```c
#define TRUE 1

#include <sys/socket.h>
#include <sys/types.h>
#include <sys/signal.h>

#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>

#include <netdb.h>
#include <errno.h>
#include <stdio.h>


#define DEFDATALEN      56
#define MAXIPLEN    60
#define MAXICMPLEN  76

static void ping(const char *host);

/* common routines */
static int in_cksum(unsigned short *buf, int sz)
{
    int nleft = sz;
    int sum = 0;
    unsigned short *w = buf;
    unsigned short ans = 0;

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1) {
        *(unsigned char *) (&ans) = *(unsigned char *) w;
        sum += ans;
    }

    sum = (sum >> 16) + (sum & 0xFFFF);
    sum += (sum >> 16);
    ans = ~sum;
    return (ans);
}

/* simple version */
static const char *ping_usage = "ping host\n"
    "Send ICMP ECHO_REQUEST packets to network hosts\n"
    ;

static char *hostname = NULL;

static void noresp(int ign)
{
    printf("No response from %s\n", hostname);
    exit(0);
}

static void ping(const char *host)
{
    struct hostent *h;
    struct sockaddr_in pingaddr;
```

```
    struct icmp *pkt;
    int pingsock, c;
    char packet[DEFDATALEN + MAXIPLEN + MAXICMPLEN];

    if ((pingsock = socket(AF_INET, SOCK_RAW, 1)) < 0) {    /* 1 == ICMP */
        perror("ping: creating a raw socket");
        exit(1);
    }
#void * memset ( void * ptr, int value, size_t num );
#Fill block of memory
#Sets the first num bytes of the block of memory pointed by ptr to the specified
    memset(&pingaddr, 0, sizeof(struct sockaddr_in));

    pingaddr.sin_family = AF_INET;
    if (!(h = gethostbyname(host))) {
        fprintf(stderr, "ping: unknown host %s\n", host);
        exit(1);
    }
#void * memcpy ( void * destination, const void * source, size_t num );
#opy block of memory
#Copies the values of num bytes from the location pointed by source directly to
    memcpy(&pingaddr.sin_addr, h->h_addr, sizeof(pingaddr.sin_addr));
    hostname = h->h_name;

    pkt = (struct icmp *) packet;
    memset(pkt, 0, sizeof(packet));
    pkt->icmp_type = ICMP_ECHO;
    pkt->icmp_cksum = in_cksum((unsigned short *) pkt, sizeof(packet));

    c = sendto(pingsock, packet, sizeof(packet), 0,
               (struct sockaddr *) &pingaddr, sizeof(struct sockaddr_in));

    if (c < 0 || c != sizeof(packet)) {
        if (c < 0)
            perror("ping: sendto");
        fprintf(stderr, "ping: write incomplete\n");
        exit(1);
    }

    signal(SIGALRM, noresp);
    alarm(5);                   /* give the host 5000ms to respond */
    /* listen for replies */
    while (1) {
        struct sockaddr_in from;
        size_t fromlen = sizeof(from);

        if ((c = recvfrom(pingsock, packet, sizeof(packet), 0,
                          (struct sockaddr *) &from, &fromlen)) < 0) {
            if (errno == EINTR)
                continue;
            perror("ping: recvfrom");
            continue;
        }
        if (c >= 76) {          /* ip + icmp */
            struct iphdr *iphdr = (struct iphdr *) packet;

            pkt = (struct icmp *) (packet + (iphdr->ihl << 2)); /* skip ip hdr */
            if (pkt->icmp_type == ICMP_ECHOREPLY)
                break;
        }
    }
    printf("%s is alive!\n", hostname);
```

```
    return;
}

void usage(const char ping_usage[])
{
    printf( ping_usage );
}

int main(int argc, char **argv)
{
    argc--;
    argv++;
    if (argc < 1)
    {
        usage(ping_usage);
        exit( -1 );
    }
    ping(*argv);
    exit(TRUE);
}
```