```c
/*
    Simple udp server
*/

#include<stdio.h> //printf
#include<string.h> //memset
#include<stdlib.h> //exit(0);
#include<arpa/inet.h>
#include<sys/socket.h>

#define BUFLEN 512  //Max length of buffer
#define PORT 8888   //The port on which to listen for incoming data

void die(char *s)
{
    perror(s);
    exit(1);
}

int main(void)
{
    struct sockaddr_in si_me, si_other;

    int s, i, slen = sizeof(si_other) , recv_len;
    char buf[BUFLEN];

    //create a UDP socket
    if ((s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1)
    {
        die("socket");
    }

    // zero out the structure
    memset((char *) &si_me, 0, sizeof(si_me));

    si_me.sin_family = AF_INET;
    si_me.sin_port = htons(PORT);
    si_me.sin_addr.s_addr = htonl(INADDR_ANY);

    //bind socket to port
    if( bind(s , (struct sockaddr*)&si_me, sizeof(si_me) ) == -1)
    {
        die("bind");
    }

    //keep listening for data
    while(1)
    {
        printf("Waiting for data...");
        fflush(stdout);

        //try to receive some data, this is a blocking call
        if ((recv_len = recvfrom(s, buf, BUFLEN, 0, (struct sockaddr *) &si_other, &slen)) == -1)
        {
            die("recvfrom()");
        }
         buf[recv_len-1] = '\0';
        //print details of the client/peer and the data received
        printf("Received packet from %s:%d\n", inet_ntoa(si_other.sin_addr), ntohs(si_other.sin_port));
        printf("Data: %s\n" , buf);

        //now reply the client with the same data
        if (sendto(s, buf, recv_len, 0, (struct sockaddr*) &si_other, slen) == -1)
        {
            die("sendto()");
        }
    }

    close(s);
    return 0;
}
```