OS 2019 Homework2

Multithreaded String Searching

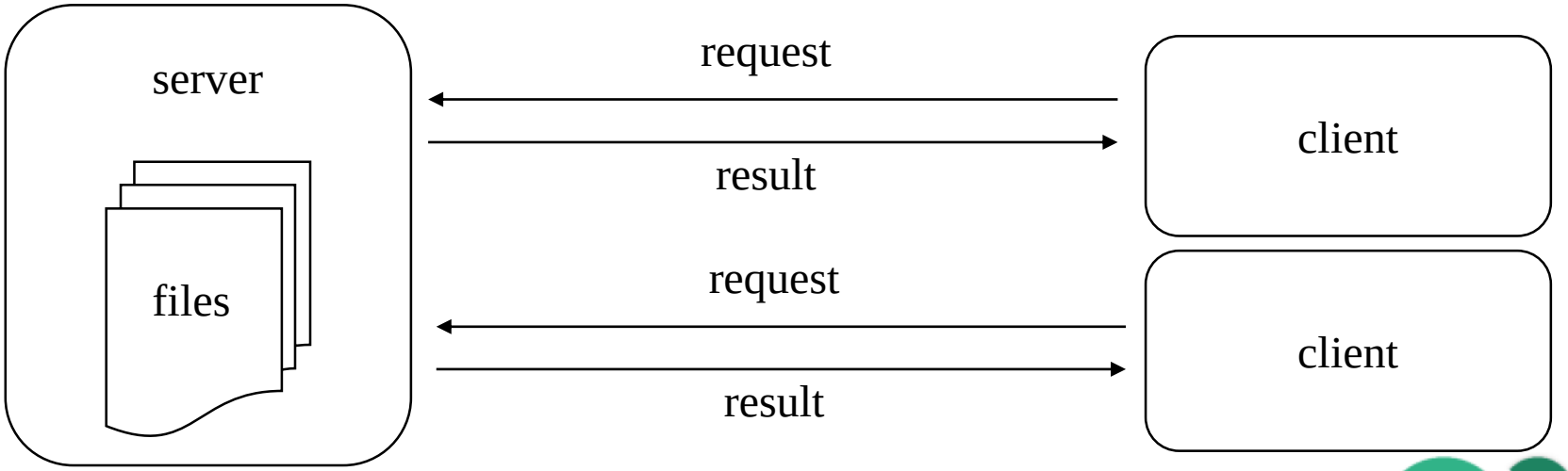
(Due date: 2019/11/21 23:59)

Objectives

- Understand multi-thread programming
- Understand client-sever model

Overview

- Files stored in the server
- Each client sends a request to the server to search for one or more strings in these files
- Server returns the search results



Requirements - Server

- Connected with clients via Internet sockets
- Multi-threaded: a main thread + a pool of worker thread
- Main thread
  - Responsible for receiving client requests
  - Insert the requests in the *request queue*
- Worker threads
  - Remove entries from the request queue and search all files for the queried strings
  - Return results to the clients
- Lock is required to prevent race condition

Race condition

- If more than one worker threads attempt to access the same resource, e.g., global variable, simultaneously, there might be an error occurred
- For example, there is a global variable A = 5, a local variable B = 0, worker threads w1, w2

```
A = A - 1      (1)
B = A          (2)
```

Expected flow:
w1 execute (1)
w1 execute (2)
w2 execute (1)
w2 execute (2)
---
w1: B = 4
w2: B = 3

Not expected flow:
w1 execute (1)
w2 execute (1)
w1 execute (2)
w2 execute (2)
---
w1: B = 3
w2: B = 3

Lock

- To prevent race condition, we might need to use a lock
- For example

```
lock
A = A - 1      (1)
B = A          (2)
unlock
```

- The detail could be found in the reference page

Requirements - Server

- Parameter Format
  - ./server -r *ROOT* -p *PORT* -n *THREAD_NUMBER*
  - *ROOT*: The path of the files
    - e.g., if the /test directory is set as ROOT, the server will search all files under /test for the strings
  - *PORT*: The port which the server is listening to
  - *THREAD_NUMBER*: number of (worker) threads in the thread pool

- Server should print out the information once receiving a request
  - Output Format: Query *"QUERY_STRING" ["QUERY_STRING"]** \n

Requirements - Client

- Parameter Format
  - ./client -h *LOCALHOST* -p *PORT*
  - *LOCALHOST*: Server would be running on localhost (127.0.0.1)
  - *PORT*: The port which the server is listening to
- Send Requests
  - Request Format: Query *"QUERY_STRING" ["QUERY_STRING"]**
    - *QUERY_STRING*: The string(s) that the client tries to query
    - The maximum size of the *QUERY_STRING* is 128 bytes
  - Clients should support one or more requests
  - Clients should be multi-threaded, create a thread to handle a request

Requirements - Client

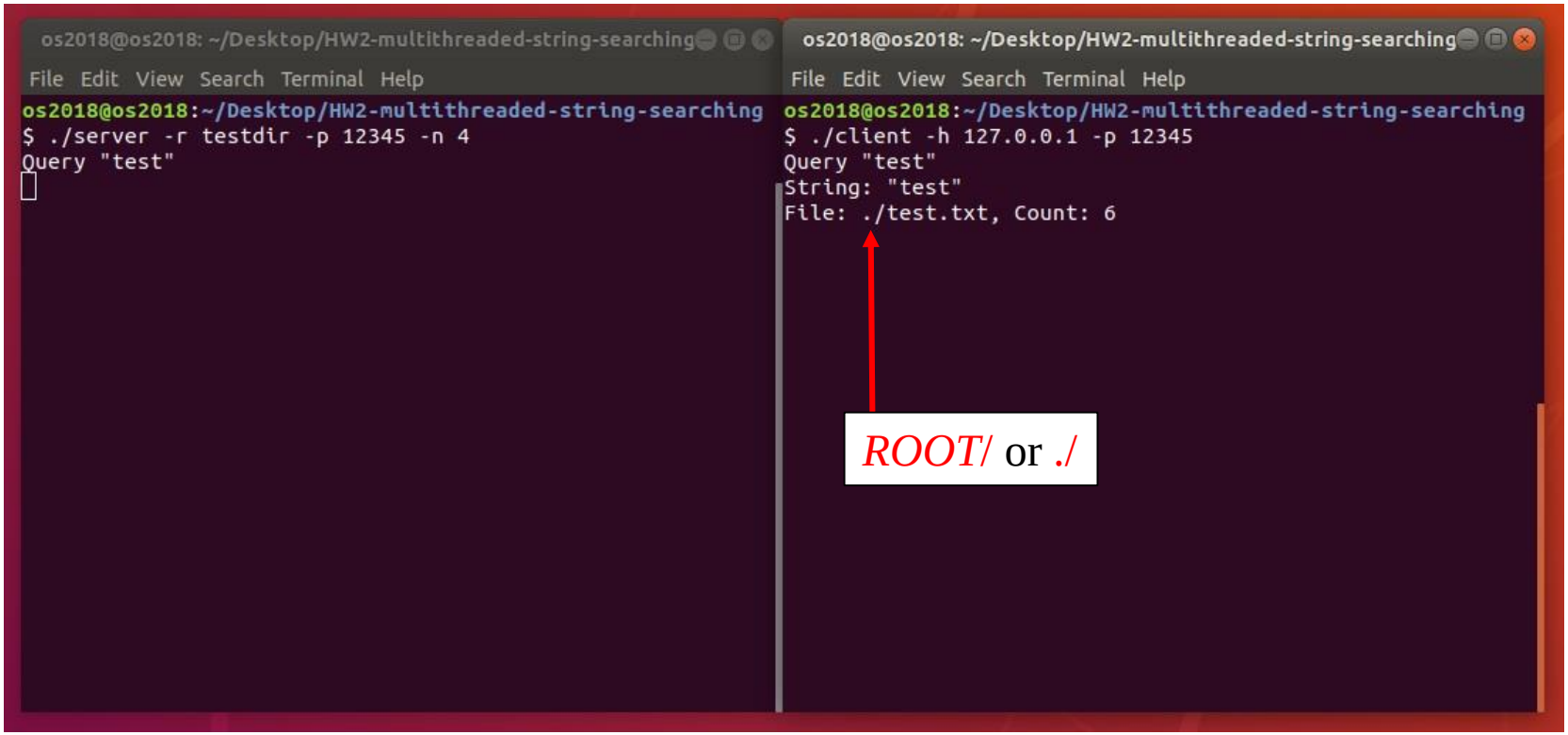
- Clients should output the result for each string it queried
  - If a string exists in one or more files, the client prints out the file path(s) and the count(s) of the string (i.e., the number of occurrences in each of the files)

    Format: String: “*QUERY_STRING*”\n
    File: *FILE_PATH*, Count: *STRING_COUNT*\n
    [File: *FILE_PATH*, Count: *STRING_COUNT*\n]*

  - If the string does not exist in any files, print “Not found”

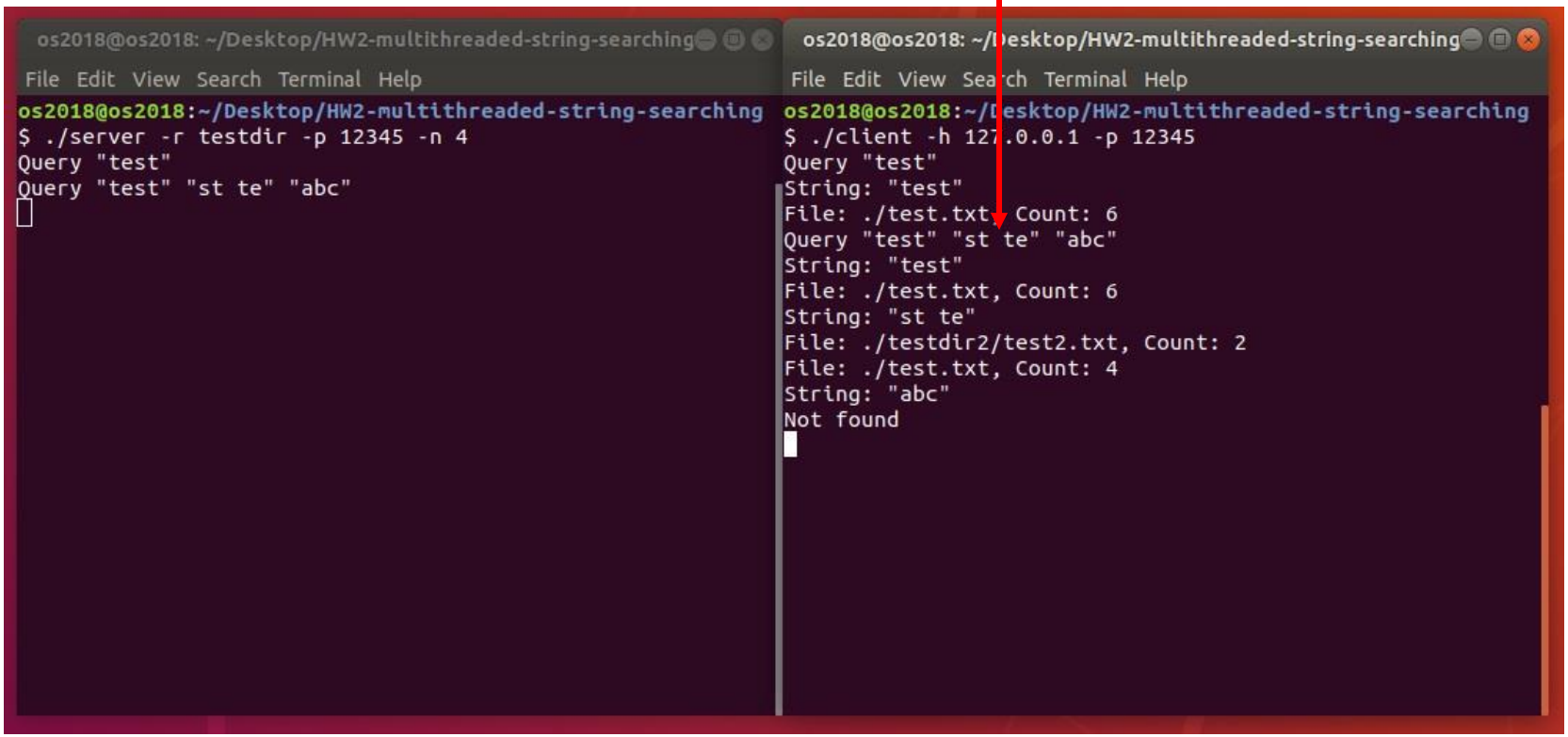
    Format: String: “*QUERY_STRING*”\n
    Not found\n

Example

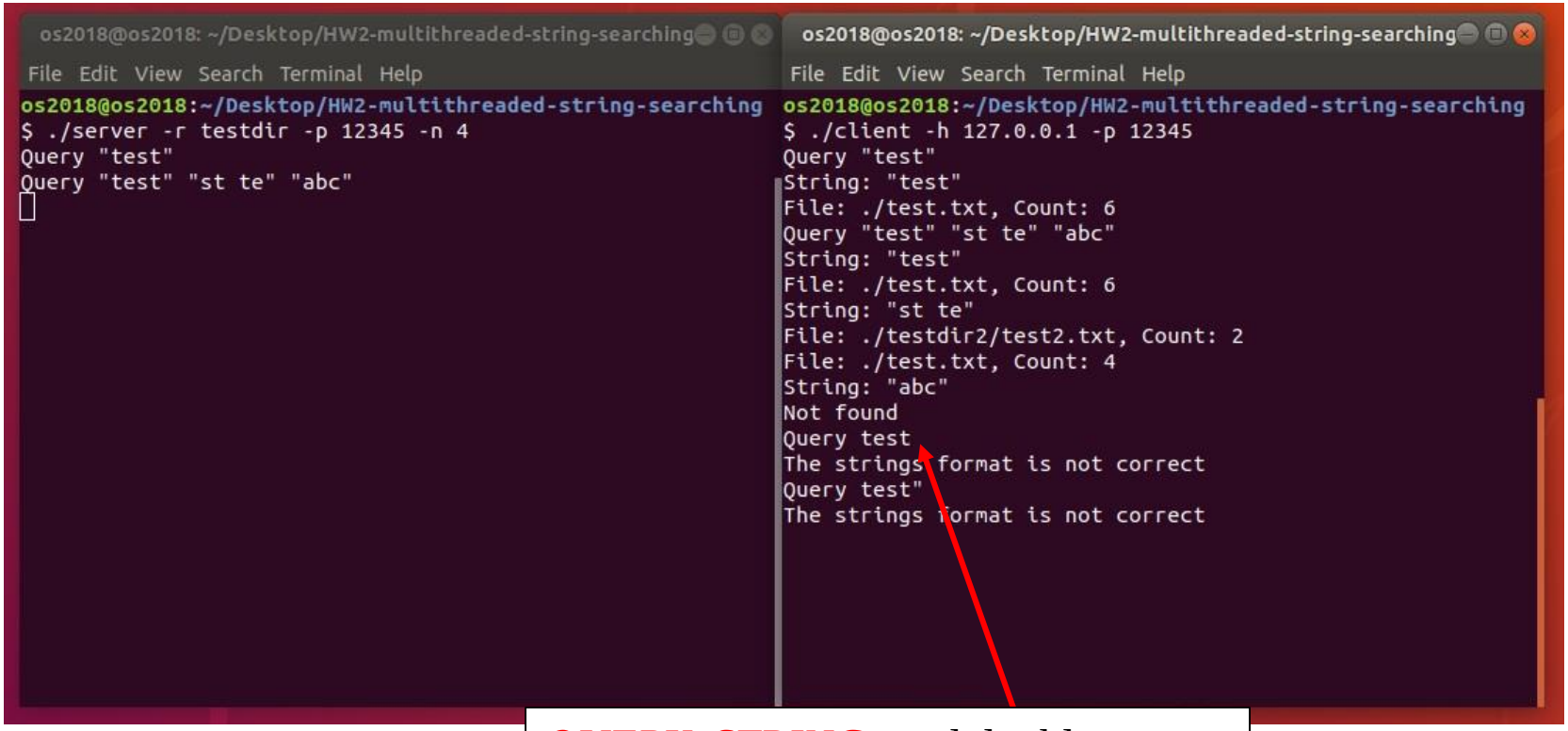


Example

QUERY_STRING might include space

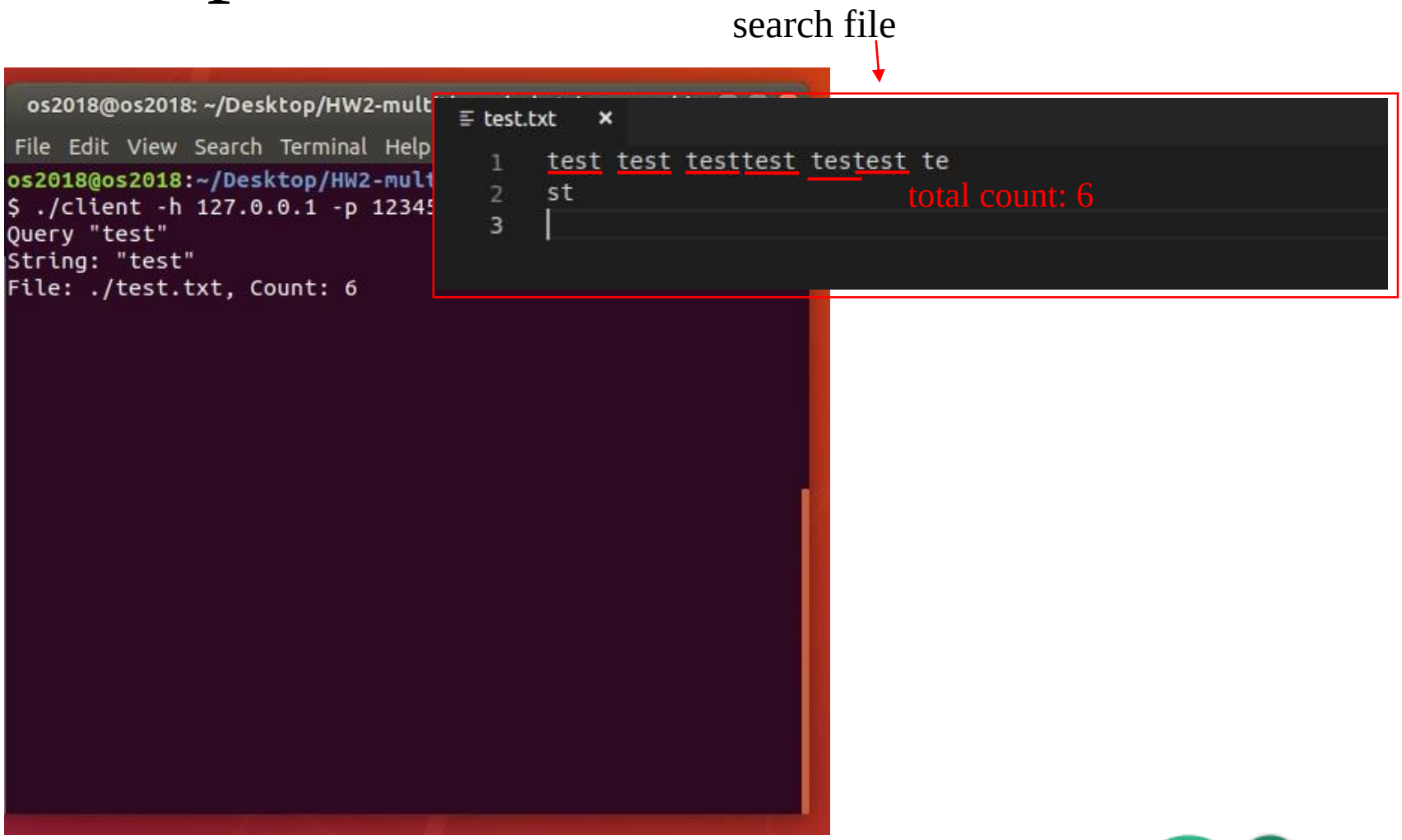


Example



QUERY_STRING need double quotes

Example



Reference

- Thread pool
- pthread
- socket
- Lock
  - pthread_mutex_lock , pthread_mutex_init
  - pthread_spin_lock , pthread_spin_init