**20/10/2021 CN LAB 6 2019103573**

**IMPLEMENTING THE DNS ITERATIVE**

**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 ROOTPORT 8041

extern int errno;

int main()

{

    int socketfd = 0, clientfd = 0, sentbytes, recvbytes;

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

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

    char 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(ROOTPORT);

    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;

    }

    printf("ROOT DNS RESOLVER STARTED AT PORT : %d\n", ROOTPORT);

    while (1)

    {

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

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

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

        FILE \*fd = fopen("rootdns.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, 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, "\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);

        }

        else

        {

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

            strcat(filebuff, lastbuff);

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

            close(fdes);

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

            sentbytes = sendto(socketfd, ip, strlen(ip) + 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 TLDPORT 8042

extern int errno;

int main()

{

    int socketfd = 0, clientfd = 0, sentbytes, recvbytes;

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

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

    char 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(TLDPORT);

    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;

    }

    printf("TOP LEVEL DOMAIN SERVER %d\n", TLDPORT);

    while (1)

    {

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

        &client\_addr, &length);

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

        FILE \*fd = fopen("tlddns.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, 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, "\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);

        }

        else

        {

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

            strcat(filebuff, lastbuff);

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

            close(fdes);

            fprintf(stdout, "AUTHORITATIVE SERVER IP : %s\n\n",

            ip);

            sentbytes = sendto(socketfd, ip, strlen(ip) + 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 AUTHPORT 8043

extern int errno;

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, clientfd = 0, sentbytes, recvbytes;

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

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

    char 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(AUTHPORT);

    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;

    }

    printf("AUTHORITATIVE DNS SERVER PORT : %d\n",AUTHPORT);

    while(1) {

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

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

        fprintf(stdout, "DNS QUERY : %s\n", buffer);

        FILE \*fd = fopen("authdns.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);

        }

        else

        {

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

            strcat(filebuff, lastbuff);

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

            close(fdes);

            fprintf(stdout, "AUTHORITATIVE SERVER IP : %s\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 TLDPORT 8042

#define AUTHPORT 8043

#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, tld\_addr, auth\_addr, client\_addr;

    char buffer[512], root[20], tld[30], auth[100];

    char rootip[30], tldip[30], authip[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;

    }

    printf(" LOCAL DNS PORT : %d\n", PORT);

    while (1)

    {

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

        &client\_addr, &length);

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

        {

            fprintf(stdout, "exiting");

            break;

        }

        fprintf(stdout, "Request from client : %s\n\n", buffer);

        strcpy(auth, buffer);

        int i = 0, j = 0, k = 0;

        while (buffer[i++] != '.')

        ;

        while (buffer[i] != '.')

        {

            tld[j++] = buffer[i++];

        }

        tld[j++] = buffer[i++];

        while (buffer[i] != ' ' && buffer[i] != '\0')

        {

            tld[j++] = buffer[i];

            root[k++] = buffer[i];

            i++;

        }

        tld[j] = '\0';

        root[k] = '\0';

        //fprintf(stdout, "\t\t[RESOLVING DNS QUERY]\n\n");

        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, root, strlen(root) + 1, 0,

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

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

        //fprintf(stdout, "[ROOT DNS SERVER]\n\n");

        fprintf(stdout, " [#] TLD server IP for %s:%s\n\n", root, rootip);

        close(rootfd);

        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, tld, strlen(tld) + 1, 0, (struct sockaddr \*)&tld\_addr, length);

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

        //fprintf(stdout,"[TLD SERVER]\n\n");

        fprintf(stdout, " [#] Auth server IP for %s:%s\n\n", tld, tldip);

        close(tldfd);

        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, auth, strlen(auth) + 1, 0,

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

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

        // fprintf(stdout, "[AUTHORITATIVE SERVER]\n\n");

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

            fprintf(stdout, "DNS RECORDS NOT FOUND : %s \n", auth);

        else

            fprintf(stdout, " [#] Server IP for %s: %s\n\n", auth,authip);

        close(authfd);

        sentbytes = sendto(socketfd, authip, strlen(authip) + 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);

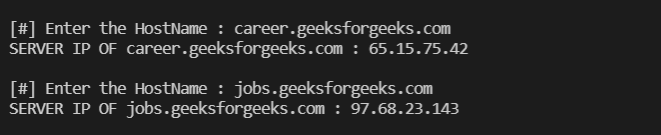
}

     close(socketfd);

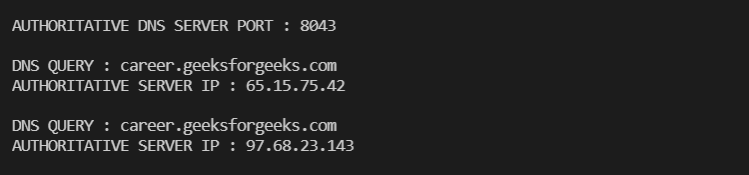
     return 0;

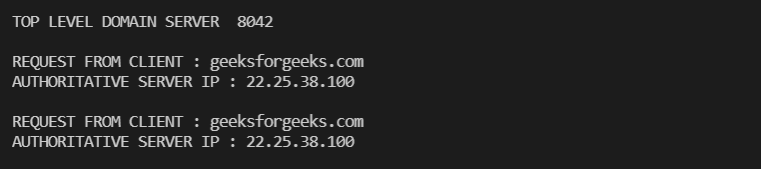
}

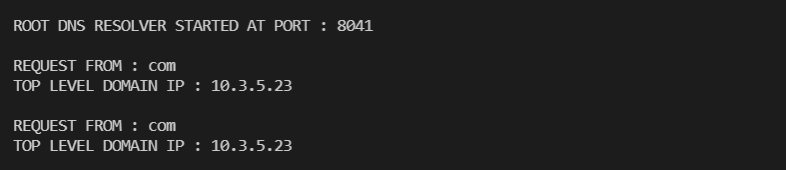
**OUTPUT :-**

**CLIENT**

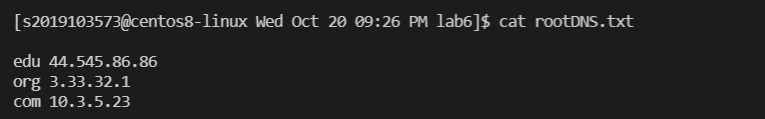
**LOCAL DNS**

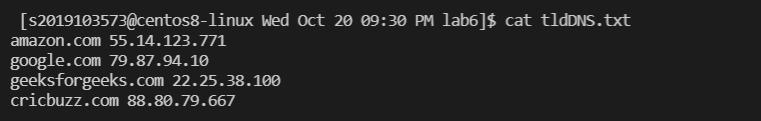
**AUTHORIZATION**

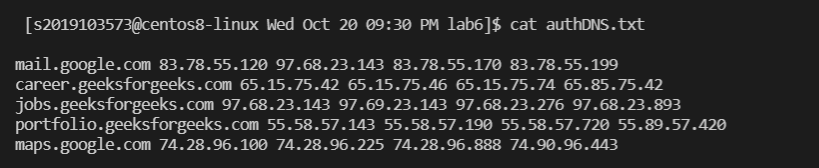
**TOP LEVEL DOMAIN**

**ROOT**

**TEXT FILES**

**ROOT - rootDNS.txt**

**TLP - tldDNS.txt**

**AUTHORIZATION - authDNS.txt**