网络编程入门篇

利用 socket 实现 TCP 服务器

目录

- 基础知识
- 具体示例
- 示例代码讲解

RT-Thread

基础知识

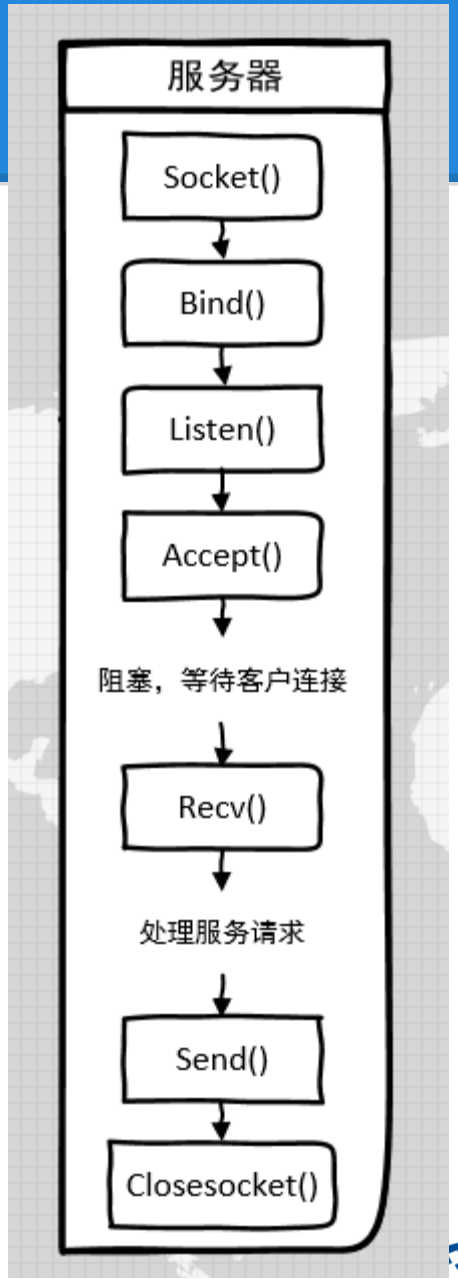
基础知识

- socket 编程一般采用客户端-服务器模式（即由客户进程向服务器进程发出请求，服务器进程执行请求的任务并将执行结果返回给客户进程的模式。）
- 今天我们要讲的就是如何利用 socket 编程实现基于 TCP 协议通信的服务器。
- 首先我们先向大家展示 socket 编程的流程，然后给出具体的示例，最后讲解一下示例代码。

基础知识

TCP 服务器的 socket 编程流程，如右图所示：

- 创建 socket
- 将创建的 socket 绑定到一个 IP 地址和端口号上
- 设置 socket 为监听模式
- 接收请求并返回 socket
- 与客户端进行通信
- 关闭 socket



具体示例

工程配置

- RT-Thread samples 软件包中已有一份该示例代码 tcpserver.c，可以通过 env 配置将示例代码加入到项目中。
- 按照下面的路径获取 tcp server 的示例代码：

```
RT-Thread online packages  --->
    miscellaneous packages  --->
        samples: RT-Thread kernel and components samples  --->
            network sample options  --->
                [*] [network] tcp server
```

- 保存并更新软件包 pkgs --update
- 编译工程 scons
- 然后运行qemu

RT-Thread

查看 ip 地址

- 在 qemu 运行起来之后，输入命令 ifconfig 查看系统的 IP 地址。

```
msh />ifconfig
network interface: e0 (Default)
MTU: 1500
MAC: 52 54 00 11 22 33
FLAGS: UP LINK_UP ETHARP BROADCAST
ip address: 192.168.137.117
gw address: 192.168.137.1
net mask  : 255.255.255.0

ipv6 link-local: FE80::5054:FF:FE11:2233 state:30 VALID
ipv6[1] address: :: state:00 INVALID
ipv6[2] address: :: state:00 INVALID

dns server #0: 192.168.137.1
dns server #1: 0.0.0.0
msh />
```

RT-Thread

运行示例代码

- 在 msh 命令行下输入命令 tcpserv 即可让示例代码运行。

```
msh /> tcpserv
TCPServer Waiting for client on port 5000...
```

- 示例代码会在 qemu 上启动一个 TCP 服务器，端口号是 5000。

开启TCP客户端

- 这样我们可以在电脑上搭建一个 TCP 客户端来连接已经处于运行状态的 TCP 服务器。这里以网络调试助手 IPOP 为例。

- 1. 如果是 QEMU 平台的话要先绑定网卡和 IP 地址。
  - 选择和开发板处于同一网段的网卡（这里就是TAP网卡了）
  - 绑定一个和开发板处于同一网段的ip地址
    - 例如开发板ip为192.168.137.117，绑定的IP地址可以是192.168.137.35
  - 已经绑定过的话就不需要绑定了。

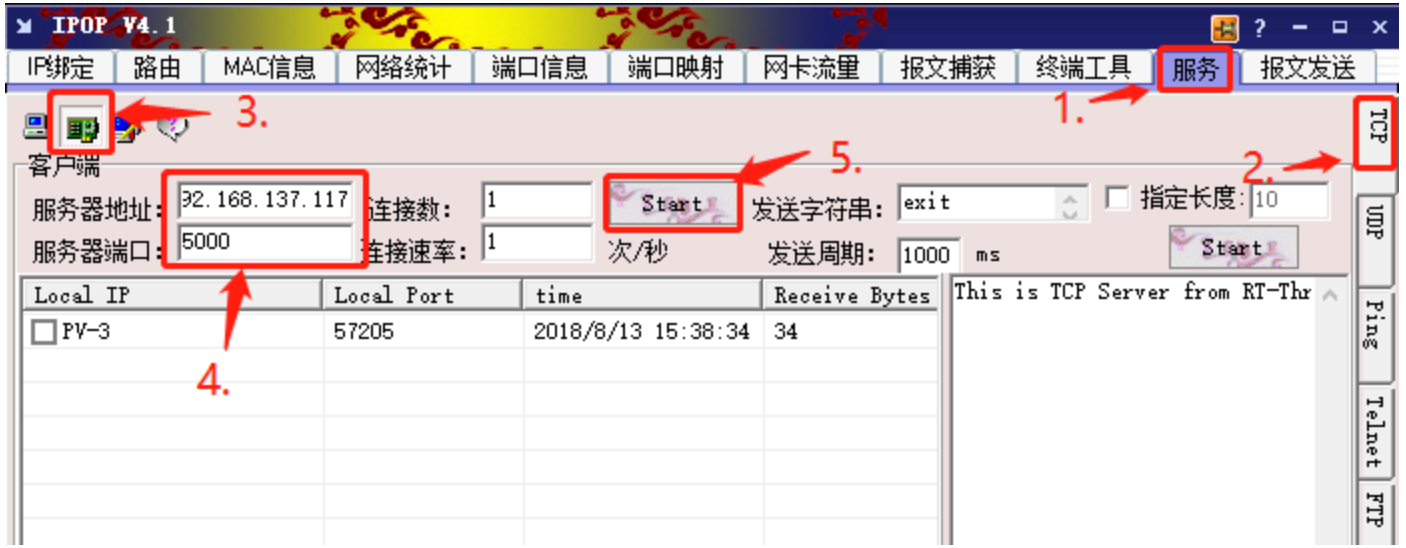
开启TCP客户端

- 这样我们可以 行状态的 TCP 服务器。这里
- 1. 如果是 QE
  - 选择和开发
  - 绑定一个和
    - 例如开发
  - 已经绑定过

RT-Thread

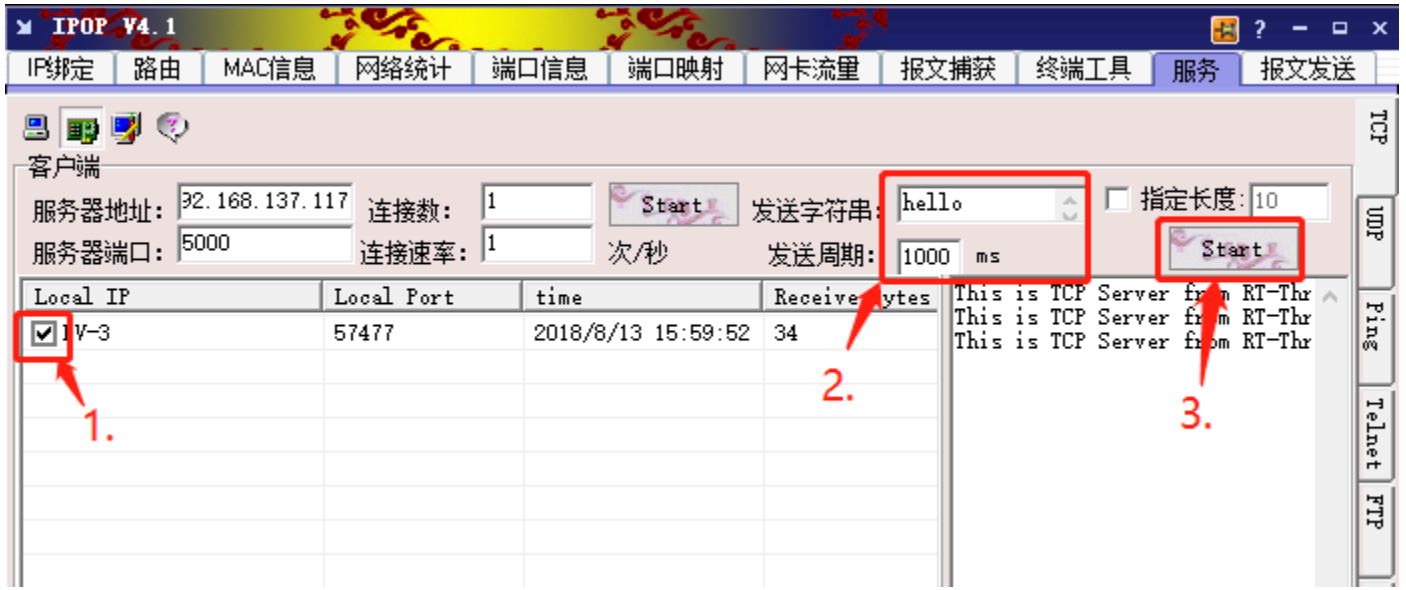
开启TCP客户端

- 然后是开启TCP客户端连接服务器



RT-Thread

从客户端发消息给服务器



发送字符 'q' 断开连接，发送 “exit” 关闭服务器

```
msh />tcpserv

TCPServer Waiting for client on port 5000...
I got a connection from (192.168.137.1 , 57205)
I got a connection from (192.168.137.1 , 57253)
msh />
```

RT-Thread

注意事项

- 电脑需要关闭防火墙

示例代码讲解

示例代码讲解

```
/*
* 程序清单：tcp 服务端
 *
 * 这是一个 tcp 服务端的例程
 * 导出 tcpserv 命令到控制终端
 * 命令调用格式：tcpserv
 * 无参数
 * 程序功能：作为一个服务端，接收并显示客户端发来的数据 ， 接收到 exit 退出程序
*/
#include <rtthread.h>
#include <sys/socket.h> /* 使用BSD socket， 需要包含socket.h头文件 */
#include <netdb.h>
#include <string.h>
#define BUFSZ      (1024)

static const char send_data[] = "This is TCP Server from RT-Thread."; /* 发送用到的数据 */
static void tcpserv(int argc, char **argv)
{
    char *recv_data; /* 用于接收的指针，后面会做一次动态分配以请求可用内存 */
    socklen_t sin_size;
    int sock, connected, bytes_received;
    struct sockaddr_in server_addr, client_addr;
    rt_bool_t stop = RT_FALSE; /* 停止标志 */
    int ret;
```

RT-Thread

示例代码讲解

```
recv_data = rt_malloc(BUFSZ + 1); /* 分配接收用的数据缓冲 */
if (recv_data == RT_NULL)
{
    rt_kprintf("No memory\n");
    return;
}

/* 一个socket在使用前，需要预先创建出来，指定SOCK_STREAM为TCP的socket */
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
    /* 创建失败的错误处理 */
    rt_kprintf("Socket error\n");

    /* 释放已分配的接收缓冲 */
    rt_free(recv_data);
    return;
}

/* 初始化服务端地址 */
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(5000); /* 服务端工作的端口 */
server_addr.sin_addr.s_addr = INADDR_ANY;
rt_memset(&(server_addr.sin_zero), 0, sizeof(server_addr.sin_zero));
```

示例代码讲解

```
/* 绑定socket到服务端地址 */
if (bind(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr)) == -1)
{
    /* 绑定失败 */
    rt_kprintf("Unable to bind\n");

    /* 释放已分配的接收缓冲 */
    rt_free(recv_data);
    return;
}

/* 在socket上进行监听 */
if (listen(sock, 5) == -1)
{
    rt_kprintf("Listen error\n");

    /* release recv buffer */
    rt_free(recv_data);
    return;
}

rt_kprintf("\nTCPServer Waiting for client on port 5000...\n");
while (stop != RT_TRUE)
{
    sin_size = sizeof(struct sockaddr_in);
```

示例代码讲解

```
/* 接受一个客户端连接socket的请求，这个函数调用是阻塞式的 */
connected = accept(sock, (struct sockaddr *)&client_addr, &sin_size);
/* 返回的是连接成功的socket */
if (connected < 0)
{
    rt_kprintf("accept connection failed! errno = %d\n", errno);
    continue;
}

/* 接受返回的client_addr指向了客户端的地址信息 */
rt_kprintf("I got a connection from (%s , %d)\n",
           inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

/* 客户端连接的处理 */
while (1)
{
    /* 发送数据到connected socket */
    ret = send(connected, send_data, strlen(send_data), 0);
    if (ret < 0)
    {
        /* 发送失败，关闭这个连接 */
        closesocket(connected);
        rt_kprintf("\nsend error,close the socket.\r\n");
        break;
    }
```

RT-Thread

示例代码讲解

```
else if (ret == 0)
{
    /* 打印send函数返回值为0的警告信息 */
    rt_kprintf("\n Send warning,send function return 0.\r\n");
}

/* 从connected socket中接收数据，接收buffer是1024大小，但并不一定能够收到1024大小的数据 */
bytes_received = recv(connected, recv_data, BUFSZ, 0);
if (bytes_received < 0)
{
    /* 接收失败，关闭这个connected socket */
    closesocket(connected);
    break;
}
else if (bytes_received == 0)
{
    /* 打印recv函数返回值为0的警告信息 */
    rt_kprintf("\nReceived warning,recv function return 0.\r\n");
    closesocket(connected);
    break;
}

/* 有接收到数据，把末端清零 */
recv_data[bytes_received] = '\0';
if (strcmp(recv_data, "q") == 0 || strcmp(recv_data, "Q") == 0)
```

示例代码讲解

```
        {
            /* 如果是首字母是q或Q，关闭这个连接 */

            closesocket(connected);
            break;
        }
        else if (strcmp(recv_data, "exit") == 0)
        {
            /* 如果接收的是exit，则关闭整个服务端 */
            closesocket(connected);
            stop = RT_TRUE;
            break;
        }
        else
        {
            /* 在控制终端显示收到的数据 */
            rt_kprintf("RECEIVED DATA = %s \n", recv_data);
        }
      }
    }
    /* 退出服务 */
    closesocket(sock);

    /* 释放接收缓冲 */
    rt_free(recv_data);
    return ;
}
MSH_CMD_EXPORT(tcpserv, a tcp server sample);
```