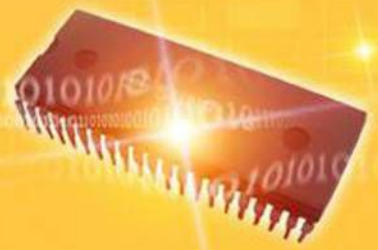
嵌入式系统工程师



TCP网络编程

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP客户端-connect、send、recv
- TCP服务器-bind、listen、accept
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP客户端-connect、send、recv
- TCP服务器-bind、listen、accept
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com

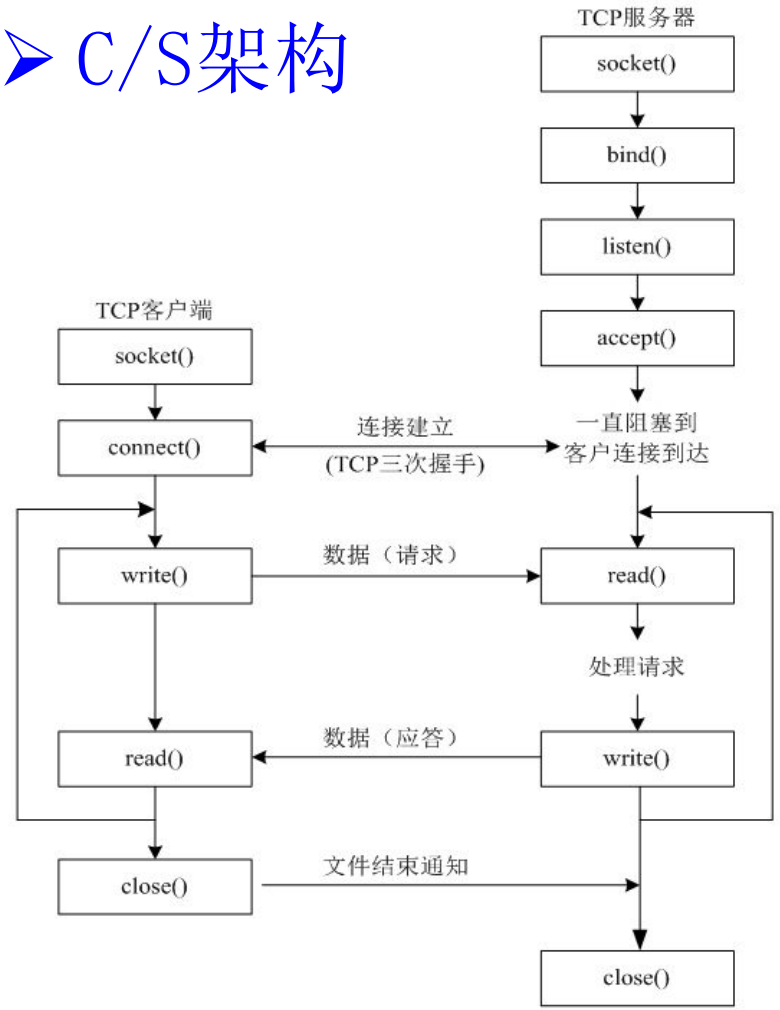
TCP介绍

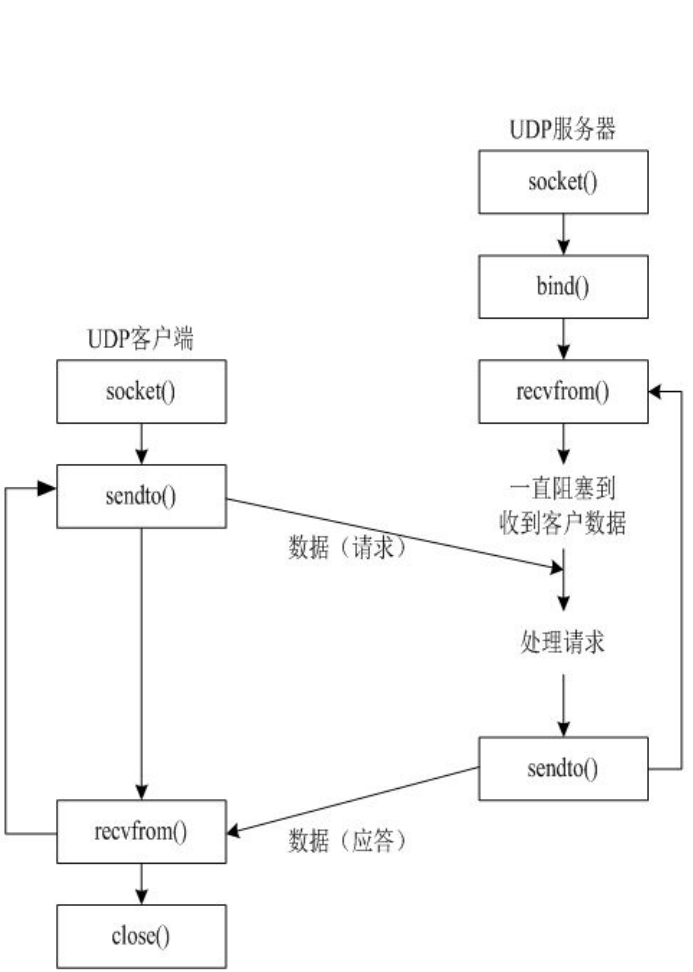
- TCP回顾
  - 面向连接的流式协议;可靠、出错重传、且每收到一个数据都要给出相应的确认
  - 通信之前需要建立链接
  - 服务器被动链接，客户端是主动链接
- TCP与UDP的差异

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

TCP介绍

- C/S架构





大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP客户端-connect、send、recv
- TCP服务器-bind、listen、accept
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

创建TCP套接字

- UDP套接字回顾

```
16      int sockfd;
17      sockfd = socket(AF_INET, SOCK_DGRAM, 0);
18      if(sockfd < 0)
19      {
20          perror("socket");
21          exit(-1);
22      }
```

- 创建TCP套接字

```
2       int sockfd;
3       sockfd = socket(AF_INET, SOCK_STREAM, 0);
4       if(sockfd < 0)
5       {
6           perror("socket");
7           exit(-1);
8       }
```

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP客户端-connect、send、recv
- TCP服务器-bind、listen、accept
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

TCP客户端

- 做为客户端需要具备的条件
  - 知道“服务器”的ip、port
  - 主动连接“服务器”
- 需要用到的函数
  - socket—创建“主动TCP套接字”
  - connect—连接“服务器”
  - send—发送数据到“服务器”
  - recv—接受“服务器”的响应
  - close—关闭连接

连接—connect

- int connect(int sockfd,
  const struct sockaddr *addr,
  socklen_t len);
- 功能：主动跟服务器建立链接
- 参数：
  - sockfd：socket套接字
  - 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：其他

连接—connect

- 注意：
  - connect建立连接之后不会产生新的套接字
  - 连接成功后才可以开始传输TCP数据
- 头文件：#include <sys/socket.h>

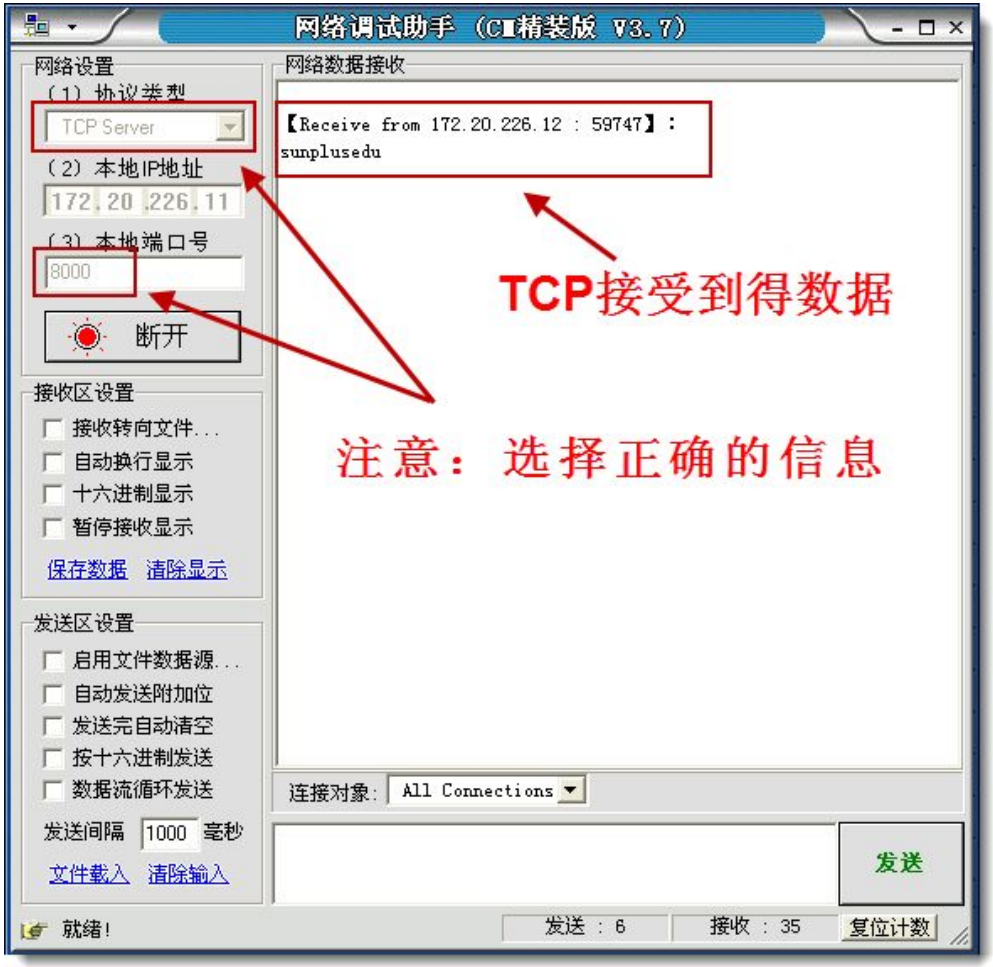
发送数据—send

- ssize_t send(int sockfd, const void* buf, size_t nbytes, int flags);
- 功能：用于发送数据
- 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：成功发送的字节数
- 头文件：#include <sys/socket.h>
- 注意：不能用TCP协议发送0长度的数据包

发送数据—demo

- 发送数据如图所示：

```
root@edu-T: ~/share/net/TCP_client
root@edu-T:~/share/net/TCP_client# ./a.out
send data to 172.20.226.11:8000
send:sunplusedu
```

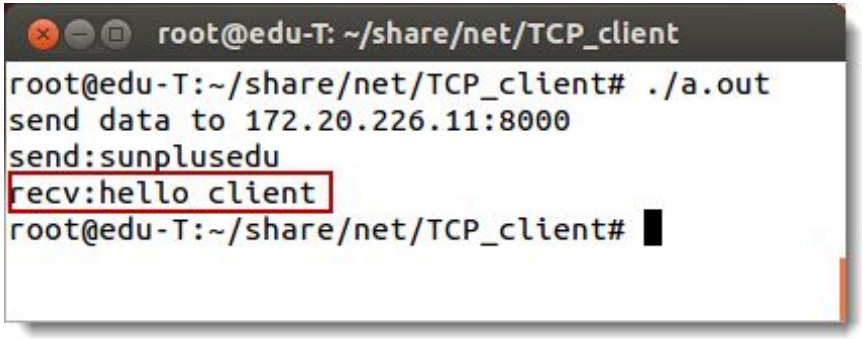


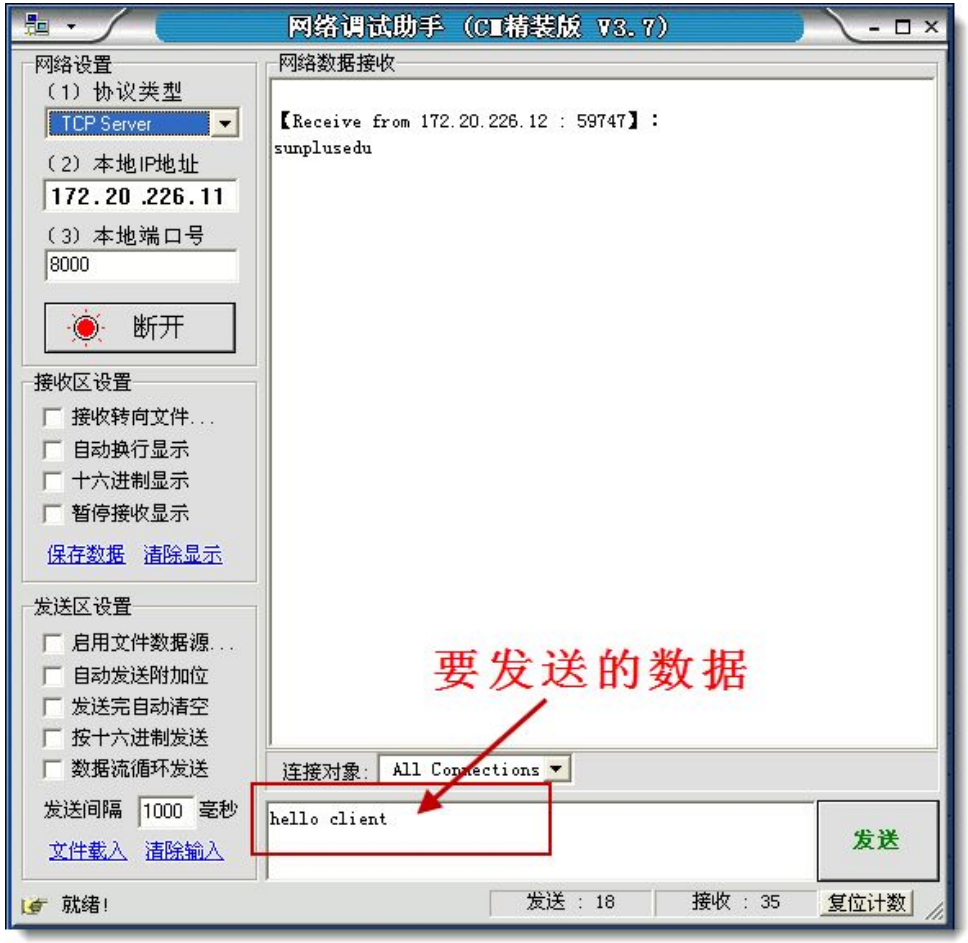
接收数据—recv

- ssize_t recv(int sockfd, void *buf,
  size_t nbytes, int flags);
- 功能：用于接收网络数据
- 参数：
  - sockfd：套接字
  - 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：成功接收到字节数
- 头文件：#include <sys/socket.h>

接收数据—demo

- 接收数据如图所示：





客户端-code

```c
#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(int argc, char *argv[])
{
    unsigned short port = 8000;                 //服务器的端口号
    char *server_ip = "172.20.226.11";          //服务器的IP
    if( argc > 1 )      //函数参数，可以更改服务器的IP
    {
        server_ip = argv[1];
    }
    if( argc > 2 )      //函数参数，可以更改服务器的端口号
    {
        port = atoi(argv[2]);
    }

    int sockfd = 0;
    sockfd = socket(AF_INET, SOCK_STREAM, 0);//创建通行端点：套接字
    if(sockfd < 0)
    {
        perror("socket");
        exit(-1);
    }
```

给main传参

1.创建TCP套接字

客户端-code

```
29      struct sockaddr_in server_addr;
30      bzero(&server_addr, sizeof(server_addr));//初始化服务器地址
31      server_addr.sin_family = AF_INET;
32      server_addr.sin_port = htons(port);
33      inet_pton(AF_INET, server_ip, &server_addr.sin_addr);
34
35      int err_log = connect(sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr));
36      if(err_log != 0)
37      {
38          perror("connect");
39          close(sockfd);
40          exit(-1);
41      }
42
43      char send_buf[512] = "";
44      char recv_buf[512] = "";
45      printf("send data to %s:%d\n", server_ip, port);
46
47      printf("send:");
48      fgets(send_buf, sizeof(send_buf), stdin);
49      send_buf[strlen(send_buf)-1] = 0;//去掉'\n'
50      send(sockfd, send_buf, strlen(send_buf), 0);//向服务器发送数据
51
52      recv(sockfd, recv_buf, sizeof(recv_buf), 0);//接受服务器的响应
53      printf("recv:%s\n", recv_buf);
54      close(sockfd);
55  }
```

2.设置要连接的IP、端口

3.连接服务器

4.发送消息

5.接收数据

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP客户端-connect、send、recv
- TCP服务器-bind、listen、accept
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

TCP服务器

- 做为TCP服务器需要具备的条件
  - 具备一个可以确知的地址
  - 让操作系统知道是一个服务器，而不是客户端
  - 等待连接的到来
- 对于面向连接的TCP协议来说，连接的建立才真正意味着数据通信的开始

服务器—bind

➢ bind示例

```
18      int err_log = 0;
19      unsigned short port = 8000;
20      struct sockaddr_in my_addr;
21
22      bzero(&my_addr, sizeof(my_addr));
23      my_addr.sin_family = AF_INET;
24      my_addr.sin_port    = htons(port);
25      my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
26
27      err_log = bind(sockfd, (struct sockaddr*)&my_addr, sizeof(my_addr));
28      if( err_log != 0)
29      {
30          perror("binding");
31          close(sockfd);
32          exit(-1);
33      }
```

服务器—listen

- int listen(int sockfd, int backlog);
- 功能：
  - 将套接字由主动修改为被动
  - 使操作系统为该套接字设置一个连接队列，用来记录所有连接到该套接字的连接
- 参数：
  - sockfd： socket监听套接字
  - backlog：连接队列的长度
- 返回值：
  - 成功：返回0
  - 失败：其他
- 头文件：#include <sys/socket.h>

服务器—accept

- int accept(int sockfd,struct sockaddr *cliaddr, socklen_t *addrlen);
- 功能：从已连接队列中取出一个已经建立的连接，如果没有任何连接可用，则进入睡眠等待(阻塞)
- 参数：
  - sockfd： socket监听套接字
  - cliaddr：用于存放客户端套接字地址结构
  - addrlen：套接字地址结构体长度的地址
- 返回值：已连接套接字
- 头文件：#include <sys/socket.h>
- 注意：返回的是一个已连接套接字，这个套接字代表当前这个连接

服务器—code

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <sys/socket.h>
6  #include <netinet/in.h>
7  #include <arpa/inet.h>
8  int main(int argc, char *argv[])
9  {
10     unsigned short port = 8000;
11     if(argc > 1)
12     {
13         port = atoi(argv[1]);
14     }
15
16     int sockfd = socket(AF_INET, SOCK_STREAM, 0);
17     if(sockfd < 0)
18     {
19         perror("socket");
20         exit(-1);
21     }
22
23     struct sockaddr_in my_addr;
24     bzero(&my_addr, sizeof(my_addr));
25     my_addr.sin_family = AF_INET;
26     my_addr.sin_port   = htons(port);
27     my_addr.sin_addr.s_addr = htonl(INADDR_ANY);
```

1.创建TCP套接字

2.组织本地信息

服务器—code

```
28
29      int err_log = bind(sockfd, (struct sockaddr*)&my_addr, sizeof(my_addr));
30      if( err_log != 0)
31      {
32          perror("binding");
33          close(sockfd);
34          exit(-1);
35      }
36
37      err_log = listen(sockfd, 10);
38      if(err_log != 0)
39      {
40          perror("listen");
41          close(sockfd);
42          exit(-1);
43      }
44
45      printf("listen client @port=%d...\n",port);
46
47      while(1)
48      {
49
50          struct sockaddr_in client_addr;
51          char cli_ip[INET_ADDRSTRLEN] = "";
52          socklen_t cliaddr_len = sizeof(client_addr);
53
```

3.绑定信息

4.“主动”变“被动”

服务器—code

```
54          int connfd;
55          connfd = accept(sockfd, (struct sockaddr*)&client_addr, &cliaddr_len);
56          if(connfd < 0)
57          {
58              perror("accept");
59              continue;
60          }
61
62          inet_ntop(AF_INET, &client_addr.sin_addr, cli_ip, INET_ADDRSTRLEN);
63          printf("----------------------------------------\n");
64          printf("client ip=%s,port=%d\n", cli_ip,ntohs(client_addr.sin_port));
65
66          char recv_buf[2048] = "";
67          while( recv(connfd, recv_buf, sizeof(recv_buf), 0) > 0 )
68          {
69              printf("\nrecv data:\n");
70              printf("%s\n",recv_buf);
71          }
72
73          close(connfd);      //关闭已连接套接字
74          printf("client closed!\n");
75      }
76      close(sockfd);          //关闭监听套接字
77      return 0;
78  }
```

5.等待连接的到来

6.转换并打印信息

7.接收信息

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP编程-connect、send、recv
- TCP编程-bind、listen、accept
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

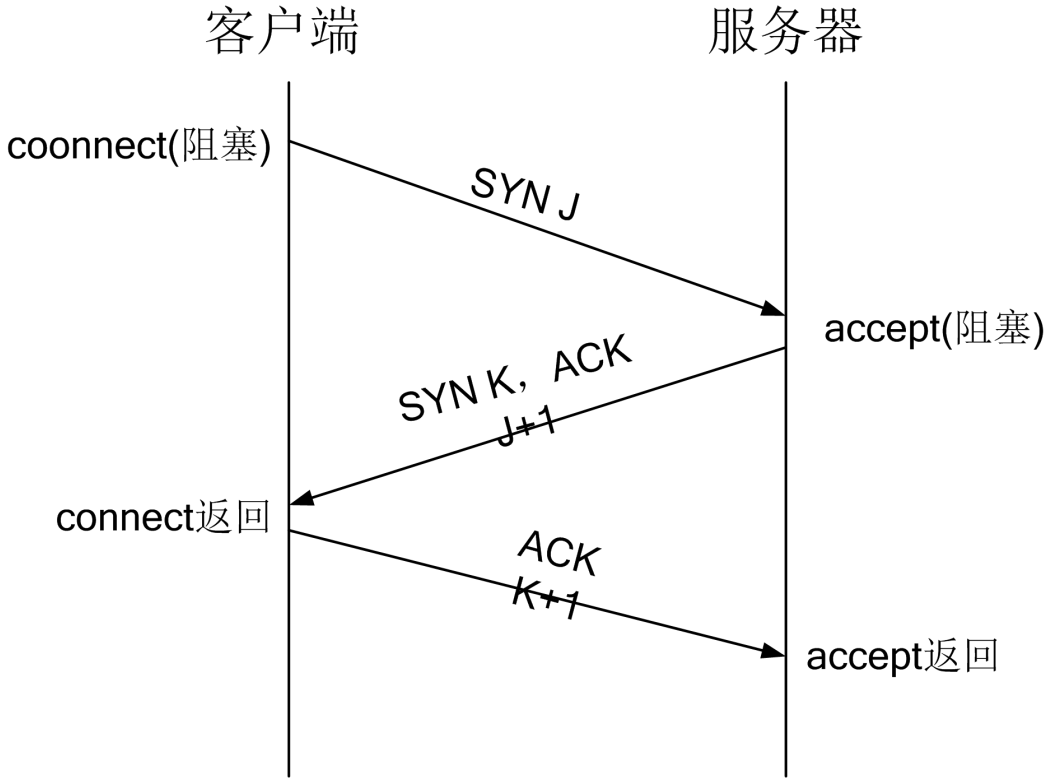
关闭连接—close

- 使用close函数即可关闭套接字
- 关闭一个代表已连接套接字将导致另一端接收到一个0长度的数据包
- 做服务器时
  - 关闭监听套接字将导致服务器无法接收新的连接，但不会影响已经建立的连接
  - 关闭accept返回的已连接套接字将导致它所代表的连接被关闭，但不会影响服务器的监听
- 做客户端时

关闭连接就是关闭连接，不意味着其他

TCP—三次握手

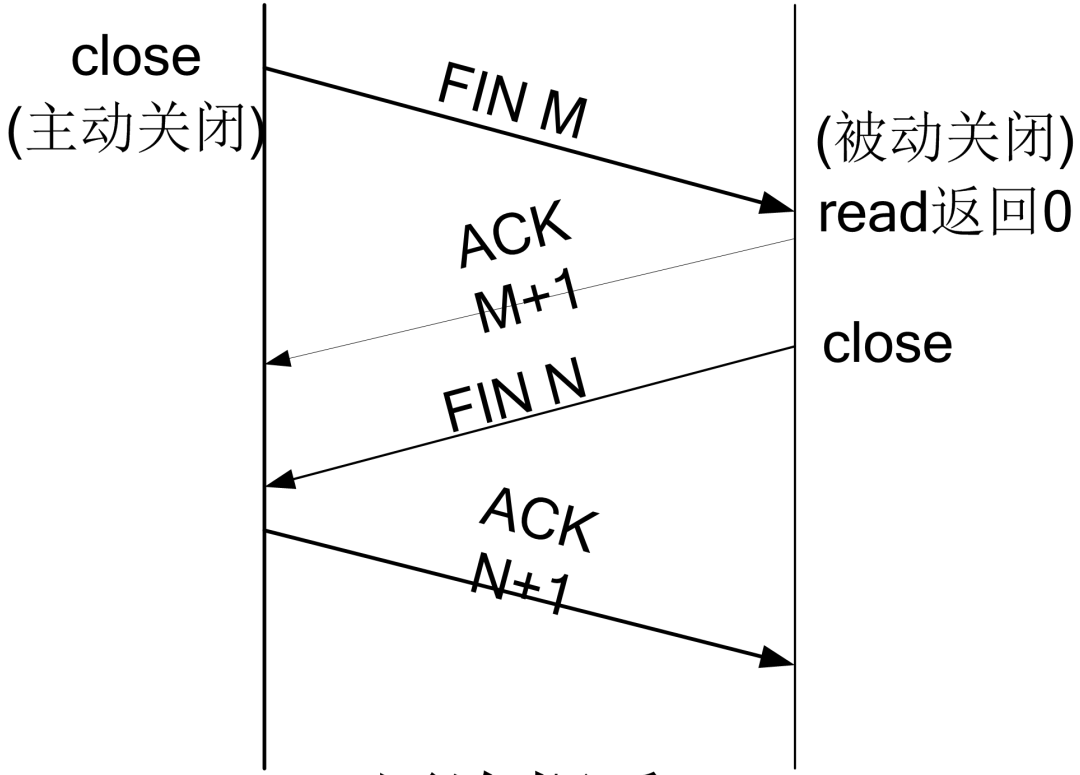
- TCP面向连接的三次握手过程



三次握手

TCP—四次挥手

- TCP面向连接的四次挥手过程



四次挥手

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP编程-connect、send
- TCP编程-bind、listen、accept、recv
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

TCP服务器并发—进程

- 多进程：fork()

```
1  #include <头文件>
2  int main(int argc, char *argv[])
3  {
4      创建套接字sockfd
5      绑定(bind)套接字sockfd
6      监听(listen)套接字sockfd
7
8      while(1)
9      {
10         int connfd = accept();
11
12         if(fork() == 0)      //子进程
13         {
14             close(sockfd);//关闭监听套接字sockfd
15
16             fun();           //服务客户端的具体事件在fun里实现
17
18             close(connfd);//关闭已连接套接字connfd
19             exit(0);         //结束子进程
20         }
21         close(connfd);       //关闭已连接套接字connfd
22     }
23     close(sockfd);
24     return 0;
25 }
```

TCP服务器并发—线程

- 多线程:pthread_create()

```
1  #include <头文件>
2  int main(int argc, char *argv[])
3  {
4      创建套接字sockfd
5      绑定(bind)套接字sockfd
6      监听(listen)套接字sockfd
7
8      while(1)
9      {
10         int connfd = accept();
11         pthread_t tid;
12         pthread_create(&tid, NULL, (void *)client_fun, (void *)connfd);
13         pthread_detach(tid);
14     }
15     close(sockfd);//关闭监听套接字
16     return 0;
17 }
18 void *client_fun(void *arg)
19 {
20     int connfd = (int)arg;
21     fun();//服务于客户端的具体程序
22     close(connfd);
23 }
```

大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP编程-connect、send
- TCP编程-bind、listen、accept、recv
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

Webserver—介绍

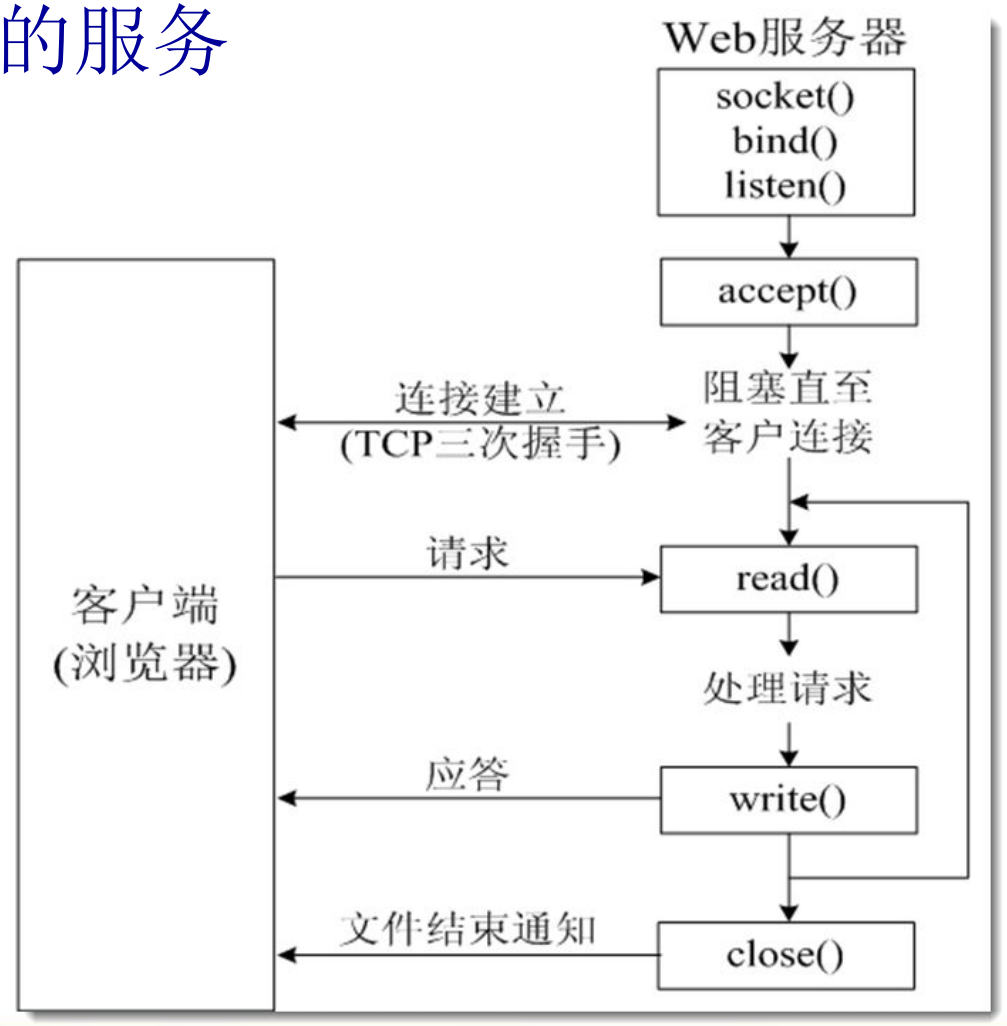
- Web服务器又称WWW服务器、网站服务器等
- 特点
  - 使用HTTP协议与客户机浏览器进行信息交流
  - 不仅能存储信息，还能在用户通过web浏览器提供的信息的基础上运行脚本和程序
  - 该服务器需可安装在UNIX、Linux或Windows等操作系统上
  - 著名的服务器有Apache、Tomcat、 IIS等

Webserver—HTTP协议

- 概念
  - 一种详细规定了浏览器和万维网服务器之间互相通信的规则，通过因特网传送万维网文档的数据传送协议
- 特点
  - 支持C/S架构
  - 简单快速：客户向服务器请求服务时，只需传送请求方法和路径，常用方法:GET、POST
  - 无连接：限制每次连接只处理一个请求
  - 无状态：即如果后续处理需要前面的信息，它必须重传，这样可能导致每次连接传送的数据量会增大

Webserver—通信过程

- 面向连接的服务



大纲

- TCP介绍、编程流程
- TCP编程-socket
- TCP编程-connect、send
- TCP编程-bind、listen、accept、recv
- TCP编程-close、三次握手、四次挥手
- TCP并发服务器
- Web服务器介绍
- Web编程开发

Webserver—网页浏览

- 网页浏览（使用GET方式）


```
GET /index.html HTTP/1.1
Accept:image/gif.image/jpeg,*/*
Accept-Language:zh-cn
Connection:Keep-Alive
Host:localhost
Accept-Encoding:gzip,deflate
```

客户端浏览器请求　　服务器接收到的数据

Webserver—网页浏览

- 服务器应答的格式：请求成功

```
HTTP/1.1 200 OK
Server:Apache Tomcat/5.0.12
Content-Type:text/html
Content-Length:28
<body>
Hello HTTP!
</body>
```

Webserver—网页浏览

- 服务器应答的格式：请求失败

```
HTTP/1.1 404 Not Fount
Server:Apache Tomcat/5.0.12
Content-Type:text/html
Content-Length:40
<HTML><BODY>File not fount</BODY></HTML>
```

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com