嵌入式系统工程师

UDP网络编程

大纲

- 概述
- 基本UDP编程
- UDP广播

大纲

- 概述
- 基本UDP编程
- UDP广播

- UDP协议

  UDP是面向无连接的用户数据报协议，在传输数前不需要先建立连接。目地主机的运输层收到UDP报文后，不需要给出任何确认

- UDP协议与TCP协议的差异

	TCP	UDP
是否面向连接	✓	✗
是否可靠	✓	✗
是否广播，多播	✗	✓
效率	低	高

概述

- 如何在TCP和UDP之间取舍
  - 广播和多播应用必须使用UDP
  - 简单的请求-应答应用程序可以使用UDP
  - 对于海量数据传输不应该使用UDP

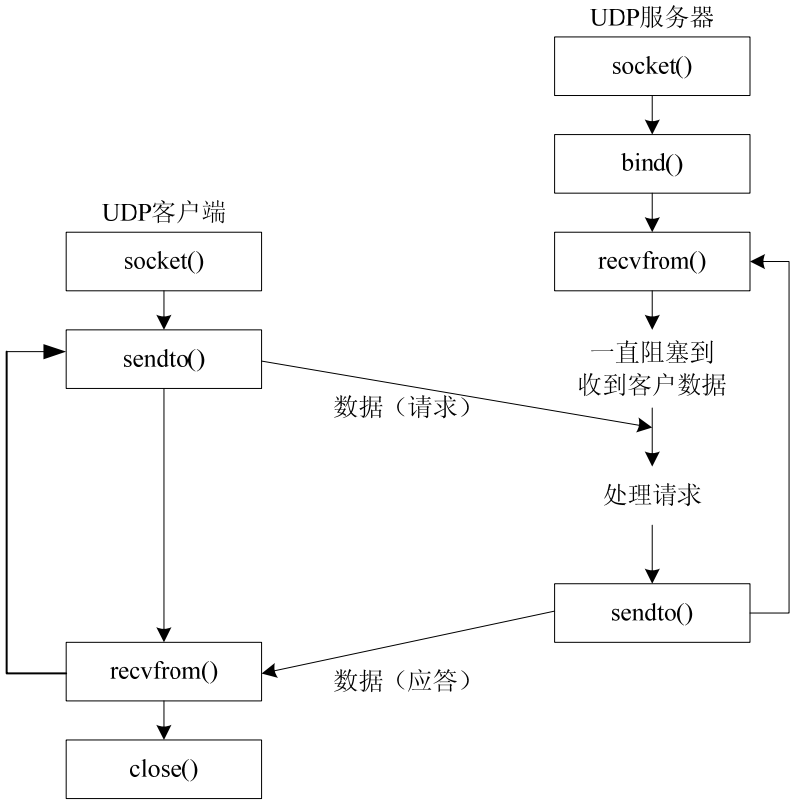
- UDP的使用场合
  - DNS、NFS、流媒体传输等等

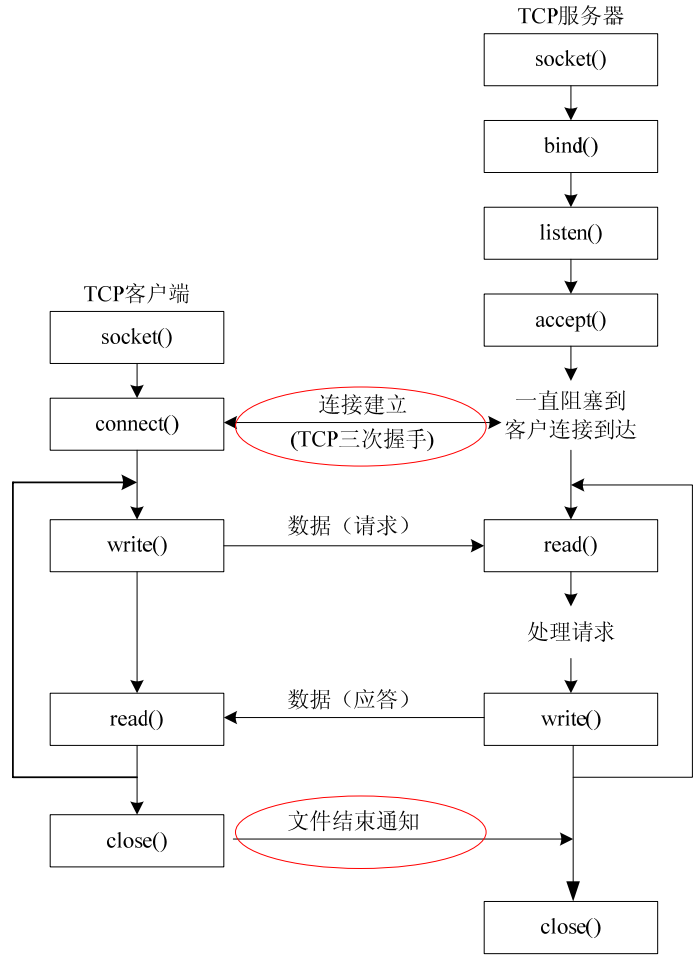
大纲

- 概述
- 基本UDP编程
- UDP广播

基本UDP编程

- 典型的UDP客户/服务器程序





基本UDP编程

- ssize_t recvfrom(int sockfd, void *buf, size_t nbytes, int flags, struct sockaddr *from, socklen_t *addrlen);
- 功能:
  - 用于接收数据
- 参数:
  - sockfd: 套接字
  - buf: 接收数据缓冲区
  - nbytes: 接收数据缓冲区的大小
  - flags: 套接字标志(常为0)
  - from: 用于存放发送方信息的地址结构体指针
  - addrlen: from所指内容的长度

基本UDP编程

- 注意:
  - struct sockaddr *from, socklen_t *addrlen
    - 类似于accept函数的最后两个参数
    - 通过from和addrlen参数存放数据来源
    - 可以为NULL，表示不关心数据来源
- 返回值:
  - 成功:接收到的字符数
  - 失败: -1

基本UDP编程

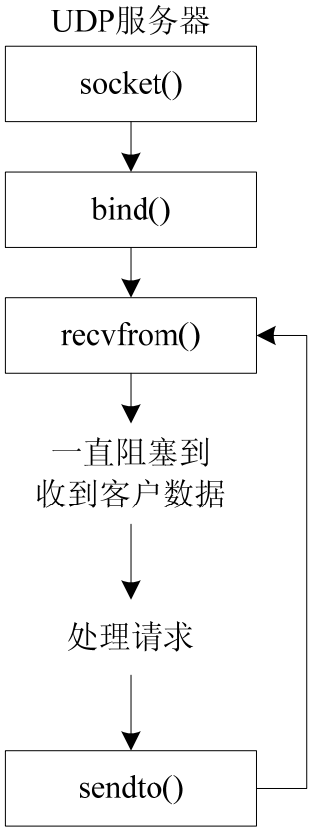
- ssize_t sendto(int sockfd, const void *buf, size_t nbytes,int flags, const struct sockaddr *to,socklen_t addrlen);
- 功能:
  - 用于发送数据
- 参数:
  - sockfd: 套接字
  - buf: 发送数据缓冲区
  - nbytes:发送数据缓冲区的大小
  - flags: 一般为0
  - to: 指向目的主机地址结构体的指针
  - addrlen: to所指向内容的长度

基本UDP编程

- 注意:
  - const struct sockaddr *to, socklen_t addrlen
    - 类似于connect函数的最后两个参数
    - 通过to和addrlen确定目的地址
  - 发送一个0长度的UDP数据包是可行的
- 返回值:
  - 成功: 发送的字符数
  - 失败: -1

基本UDP编程

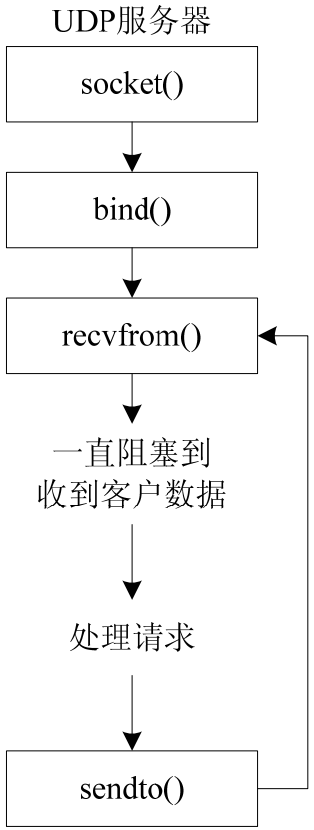
➢ UDP Echo Server




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

int main(int argc, char *argv[])
{
    int sockfd = 0;
    int error_log = 0;
    struct sockaddr_in bind_addr;
    unsigned short port = 8000;

    if(argc > 1)
    {
        port = atoi(argv[1]);
    }

    printf("UDP Server Started!\n");
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if(sockfd < 0)
    {
        perror("socket error");
        exit(-1);
    }
```

基本UDP编程

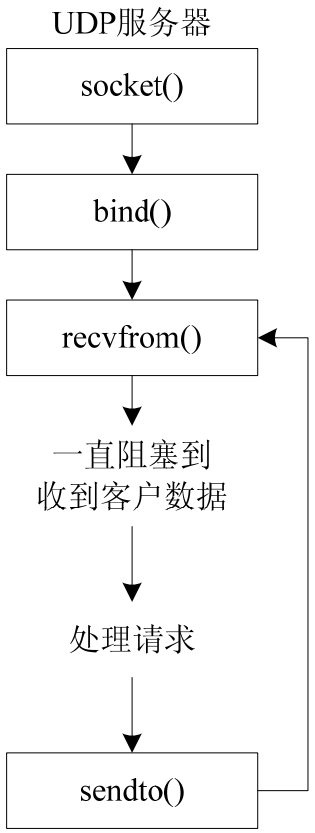
➢ UDP Echo Server




```
bzero(&bind_addr, sizeof(bind_addr));
bind_addr.sin_family = AF_INET;
bind_addr.sin_port   = htons(port);
bind_addr.sin_addr.s_addr = htonl(INADDR_ANY);

printf("Binding server to port %d\n", port);
error_log = bind(sockfd, (struct sockaddr*)&bind_addr, sizeof(bind_addr));
if(error_log != 0)
{
    perror("bind error");
    close(sockfd);
    exit(-1);
}
```

基本UDP编程

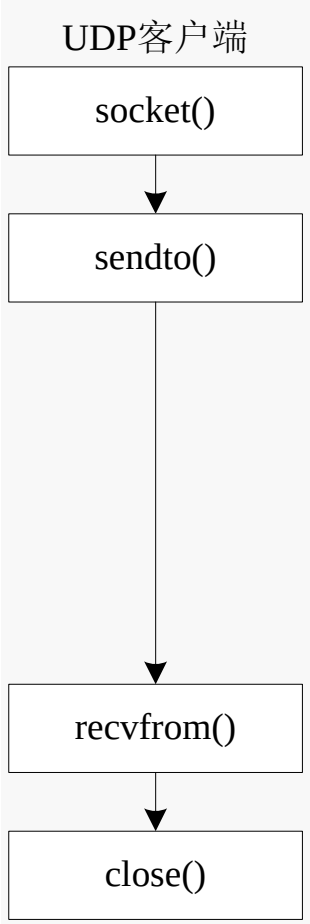
➢ UDP Echo Server




```
    printf("waiting data from other client...\n");
    while(1)
    {
        char recv_buf[1024] = "";
        char cli_ip[INET_ADDRSTRLEN] = "";
        int recv_len = 0;
        struct sockaddr_in client_addr;
        socklen_t client_addr_len = sizeof(client_addr);

        recv_len = recvfrom(sockfd, recv_buf, sizeof(recv_buf), 0,\
        (struct sockaddr*)&client_addr, &client_addr_len);

        inet_ntop(AF_INET, &client_addr.sin_addr, cli_ip, INET_ADDRSTRLEN)
        printf("client ip = %s\n", cli_ip);

        sendto(sockfd, recv_buf, recv_len, 0,\
        (struct sockaddr*)&client_addr, client_addr_len);
    }

    close(sockfd);
    return 0;
}
```

基本UDP编程

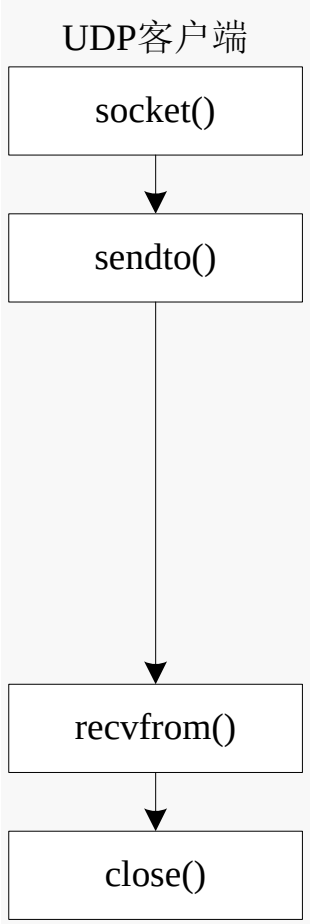
➢ UDP Client




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

int main(int argc, char *argv[])
{
    int sockfd = 0;
    struct sockaddr_in server_addr;
    unsigned short port = 8000;
    char *ser_ip = "172.20.223.151";

    if( argc > 1 )                                  //服务器ip地址
    {
        ser_ip = argv[1];
    }
    if( argc > 2 )                                  //服务器端口
    {
        port = atoi(argv[2]);
    }

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);//创建UDP套接字
    if(sockfd < 0)
    {
        perror("socket error!");
        exit(-1);
    }
```

基本UDP编程

➢ UDP Client




```
    bzero(&server_addr, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port   = htons(port);
    inet_pton(AF_INET, ser_ip, &server_addr.sin_addr);

    printf("ready send data to UDP server %s:%d!\n", ser_ip, port);
    while(1)
    {
        char send_buf[2048] = "";
        int len = 0;

        fgets(send_buf, sizeof(send_buf), stdin);
        send_buf[strlen(send_buf)-1] = '\0';

        sendto(sockfd, send_buf, strlen(send_buf), 0,\
        (struct sockaddr*)&server_addr, sizeof(server_addr));

        len = recvfrom(sockfd, send_buf, sizeof(send_buf), 0, NULL, NULL);
        printf("%s\n", send_buf);
    }

    close(sockfd);
    return 0;
}
```

练习客户端和服务器

大纲

- 概述
- 基本UDP编程
- UDP广播

UDP广播

UDP广播概述

- 广播：由一台主机向该主机所在子网内的所有主机发送数据的方式
- 广播只能用UDP或原始IP实现，不能用TCP
- 广播的用途
  - 单个服务器与多个客户主机通信时减少分组流通
  - 地址解析协议（ARP）
  - 动态主机配置协议（DHCP）
  - 网络时间协议（NTP）

UDP广播

- 广播地址
  - {子网ID，主机ID}
    - 子网ID表示由子网掩码中1覆盖的连续位
    - 主机ID表示由子网掩码中0覆盖的连续位
  - 子网定向广播地址：主机ID全1
    - 例如：对于192.168.220.0/24子网，192.168.220.255即为其定向广播地址
    - 通常路由器不转发该广播
  - 受限广播地址：255.255.255.255
    - 路由器从不转发该广播
    - 通常在DHCP等应用中把该地址当做宿主地址，因为此时客户主机还不知道所处子网的信息

UDP广播

- UDP广播的特点
  - 处于同一子网的所有主机都必须处理数据
  - UDP数据包会沿协议栈向上一直到UDP层
  - 运行音视频等较高速率工作的应用，会带来大负
  - 局限于局域网内使用

UDP广播

➢ 套接口选项

- int setsockopt(int sockfd, int level,int optname, const void *optval, socklen_t optlen);
- 成功执行返回0，否则返回-1

level	optname	说明	optval类型
SOL_SOCKET	SO_BROADCAST	允许发送广播数据包	int
	SO_RCVBUF	接收缓冲区大小	int
	SO_SNDBUF	发送缓冲区大小	int

UDP广播

- 广播示例：

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

int main(int argc, char *argv[])
{
    int sockfd = 0;
    char buf[1024] = "";
    unsigned short port = 8000;
    struct sockaddr_in send_addr;

    bzero(&send_addr, sizeof(send_addr));
    send_addr.sin_family = AF_INET;
    send_addr.sin_port = htons(port);

    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if(sockfd < 0)
    {
        perror("socket error");
        exit(1);
    }
```

UDP广播

➢ 广播示例

```
    if(argc > 1)
    {
        send_addr.sin_addr.s_addr = inet_addr(argv[1]);
    }
    else
    {
        printf("not have a server IP\n");
        exit(1);
    }

    int yes = 1;
    setsockopt(sockfd, SOL_SOCKET, SO_BROADCAST, &yes, sizeof(yes));

    strcpy(buf, "boardcast sucess");
    int len = sendto(sockfd, buf, strlen(buf), 0,\
        (struct sockaddr *)&send_addr, sizeof(send_addr));
    if(len < 0)
    {
        printf("send error\n");
        close(sockfd);
        exit(1);
    }

    return 0;
}
```