# CENG-393
# HOMEWORK

*Kaan Kalan*
*201611032*

I hereby declare that_this homework is my own work
and is not copied from αny other student´s work or other sources.

Kaan Kalan

**YOUTUBE Video Explanation:** **https://youtu.be/Gh3X_BAZRQo**

*28.12.2020*

***Abstract***

In this report, there is a detailed explanation of CENG393 lab sessions homework(HW). The purpose of this HW implementing a simple student grade information system. Students are designed as a client-side and server-side can be our university SQL system. The client enters the server by connecting to the server IP and enters Username and Password. If information is correct, the server sends the grades to the client. Otherwise, sends a request to enter wrong entered info again.

## 1) Methodologies

Implementation steps has been defined in this section

### 1.1 Server-Side

The main management task has done on the server-side.

#### 1.1.1 Read File

As a first part, grades.txt file has been read by the server, and all ID information attached to the id string. Strings have defined like `'char* id[100], char* password[100], int grades[100]'` and as the array gets input, it has been expanded using malloc. So dynamic memory allocation has used to save the waste of memory.

For example:

```
      fscanf(fp,"%s",readed_text);
      id[k]=malloc(strlen(readed_text)+1);
      if(id[k])
            strcpy(id[k],readed_text);
```

#### 1.1.2 Learn the client ID

Accept function has implemented like:

```
socklen_t socksize=sizeof(struct sockaddr_in);
client[i]= accept(sockfd,(struct sockaddr *)&dest[i], &socksize);
```

to hold accepted clients IP. Because when an request has answered by server, server prints out all records.[1] We reached to client id with using : `inet_ntoa(dest->sin_addr)`

#### 1.1.3 Managing which client entered what

Another problem is What if one client enters ID and the other enters the password? Are we gonna send the grade last client? Of course no. To solve that problem, `int client_steps[100]` has defined. This array holds, which client entered Username, who has not. When a request arrived from a client we check this array.

- If `client[i]` sends an info and `client_steps[i]` is 1, That means, this user entered Username correctly before. Assume current request as a password
- If `client[i]` sends an info and `client_steps[i]` is -1, That means this user entered Username wrongly before. Assume current request as a password
- If `client[i]` sends an info and `client_steps[i]` is 0, That means this user has not entered the Username yet. Assume current request as a Username.

### 1.1.4 Sending answer to the client, (what client has to enter know)

In this homework, answers the requests with codifying the integers,

- If information is correct, the server sends the user's grade as an integer.
- If the user didn't enter the password yet, the server sent -3 as an integer
- If the password is wrong, the server sends -2 as an integer
- If the username is wrong, the server sends -1 as an integer

These send tasks have done when a client sends input to the socket.

### 1.1.5 Checking the is entered info in the grades.txt

When we take information from `client[i],` after decided whether is that password or Username, we have to check is this info in the grades.txt. To do that task, an `int is_correct(char** id, char*taken _id, int k)` function has been implemented.

- `char** id` means grades.txt file ID array or Password array.
- `char* taken_id` means taken info from the client( whether ID or Password).
- `int k` means the length of taken information from grades.txt. More clearly, used array length of ID and Password arrays. <u>That's an important point</u> because when we didn't do that, if sent info is not except in the ID or Password array(database arrays), the function keeps trying to reach the unused memory blocks and we are taking <u>Segmentation Fauld</u> from Linux. To solve that we sending k and our loop checks until reach the k'th number or array.[2]

If taken_id excepts in the id array, the function returns the index number of this id.
Else function return -5 to main of server code.

### 1.1.6 What if entered User name and Password except in the array but doesn't match with each other

To prevent this situation, an `int cl_holder[100]` array has been implemented. This array holds which the user finds the username at which position(Database index). Thanks to this information, when the client sends the password, we can check the password's index and Username(cl_holder[clientnumber]) index are matching or not. If they match, we can send the grade of the user. If they didn't match we send -2(wrong password)

### 1.1.7 Print out the wrong entered ID in the server

When we check the sample output of the homework, we can recognize, when the client entered the wrong username, we have to print out the wrong entered username. To solve that problem, `char *wrong_ID[50]` array has been implemented. This array holds the i'th client entered this Username before. Thanks to this information, we can report all activities that have done on the server-side.

### 1.1.8 Printing part of string in a single line

When we take an input from client(buf), C probably add '\n' to taken string. To prevent this, I printed the string this way:

```
printf(" incorrect username %.*s to %s\n",(int)strlen(wrong_ID[i]), wrong_ID[i],
inet_ntoa(...
```

%.*s means print the `wrong_ID[i]` string until `strlen(wrong_ID[i])` char. It means dont print the last character of string [3].

### 1.2 Client-Side

The Client-side is more clear than the server-side. There are just two tasks on the client-side. Send username or password to the server and listen to the server's response.

#### 1.2.1 Managing servers response

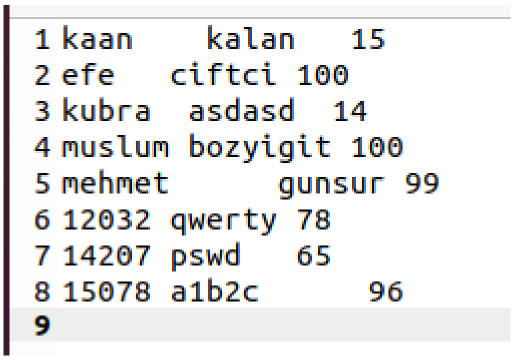
When the server sends a response to the client, the Client must get the user to know about what the user going to do. To do that, when the client receives the integer from the server, checks is that -3, -2, -1, or up to 0. Actually, the process doesn't change with these responses. The client keeps send a string to the server, but the user knows what the server expects from him.

- If taken input ( buf) is greater than 0 prints and close the socket. The task is successfully done!
- If the buf is equal to -3, servers expect the password. Prints out 'Password:'
- If the buf is equal to -2, servers expect the password again. Prints out "Password is incorrect, Password:".
- If the buf is equal to -1, servers expect the username and password again. Prints out 'Username does not exist, Username:'.

#### 1.2.2 Printing out multiple strings

When we want to print out the messages, printf function prints the string after the socket connection ended. After some research, the problem was output has been buffered.[4] To prevent that we can add the '\n' end of the string or we can add `setvbuf(stdout,NULL,_IONBF,0);` before every print statement. So the problem is solved with this solution.

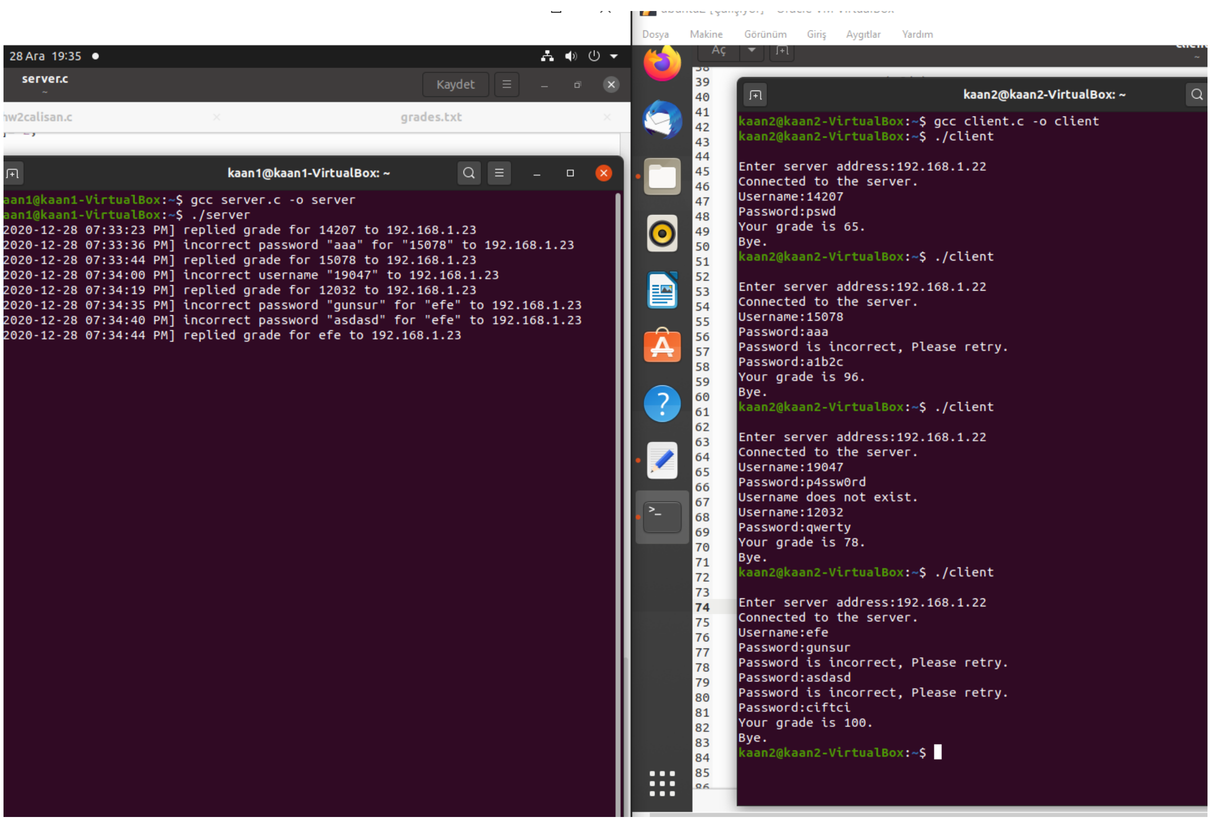
## 2. OUTPUTS



```
kaan     kalan   15
efe   ciftci 100
kubra   asdasd   14
muslum bozyigit 100
mehmet      gunsur 99
12032 qwerty 78
14207 pswd   65
15078 a1b2c      96

```

*Figure 1. grades.txt file*



```
aan1@kaan1-VirtualBox:~$ gcc server.c -o server
aan1@kaan1-VirtualBox:~$ ./server
2020-12-28 07:33:23 PM] replied grade for 14207 to 192.168.1.23
2020-12-28 07:33:36 PM] incorrect password "aaa" for "15078" to 192.168.1.23
2020-12-28 07:33:44 PM] replied grade for 15078 to 192.168.1.23
2020-12-28 07:34:00 PM] incorrect username "19047" to 192.168.1.23
2020-12-28 07:34:19 PM] replied grade for 12032 to 192.168.1.23
2020-12-28 07:34:35 PM] incorrect password "gunsur" for "efe" to 192.168.1.23
2020-12-28 07:34:40 PM] incorrect password "asdasd" for "efe" to 192.168.1.23
2020-12-28 07:34:44 PM] replied grade for efe to 192.168.1.23
```

```
kaan2@kaan2-VirtualBox:~$ gcc client.c -o client
kaan2@kaan2-VirtualBox:~$ ./client

Enter server address:192.168.1.22
Connected to the server.
Username:14207
Password:pswd
Your grade is 65.
Bye.
kaan2@kaan2-VirtualBox:~$ ./client

Enter server address:192.168.1.22
Connected to the server.
Username:15078
Password:aaa
Password is incorrect, Please retry.
Password:a1b2c
Your grade is 96.
Bye.
kaan2@kaan2-VirtualBox:~$ ./client

Enter server address:192.168.1.22
Connected to the server.
Username:19047
Password:p4ssw0rd
Username does not exist.
Username:12032
Password:qwerty
Your grade is 78.
Bye.
kaan2@kaan2-VirtualBox:~$ ./client

Enter server address:192.168.1.22
Connected to the server.
Username:efe
Password:gunsur
Password is incorrect, Please retry.
Password:asdasd
Password is incorrect, Please retry.
Password:ciftci
Your grade is 100.
Bye.
kaan2@kaan2-VirtualBox:~$
```

*Figure 2. Sample Output*

## References

1. C Programming/Networking in UNIX - Wikibooks, open books for an open world. (2020). Retrieved 26 December 2020, from https://en.wikibooks.org/wiki/C_Programming/Networking_in_UNIX#A_simple_server

2. Core Dump (Segmentation fault) in C/C++ - GeeksforGeeks. (2017). Retrieved 28 December 2020, from https://www.geeksforgeeks.org/core-dump-segmentation-fault-c-cpp/#:~:text=Core%20Dump%2FSegmentation%20fault%20is,an%20error%20indicating%20memory%20corruption.

3.[duplicate], G., Buyukliev, B., & Scurtu, M. (2010). Get a substring of a char*. Retrieved 27 December 2020, from https://stackoverflow.com/questions/4214314/get-a-substring-of-a-char

4. console, p., T., M., & T., M. (2012). printf not printing on console. Retrieved 26 December 2020, from https://stackoverflow.com/questions/13035075/printf-not-printing-on-console