**20/10/2021 CN LAB 7 2019103573**

**IMPLEMENTING THE DNS RECURSIVE**

**ROOT DNS SERVER**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#define PORT 8041

#define TLDPORT 8042

int main()

{

    int socketfd = 0, tldfd = 0;

    socklen\_t length = sizeof(struct sockaddr\_in);

    struct sockaddr\_in host\_addr, tld\_addr, client\_addr;

    char buffer[64];

    char reqip[30];

    int recvbytes, sentbytes;

    socketfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "Error in socket creation.\n");

        return -1;

    }

    host\_addr.sin\_family = AF\_INET;

    host\_addr.sin\_port = htons(PORT);

    inet\_pton(AF\_INET, "127.0.0.1", &host\_addr.sin\_addr);

    if (bind(socketfd, (struct sockaddr \*)&host\_addr, sizeof(host\_addr)) < 0)

    {

        fprintf(stderr, "Error in binding port to socket.\n");

        return -1;

    }

    fprintf(stdout, " [-] ROOT DNS SERVER PORT : %d\n", PORT);

    while (1)

    {

        printf("\n-------------------------------------------------\n");

        recvbytes = recvfrom(socketfd, buffer, sizeof(buffer), 0,(struct sockaddr\*)

        &client\_addr, &length);

        if (strncmp(buffer, "exit", sizeof("exit")) == 0)

            break;

        fprintf(stdout, " [-] REQUEST FROM CLIENT : %s\n", buffer);

        char domain[6];

        int i = 0, j = 0;

        while (buffer[i++] != '.')

        ;

        while (buffer[i++] != '.')

        ;

        while (buffer[i] != '\0')

        domain[j++] = buffer[i++];

        domain[j] = '\0';

        FILE \*fd = fopen("root.txt", "r");

if (!fd)

        {

            fprintf(stderr, "Could not access DNS records.\n");

            sendto(socketfd, "ERROR", strlen("ERROR") + 1, 0,

            (struct sockaddr \*)&client\_addr, length);

            continue;

        }

        char linebuff[40], filebuff[400], ip[20], tempbuff[40],lastbuff[40];

        char \*temp, \*iptemp;

        int flag = 0;

        linebuff[0] = '\0';

        lastbuff[0] = '\0';

        filebuff[0] = '\0';

        ip[0] = '\0';

        while (fgets(linebuff, sizeof(linebuff), fd))

        {

            strcpy(tempbuff, linebuff);

            temp = strtok(tempbuff, " ");

            if (flag == 0 && strncmp(temp, domain,strlen(domain)) == 0)

            {

                flag = 1;

                strcpy(lastbuff, linebuff);

                iptemp = strtok(NULL, "\n");

                for (i = 0; \*iptemp != '\0'; i++, iptemp++)

                    ip[i] = \*iptemp;

                ip[i] = '\0';

            }

            else

            {

                strcat(filebuff, linebuff);

            }

        }

        fclose(fd);

        if (flag == 0)

        {

            sentbytes = sendto(socketfd, "404", strlen("404") + 1, 0, (struct sockaddr\*)

            &client\_addr, length);

            continue;

        }

        else

        {

            int fdes = open("root.txt", O\_WRONLY);

            strcat(filebuff, lastbuff);

            write(fdes, filebuff, strlen(filebuff));

            close(fdes);

            fprintf(stdout, "IP for TOP LEVEL DOMAIN SERVER of %s: %s\n\n\n ", domain, ip);

        }

        fprintf(stdout, "Querying TOP LEVEL DOMAIN DNS\n");

        //REQUEST TO TLD DNS

        tldfd = socket(AF\_INET, SOCK\_DGRAM, 0);

        if (tldfd < 0)

        {

            fprintf(stderr, "Error in socket creation.\n");

            return -1;

        }

        tld\_addr.sin\_family = AF\_INET;

        tld\_addr.sin\_port = htons(TLDPORT);

        inet\_pton(AF\_INET, "127.0.0.1", &tld\_addr.sin\_addr);

        sentbytes = sendto(tldfd, buffer, strlen(buffer) + 1, 0,(struct sockaddr\*)

        &tld\_addr, length);

        recvbytes = recvfrom(tldfd, reqip, sizeof(reqip), 0, NULL,NULL);

        fprintf(stdout, "Server IP for %s: %s\n >>>> Returning to local DNS\n\n", buffer, reqip);

        close(tldfd);

        sentbytes = sendto(socketfd, reqip, strlen(reqip) + 1, 0,(struct sockaddr\*)

        &client\_addr, length);

    }

    close(socketfd);

    return 0;

}

**TOP LEVEL DOMAIN**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#define PORT 8042

#define AUTHPORT 8043

int main()

{

    int socketfd = 0, authfd = 0;

    socklen\_t length = sizeof(struct sockaddr\_in);

    struct sockaddr\_in host\_addr, auth\_addr, client\_addr;

    char buffer[64];

    char reqip[30];

    int recvbytes, sentbytes;

    socketfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "Error in socket creation.\n");

        return -1;

    }

    host\_addr.sin\_family = AF\_INET;

    host\_addr.sin\_port = htons(PORT);

    inet\_pton(AF\_INET, "127.0.0.1", &host\_addr.sin\_addr);

    if (bind(socketfd, (struct sockaddr \*)&host\_addr, sizeof(host\_addr)) < 0)

    {

        fprintf(stderr, "Error in binding port to socket.\n");

        return -1;

    }

    fprintf(stdout, " [-] TOP LEVEL DOMAIN DNS SERVER PORT : %d\n", PORT);

    while (1)

    {

        printf("\n-------------------------------------------------\n");

        recvbytes = recvfrom(socketfd, buffer, sizeof(buffer), 0,(struct sockaddr\*)

        &client\_addr, &length);

        if (strncmp(buffer, "exit", sizeof("exit")) == 0)

            break;

        fprintf(stdout, " REQUEST FROM CLIENT : %s\n", buffer);

        char domain[20];

        domain[0] = '\0';

        int i = 0, j = 0;

        while (buffer[i++] != '.');

        while (buffer[i] != '\0')

        domain[j++] = buffer[i++];

        domain[j] = '\0';

        fprintf(stdout, " [-] DOMAIN : %s\n", domain);

        FILE \*fd = fopen("tld.txt", "r");

        if (!fd)

        {

            fprintf(stderr, "Could not access DNS records.\n");

            sendto(socketfd, "ERROR", strlen("ERROR") + 1, 0,

            (struct sockaddr \*)&client\_addr, length);

            continue;

        }

        char linebuff[40], filebuff[400], ip[20], tempbuff[40],lastbuff[40];

        char \*temp, \*iptemp;

        int flag = 0;

        linebuff[0] = '\0';

        lastbuff[0] = '\0';

        filebuff[0] = '\0';

        ip[0] = '\0';

        while (fgets(linebuff, sizeof(linebuff), fd))

        {

            strcpy(tempbuff, linebuff);

            temp = strtok(tempbuff, " ");

            if (flag == 0 && strncmp(temp, domain, strlen(domain)) == 0)

            {

                flag = 1;

                strcpy(lastbuff, linebuff);

                iptemp = strtok(NULL, "\n");

                for (i = 0; \*iptemp != '\0'; i++, iptemp++)

                    ip[i] = \*iptemp;

                ip[i] = '\0';

            }

            else

            {

                strcat(filebuff, linebuff);

            }

        }

        fclose(fd);

        if (flag == 0)

        {

            sentbytes = sendto(socketfd, "404", strlen("404") + 1, 0, (struct sockaddr\*)

            &client\_addr, length);

            continue;

        }

        else

        {

            int fdes = open("tld.txt", O\_WRONLY);

            strcat(filebuff, lastbuff);

            write(fdes, filebuff, strlen(filebuff));

            close(fdes);

            fprintf(stdout, " IP for AUTHORITATIVE SERVER of %s: %s\n", domain, ip);

        }

        fprintf(stdout, "Querying AUTH DNS\n");

        //REQUEST TO AUTH DNS

        authfd = socket(AF\_INET, SOCK\_DGRAM, 0);

        if (authfd < 0)

        {

            fprintf(stderr, "Error in socket creation.\n");

            return -1;

        }

        auth\_addr.sin\_family = AF\_INET;

        auth\_addr.sin\_port = htons(AUTHPORT);

        inet\_pton(AF\_INET, "127.0.0.1", &auth\_addr.sin\_addr);

        sentbytes = sendto(authfd, buffer, strlen(buffer) + 1, 0,(struct sockaddr\*)

        &auth\_addr, length);

        recvbytes = recvfrom(authfd, reqip, sizeof(reqip), 0, NULL,NULL);

        fprintf(stdout, "Server IP for %s: %s\n\n >>>> Returning to root DNS...\n\n ", buffer, reqip);

        close(authfd);

        sentbytes = sendto(socketfd, reqip, strlen(reqip) + 1, 0,(struct sockaddr\*)

        &client\_addr, length);

    }

    close(socketfd);

    return 0;

}

**AUTHORIZATION SERVER CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#define IPLOOKUP\_TABLE\_COUNT 4

#define IP\_FOR\_EACH\_DNS\_RECORDS 3

#define PORT 8043

typedef struct

{

    char \*key;

    int value;

} keyValuePairs;

keyValuePairs ip\_lookuptable[] = {

{"www.cricbuzz.com", 0},

{"mail.google.com", 0},

{"cric.cricbuzz.com", 0}};

int rotate\_dns\_ip(char \*domain\_name)

{

    for (int i = 0; i < IPLOOKUP\_TABLE\_COUNT; i++)

    {

        if (strcmp(domain\_name, ip\_lookuptable[i].key) == 0)

        {

            int value = ip\_lookuptable[i].value;

            ip\_lookuptable[i].value++;

            return value;

        }

    }

    return -1;

}

int main()

{

    int socketfd = 0;

    socklen\_t length = sizeof(struct sockaddr\_in);

    struct sockaddr\_in host\_addr, client\_addr;

    char buffer[64];

    char reqip[30];

    int recvbytes, sentbytes;

    socketfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "Error in socket creation.\n");

        return -1;

    }

    host\_addr.sin\_family = AF\_INET;

    host\_addr.sin\_port = htons(PORT);

    inet\_pton(AF\_INET, "127.0.0.1", &host\_addr.sin\_addr);

    if (bind(socketfd, (struct sockaddr \*)&host\_addr, sizeof(host\_addr)) < 0)

    {

        fprintf(stderr, "Error in binding port to socket.\n");

        return -1;

    }

    fprintf(stdout," [-] AUTHORITATIVE DNS SERVER PORT : %d\n",PORT);

    while(1) {

    printf("\n-------------------------------------------------\n");

    recvbytes = recvfrom(socketfd, buffer, sizeof(buffer), 0,(struct sockaddr\*)

    &client\_addr, &length);

    if (strncmp(buffer, "exit", sizeof("exit")) == 0)

        break;

fprintf(stdout, " REQUEST FROM CLIENT : %s\n", buffer);

    FILE \*fd = fopen("auth.txt", "r");

    if (!fd)

    {

        fprintf(stderr, "Could not access DNS records.\n");

        sendto(socketfd, "ERROR", strlen("ERROR") + 1, 0,

        (struct sockaddr \*)&client\_addr, length);

        continue;

    }

    char linebuff[80], filebuff[400], ip[40], tempbuff[80],

    lastbuff[80];

    char \*temp, \*iptemp;

    int flag = 0, i;

    linebuff[0] = '\0';

    lastbuff[0] = '\0';

    filebuff[0] = '\0';

    ip[0] = '\0';

    while (fgets(linebuff, sizeof(linebuff), fd))

    {

        strcpy(tempbuff, linebuff);

        temp = strtok(tempbuff, " ");

        if (flag == 0 && strncmp(temp, buffer, strlen(temp)) == 0)

        {

            flag = 1;

            strcpy(lastbuff, linebuff);

            iptemp = strtok(NULL, " ");

            int counter = 0;

            int curr\_pointer =

            rotate\_dns\_ip(buffer) % IP\_FOR\_EACH\_DNS\_RECORDS;

            int i = 0;

            while (1)

            {

                for (i = 0; \*iptemp != ' ' && \*iptemp !='\0'; i++, iptemp++)

                    ip[i] = \*iptemp;

                if (\*iptemp == '\n' || counter == curr\_pointer)

                    break;

                counter++;

                iptemp = strtok(NULL, " ");

            }

                ip[i] = '\0';

            }

            else

            {

                strcat(filebuff, linebuff);

            }

        }

        fclose(fd);

        if (flag == 0)

        {

            sentbytes = sendto(socketfd, "404", strlen("404") + 1, 0, (struct sockaddr\*)

            &client\_addr, length);

            continue;

        }

        else

        {

            int fdes = open("auth.txt", O\_WRONLY);

            strcat(filebuff, lastbuff);

            write(fdes, filebuff, strlen(filebuff));

            close(fdes);

            fprintf(stdout, " Requested IP is : %s\n >>>> Returning to TLD DNS\n\n ", ip);

            sentbytes = sendto(socketfd, ip, strlen(ip) + 1, 0,(struct sockaddr\*)

            &client\_addr, length);

        }

    }

    close(socketfd);

    return 0;

}

**LOCAL DNS**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <errno.h>

#define ROOTPORT 8041

#define PORT 8044

int main()

{

    int socketfd = 0, localfd = 0;

    int rootfd = 0, tldfd = 0, authfd = 0;

    socklen\_t length = sizeof(struct sockaddr\_in);

    struct sockaddr\_in host\_addr, root\_addr, client\_addr;

    char buffer[512];

    char reqip[30];

    int recvbytes, sentbytes;

    socketfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "Error in socket creation.\n");

        return -1;

    }

    host\_addr.sin\_family = AF\_INET;

    host\_addr.sin\_port = htons(PORT);

    inet\_pton(AF\_INET, "127.0.0.1", &host\_addr.sin\_addr);

    if (bind(socketfd, (struct sockaddr \*)&host\_addr, sizeof(host\_addr)) < 0)

    {

        fprintf(stderr, "Error in binding port to socket.\n");

        return -1;

    }

    fprintf(stdout, " [-] SERVER STARTED AT PORT : %d\n", PORT);

    while (1)

    {

        printf("\n-------------------------------------------------\n");

        recvbytes = recvfrom(socketfd, buffer, sizeof(buffer), 0,

        (struct sockaddr \*)&client\_addr, &length);

        if (strncmp(buffer, "exit", sizeof("exit")) == 0)

            break;

        fprintf(stdout, " [+] REQUEST FROM CLIENT : %s\n", buffer);

        //REQUEST TO ROOT DNS

        rootfd = socket(AF\_INET, SOCK\_DGRAM, 0);

        if (rootfd < 0)

        {

            fprintf(stderr, "Error in socket creation.\n");

            return -1;

        }

        root\_addr.sin\_family = AF\_INET;

        root\_addr.sin\_port = htons(ROOTPORT);

        inet\_pton(AF\_INET, "127.0.0.1", &root\_addr.sin\_addr);

        sentbytes = sendto(rootfd, buffer, strlen(buffer) + 1, 0,

        (struct sockaddr \*)&root\_addr, length);

        recvbytes = recvfrom(rootfd, reqip, sizeof(reqip), 0, NULL, NULL);

        fprintf(stdout, " Server IP for %s: %s\n >>>> Returning dns query results to client...\n\n ",

        buffer, reqip);

        close(rootfd);

        sentbytes = sendto(socketfd, reqip, strlen(reqip) + 1, 0,

        (struct sockaddr \*)&client\_addr, length);

    }

    close(socketfd);

    return 0;

}

**CLIENT**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#define LOCALDNS 8044

int main()

{

    int socketfd = 0, sentbytes, recvbytes;

    struct sockaddr\_in host\_addr;

    char input[20], buffer[20];

    socketfd = socket(AF\_INET, SOCK\_DGRAM, 0);

    if (socketfd < 0)

    {

        fprintf(stderr, "Error in socket creation.\n");

        return -1;

    }

    host\_addr.sin\_family = AF\_INET;

    host\_addr.sin\_port = htons(LOCALDNS);

    inet\_pton(AF\_INET, "127.0.0.1", &host\_addr.sin\_addr);

    while (1)

    {

        fprintf(stdout, "\n [-] Enter the HostName: ");

        scanf("%s", input);

        sentbytes = sendto(socketfd, input, strlen(input) + 1, 0,

        (struct sockaddr \*)&host\_addr, sizeof(host\_addr));

        if (strncmp(input, "exit", sizeof("exit")) == 0)

            break;

        recvbytes = recvfrom(socketfd, buffer, sizeof(buffer), 0, NULL, NULL);

        if (strcmp("404", buffer) == 0)

            printf("DNS RECORDS NOT FOUND FOR %s\n", input);

        else

            printf("SERVER IP OF %s : %s\n", input, buffer);

        printf("\n\n------------------------------------------------\n\n");

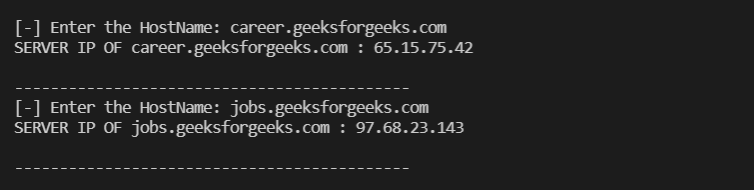
    }

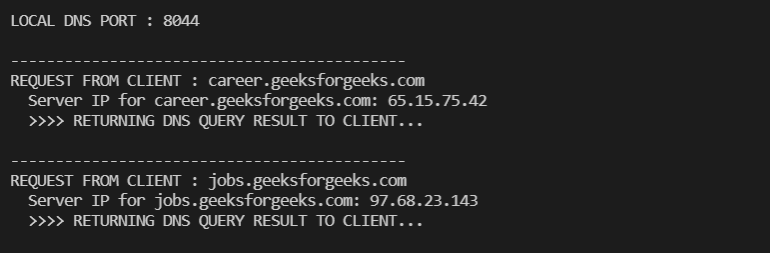
    close(socketfd);

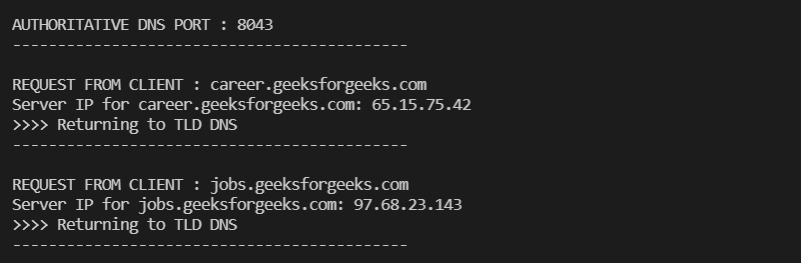
    return 0;

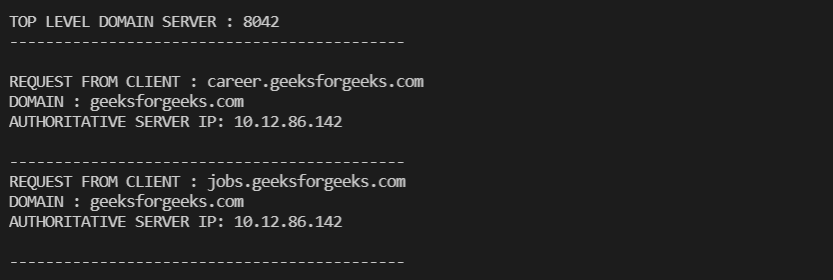
}

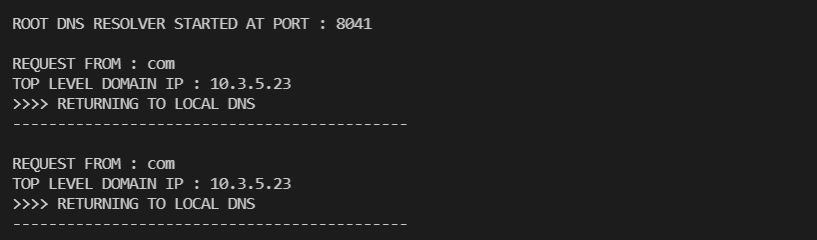
**OUTPUT :-**

**CLIENT**

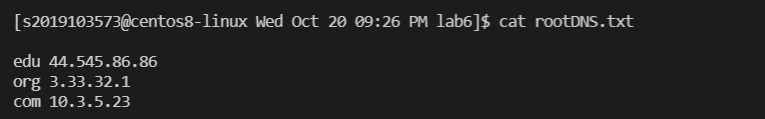
**LOCAL DNS**

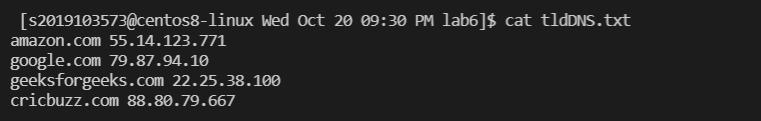
**AUTHORIZATION**

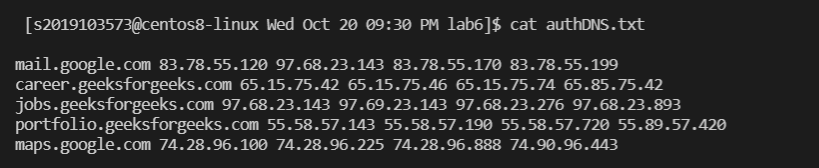
**TOP LEVEL DOMAIN**

**ROOT**

**TEXT FILES**

**ROOT - rootDNS.txt**

**TLP - tldDNS.txt**

**AUTHORIZATION - authDNS.txt**