IMPLEMENTING WEB CACHE USING PROXY SERVER

SERVER

```c
#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);
    }
}
```

```c
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

```c
#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

```
Bind Successful
Server Listening
Proxy Connection Accepted

----------------------

Proxy: www.google.com 02/10/2021

Date matches

----------------------

Proxy: www.google.com 04/10/2021

Date does not match with cache

----------------------

Proxy: www.yahoo.com 06/10/2021

Date matches

----------------------

Proxy: exit

Server Exit
```

CLIENT

```
Connection Established

------------------------

Enter URL: www.google.com

Server Response:

www.boat.com 03/10/2021
www.google.com 02/10/2021
www.yahoo.com 06/10/2021

------------------------

Enter URL: www.yahoo.com

Server Response:

www.boat.com 03/10/2021
www.google.com 06/10/2021
www.yahoo.com 06/10/2021

------------------------

Enter URL: www.boat.com

Server Response:

www.boat.com 03/10/2021
www.google.com 02/10/2021
www.yahoo.com 06/10/2021

Enter URL: exit

Client Exit
```

PROXY

```
Connection to server established
Bind Successful
Proxy Listening
Client Connection Successful

----------------------

Client: www.yahoo.com

URL found in Cache
Server: Date matches

----------------------

Client: www.google.com

URL found in Cache
Server: www.google.com 06/10/2021

Date does not match with that of server
Updating the time for the given url in cache file:

----------------------

Client: www.google.com

URL found in Cache
Server: Date matches

----------------------

Client: exit

Proxy Exit
```