
    char *server_ip_addr = "127.0.0.1";
    int server_ip_port = 996;

    //要发送给server的数据
    char *send_message = "hello";

    //创建client端的socket套接口
    int socket_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_fd < 0) {
        fprintf(stderr, "socket error %s errno: %d\n", strerror(errno),
errno);
    }

    //初始化sockaddr_in结构体
    struct sockaddr_in t_sockaddr;
    memset(&t_sockaddr, 0, sizeof(struct sockaddr_in));
    t_sockaddr.sin_family = AF_INET;
    t_sockaddr.sin_port = htons(server_ip_port);
    inet_pton(AF_INET, server_ip_addr, &t_sockaddr.sin_addr);

    //连接
    if((connect(socket_fd, (struct sockaddr*)&t_sockaddr, sizeof(struct
sockaddr))) < 0 ) {
        fprintf(stderr, "connect error %s errno: %d\n", strerror(errno),
errno);
        return 0;
    }

    //向server发送数据
    if((send(socket_fd, send_message, strlen(send_message), 0)) < 0) {
        fprintf(stderr, "send message error: %s errno : %d",
strerror(errno), errno);
        return 0;
    }

    //关闭套接字
    close(socket_fd);
    socket_fd = -1;

    return 0;
}
```