Copy code

```
//TCP Day Time Client

#include "unp.h"

int main(int argc, char **argv) {
    if (argc != 2)
        err_quit("usage: a.out <IPaddress>");

    char receive[30];

    int sockfd = Socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in serverAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    Inet_pton(AF_INET, argv[1], &serverAddress.sin_addr);
    serverAddress.sin_port = htons(13);

    Connect(sockfd, (SA *)&serverAddress, sizeof(serverAddress));

    Recv(sockfd, receive, 29, 0);
    receive[30] = '\0';

    printf("%s", receive);
}
```

Copy code

```
//TCP Day Time Server

#include <time.h>
#include "unp.h"

int main(int argc, char **argv) {
    int listenfd = Socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in serverAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = INADDR_ANY;
    serverAddress.sin_port = htons(13);

    Bind(listenfd, (SA *)&serverAddress, sizeof(serverAddress));

    Listen(listenfd, LISTENQ);

    for (;;) {
        int client = Accept(listenfd, NULL, NULL);
        time_t ticks = time(NULL);
        Send(client, ctime(&ticks), 30, 0);
        Close(client);
    }
}
```

Copy code

```
//Determine Host Byte Order

#include "unp.h"

int main() {
    uint32_t num = 0x01234567;
    uint8_t *n = (uint8_t *) &num;

    if (*n == 0x67)
        printf("Little Endian\n");
    else if (*n == 0x01)
        printf("Big Endian\n");
    else
        printf("Unknown byte order\n");

    return 0;
}
```

Copy code

```
//TCP Concurrent Server

int listenfd = Socket(...);

/* fill in sockaddr_in{} with server's well-known port */

Bind(listenfd, ...);

Listen(listenfd, LISTENQ);

for (;;) {
    int client = Accept(listenfd, ...);

    if (Fork() == 0) {
        Close(listenfd);
        doit(client);
        Close(client);
        exit(0);
    }

    Close(client);
}
```

Copy code

```c
//Address Family of a Socket

#include "unp.h"

int sockfd_to_family(int sockfd) {
    struct sockaddr_storage ss;
    socklen_t len = sizeof(ss);

    Getsockname(sockfd, (SA *)&ss, &len);

    return (ss.ss_family);
}
```

Copy code

```
//TCP Echo Client : main()

#include "unp.h"

int main(int argc, char **argv) {
    if (argc != 2)
        err_quit("usage: tcpcli <IPaddress>");

    int sockfd = Socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in serverAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(SERV_PORT);
    Inet_pton(AF_INET, argv[1], &serverAddress.sin_addr);

    Connect(sockfd, (SA *)&serverAddress, sizeof(serverAddress));

    str_cli(stdin, sockfd);
}
```

Copy code

```
//TCP Echo Client : str_cli()

#include "unp.h"

void str_cli(FILE *fp, int sockfd) {
    char sendline[MAXLINE], recvline[MAXLINE];

    while (Fgets(sendline, MAXLINE, fp) != NULL) {
        Writen(sockfd, sendline, strlen (sendline));
        Readline(sockfd, recvline, MAXLINE);
        Fputs(recvline, stdout);
    }
}
```

Copy code

```
//TCP Echo Server : main()

#include "unp.h"

int main(int argc, char **argv) {
    int listenfd = Socket(AF_INET, SOCK_STREAM, 0);

    struct sockaddr_in clientAddress, serverAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddress.sin_port = htons(SERV_PORT);

    Bind(listenfd, (SA *)&serverAddress, sizeof(serverAddress));

    Listen(listenfd, LISTENQ);

    for (;;) {
        socklen_t clilen = sizeof(clientAddress);
        int client = Accept(listenfd, (SA *)&clientAddress, &clilen);
        str_echo(client);
        Close(client);
    }
}
```

Copy code

```
//TCP Echo Server : str_echo()

#include "unp.h"

void str_echo(int sockfd) {
    ssize_t n;
    char buf[MAXLINE];

    while ((n = read(sockfd, buf, MAXLINE)) > 0) {
        Writen(sockfd, buf, n);
    }
}
```

Copy code

```
// UDP Echo Client : main()

#include "unp.h"

int main(int argc, char **argv) {
    if (argc != 2)
        err_quit("usage: udpcli <IPaddress>");

    int sockfd = Socket(AF_INET, SOCK_DGRAM, 0);

    struct sockaddr_in serverAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(SERV_PORT);
    Inet_pton(AF_INET, argv[1], &serverAddress.sin_addr);

    dg_cli(stdin, sockfd, (SA *)&serverAddress, sizeof(serverAddress));
}
```

Copy code

```c
// UDP Echo Client : dg_cli()
#include "unp.h"

void dg_cli(FILE *fp, int sockfd, const SA *serverAddress, socklen_t servlen) {
    char sendline[MAXLINE], recvline[MAXLINE + 1];

    while (Fgets(sendline, MAXLINE, fp) != NULL) {
        Sendto(sockfd, sendline, strlen(sendline), 0, serverAddress, servlen);
        int n = Recvfrom(sockfd, recvline, MAXLINE, 0, NULL, NULL);
        recvline[n] = '\0';
        printf("%s\n", recvline);
    }
}
```

Copy code

```
//UDP Echo Server : main()

#include "unp.h"

int main(int argc, char **argv) {
    int sockfd = Socket(AF_INET, SOCK_DGRAM, 0);

    struct sockaddr_in serverAddress, clientAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddress.sin_port = htons(SERV_PORT);

    Bind(sockfd, (SA *)&serverAddress, sizeof(serverAddress));

    dg_echo(sockfd, (SA *)&clientAddress, sizeof(clientAddress));
}
```

Copy code

```
//UDP Echo Server : dg_echo()

#include "unp.h"

void dg_echo(int sockfd, SA *pcliaddr, socklen_t clilen) {
    char mesg[MAXLINE];

    for (;;) {
        socklen_t len = clilen;
        int n = Recvfrom(sockfd, mesg, MAXLINE, 0, pcliaddr, &len);
        Sendto(sockfd, mesg, n, 0, pcliaddr, len);
    }
}
```

Copy code

```
//To write a fixed number of datagrams to the server

#include "unp.h"
#define NDG 2000
#define DGLEN 1400

void dg_cli(FILE *fp, int sockfd, const SA *pservaddr, socklen_t servlen) {
    char sendline[DGLEN];
    for (int i = 0; i < NDG; i++) {
        Sendto(sockfd, sendline, DGLEN, 0, pservaddr, servlen);
    }
}
```

Copy code

```
//To count the received datagrams

#include "unp.h"

static void recvfrom_int(int);
static int count;

void dg_echo(int sockfd, SA *pcliaddr, socklen_t clilen) {
    char mesg[MAXLINE];
    Signal(SIGINT, recvfrom_int);

    for (;;) {
        socklen_t len = clilen;
        Recvfrom(sockfd, mesg, MAXLINE, 0, pcliaddr, &len);
        count++;
    }
}

static void recvfrom_int(int signo) {
    printf("\nReceived %d datagrams\n", count);
    exit(0);
}
```

Copy code

```
//Increasing the size of the socket receive queue

#include "unp.h"

void dg_echo(int sockfd, SA *pcliaddr, socklen_t clilen) {

    int receiveQueueSize = 256 * 1024; // 256KB
    Setsockopt(sockfd, SOL_SOCKET, SO_RCVBUF, &receiveQueueSize,
sizeof(receiveQueueSize));

    //// rest of the code
}
```

Copy code

```c
// UDP program that uses connect to determine outgoing interface

#include "unp.h"

int main(int argc, char **argv) {
    if (argc != 2)
        err_quit("usage: udpcli <IPaddress>");

    int sockfd = Socket(AF_INET, SOCK_DGRAM, 0);

    struct sockaddr_in serverAddress, clientAddress;
    bzero(&serverAddress, sizeof(serverAddress));
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_port = htons(SERV_PORT);
    Inet_pton(AF_INET, argv[1], &serverAddress.sin_addr);

    Connect(sockfd, (SA *)&serverAddress, sizeof(serverAddress));

    socklen_t len = sizeof(clientAddress);
    Getsockname(sockfd, (SA *)&clientAddress, &len);
    printf("Outgoing interface IP: %s\n", Sock_ntop((SA *)&clientAddress, len));

    // rest of the code
}
```

Copy code

```
// daemon_inetd function: daemonizes process run by inetd

#include "unp.h"
#include <syslog.h>

extern int daemon_proc; /* defined in error.c */

void daemon_inetd(const char *pname, int facility) {
    daemon_proc = 1; /* for our err_XXX() functions */
    openlog(pname, LOG_PID, facility);
}
```