

# COSC 3360 - Fundamentals of Operating Systems

[Dashboard](#) / [My courses](#) / [COSC3360SU2022](#) / [PROGRAMMING ASSIGNMENTS](#) / [Programming Assignment 2](#)


 [Description](#)



 [Submission](#)

 [Edit](#)

 [Submission view](#)

## Programming Assignment 2

 **Due date:** Thursday, 7 July 2022, 11:59 PM

 **Requested files:** client.cpp, server.cpp ( [Download](#))

**Type of work:**  Individual work

**Similarity Threshold:** 90%

### Objective:

This assignment will introduce you to interprocess communication mechanisms in UNIX using sockets.

### Problem:

You must write two programs to implement a distributed version of the multiple patterns search solution based on the Rabin–Karp algorithm you created for [programming assignment 1](#).

These programs are:

### The server program:

The user will execute this program using the following syntax:

```
./exec_filename port_no < input_filename
```

where exec\_filename is the name of your executable file, port\_no is the port number to create the socket, and input\_filename is the file's name with the text (string) the program uses to find the patterns it receives from the client program. The port number will be available to the server program as a command-line argument.

The server program reads from STDIN the text used to search the patterns it receives from the client program. This program receives multiple requests from the client program using sockets. Therefore, to handle these requests simultaneously, the server program creates a child process per request. For this reason, the parent process needs to handle zombies processes by implementing the fireman() call in the primer.

Each child process executes the following tasks:

1. Receives a pattern from the client program.
2. Uses the Rabin-Karp algorithm to determine the positions where the received pattern from the client appears in the text.
3. Returns the list of positions to the client program using sockets.

**Input Format:** Your program should read its input from STDIN (C++ cin) (Moodle uses input redirection to send the information from the input file to STDIN).

The input file has the text that the program uses to find the positions of the patterns received from the client.

**Example Input File:**

Carlos Alberto Rincon Castro

The server program will not print any information to STDOUT.

## The client program:

The user will execute this program using the following syntax:

`./exec_filename hostname port_no < input_filename`

where `exec_filename` is the name of your executable file, `hostname` is the address where the server program is located, `port_no` is the port number used by the server program, and `input_filename` is the name of the file with the list of patterns to search. The `hostname` and the `port` number will be available to the client as command-line arguments.

This program creates  $m$  child threads (where  $m$  is the number of patterns in the input file). Each child thread determines the positions where the assigned pattern appears in the text following the steps presented below:

1. Create a socket to communicate with the server program.
2. Send the pattern to the server program using sockets.
3. Wait for the list of positions from the server program.
4. Write the received information into a memory location accessible by the main thread.

**Input Format:** Your program should read its input from STDIN (C++ `cin`) (Moodle uses input redirection to send the information from the input file to STDIN).

The input file has  $m$  strings (one per line) representing the patterns to search.

**Example Input File:**

Ca
o
Rincon
w

Finally, after receiving the list of positions from the child threads, the main thread prints the search results. Given the previous input file, the expected output is:

Pattern "Ca" in the input text at position 0
Pattern "Ca" in the input text at position 22
Pattern "o" in the input text at position 4
Pattern "o" in the input text at position 13
Pattern "o" in the input text at position 19
Pattern "o" in the input text at position 27
Pattern "Rincon" in the input text at position 15
Pattern "w" not found

## Notes:

- You can safely assume that the input files will always be in the proper format.
- You must use the output statement format based on the example above.
- For the client program, you must use POSIX Threads and stream socket. A penalty of 100% will be applied to submissions not using POSIX Threads and Stream Sockets.
- You must use multiple processes (fork) and stream socket for the server program. A penalty of 100% will be applied to submissions not using multiple processes and Stream Sockets.

- The Moodle server will kill your server program after it is done executing each test case.

## Requested files

client.cpp

```
1 // Write your code here
```

server.cpp

```
1 // Write your code here
```

[VPL](#)

[◀ Programming Assignment 1](#)

Jump to...

[Theory Part - Practice Exam 2 ▶](#)