

Program 1 : Scanning WiFi Hotspots using C.

Objective:

Write a program to list all the Wi-Fi hotspots available and give total count of them using C programming.

Modules Covered :

- Identifies the Wi-Fi hot-spots in its wireless range.
- Gives a counts of Wi-Fi hotspots.

```
#include "include/interface_helper.c"
#include "include/wifi_scan.h"
#include <stdio.h> // printf, scanf
#include <string.h>

// function to convert bssid to printable hardware mac address
const char *bssid_to_string(const uint8_t bssid[BSSID_LENGTH], char
bssid_string[BSSID_STRING_LENGTH])
{
    snprintf(bssid_string, BSSID_STRING_LENGTH, "%02x:%02x:%02x:%02x:%02x:%02x",
        bssid[0], bssid[1], bssid[2], bssid[3], bssid[4], bssid[5]);
    return bssid_string;
}

const int BSS_INFOS = 10; //the maximum amounts of APs (Access Points) we want to store

void Usage(char **argv);

int main(int argc, char **argv)
{
    struct wifi_scan *wifi=NULL; // this stores all the library information
    struct bss_info bss[BSS_INFOS]; // this is where we are going to keep informatoin
    about APs (Access Points)
    char mac[BSSID_STRING_LENGTH]; // a placeholder where we convert BSSID to printable
    hardware mac address
    int status, i, count;

    if(argc != 2) // if the user doesn't know how to use, we display usage instructions
    {
        Usage(argv);
        return 0;
    }

    printf("Triggering scan needs permissions.\n");

    printf("### Close the program with ctrl+c when you're done ###\n\n");

    // initialize the library with network interface argv[1] (e.g. wlan0 / wlp3s0)
    wifi = wifi_scan_init(argv[1]);

    /*
        Using wifi_scan_all function, it returns no. of hotspots
    */
    status = wifi_scan_all(wifi, bss, BSS_INFOS);

    // it may happen that device is unreachable (e.g. the device works in such way that it
    doesn't respond while scanning)
    if(status < 0) {
        perror("Unable to get scan data");
    }
}
```



```

x
MCWC_Program_WIFI_AP: ./wifi_scan_all
+ x ...IFI_AP: ./wifi_scan_all
sidx1024@sidx1024-X450JN:~/Documents/MCWC_Program_WIFI_AP$ gcc wifi_scan_all.c -lmnl -o wifi_scan_all
sidx1024@sidx1024-X450JN:~/Documents/MCWC_Program_WIFI_AP$ sudo ./wifi_scan_all wlp3s0
Triggering scan needs permissions.
### Close the program with ctrl+c when you're done ###

MAC ADDRESS      AP NAME           RSS(dbm)  LAST SEEN  STATUS
cc:d3:1e:50:4a:18  Jio of Siddharth  -58       128        associated
90:8d:78:70:b9:47  nirav             -45       88
f0:d7:aa:aa:20:ac  ADYYTW90b0coNSlQbHVz -58      2656
a8:6b:ad:36:95:d4  R 2 D 2          -40       2632
90:8d:78:cd:da:2f  Bring Beer and get access -89      28972
c8:3a:35:52:00:c8  Don't ask for wifi -86       28324
98:de:d0:bd:1c:62  Don              -88       1676
7c:91:22:8f:04:48  Prohibited       -95       26564
f4:f2:6d:47:46:7c  Prohibited       -84       3944

-----
Total hotspot(s) count: 9
-----
sidx1024@sidx1024-X450JN:~/Documents/MCWC_Program_WIFI_AP$

```

Full code available at:
<https://github.com/siddharth1024/MCWC-WiFi-MiniProject>

Program 2 : File Transfer using C

Objective:

Write a program to transfer a file between two Wi-Fi devices over any available WiFi hotspot, using C programming.

Modules Covered :

- Also performs file transfer between given two Wi-Fi devices –across different Wi-Fi hotspots.

server.c

```
#include <stdio.h>
#include <string.h>
#include <unistd.h> // write(), close(), sleep()
#include <arpa/inet.h> // listen(), accept(), bind(), htons()

struct sockaddr_in c_addr;
char fname[100];

void SendFileToClient(int *arg) {
    int connfd = (int) *arg;
    printf("Connection accepted and id: %d\n", connfd);
    printf("Connected to Client: %s:%d\n", inet_ntoa(c_addr.sin_addr),
    ntohs(c_addr.sin_port));
    write(connfd, fname, 256);

    FILE *fp = fopen(fname, "rb");
    if (fp == NULL) {
        printf("File open error");
    }

    /*Read data from file and send it */
    while (1) {
        /*First read file in chunks of 1024 bytes */
        unsigned char buff[1024] = {
            0
        };
        int nread = fread(buff, 1, 1024, fp);
        printf("Bytes read %d \n", nread);

        /*If read was success, send data. */
        if (nread > 0) {
            printf("Sending \n");
            write(connfd, buff, nread);
        }
        if (nread < 1024) {
            if (feof(fp)) {
                printf("End of file\n");
                printf("File transfer completed for id: %d\n", connfd);
            }
            if (ferror(fp))
                printf("Error reading\n");
            break;
        }
    }
    printf("Closing Connection for id: %d\n", connfd);
    close(connfd);
    shutdown(connfd, SHUT_WR);
    sleep(2);
}
```

```

int main(int argc, char *argv[]) {
    int connfd = 0, err;
    pthread_t tid;
    struct sockaddr_in serv_addr;
    int listenfd = 0, ret;
    char sendBuff[1024];
    int numrv;
    size_t clen = 0, slen = 0;

    /*
        STEP 1 : Open a socket to listen
    */

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenfd < 0) {
        printf("Error in socket creation\n");
        return -1;
    }

    printf("Socket retrieve success\n");

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);
    slen = sizeof(serv_addr);

    /*
        STEP 2 : Binding for socket
    */

    ret = bind(listenfd, (struct sockaddr *) &serv_addr, slen);
    if (ret < 0) {
        printf("Error in bind\n");
        return -1;
    }

    /*
        STEP 3 : Start listening for client requests
    */

    if (listen(listenfd, 10) == -1) {
        printf("Failed to listen\n");
        return -1;
    }

    if (argc < 2) {
        printf("Enter file name to send: ");
        scanf("%s", fname);
    } else
        strcpy(fname, argv[1]);

    /*
        STEP 4 : Accept any new connections, create new thread for each connection. Do
        until user stops it.
    */

    while (1) {
        clen = sizeof(c_addr);
        printf("Waiting...\n");
        connfd = accept(listenfd, (struct sockaddr *) &c_addr, (socklen_t *) &clen);
        if (connfd < 0) {
            printf("Error in accept\n");
            continue;
        }
        err = pthread_create( &tid, NULL, &SendFileToClient, &connfd);
    }
}

```

```

        if (err != 0)
            printf("\nCan't create thread :[%s]", strerror(err));
    }
    /*
     * STEP 5 : Close and release socket.
     */

    close(connfd);
    return 0;
}

```

client.c

```

#include <stdio.h>
#include <string.h>
#include <unistd.h> // read(sockfd, fname, 256);
#include <arpa/inet.h> // inet_xxx, ntohs

int main(int argc, char *argv[]) {

    int sockfd = 0;
    int bytesReceived = 0;

    /*
     * STEP 1 : Create a buffer to store recieved data.
     */

    char recvBuff[1024];

    /*
     * STEP 2 : Clear buffer and fill it with zeros.
     */

    memset(recvBuff, '0', sizeof(recvBuff));
    struct sockaddr_in serv_addr;

    /* Create a socket first */

    /*
     * STEP 3 : Open a socket
     */

    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        printf("\nError : Could not create socket \n");
        return 1;
    }

    /* Initialize sockaddr_in data structure */
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000); // port
    char ip[50];

    /* Ask IP address from user if not entered in command line arguments */
    if (argc < 2) {
        printf("Enter IP address to connect: ");
        scanf("%s", ip);
    }
    else {
        strcpy(ip, argv[1]);
    }

    serv_addr.sin_addr.s_addr = inet_addr(ip);

```

```

/*
    STEP 4 : Attempt connection
*/

/* Attempt a connection */
if (connect(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
    printf("\n Error : Connect Failed \n");
    return 1;
}

printf("Connected to ip: %s : %d\n", inet_ntoa(serv_addr.sin_addr),
ntohs(serv_addr.sin_port));

/* Create file where data will be stored */
FILE *fp;
char fname[100];
read(sockfd, fname, 256);
printf("File Name: %s\n", fname);
printf("Receiving file...");
fp = fopen(fname, "ab");
if (NULL == fp) {
    printf("Error opening file");
    return 1;
}
long double sz = 1;
/*
    STEP 5 : Start receiving data using read() method, 1024 bytes at a time.
    And write buffer to file.
*/

/* Receive data in chunks of 1024 bytes */
while ((bytesReceived = read(sockfd, recvBuff, 1024)) > 0) {
    sz++;
    printf("Received: %Le Mb", (sz / 1024));
    fflush(stdout);
    fwrite(recvBuff, 1, bytesReceived, fp);
}

if (bytesReceived < 0) {
    printf("\n Read Error \n");
}

printf("\nFile OK....Completed\n");
return 0;
}

```

OUTPUT:

```
server: ./server
+ * server: ./server
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket$ cd server
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/server$ gcc server.c -o server -lpthread
server.c: In function 'main':
server.c:101:11: warning: implicit declaration of function 'pthread_create' [-Wimplicit-function-declaration]
    err = pthread_create( &tid, NULL, &SendFileToClient, &connfd);
            ^
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/server$ ./server
Socket retrieve success
Enter file name to send: bird.jpg
Waiting...
Waiting...
Connection accepted and id: 4
Connected to Client: 192.168.15.131:44270
Segmentation fault (core dumped)
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/server$ ./server
Socket retrieve success
Enter file name to send: bird.jpg
Waiting...
Waiting...
Connection accepted and id: 4
Connected to Client: 192.168.15.131:44286
Segmentation fault (core dumped)
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/server$
```

Screenshot : Server Side

```
client: ./client
+ * client: ./client
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/client$ gcc client.c -o client
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/client$ ./client 192.168.15.131

Connected to ip: 192.168.15.131 : 5000
File Name: bird.jpg
Receiving file...
File OK....Completed
sidx1024@sidx1024-X450JN:~/Documents/TCP_C_Socket/client$
```

Screenshot : Client Side

Full code available at:
<https://github.com/siddharth1024/MCWC-WiFi-MiniProject>

Useful Linux Commands for Wi-Fi Hotspot Scanning

Objective:

Use various commands for hotspot discovery

Using iw

```
sudo iw dev wlp3s0 scan | grep SSID
```

Using nmcli

```
nmcli dev wifi
```

Full code available at:

<https://github.com/siddharth1024/MCWC-WiFi-MiniProject>