# Online Judge or Autograder

**Tutorial**
**Computing Lab 2, Spring 2021**

# Online Judge or Autograder

Online judges are systems designed for the reliable evaluation of algorithm source code submitted by users.

Online judges takes source code, Compile it, check for compilation errors, execute your code by providing some secret inputs if required and collect your code's output and finally match your output with its hidden test cases.

**FTP command -**

CODEJUD - This command will take a c/c++ file from client and server will compile , execute and match the output with given test cases and will notify back to client about any error or correctness of c/c++ file.

# CODEJUD

1. **Compilation Phase -** This phase will take a String that contains the source code and coding language (c/c++).This is the file client wants to test. This phase will compile the input and send compilation error messages to clients if any. Else it will create an object file and send a compilation success message to the client.

2. **Execution Phase** - This phase will take object file , time Limit(1sec). This phase will execute your code by providing some secret inputs if required. This phase will decide if the input file has any error (TLE or Runtime error). If it succeeds then it will create a new output.txt file and send an execution success message to the client.

3. **Matching Phase** - This phase will compare Correct testcase.txt file with the actual output.txt file, if it is matched then the successful message will be passed to client, else the wrong output message will be passed to client.

# Feedback from CODEJUD

1. **Compilation phase -**

   COMPILE_ERROR - The judge's compiler can not compile client's source code.

   COMPILE_SUCCESS - The judge's compiler compile code successfully.

2. **Execution Phase -**

   TIME LIMIT EXCEEDED - Client's program failed to finish executing before the established time limit (1 sec) for the problem.

   RUN_ERROR - The judge find error occurs during program execution(run-time).

   RUN_SUCCESS - The judge find no error during program execution(run-time).

3. **Matching Phase -**

   ACCEPTED - All the test cases passed.

   WRONG_ANSWER - Test cases failed.

# system() call

```
int system(const char *command);
```

system() - It is use to invoke an operating system command from a C/C++ program.

The C library function int system(const char *command) passes the command name or program name specified by command to the host environment to be executed by the command processor and returns after the command has been completed.

Can invoke compiler command etc. from program using system().

stdlib.h or cstdlib needs to be included to call system.

# Problem Description

Write two separate c programs, one for server (handles requests from multiple users)  and other one for client.

The server process uses the “select” system call to see which clients are making the requests and serve the request of multiple clients.

In order to transfer files etc. you can use ftp. And you can reuse your previous ftp setup.

Implement the FTP commands RETR, STOR, LIST, ABOR, QUIT, DELE, CODEJUD.

# Workflow

**Working of server and client is as follows:**

1. Server accepts connections from clients.
2. Server receives control information like FTP commands.
3. Server checks for valid commands, and executes them and give output to client based on command.
4. Client receives the response and displays it to the user..
5. In order to transfer files etc. you can use ftp. And you can reuse your previous ftp setup.
6. Handle corner cases.
7. Use select to handle multiple client requests.

# Workflow

**Implement the following ftp commands:**

1. RETR <filename>
2. STOR <filename>
3. LIST
4. ABOR (Optional)
5. QUIT
6. DELE <filename>
7. CODEJUD <filename> <ext(c/c++)>

**Marking Scheme :- 100 marks**

1.RETR,STOR,LIST,ABOR,QUIT,DELE :- 15 marks

2. CODEJUD :- 15 + 15 + 15 (Compilation, Execution, Matching).

3. Concurrency Handling : 20 marks

4. Error Handling : 10

5. .Coding Style : 10

# Submission

Deadline : **February 3rd, 2021 [2 PM]**

1. Create a folder with the name “<roll_number>_Assign4”.
2. Create file server.c.
3. Create file client.c.
4. Create readme file with 1-2 lines about each possible error case that you have considered with an example.
5. Compress the folder into zip or tar format and submit compressed folder in Moodle