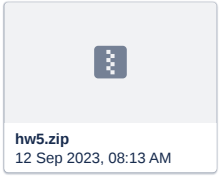
5. IPC with Message Passing & Shared Memory



hw5.zip
12 Sep 2023, 08:13 AM

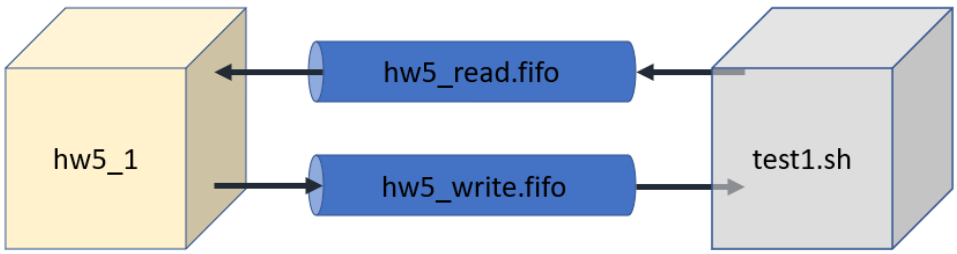
Overview

The Inter-Process Communication (IPC) mechanism in operating systems allows different processes to communicate and exchange data. Two common approaches of IPC are Message Passing and Shared Memory.

Part I

Descriptions

In Part I, you are required to use FIFO to implement message passing. FIFO, also known as FIFO queues or named pipes, is a communication mechanism that allows processes to send and receive data in a structured manner. It operates based on the principle of a queue, where the first message sent is the first to be received (hence "First-In-First-Out").



API references : mkfifo, open, read, write, sleep, close, unlink.

Requirement & Testing

1. Create and compile your C/C++ program.

   `gcc hw5_1_yourID.c -o hw5_1`

   or

   `g++ hw5_1_yourID.cpp -o hw5_1`

2. Run your program first.

   `./hw5_1`

   At this point, your program must have already created two FIFOs: `./hw5_read.fifo` and `./hw5_write.fifo`, and it should be attempting to read data from `hw5_read.fifo`.

3. Launch another terminal and execute the provided `test1.sh` script.

   `./test1.sh`

4. `test1.sh` will start to send a test case to your program.

   a. After receiving the test data string, you should first remove the trailing `'\n'` character from the string.

   b. Sort the characters within the string in lexicographical order.

   c. Append a `'\n'` character to the end of the string.

   d. Send it back to `test1.sh` using `./hw5_write.fifo`.

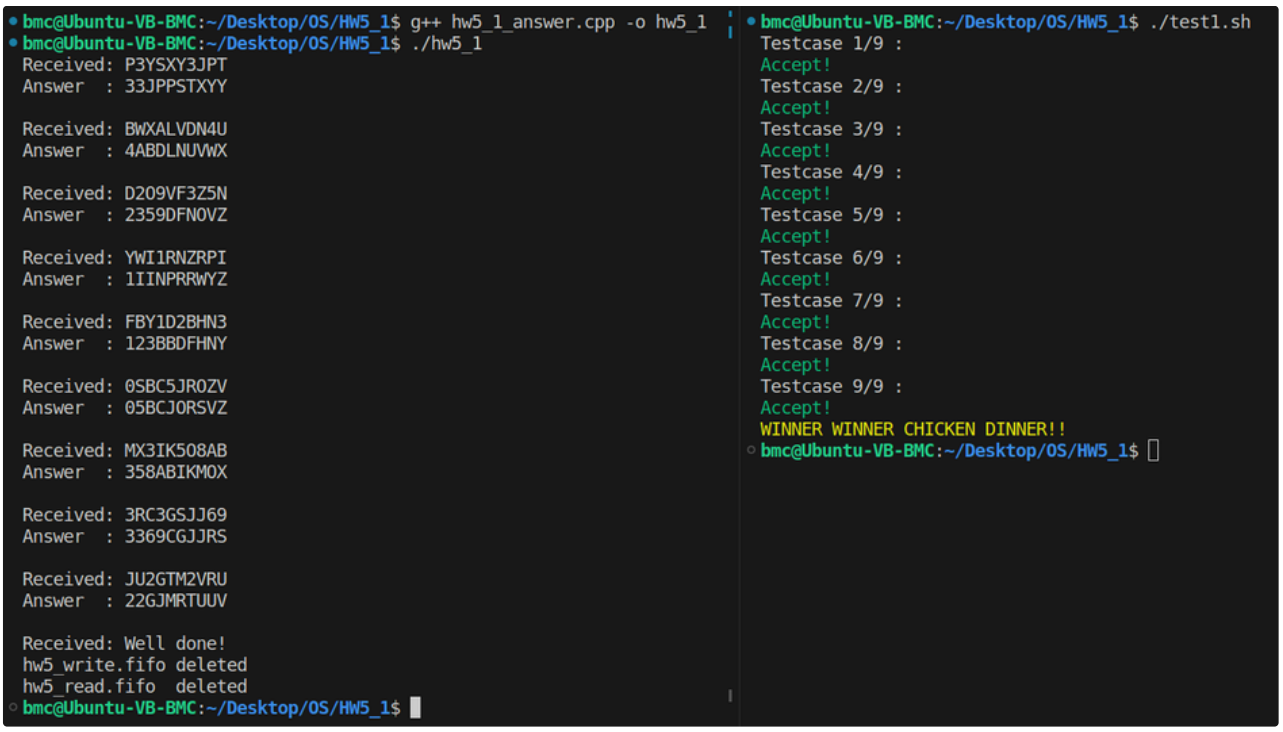
   e. Call `sleep(1)` to wait for one second to ensure that `test1.sh` has received the string.

5. `test1.sh` will check if the sorted characters are correct, and if they are, it will mark the test case with `"Accept!"`.

6. Your program will attempt to read data from `hw5_read.fifo` again and repeat steps 4 and 5.

7. When you receive the string `"Well done!\n"` and `test1.sh` outputs `"WINNER WINNER CHICKEN DINNER!!"`, it means you have passed all the test cases.

8. Finally, you must call `close()` and `unlink()` functions to close and delete two FIFOs.



Note

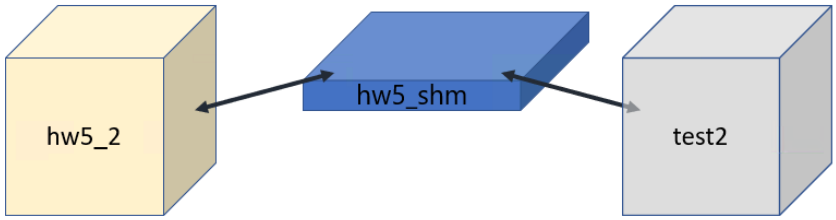
1. You can use C or C++ in this part. The file name should be `hw5_1_yourStudentID.c[pp]` .
2. Do not modify `test1.sh` .

3. If you cannot execute `test1.sh`, try `chmod`.
4. You can output any debug information in your program. The output `"WINNER WINNER CHICKEN DINNER!!"` in `test1.sh` is the only one that will affect your scores.
5. Be careful when handling `'\n'` or `'\0'` characters.

Part II

Descriptions

In Part II, you are required to use shared memory to achieve message communication between two processes. Shared memory allows multiple processes to share a portion of their virtual memory space, enabling efficient communication and data exchange.



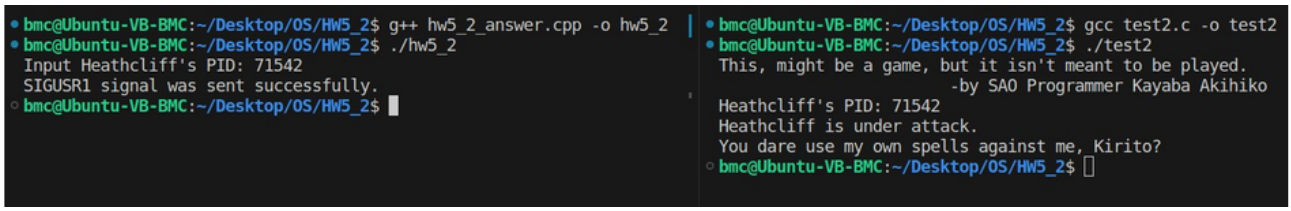
API references：shm_open, shm_unlink, ftruncate, mmap, munmap, kill, sleep, close.

Requirement & Testing

1. Compile the supplied `test2.c`.
   `gcc test2.c -o test2`
2. Run `test2`.
3. Launch another terminal. Create and compile your C/C++ program.
   `gcc hw5_2_yourID.c -o hw5_2`
   or
   `g++ hw5_2_yourID.cpp -o hw5_2`
4. Run your program. Enter the process ID (PID) of `test2`.
5. Your program should create a shared memory segment named `hw5_shm` using `shm_open()`, `ftruncate()` and `mmap()`. This will result in the creation of a shared memory object, `/dev/shm/hw5_shm`, on your system.
6. Write some data(?) to the shared memory.
7. Send a `SIGUSR1` signal to `test2`.
8. Call `sleep(1)`, then use `munmap()`, `close()` and `shm_unlink()` to release the shared memory.

Goal：Observe *Heathcliff*'s behavior (`test2`) and defeat him.

If successful, `test2` will output `"You dare use my own spells against me, Kirito?"` and then exit.



Note

1. You can use C or C++ in this part. The file name should be `hw5_2_yourStudentID.c[pp]` .
2. Do not modify `test2.c` .
3. You can output any debug information in your program. The output `“You dare use my own spells against me, Kirito?"` in `test2` is the only one that will affect your scores.

Report

You need to write a report answering the following questions :

- Part I (FIFO)
  a. A screenshot of your test results.
  b. What might happen if your program didn't call `sleep(1)` ? Why?
  c. What happens when a process writes to a FIFO, but there is no process reading from it?
- Part II (shared memory)
  a. A screenshot of your test results.
  b. How did you defeat *Heathcliff* ?
  c. What might happen if you reverse steps 6 and 7, meaning, sending `SIGUSR1` before writing the data?
- Any difficulties you encountered during the coding?

Submission

Please submit a **zip** file to E3, which contains your program sources and report.

- Make sure your code can be compiled and run on Ubuntu 20.04.4 LTS (or Ubuntu 22.04 LTS)
- Make sure your testing output is correct as mentioned.
- Your report should be submitted in PDF format.
- The structure of the zip file should be as the following:

```
<stduent_id>.zip
    |- <student_id>/
        |- hw5_1_<student_id>.c[pp]
        |- hw5_2_<student_id>.c[pp]
        |- hw5_<stduent_id>.pdf
```

For questions, please contact TA Mr. Chang <m2955121314.11@nycu.edu.tw>