**05/10/2021 CN LAB 5 2019103573**

**IMPLEMENTING WEB CACHE USING PROXY SERVER**

**SERVER**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <string.h>

#define SIZE 256

void cache(int proxy\_socket) {

    char buf[SIZE];

    int f = 0, i;

    FILE \*fp;

    char \* line = NULL;

    size\_t len = 0;

    fp = fopen("sample.txt","r+");

    while(1) {

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

        rewind(fp);

        read(proxy\_socket, buf, sizeof(buf));

        printf("\nProxy: %s\n", buf);

        if(strcmp(buf, "exit") == 0) {

            printf("\nServer Exit\n");

            break;

        }

        while ((getline(&line, &len, fp)) != -1) {

            if(strcmp(buf, line) == 0) {

                printf("Date matches\n");

                strcpy(buf, "Date matches");

            }

            else {

                for(i=0; i<strlen(buf) && buf[i] == line[i] && buf[i] != ' '; i++);

                if(buf[i] == ' ') {

                    printf("Date does not match\n");

                    strcpy(buf, line);

                }

                else if(i == strlen(buf)) {

                    printf("\nURL is not present in cache\n");

                    strcpy(buf, line);

                }

            }

        }

        write(proxy\_socket, buf, sizeof(buf));

        bzero(buf, sizeof(buf));

    }

}

int main() {

    int server\_socket;

    server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

    if(server\_socket == -1) {

        printf("Socket Creation failed\n");

        exit(0);

    }

    struct sockaddr\_in server\_address;

    server\_address.sin\_family = AF\_INET;

    server\_address.sin\_port = htons(8510);

    server\_address.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if(bind(server\_socket, (struct sockaddr \*) &server\_address, sizeof(server\_address)) != 0) {

        printf("Bind Failed\n");

        exit(0);

    }

    else {

        printf("Bind Successful\n");

    }

    if(listen(server\_socket, 3 != 0)) {

        printf("Listen Failed\n");

        exit(0);

    }

    else {

        printf("Listening\n");

    }

    int proxy\_socket;

    proxy\_socket = accept(server\_socket, NULL, NULL);

    if(proxy\_socket < 0) {

        printf("Accept Failed\n");

        exit(0);

    }

    else {

        printf("Proxy Accepted\n");

    }

    cache(proxy\_socket);

    close(server\_socket);

    return 0;

}

**CLIENT**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <string.h>

#define SIZE 256

void cache(int client\_socket) {

    char buf[SIZE];

    int n;

    while(1) {

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

        bzero(buf, sizeof(buf));

        n = 0;

        printf("\nEnter URL: ");

        while((buf[n++] = getchar()) != '\n');

        write(client\_socket, buf, sizeof(buf));

        buf[n-1] = '\0';

        if(strcmp(buf, "exit") == 0) {

            printf("\nClient Exit\n");

            break;

        }

        bzero(buf, sizeof(buf));

        read(client\_socket, buf, sizeof(buf));

        printf("\nServer Response:\n\n%s\n", buf);

    }

}

int main() {

    int client\_socket;

    client\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

    struct sockaddr\_in server\_address;

    server\_address.sin\_family = AF\_INET;

    server\_address.sin\_port = htons(8500);

    server\_address.sin\_addr.s\_addr = INADDR\_ANY;

    int conn\_status = connect(client\_socket, (struct sockaddr \*) &server\_address, sizeof(server\_address));

    if(conn\_status == -1) {

        printf("Connection Failed\n");

        exit(0);

    }

    printf("Connection Established\n");

    cache(client\_socket);

    close(client\_socket);

    return 0;

}

**PROXY**

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <unistd.h>

#include <string.h>

#define SIZE 256

void cache(int proxy\_socket) {

    int f = 0, i, pos;

    char buf[SIZE];

    FILE \*fp;

    char \* line = NULL;

    size\_t len = 0;

    fp = fopen("sample.txt","r+");

    int socket\_fd;

    socket\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

    if(socket\_fd == -1) {

        printf("Socket Creation failed\n");

        exit(0);

    }

    struct sockaddr\_in server\_address;

    server\_address.sin\_family = AF\_INET;

    server\_address.sin\_port = htons(8500);

    server\_address.sin\_addr.s\_addr = htonl(INADDR\_ANY);

    if(bind(socket\_fd, (struct sockaddr \*) &server\_address, sizeof(server\_address)) != 0) {

        printf("Bind Failed\n");

        exit(0);

    }

    else {

        printf("Bind Successful\n");

    }

    if(listen(socket\_fd, 3 != 0)) {

        printf("Listen Failed\n");

        exit(0);

    }

    else {

        printf("Listening\n");

    }

    int client\_socket;

    client\_socket = accept(socket\_fd, NULL, NULL);

    if(client\_socket < 0) {

        printf("Accept Failed\n");

        exit(0);

    }

    else {

        printf("Client Accepted\n");

    }

    while(1) {

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

        rewind(fp);

        f = 0;

        bzero(buf, sizeof(buf));

        read(client\_socket, buf, sizeof(buf));

        printf("\nClient: %s\n", buf);

        buf[strlen(buf)-1] = '\0';

        if(strcmp(buf, "exit") == 0) {

            printf("Proxy Exit\n");

            write(proxy\_socket, buf, sizeof(buf));

            break;

        }

        while ((getline(&line, &len, fp)) != -1) {

            for(i=0; i < strlen(buf)-1; i++) {

                if(buf[i] != line[i]) {

                    break;

                }

            }

            if(i == strlen(buf)-1) {

                pos = ftell(fp) - strlen(line);

                f = 1;

                break;

            }

        }

        if(f == 1) {

            printf("URL found\n");

            strcpy(buf, line);

            write(proxy\_socket, buf, sizeof(buf));

            bzero(buf, sizeof(buf));

            read(proxy\_socket, buf, sizeof(buf));

            printf("Server: %s\n", buf);

            if(strcmp(buf, "Date matches") == 0) {

                strcpy(buf, line);

            }

            else {

                printf("Date does not match\n");

                printf("Updating cache file:\n");

                fseek(fp, pos, SEEK\_SET);

                fputs(buf, fp);

            }

        }

        else {

            printf("URL not found\n");

            write(proxy\_socket, buf, sizeof(buf));

            bzero(buf, sizeof(buf));

            read(proxy\_socket, buf, sizeof(buf));

            printf("Server: %s\n", buf);

            printf("Updating cache file:\n");

            fseek(fp, 0, SEEK\_END);

            fputs(buf, fp);

        }

        bzero(buf, sizeof(buf));

        rewind(fp);

        while ((getline(&line, &len, fp)) != -1) {

            strcat(buf, line);

        }

        write(client\_socket, buf, sizeof(buf));

    }

    close(proxy\_socket);

    fclose(fp);

}

int main() {

    int proxy\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

    struct sockaddr\_in server\_address;

    server\_address.sin\_family = AF\_INET;

    server\_address.sin\_port = htons(8510);

    server\_address.sin\_addr.s\_addr = INADDR\_ANY;

    int conn\_status = connect(proxy\_socket, (struct sockaddr \*) &server\_address, sizeof(server\_address));

    if(conn\_status == -1) {

        printf("Connection to server failed\n");

        exit(0);

    }

    printf("Connection to server established\n");

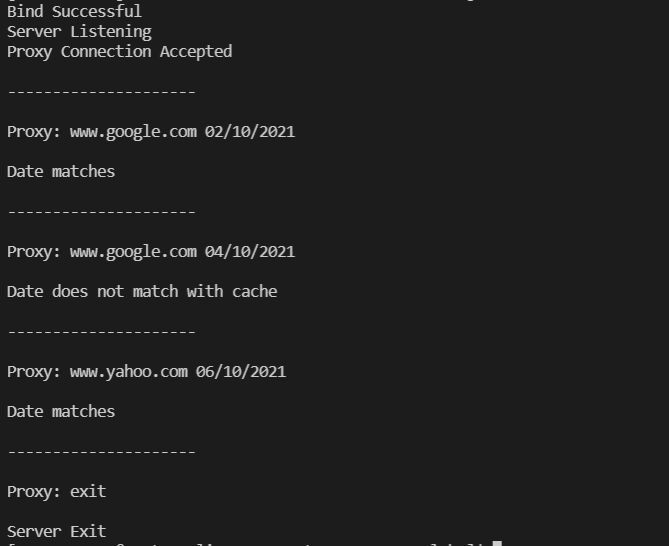
    cache(proxy\_socket);

    close(proxy\_socket);

    return 0;

}

**OUTPUT :-**

**SERVER**

**CLIENT**

**PROXY**