Socket编程实例

TCP C/S

```
include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define MAXLINE 80                                  //×î´óÊý¾Ý³¤¶È
#define SERV_PORT 6666                              //·þÎñÆ÷¶Ë¿ÚºÅ
int main(void)
{
    struct sockaddr_in servaddr, cliaddr;       //¶¨Òå·þÎñÆ÷Óë¿Í»§¶ËµØÖ·½á¹¹Ìå
    socklen_t cliaddr_len;                          //¿Í»§¶ËµØÖ·³¤¶È
    int listenfd, connfd;
    char buf[MAXLINE];
    char str[INET_ADDRSTRLEN];
    int i, n;
    //socket()
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    bzero(&servaddr, sizeof(servaddr));         //½«·þÎñÆ÷¶ËµØÖ·ÇåÁã
    servaddr.sin_family = AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);
    //bind()
    bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
    //listen()
    listen(listenfd, 20);
    printf("Accepting connections ...\n");
    while (1) {
        cliaddr_len = sizeof(cliaddr);
```

```
            //accept()
             connfd = accept(listenfd, (struct sockaddr *)&cliaddr, &cliaddr_len);
            //recv()
             n = recv(connfd, buf, MAXLINE, 0);
             printf("received from %s at PORT %d\n",
                    inet_ntop(AF_INET, &cliaddr.sin_addr, str, sizeof(str)),
                    ntohs(cliaddr.sin_port));
             for (i = 0; i < n; i++)
                    buf[i] = toupper(buf[i]);
            //send()
             send(connfd, buf, n, 0);
             //¹Ø±ÕÁ¬½Ó
             close(connfd);
      }
      return 0;
}



#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define MAXLINE 80
#define SERV_PORT 6666
int main(int argc, char *argv[])
{
      struct sockaddr_in servaddr;          //¶¨Òå·þÎñÆ÷µØÖ·½á¹¹Ìå
      char buf[MAXLINE];
      int sockfd, n;
      char *str;
      if (argc != 2) {
             fputs("usage: ./client message\n", stderr);
             exit(1);
      }
      str = argv[1];
```

```
        //socket()
        sockfd = socket(AF_INET, SOCK_STREAM, 0);
        bzero(&servaddr, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        inet_pton(AF_INET, "127.0.0.1", &servaddr.sin_addr);
        servaddr.sin_port = htons(SERV_PORT);
        //connect()
        connect(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
        //send()
        send(sockfd, str, strlen(str), 0);
        //recv()
        n = recv(sockfd, buf, MAXLINE, 0);
        printf("Response from server:\n");
        write(STDOUT_FILENO, buf, n);
        close(sockfd);
        return 0;
}
```

UDP C/S

```
#include <string.h>
#include <netinet/in.h>
#include <stdio.h>
#include <unistd.h>
#include <strings.h>
#include <arpa/inet.h>
#include <ctype.h>
#define MAXLINE 80                                  //×î´óÊý¾Ý³¤¶È
#define SERV_PORT 6666                              //·þÎñÆ÷¶Ë¿ÚºÅ
int main(void)
{
        struct sockaddr_in servaddr, cliaddr;       //¶¨Òå·þÎñÆ÷Óë¿Í»§¶ËµØÖ·½á¹¹Ìå
        socklen_t cliaddr_len;                      //¿Í»§¶ËµØÖ·³¤¶È
        int sockfd;                                 //·þÎñÆ÷socketÎÄ¼þÃèÊö·û
        char buf[MAXLINE];
```

```
    char str[INET_ADDRSTRLEN];
    int i, n;
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);//´´½¨·þÎñÆ÷¶ËÌ×½Ó×ÖÎÄ¼þ
    //³õÊ¼»¯·þÎñÆ÷¶Ë¿ÚµØÖ·
    bzero(&servaddr, sizeof(servaddr));        //µØÖ·½á¹¹ÌåÇåÁã
    servaddr.sin_family = AF_INET;                    //Ö¸¶¨ÐÒé×å
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port = htons(SERV_PORT);       //Ö¸¶¨¶Ë¿ÚºÅ
    //°ó¶¨·þÎñÆ÷¶Ë¿ÚµØÖ·
    bind(sockfd, (struct sockaddr *)&servaddr, sizeof(servaddr));
    printf("Accepting connections ...\n");
    //Êý¾Ý´«Êä
    while (1) {
        cliaddr_len = sizeof(cliaddr);
        //½ÓÊÕÊý¾Ý
        n = recvfrom(sockfd, buf, MAXLINE, 0, (struct sockaddr*)&cliaddr,
            &cliaddr_len);
        if (n == -1)
            perror("recvfrom error");
        printf("received from %s at PORT %d\n",
            inet_ntop(AF_INET, &cliaddr.sin_addr, str, sizeof(str)),
            ntohs(cliaddr.sin_port));
        //·þÎñÆ÷¶Ë²Ù×÷£¬Ð¡´ª´óÐ´
        for (i = 0; i < n; i++)
            buf[i] = toupper(buf[i]);
        n = sendto(sockfd, buf, n, 0, (struct sockaddr *)&cliaddr,
            sizeof(cliaddr));
        if (n == -1)
            perror("sendto error");
    }
    close(sockfd);
    return 0;
}
#include <stdio.h>
```

```
#include <string.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <strings.h>
#include <ctype.h>
#define MAXLINE 80
#define SERV_PORT 6666
int main(int argc, char *argv[])
{
	struct sockaddr_in servaddr;
	int sockfd, n;
	char buf[MAXLINE];
	sockfd = socket(AF_INET, SOCK_DGRAM, 0);
	bzero(&servaddr, sizeof(servaddr));
	servaddr.sin_family = AF_INET;
	inet_pton(AF_INET, "127.0.0.1", &servaddr.sin_addr);
	servaddr.sin_port = htons(SERV_PORT);
	//·¢ËÍÊý¾ÝµÍ½¿Í»§¶Ë
	while (fgets(buf, MAXLINE, stdin) != NULL) {
		n = sendto(sockfd, buf, strlen(buf), 0, (struct sockaddr *)&servaddr,
			sizeof(servaddr));
		if (n == -1)
			perror("sendto error");
		//½ÓÊÕ¿Í»§¶Ë·µ»ØµÄÊý¾Ý
		n = recvfrom(sockfd, buf, MAXLINE, 0, NULL, 0);
		if (n == -1)
			perror("recvfrom error");
		//½«½ÓÊÕµ½µÄÊý¾Ý´òÓ¡µ½ÖÕ¶Ë
		send(STDOUT_FILENO, buf, n, 0);
	}
	close(sockfd);
	return 0;
}
```

DM C/S

```
#include <stdlib.h>
#include <stdio.h>
#include <stddef.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <errno.h>
#define QLEN 10
//´´½¨·þÎñÆ÷½ø³Ì£¬³É¹¦·µ»Ø0£¬³ö´í·µ»ØÐ¡ÓÚ0µÄerrno
int serv_listen(const char *name)
{
	int fd, len, err, rval;
	struct sockaddr_un un;
	//´´½¨±¾µØdomainÌ×½Ó×Ö
	if ((fd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
		return(-1);
	//É¾³ýÌ×½Ó×ÖÎÄ¼þ£¬ÜÃâÒòÎÄ¼þ´æÔÚµ¼ÖÂbind()°ó¶¨Ê§°Ü
	unlink(name);
	//³õÊ¼»¯Ì×½Ó×Ö½á¹¹ÌåµØÖ·
	memset(&un, 0, sizeof(un));
	un.sun_family = AF_UNIX;
	strcpy(un.sun_path, name);
	len = offsetof(struct sockaddr_un, sun_path) + strlen(name);
	if (bind(fd, (struct sockaddr *)&un, len) < 0) {
		rval = -2;
		goto errout;
	}
	if (listen(fd, QLEN) < 0) { //¸æÖªÄÚºËÕâÊÇÒ»¸ö·þÎñ½ø³Ì
		rval = -3;
		goto errout;
```

```
        }
        return(fd);
errout:
        err = errno;
        close(fd);
        errno = err;
        return(rval);
}
int serv_accept(int listenfd, uid_t *uidptr)
{
        int clifd, len, err, rval;
        time_t staletime;
        struct sockaddr_un un;
        struct stat statbuf;
        len = sizeof(un);
        if ((clifd = accept(listenfd, (struct sockaddr *)&un, &len)) < 0)
                return(-1);
        //´Óµ÷ÓÃµØÖ·»ñÈ¡¿Í»§¶ËµÄuid
        len -= offsetof(struct sockaddr_un, sun_path); //»ñÈ¡Â·¾¶ÃûµÄ
        un.sun_path[len] = 0;       //ÎªÂ·¾¶ÃûÖ·û´®Ìí¼ÓÖÕÖ¹·û
        if (stat(un.sun_path, &statbuf) < 0) {
                rval = -2;
                goto errout;
        }
        if (S_ISSOCK(statbuf.st_mode) == 0) {
                rval = -3;                      //Èô·µ»ØÖµÎª-3£¬ËµÃ÷Õâ²»ÊÇÒ»¸ösocketÎÄ¼þ
                goto errout;
        }
        if (uidptr != NULL)
                *uidptr = statbuf.st_uid;           //·µ»ØuidµÄµ÷ÓÃÕßÖ¸Õë
        //µ½´Ë³É¹¦»ñÈ¡Â·¾¶Ãû
        unlink(un.sun_path);
        return(clifd);
errout:
```

```
        err = errno;
        close(clifd);
        errno = err;
        return(rval);
}
int main(void)
{
        int lfd, cfd, n, i;
        uid_t cuid;
        char buf[1024];
        lfd = serv_listen("foo.socket");
        if (lfd < 0) {
                switch (lfd) {
                case -3:perror("listen"); break;
                case -2:perror("bind"); break;
                case -1:perror("socket"); break;
                }
                exit(-1);
        }
        cfd = serv_accept(lfd, &cuid);
        if (cfd < 0) {
                switch (cfd) {
                case -3:perror("not a socket"); break;
                case -2:perror("a bad filename"); break;
                case -1:perror("accept"); break;
                }
                exit(-1);
        }
        while (1) {
        r_again:
                n = read(cfd, buf, 1024);
                if (n == -1) {
                        if (errno == EINTR)
                                goto r_again;
```

```
            }
            else if (n == 0) {
                printf("the other side has been closed.\n");
                break;
            }
            for (i = 0; i < n; i++)
                buf[i] = toupper(buf[i]);
            write(cfd, buf, n);
        }
        close(cfd);
        close(lfd);
        return 0;
}

#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <errno.h>
#define CLI_PATH "/var/tmp/" /* +5 for pid = 14 chars */
//´´½¨¿Í»§¶Ë½ø³Ì£¬³É¹¦·µ»Ø0£¬³ö´í·µ»ØÐ¡ÓÚ0µÄerrno
int cli_conn(const char *name)
{
        int fd, len, err, rval;
        struct sockaddr_un un;
        //´´½¨±¾µØÌ×½Ó×Ödomain
        if ((fd = socket(AF_UNIX, SOCK_STREAM, 0)) < 0)
                return(-1);
        //Ê¹ÓÃ×Ô¶¨ÒåµØÖ·Ìî³äsocketµØÖ·½á¹¹Ìå
        memset(&un, 0, sizeof(un));
```

```
        un.sun_family = AF_UNIX;
        sprintf(un.sun_path, "%s%05d", CLI_PATH, getpid());
        len = offsetof(struct sockaddr_un, sun_path) + strlen(un.sun_path);
        unlink(un.sun_path); //±ÜÃâÒòÎÄ¼þÒÑ´æÔÚµ¼ÖÂµÄbind()Ê§°Ü
        if (bind(fd, (struct sockaddr *)&un, len) < 0) {
                rval = -2;
                goto errout;
        }
        //Ê¹ÓÃ·þÎñÆ÷½ø³ÌµØÖ·ÌîäsocketµØÖ·½á¹¹Ìå
        memset(&un, 0, sizeof(un));
        un.sun_family = AF_UNIX;
        strcpy(un.sun_path, name);
        len = offsetof(struct sockaddr_un, sun_path) + strlen(name);
        if (connect(fd, (struct sockaddr *)&un, len) < 0) {
                rval = -4;
                goto errout;
        }
        return(fd);
errout:
        err = errno;
        close(fd);
        errno = err;
        return(rval);
}
int main(void)
{
        int fd, n;
        char buf[1024];
        fd = cli_conn("foo.socket");            //Ì×½Ó×ÖÎÄ¼þÎªfoo.socket
        if (fd < 0) {                                   //ÈÝ´í´¦Àí
                switch (fd) {
                case -4:perror("connect"); break;
                case -3:perror("listen"); break;
                case -2:perror("bind"); break;
```

```
            case -1:perror("socket"); break;
            }
            exit(-1);
        }
        while (fgets(buf, sizeof(buf), stdin) != NULL) {
            write(fd, buf, strlen(buf));
            n = read(fd, buf, sizeof(buf));
            write(STDOUT_FILENO, buf, n);
        }
        close(fd);
        return 0;
}
```