


第9章 网络程序设计
——套接字

主　讲：王小银

10.2 socket套接字

socket：是一种通信机制，通过该机制，客户端/服务器系统的开发工作既可以在本地单机上进行，也可以跨机器进行。

常用的套接字有三种类型：

1. 流式套接字(SOCK_STREAM)：提供了一个可靠的、面向连接的数据传输服务，数据无差错、无重复的发送且按发送顺序接收。

2. 数据报套接字(SOCK_DGRAM)：提供了一种无连接的服务，数据通过相互独立的报文进行传输，是无序的，并且不保证可靠，无差错。

3. 原始套接字(SOCK_RAW)：主要用于一些协议的开发，可以进行比较底层的操作。

创建一个新的套接字

函数名称	socket
函数功能	创建一个新的socket
头文件	#include <sys/types.h> #include <sys/socket.h>
函数原型	int socket(int domain, int type, int protocol);
参数	domain：　　socket的通信domain； type：　　指定socket的类型； protocol：　　与该套接字一起使用的特定协议。
返回值	0：　　成功； -1：　　失败。

绑定地址

函数名称	bind
函数功能	将一个socket绑定到一个地址上，使得客户端可以连接
头文件	#include <sys/socket.h>
函数原型	int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
参数	sockfd：上一个socket()系统调用中获得的文件描述符； addr：指向包含绑定地址的结构的指针； addrlen：地址结构的大小。
返回值	0：成功； -1：失败。

代码	说明
EBADF	文件描述符无效
ENOTSOCK	文件描述符代表的不是一个套接字
EINVAL	文件描述符是一个已经命名的套接字
EADDRNOTAVAIL	地址不可用
EADDRINUSE	地址已经绑定了一个套接字

监听请求

函数名称	listen
函数功能	在一个监听socket上接受一个连接，并返回对等的socket地址
头文件	#include <sys/socket.h>
函数原型	int listen(int sockfd, int backlog);
参数	sockfd：　　socket系统调用中获得的文件描述符； backlog：　　sockfd挂起的已连接队列可以增长的最大长度。
返回值	0：　　成功； -1：　　失败。

接受连接

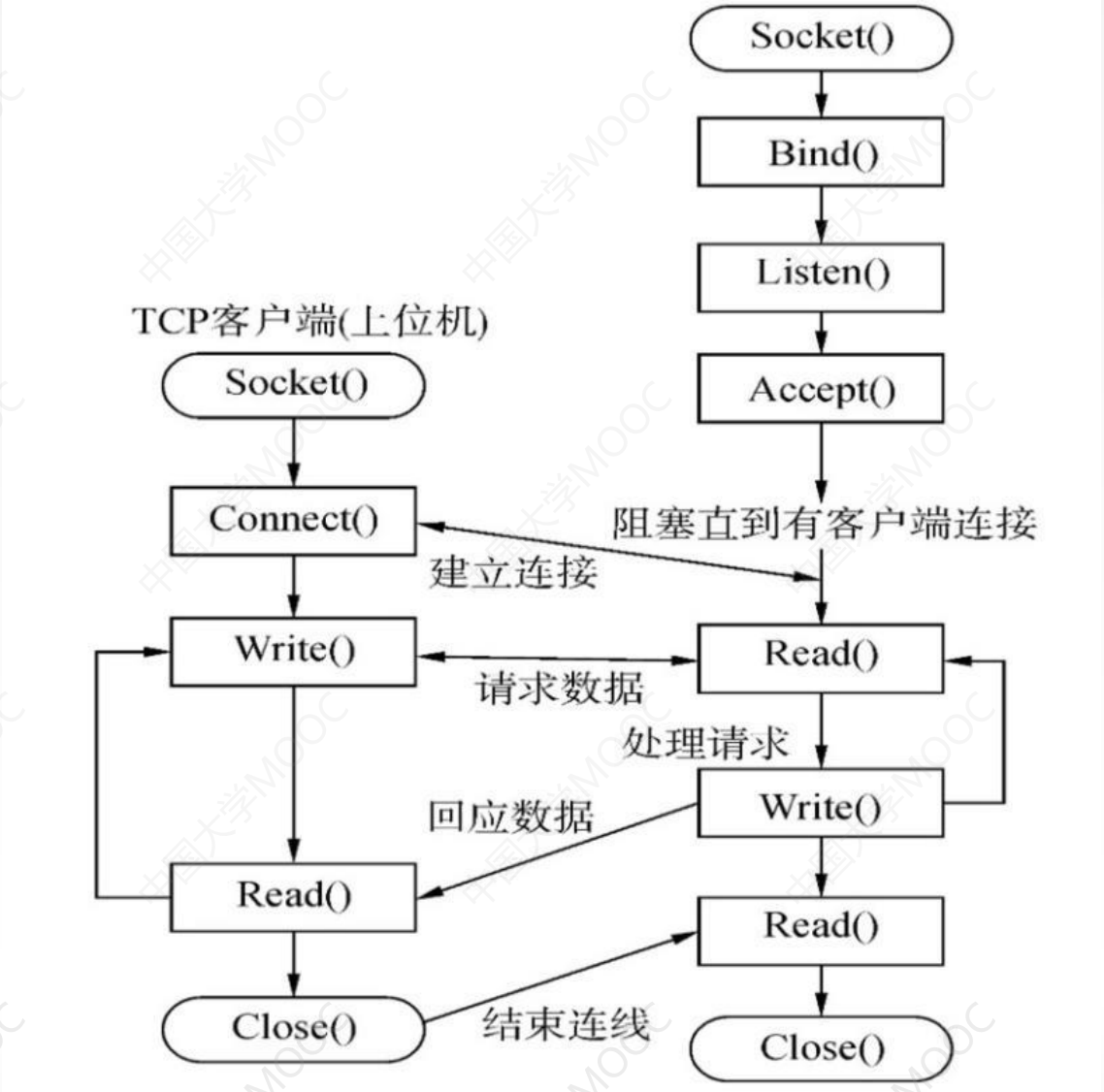
函数名称	accept
函数功能	允许一个socket接受来自其他socket的接入连接
头文件	#include <sys/socket.h>
函数原型	int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
参数	sockfd：　　socket()系统调用中获得的文件描述符； addr：　　指向保存连接对端的地址，即客户端的地址； addrlen：　　指定地址结构的大小。
返回值	0：　　成功； -1：　　失败。

建立客户端连接

函数名称	connect
函数功能	建立与另一个socket之间的连接
头文件	#include <sys/socket.h>
函数原型	int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
参数	sockfd：socket系统调用中获得的文件描述符； addr： 指向的是服务器的地址； addrlen：指定地址结构的大小。
返回值	0： 成功； -1： 失败。

代码	说明
EBADF	文件描述符无效
EALREADY	套接字上已经有了一个正在使用的连接
ETIMEDOUT	连接超时
ECONNREFUSED	连接请求被服务器拒绝

socket通信流程


Socket()
Bind()
Listen()
Accept()
TCP客户端(上位机)
Socket()
Connect()
阻塞直到有客户端连接
建立连接
Write()
Read()
请求数据
处理请求
Write()
回应数据
Read()
Read()
Close()
结束连线
Close()

示例程序——服务器端


```
#define BACKLOG 1
#define MAXRECVLEN 1024

int main(int argc, char *argv[])
{ char buf[MAXRECVLEN];
  int listenfd, connectfd;
  struct sockaddr_in server;
  struct sockaddr_in client;
  socklen_t addrlen;
   char *ip = argv[1];
  int port = atoi(argv[2]);
 if (argc != 3)
 {  printf("argument error, please input ip and port\n");
   exit(1);
}
```

```
if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
 {  perror("socket() error. Failed to initiate a socket");
    exit(1);
}

bzero(&server, sizeof(server));
server.sin_family = AF_INET;
server.sin_port = htons(port);
inet_pton(AF_INET, ip, &server.sin_addr);

if (bind(listenfd, (struct sockaddr *) &server, sizeof(server)) == -1)
 {
perror("Bind() error.");
exit(1);
}
```

示例程序——服务器端

```
if (listen(listenfd, BACKLOG) == -1)
 {
perror("listen() error. \n");
exit(1);
}

addrlen = sizeof(client);
if ((connectfd = accept(listenfd,
          (struct sockaddr *) &client, &addrlen)) == -1)
{
  perror("accept() error. \n");
  exit(1);
}
```

```
bzero(buf, MAXRECVLEN);
int ret = recv(connectfd, buf, MAXRECVLEN, 0);
if (ret > 0)
     printf("%s", buf);
else
    close(connectfd);

send(connectfd, buf, ret, 0);
close(listenfd);

  return 0;
}
```

示例程序——客户端


```
#define MAXDATASIZE 100
int main(int argc, char *argv[])
{
int sockfd;
int num;
char buf[MAXDATASIZE];
struct sockaddr_in server;

char *ip = argv[1];
int port = atoi(argv[2]);

if (argc != 3)
 { printf("argument error, please input ip and port\n");
   exit(1);
}
```

```
if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1)
{
printf("socket() error\n");
 exit(1);
}

bzero(&server, sizeof(server));
server.sin_family = AF_INET;
server.sin_port = htons(port);
inet_pton(AF_INET, ip, &server.sin_addr);

if (connect(sockfd, (struct sockaddr *) &server, sizeof(server)) == -1)
{    printf("connect() error\n");
       exit(1);
}
```

示例程序——客户端

```
char *str = "hello world\n";
if ((num = send(sockfd, str, strlen(str), 0)) == -1)
 {  printf("send() error\n");
    exit(1);
}

if ((num = recv(sockfd, buf, MAXDATASIZE, 0)) == -1)
 {    printf("recv() error\n");
      exit(1);
}
buf[num - 1] = '\0';
printf("recv message: %s\n", buf);
close(sockfd);
return 0;
}
```

谢谢大家！