网络编程入门篇

使用 socket 实现 UDP 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

- 和TCP协议一样，UDP 协议也是用于客户端-服务器模式的一种传输协议，如今的很多通信软件都是利用这个协议实现的，如腾讯 QQ 发送消息,视频聊天用的就是 UDP 协议。

- 今天我们要讲的就是如何利用 socket 编程实现基于 UDP 协议通信的服务器。

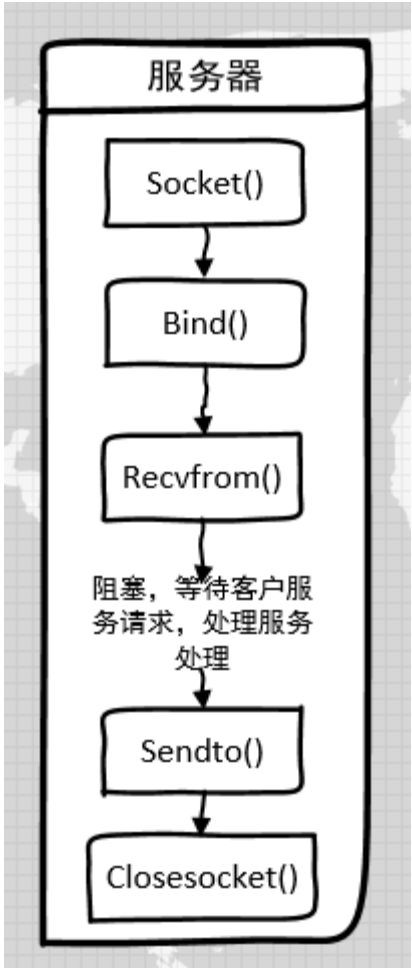
- 与开发 TCP 服务器一样，我们先将 socket 编程的流程列出来，然后给出具体的示例，最后讲解一下示例代码。

基础知识

UDP 服务器的 socket 编程流程

- 创建 socket
- 将创建的 socket 绑定到一个 IP 地址和端口号上
- 等待接收数据报，处理完成后将结果返回到客户端
- 关闭 socket

如右图所示：



具体示例

工程配置

- RT-Thread samples 软件包中已有一份该示例代码 udpserver.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 udp server 的示例代码：

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

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

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qemu运行起来后，在 msh 命令行下输入命令 udpserv 即可让示例代码运行。

```
msh />udpserv
UDPServer Waiting for client on port 5000...
```

- 这样会在 qemu 上启动一个 UDP 服务器，端口号是 5000。

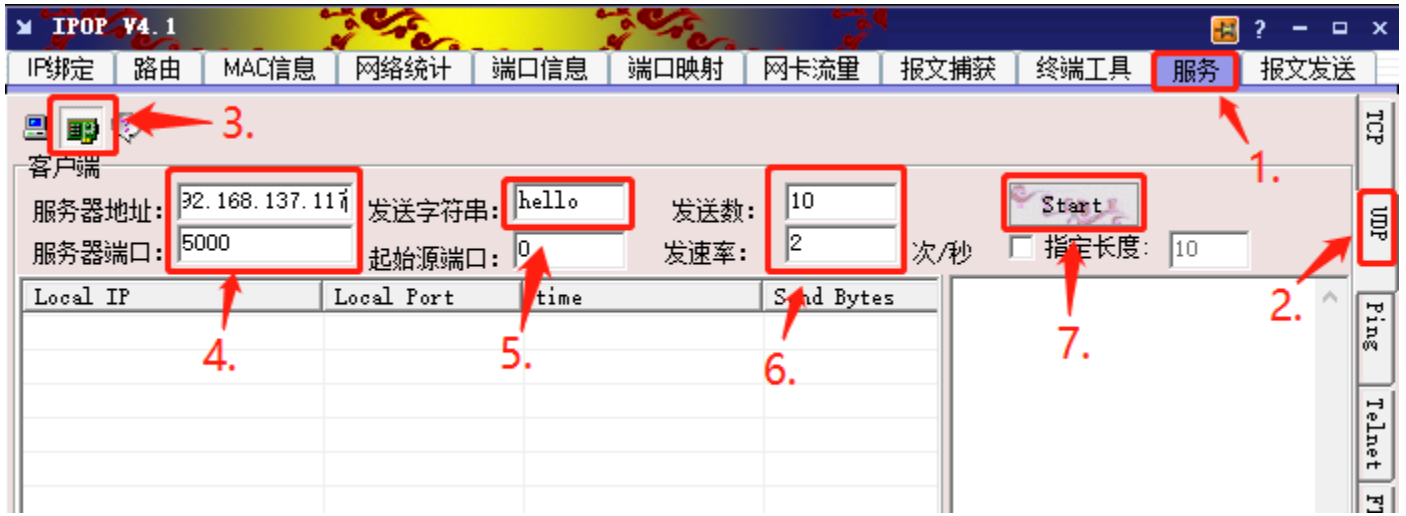
搭建UDP客户端

- 这样我们可以 行状态的 UDP 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

搭建UDP客户端

- 然后是搭建UDP客户端给服务器发送消息



RT-Thread

从客户端发消息给服务器

我们先发送 10 条 hello 信息，然后发送 “exit” 关闭服务器

```
msh />udpserv
UDPServer Waiting for client on port 5000...

(192.168.137.1 , 61820) said : hello
(192.168.137.1 , 61821) said : hello
(192.168.137.1 , 61822) said : hello
(192.168.137.1 , 61823) said : hello
(192.168.137.1 , 61824) said : hello
(192.168.137.1 , 61825) said : hello
(192.168.137.1 , 61826) said : hello
(192.168.137.1 , 61827) said : hello
(192.168.137.1 , 61828) said : hello
(192.168.137.1 , 61829) said : hello
(192.168.137.1 , 61830) said : exitmsh />
```

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：udp 服务端
 *
 * 这是一个 udp 服务端的例程
 * 导出 udpserv 命令到控制终端
 * 命令调用格式：udpserv
 * 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 void udpserv(int argc, char **argv)
{
    int sock;
    int bytes_read;
    char *recv_data;
    socklen_t addr_len;
    struct sockaddr_in server_addr, client_addr;
```

RT-Thread

示例代码讲解

```
/* 分配接收用的数据缓冲 */
    recv_data = rt_malloc(BUFSZ);
    if (recv_data == RT_NULL)
    {
        /* 分配内存失败，返回 */
        rt_kprintf("No memory\n");
        return;
    }

    /* 创建一个socket，类型是SOCK_DGRAM，UDP类型 */
    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    {
        rt_kprintf("Socket error\n");

        /* 释放接收用的数据缓冲 */
        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);
    server_addr.sin_addr.s_addr = INADDR_ANY;
    rt_memset(&(server_addr.sin_zero), 0, sizeof(server_addr.sin_zero));
```

RT-Thread

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)
    {
        /* 绑定地址失败 */
        rt_kprintf("Bind error\n");

        /* 释放接收用的数据缓冲 */
        rt_free(recv_data);
        return;
    }

    addr_len = sizeof(struct sockaddr);
    rt_kprintf("UDPServer Waiting for client on port 5000...\n");

    while (1)
    {
        /* 从sock中收取最大BUFSZ - 1字节数据 */
        bytes_read = recvfrom(sock, recv_data, BUFSZ - 1, 0,
                              (struct sockaddr *)&client_addr, &addr_len);
        /* UDP不同于TCP，它基本不会出现收取的数据失败的情况，除非设置了超时等待 */
        recv_data[bytes_read] = '\0'; /* 把末端清零 */
```

示例代码讲解

```
        /* 输出接收的数据 */
        rt_kprintf("\n(%s , %d) said : ", inet_ntoa(client_addr.sin_addr),
                ntohs(client_addr.sin_port));
        rt_kprintf("%s", recv_data);

        /* 如果接收数据是exit，退出 */
        if (strcmp(recv_data, "exit") == 0)
        {
            closesocket(sock);

            /* 释放接收用的数据缓冲 */
            rt_free(recv_data);
            break;
        }
    }

    return;
}
MSH_CMD_EXPORT(udpserv, a udp server sample);
```

RT-Thread