```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include "Practical.h"
#include <unistd.h>

#include <sys/stat.h> //NOTE THE INCLUSION OF A NEW HEADER FILE

static const int MAXPENDING = 5; // Maximum outstanding connection requests

int main(int argc, char *argv[]) {
int numBytes = 0, char_in, count = 0, size = 0; // VARIABLES FOR FILE MANIPULATION
char recvbuffer[BUFSIZE], sendbuffer[BUFSIZE], path[200] ={'.'}, discard1[50], discard2[50]; // BUFFERS
struct stat st;  //STRUCTURE REQUIRED TO HOLD OPEN FILE ATTRIBUTES
FILE * hFile;   //FILE POINTER REQUIRED TO OPEN FILE
    .
    .
    .
    .
    .

    sscanf(recvbuffer, "%s %s %s", discard1, (path+1), discard2);  //NOTE THE SECOND ELEMENT OF PATH IS REFERENCED

    if(strcmp(path, "./favicon.ico") == 0) //CHECK IF REQUEST IS FOR FAVICON - IGNORE AND CONTINUE TO NEXT ITERATION
    {
        printf("\n\nFound favicon.ico\n\n");
        close(clntSock); // Close client socket
        continue;
    }


    if(strcmp(path, "./") == 0)  //CHECK WHAT IS IN PATH
        {

        IF ./ REPLACE WITH HOME PAGE FILE NAME

        strcpy(path, "./index.html");

        }

        hFile = fopen(path, "r");   //ATTEMPTING TO OPEN FILE IN PATH

        if (hFile == NULL)          //IF REQUESTED FILE DOES NOT EXIST
                {
                THIS IF SECTION ASSUMES REQUESTED FILE DOES NOT EXIST

                OPEN THE ERROR PAGE

                strcpy(path, "./error.html");
                stat(filename, &st);
                size = st.st_size;  //RETRIEVING FILE SIZE OF OPEN FILE

                STORE NEGATIVE HTTP HEADERS (404 RESPONSE) IN OUTGOING BUFFER

                }
        else
                {

                THIS ELSE SECTION ASSUMES REQUESTED FILE EXISTS AND IS OPEN

                stat(path, &st);
                size = st.st_size;   //RETRIEVING FILE SIZE OF OPEN FILE

                STORE POSITIVE HTTP HEADERS (200 RESPONSE) IN OUTGOING BUFFER
                }

                SEND HTTP HEADERS TO CONNECTED SOCKET
                RESET OUTGOING BUFFER

        while((char_in = fgetc(hFile))!= EOF)   //READING CONTENTS OF FILE
        CHARACTER-BY-CHARACTER
        {
        sendbuffer[count] = char_in;    //STORING EACH CHARACTER IN OUTGOING BUFFER
        count++;
        }

        sendbuffer[count] = '\0';   //NULL TERMINATE FILE CONTENTS


        SEND FILE CONTENTS  TO CONNECTED SOCKET

        RESET ALL VARIABLES AND BUFFERS (INCLUDING send/recvbuffers), CLOSE FILE AND
        CONNECTED SOCKET

   } //END FOR LOOP

}// END MAIN

```