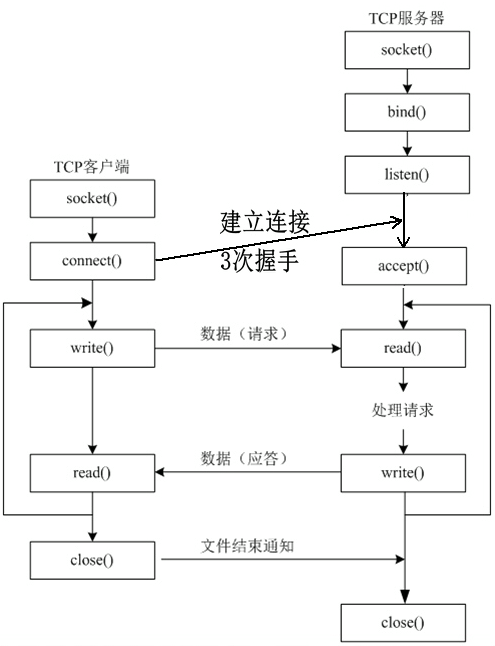
# 一、TCP概述

TCP（Transmission Control Protocol 传输控制协议）是一种面向连接的、可靠的、基于字节流的传输层通信协议。

# 二、TCP 编程的 C/S 架构

基于 TCP 的网络编程开发分为服务器端和客户端两部分，常见的核心步骤和流程如下：



# 三、TCP 客户端编程

所需头文件：#include <sys/socket.h>

**int socket(int family,int type,int protocol);**

功能：

创建一个用于网络通信的 socket[套接字](http://blog.csdn.net/tennysonsky/article/details/45047209)（描述符）

参数：

family：本示例写 AF\_INET，代表 IPv4

type：本示例写 SOCK\_STREAM，代表 TCP 数据流

protocol：这里写 0，设为 0 表示使用默认协议

返回值：

成功：[套接字](http://blog.csdn.net/tennysonsky/article/details/45047209)

失败 < 0

**int connect( int sockfd, const struct sockaddr \*addr, socklen\_t len );**

功能：

主动跟服务器建立连接

参数：

sockfd：[socket()返回的套接字](http://blog.csdn.net/tennysonsky/article/details/45047209)

addr：连接的服务器地址结构

len：地址结构体长度

返回值：

成功：0

失败：-1

**ssize\_t send(int sockfd, const void\* buf, size\_t nbytes, int flags);**

功能：

发送数据，最后一个参数为 0 时，可以用 write() 替代（ send 等同于 write ）。注意：不能用 TCP 协议发送 0 长度的数据包。

参数：

sockfd： 已建立连接的套接字

buf： 发送数据的地址

nbytes： 发送缓数据的大小(以字节为单位)

flags： 套接字标志（常为 0）

返回值：

成功：成功发送的字节数

失败 < 0

**ssize\_t recv(int sockfd, void \*buf,  size\_t nbytes, int flags);**

功能：

接收网络数据，默认的情况下，如果没有接收到数据，这个函数会阻塞，直到有数据到来。

参数：

sockfd：[套接字](http://blog.csdn.net/tennysonsky/article/details/45047209)

buf：接收网络数据的缓冲区的地址

nbytes：接收缓冲区的大小（以字节为单位）

flags：套接字标志（常为 0 ）

返回值：

成功：成功接收的字节数

失败 < 0

测试代码如下：

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <stdlib.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <netinet/in.h>

int main()

{

    unsigned short port = 8000;             // 服务器的端口号

char \*server\_ip = "127.0.0.1";      // 服务器ip地址

    int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);// 创建通信端点：套接字

    if(sockfd < 0)

    {

        perror("socket");

        exit(-1);

    }

    // 设置服务器地址结构体

    struct sockaddr\_in server\_addr;

    bzero(&server\_addr,sizeof(server\_addr)); // 初始化服务器地址

    server\_addr.sin\_family = AF\_INET;   // IPv4

    server\_addr.sin\_port = htons(port); // 端口

    inet\_pton(AF\_INET, server\_ip, &server\_addr.sin\_addr.s\_addr);    // ip

     // 主动连接服务器

    int err\_log = connect(sockfd, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

    if(err\_log != 0)

    {

        perror("connect");

        close(sockfd);

        exit(-1);

    }

    // 发送数据

    char send\_buf[512] = {0};

    printf("send data to %s:%d\n",server\_ip,port);

    printf("send:");

    fgets(send\_buf,sizeof(send\_buf),stdin); // 输入内容

    send\_buf[strlen(send\_buf)-1]='\0';

send(sockfd, send\_buf, strlen(send\_buf), 0);   // 向服务器发送信息

    // 接收数据

    char recv\_buf[512] = {0};

    recv(sockfd, recv\_buf, sizeof(recv\_buf), 0); // 接收数据

printf("%s\n", recv\_buf);

    close(sockfd);

    return 0;

}