Project: Boolean Expression Evaluation over UNIX Domain Sockets

CSCE 311 - Operating Systems

Spring 2025

Purpose

In this project, you will gain experience with interprocess communication (IPC) using UNIX domain sockets. You will implement a client-server system where the server processes boolean expressions from a large file, and the client sends truth value assignments for variables. The server evaluates each line of the file based on the provided values and returns a summary of results.

Task

You must implement two applications:

- **Server:** Reads a large file containing boolean expressions and listens on a UNIX domain socket for connections. Upon connection, it sends the client two special characters: the **unit separator (US)** and the **end of transmission (EOT)**.

  The server must:

  - Accept the following parameters (in this order):
    1. A path to a Boolean expressions file, e.g., `dat/expr_25k.txt`.
    2. Name of hosted domain socket, e.g., `bool_expr_sock`.
    3. Unit separator character, must be a single character, e.g., `,`
    4. End of transmission character, must be a single character, e.g., `.`

    ```
    ./bool-expr-server dat/expr_25k.txt bool_expr_sock , .
    ```

  - Build a mapping of the provided truth values to variables 'a' through 'z', based on the number of `true`/`false` values sent by the client.
  - Read each line of the file and attempt to evaluate the boolean expression using the provided mapping.
  - Categorize each expression as:

    True: Evaluates to true.

    False: Evaluates to false.

    Could Not Evaluate: Requires more variables than provided.

  - Return a summary to the client in the format:

    ```
    n_T US n_F US n_E EOT
    ```

    Where `n_T`, `n_F`, and `n_E` represent counts for each category and US and EOT represent the unit separator and end of transmission characters, respectively. **There should be no spaces in this message**, e.g., `101\037739\037249160\004`, to send 101, 739, 249160 with the standard US '`\037`' and EOT '`\004`'.

- **Client:** Connects to the server, receives the US and EOT characters, sends a sequence of truth values (T/F) separated by US and terminated by EOT, then receives a summary response.

  The client must:

  - Accept as parameters, in order,
    1. Name of hosted domain socket, e.g., `bool_expr_sock`.
    2. One or more T or F values; you may assume there is always one to be valid.

    ```
    ./bool-expr-client bool_expr_sock T F F T F T F T T F
    ```
  - Connect to the server and receive the US and EOT characters.
  - Send the sequence of T/F values using US as a separator and terminating with EOT.
  - Receive and display the server's response in the format:

    ```
    BoolExprClient connecting...
    Finished with 2B received, 16B sent.
    Results
    True Evaluations:       101
    False Evaluations:      739
    Could Not Evaluate:     249160
    ```

Execution

Server:

```
./bool-expr-server dat/expr_25k.txt bool_expr_sock : .
```

Client:

```
./bool-expr-client bool_expr_sock T F F T F T F T T F
```

Deliverables

You must provide a zipped directory containing at least the following files:

```
proj2/
+-- include/
|   +-- bool_expr_client.h
|   +-- bool_expr_server.h
|
+-- src/
|   +-- bool_expr_client.cc
|   +-- bool_expr_server.cc
|
+-- README.md
```

Note that any other files will be ignored. I have included two libraries (`<boolean_expression.h>` and `<domain_socket.h>`) for your use. Those will be available for your code to compile against, but you will not be able to change them because we will not copy any files other than those mentioned above.

Grading Criteria

Build (20%)

- Server correctly builds using the Makefile and libraries provided 10%.
- Client correctly builds using the Makefile and libraries provided 10%.

Execution (30%)

- Server successfully executes (15%). It must start, print the expected messages, and not crash when a client connects.
- Client successfully executes (15%). It must attempt to connect to a running server and terminate gracefully.

Correctness (40%)

- Server correctly sends, receives, and responds to messages from a client 10%.
- Client correctly receives, sends, and processes the results from a server 10%.
- Proper error handling, e.g. one application crashing does not crash the other 10%.
- The Server is robust to multiple client requests without restart 5%.
- The Server successfully runs through multiple restarts 5%.

README.md (10%)

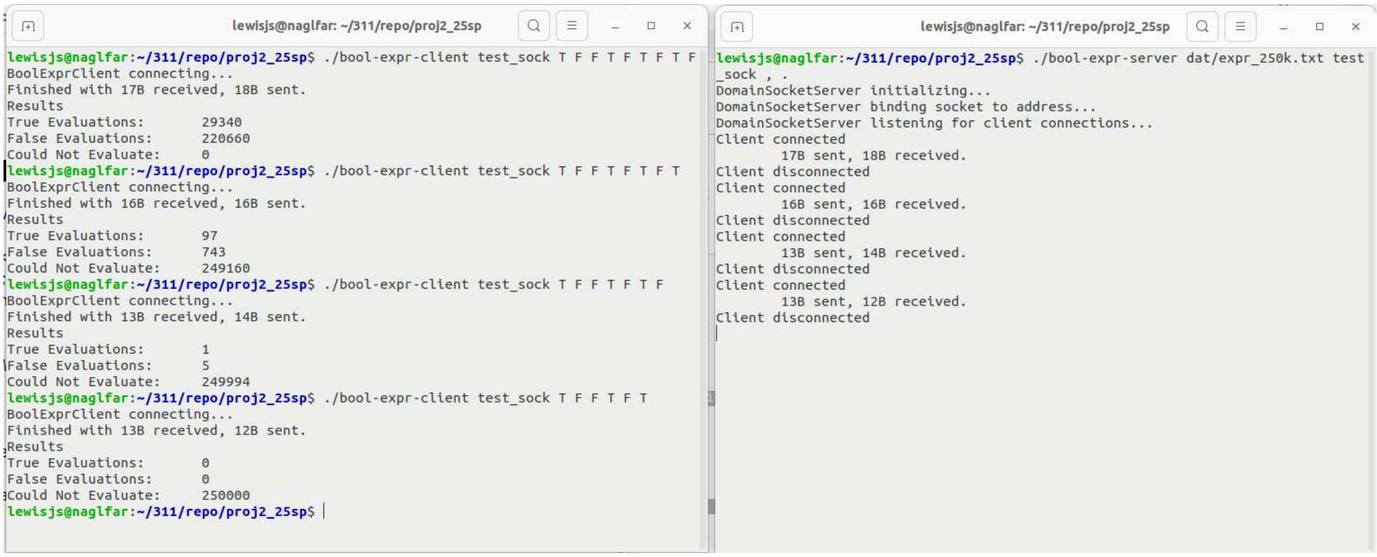
- Provide a reasonable `README.md` file, including build and execution instructions, file structure (only files you provide), and file contents for a free 10%.

Additional Notes

- The boolean expression file format will be well-defined (e.g., infix notation with ‘*‘, ‘+‘).
- Consider efficiency when handling large files.
- You must use helper libraries like ‘`std::getline`’ and ‘`std::fstream`’ for file reading.

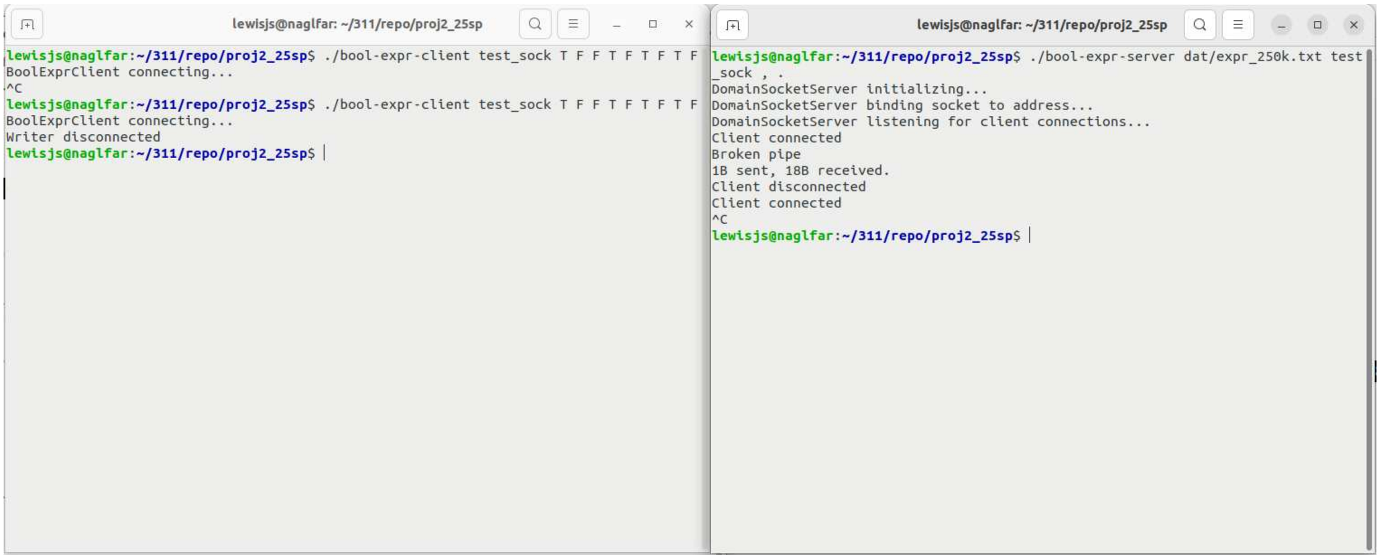
Examples

I have attached a couple images of my solution running.


lewisjs@naglfar: ~/311/repo/proj2_25sp
lewisjs@naglfar: ~/311/repo/proj2_25sp

```
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-client test_sock T F F T F T F T F
BoolExprClient connecting...
Finished with 17B received, 18B sent.
Results
True Evaluations:       29340
False Evaluations:      220660
Could Not Evaluate:     0
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-client test_sock T F F T F T F T
BoolExprClient connecting...
Finished with 16B received, 16B sent.
Results
True Evaluations:       97
False Evaluations:      743
Could Not Evaluate:     249160
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-client test_sock T F F T F T F
BoolExprClient connecting...
Finished with 13B received, 14B sent.
Results
True Evaluations:       1
False Evaluations:      5
Could Not Evaluate:     249994
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-client test_sock T F F T F T
BoolExprClient connecting...
Finished with 13B received, 12B sent.
Results
True Evaluations:       0
False Evaluations:      0
Could Not Evaluate:     250000
lewisjs@naglfar:~/311/repo/proj2_25sp$ 
```

```
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-server dat/expr_250k.txt test
_sock , .
DomainSocketServer initializing...
DomainSocketServer binding socket to address...
DomainSocketServer listening for client connections...
Client connected
        17B sent, 18B received.
Client disconnected
Client connected
        16B sent, 16B received.
Client disconnected
Client connected
        13B sent, 14B received.
Client disconnected
Client connected
        13B sent, 12B received.
Client disconnected
```


lewisjs@naglfar: ~/311/repo/proj2_25sp
lewisjs@naglfar: ~/311/repo/proj2_25sp

```
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-client test_sock T F F T F T F T F
BoolExprClient connecting...
^C
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-client test_sock T F F T F T F T F
BoolExprClient connecting...
Writer disconnected
lewisjs@naglfar:~/311/repo/proj2_25sp$ 
```

```
lewisjs@naglfar:~/311/repo/proj2_25sp$ ./bool-expr-server dat/expr_250k.txt test
_sock , .
DomainSocketServer initializing...
DomainSocketServer binding socket to address...
DomainSocketServer listening for client connections...
Client connected
Broken pipe
1B sent, 18B received.
Client disconnected
Client connected
^C
lewisjs@naglfar:~/311/repo/proj2_25sp$ 
```

In the second graphic, you can see where I first kill a client before the server can finish its request. Note the server acknowledges the broken pipe and continues. Next, I kill the server while it is working on the client request (leaving the client running).