网络编程入门篇

Select：非阻塞 Socket 编程

目录

- 基础知识
- 具体示例
- 注意事项
- 示例代码讲解

RT-Thread

基础知识

基础知识

- 在 RT-Thread 使用 socket 网络编程时，由于 socket 的 recv 和 send 的实现是阻塞式的，因此当一个任务调用 recv() 函数接收数据时，如果 socket 上并没有接收到数据，这个任务将阻塞在 recv() 函数里。这个时候，这个任务想要处理一些其他事情，就变得不可能了。
- 因此，在要求网络传输的同时，还能处理其他的数据的场景下，就需要用到 select 了，select 能够同时监视多个非阻塞 socket 的多个事件，这对于以上问题的解决有着重要的意义。

基础知识

- 下面我们先介绍一下 select 的 API，然后展示 select 使用时的流程，接着用一个简单的示例程序，给大家具体的展示 select 函数的基本用法，最后讲解一下示例程序。
- 在掌握 select 的基本用法之后，就可以从下面的应用笔记中找到 socket 非阻塞编程的具体方法了。

- 应用笔记：基于多线程的非阻塞 socket 编程

select API 说明

```
int select(int nfds, fd_set* readfds, fd_set* writefds, fd_set* errorfds, struct timeval* timeout);
```

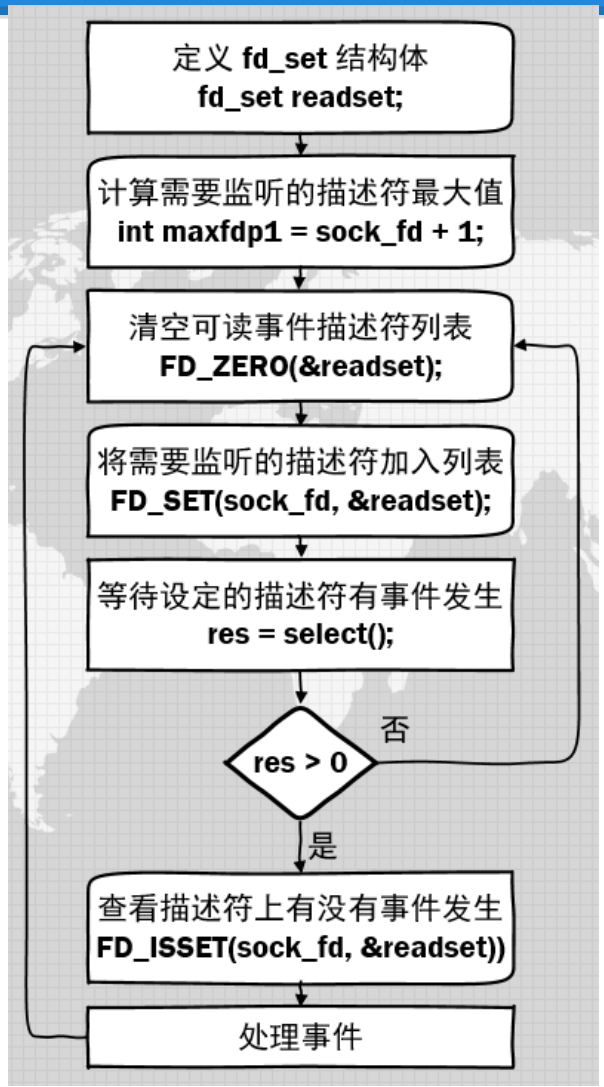
参数	描述
nfds	集合中所有文件描述符的范围，即所有文件描述符的最大值加1
readfds	需要监视读变化的文件描述符集合
writefds	需要监视写变化的文件描述符集合
errorfds	需要监视出现异常的文件描述符集合
timeout	select 的超时时间
返回值	**描述**
正值	监视的文件集合出现可读写事件或异常事件
0	等待超时，没有可读写或异常的事件
负值	select 出现错误

RT-Thread

使用流程

Select 的使用流程如右图所示：

- 定义 fd_set 结构体
- 计算 maxfdp1
- FD_ZERO
- FD_SET
- Select
- FD_ISSET



RT-Thread

具体示例

工程配置

- RT-Thread samples 软件包中已有一份该示例代码 tcpclient_select.c，可以通过 env 配置将示例代码加入到项目中。
- 按照下面的路径即可开启 select 的示例代码

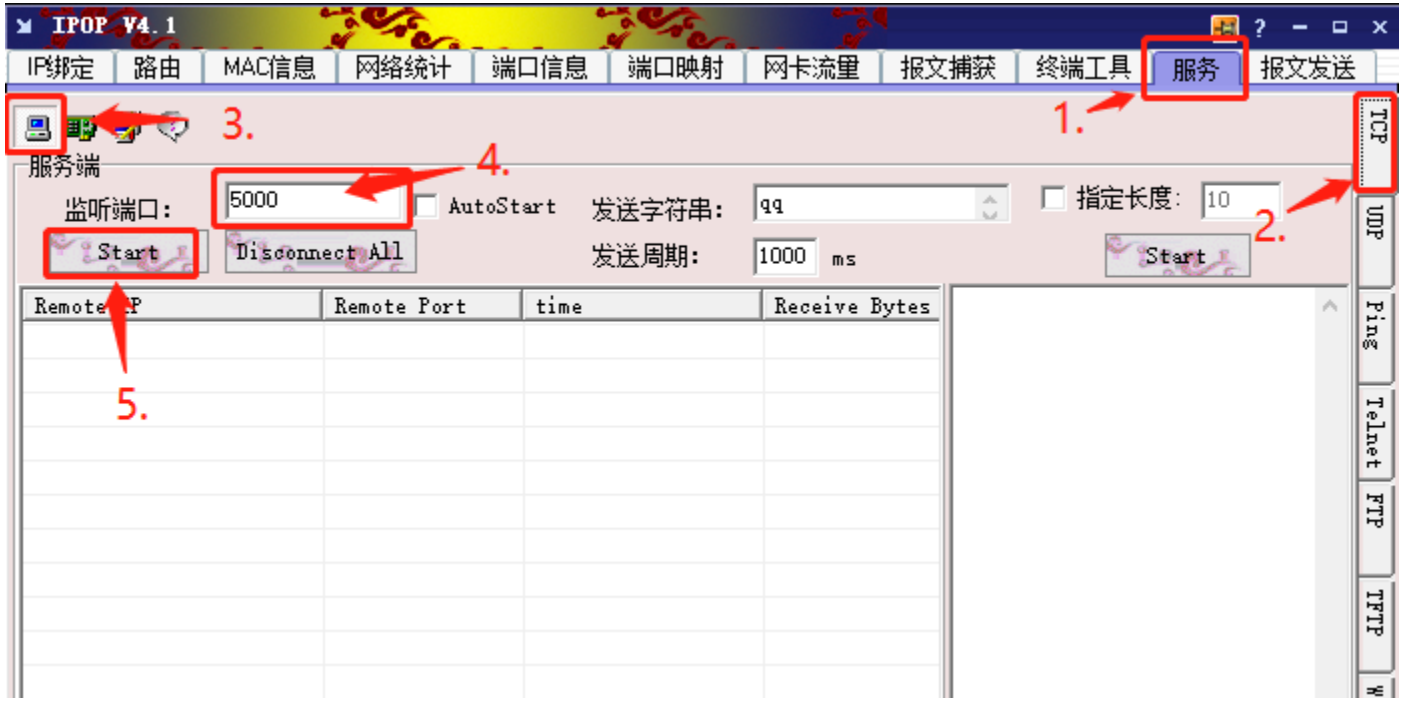
```
RT-Thread online packages  --->
    miscellaneous packages  --->
        samples: RT-Thread kernel and components samples  --->
            [*] a network_samples package for rt-thread  --->
                [*]   [network] tcp client by select api
```

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

RT-Thread

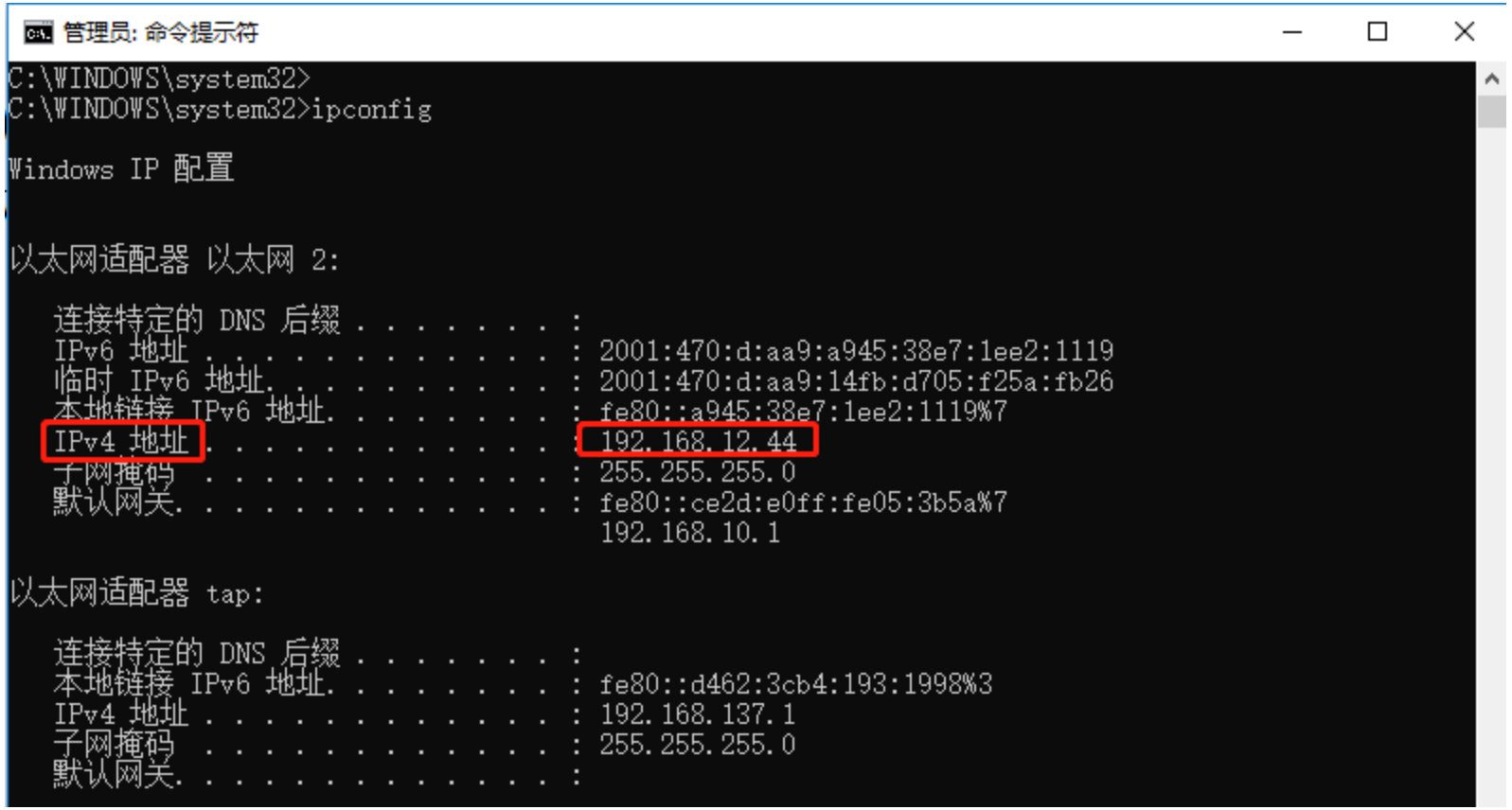
开启TCP服务器

- 在运行示例代码之前需要先在电脑上开启一个 TCP 服务器，这里以网络调试助手 IPOP 为例。



查看本机ip地址

- 在windows系统中打开命令提示符，输入 ipconfig 即可查看本机ip



RT-Thread

运行示例代码

- 在qemu运行起来后，在 msh 命令行下输入下面的命令即可让示例代码运行。

```
msh> tcpclient_select  192.168.12.44 5000
```

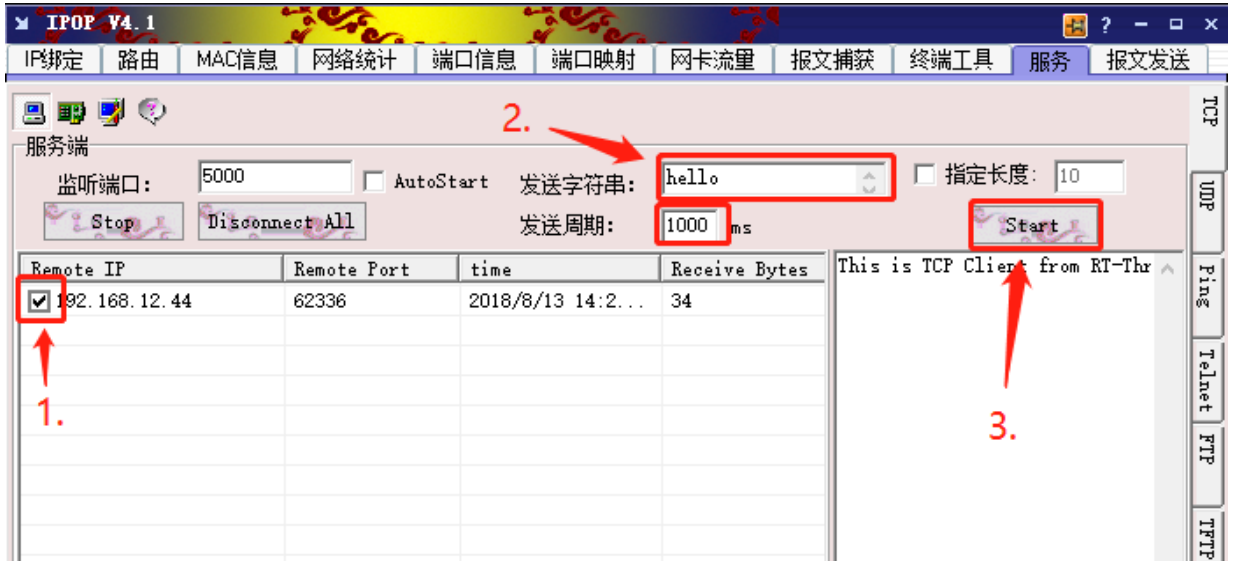
- tcpclient_select 有两个参数 URL PORT
- 其中：
- 　URL 是目标服务器的网址或 IP 地址
- 　PORT 是目标服务器的端口号

```
msh />tcpclient 192.168.12.44 5000

Received data = Welcome to TcpSrv
```

RT-Thread

从服务器发消息给客户端



发送字符 'q' 即可断开连接

```
Received data = hello
Received data = hello
Received data = hello
 got a 'q' or 'Q',close the socket.
msh />
```

RT-Thread

注意事项

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：利用 select 实现的 tcp 客户端
 *
 * 这是一个 利用 select 实现 tcp 客户端的例程
 * 导出 tcpclient_select 命令到控制终端
 * 命令调用格式：tcpclient_select URL PORT
 * URL：服务器地址 PORT:：端口号
 * 程序功能：利用 select 监听 socket 是否有数据到达，
 * 有数据到达的话再接收并显示从服务端发送过来的信息，
 * 接收到开头是 'q' 或 'Q' 的信息退出程序
*/
#include <rtthread.h>
#include <sys/socket.h> /* 使用BSD socket，需要包含socket.h头文件 */
#include <netdb.h>
#include <sys/select.h> /* 使用 dfs select 功能 */
#include <string.h>
#include <finsh.h>

#define BUFSZ 1024
```

RT-Thread

示例代码讲解

```
static const char send_data[] = "This is TCP Client from RT-Thread."; /* 发送用到的数据 */
void tcpclient_select(int argc, char **argv)
{
    int ret;
    char *recv_data;
    struct hostent *host;
    int sock, bytes_received;
    struct sockaddr_in server_addr;
    const char *url;
    int port;
    fd_set readset; //定义 fd_set 结构体
    int i, maxfdp1;  //定义最大描述符个数的变量

    if (argc < 3)
    {
        rt_kprintf("Usage: tcpclient_select URL PORT\n");
        rt_kprintf("Like: tcpclient_select 192.168.12.44 5000\n");
        return ;
    }
```

示例代码讲解

```
    url = argv[1];
    port = strtoul(argv[2], 0, 10);

    /* 通过函数入口参数url获得host地址（如果是域名，会做域名解析） */
    host = gethostbyname(url);

    /* 分配用于存放接收数据的缓冲 */
    recv_data = rt_malloc(BUFSZ);
    if (recv_data == RT_NULL)
    {
        rt_kprintf("No memory\n");
        return;
    }

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

        /* 释放接收缓冲 */
        rt_free(recv_data);
        return;
}
```

RT-Thread

示例代码讲解

```
/* 初始化预连接的服务端地址 */
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(port);
  server_addr.sin_addr = *((struct in_addr *)host->h_addr);
  rt_memset(&(server_addr.sin_zero), 0, sizeof(server_addr.sin_zero));

  /* 连接到服务端 */
  if (connect(sock, (struct sockaddr *)&server_addr, sizeof(struct sockaddr)) == -1)
  {
    /* 连接失败 */
    rt_kprintf("Connect fail!\n");
    closesocket(sock);

    /*释放接收缓冲 */
    rt_free(recv_data);
    return;
  }

  /* 获取需要监听的描述符号最大值 */
maxfdp1 = sock + 1;
```

RT-Thread

示例代码讲解

```
while (1)
  {
    /* 清空可读事件描述符列表 */
    FD_ZERO(&readset);

    /* 将需要监听可读事件的描述符加入列表 */
    FD_SET(sock, &readset);

    /* 等待设定的网络描述符有事件发生 */
    i = select(maxfdp1, &readset, 0, 0, 0);

    /* 至少有一个文件描述符有指定事件发生再向后运行 */
    if (i == 0) continue;

    /* 查看 sock 描述符上有没有发生可读事件 */
    if (FD_ISSET(sock, &readset))
    {
      /* 从sock连接中接收最大BUFSZ - 1字节数据 */
      bytes_received = recv(sock, recv_data, BUFSZ - 1, 0);
      if (bytes_received < 0)
      {
        /* 接收失败，关闭这个连接 */
        closesocket(sock);
        rt_kprintf("\nreceived error,close the socket.\r\n");
```

示例代码讲解

```
        /* 释放接收缓冲 */
        rt_free(recv_data);
        break;
    }
    else if (bytes_received == 0)
    {
        /* 默认 recv 为阻塞模式，此时收到0认为连接出错，关闭这个连接 */
        closesocket(sock);
        rt_kprintf("\nreceived error,close the socket.\r\n");

        /* 释放接收缓冲 */
        rt_free(recv_data);
        break;
    }

    /* 有接收到数据，把末端清零 */
    recv_data[bytes_received] = '\0';

    if (strncmp(recv_data, "q", 1) == 0 || strncmp(recv_data, "Q", 1) == 0)
    {
        /* 如果是首字母是q或Q，关闭这个连接 */
        closesocket(sock);
        rt_kprintf("\n got a 'q' or 'Q',close the socket.\r\n");
```

示例代码讲解

```
            /* 释放接收缓冲 */
            rt_free(recv_data);
            break;
        }
        else
        {
            /* 在控制终端显示收到的数据 */
            rt_kprintf("\nReceived data = %s ", recv_data);
        }

        /* 发送数据到sock连接 */
        ret = send(sock, send_data, strlen(send_data), 0);
        if (ret < 0)
        {
            /* 发送失败，关闭这个连接 */
            closesocket(sock);
            rt_kprintf("\nsend error,close the socket.\r\n");

            rt_free(recv_data);
            break;
        }
        else if (ret == 0)
{
```

RT-Thread

示例代码讲解

```
            /* 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");
        }
      }
   }
   return;
}

MSH_CMD_EXPORT(tcpclient_select, a tcp client sample by select api);
```

RT-Thread