Assignment 2B: Interprocess Communication with C Programming

User Documentation

Ray Krishardi Layadi - 26445549

Introduction

There are several ways to implement inter-process communication as a client-server model. This program supports 3 mechanisms which are named pipes (FIFO), message queues, and sockets. Each mechanism contains the corresponding client and server program which needs to be executed as a pair. The client program accepts input files from both command line input and input redirection. Upon receiving the input files, the client program will send the content of the files (i.e. list of Unix commands) to the server using each of the three inter-process communication mechanisms. On the other hand, the server program initially runs on the background and after receiving the Unix commands from the client via a particular inter-process communication mechanism, the server interprets and executes each of the Unix commands accordingly. Finally, after the server has executed and displayed the output of the Unix commands, the program will display a termination message to the user. The user can then press the “enter” key to exit the program.

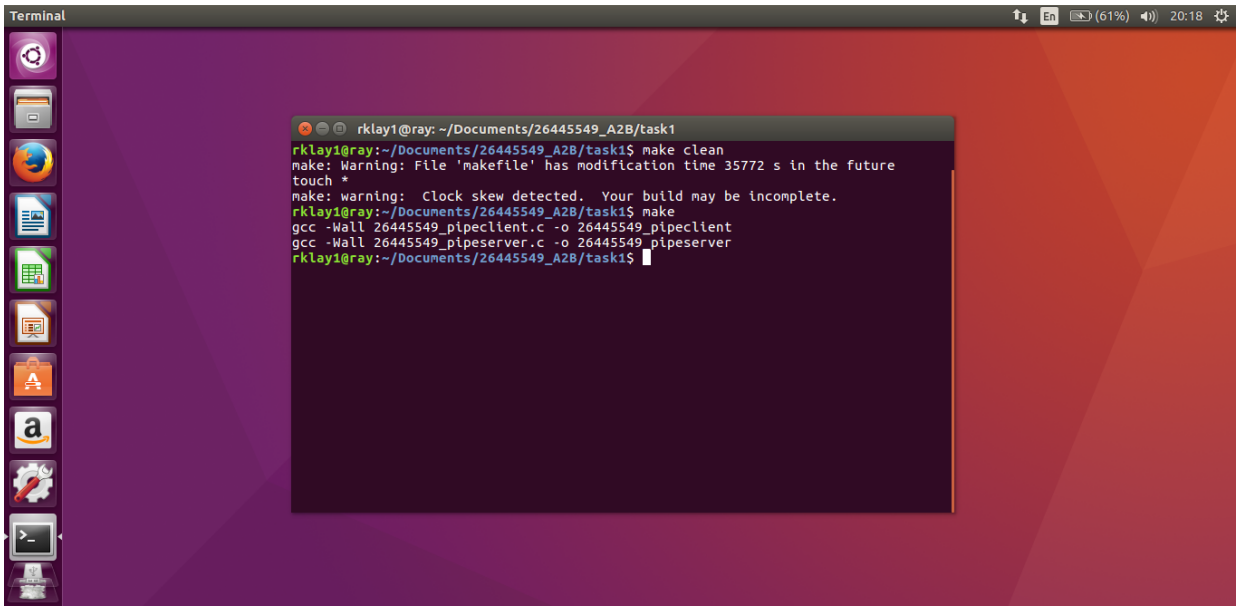
How to use each of the IPC (Inter-process communication) mechanism program

1. Build the IPC mechanism program with “make” command

macOS Sierra (version 10.12.5)

```
haha:task1 ray$ make clean
touch *
haha:task1 ray$ make
gcc -Wall 26445549_pipeclient.c -o 26445549_pipeclient
gcc -Wall 26445549_pipeserver.c -o 26445549_pipeserver
haha:task1 ray$
```

Ubuntu 16.04 LTS



Description

To build the program, the following commands are used:

- make clean
- make

Note: The “make clean” command will execute the “touch *” command which is used to fix the “make: warning: Clock skew detected. Your build may be incomplete.” warning in Ubuntu 16.04 LTS.

2. Run the IPC mechanism program by running the client and server program as a pair OR by running the main.sh shell script

1. Run the IPC mechanism program by running the client and server program as a pair

```
haha:task1 ray$ ./26445549_pipeserver &
[1] 3601
haha:task1 ray$ ./26445549_pipeclient < commands.txt
```

OR

```
haha:task1 ray$ ./26445549_pipeserver &
[1] 3601
haha:task1 ray$ ./26445549_pipeclient commands.txt
```

Description

The server program must run in the background. This could be achieved by using the ampersand symbol (&). On the other hand, the client program must run in the foreground and require input file(s) from the user which can be passed via command line input or as an input redirection. The given input file must contain list of Unix commands to be executed by the server.

2. Run the IPC mechanism program by running the main.sh shell script

```
haha:task1 ray$ chmod +x main.sh
```

```
haha:task1 ray$ ./main.sh < commands.txt
```

OR

```
haha:task1 ray$ ./main.sh commands.txt
```

Description

In order to run the main.sh shell script, execute permission must be given to the script. Hence, the command “chmod +x main.sh” is required. The script then requires input file(s) from the user which can be passed via command line input or as an input redirection. The given input file must contain list of Unix commands to be executed by the server. The script will then run the server program in background and run the client program in foreground with the given input file(s) as the parameter for the client program.

Additional functionality:

- Support multiple input file(s) from the user which can be passed via command line input or as an input redirection
    - Note: Passing multiple input file(s) as input redirection will only execute the first input file

Passing multiple input files via command line input

```
haha:task1 ray$ ./main.sh commands.txt commands.txt


client: echo FIT2100
server: FIT2100
client: date
server: Thu 19 Oct 2017 21:04:08 AEDT
client: pwd
server: /Users/ray/Desktop/Monash/Sem 2 2017/FIT2100/26445549_A2B/task1
client: ls -l commands.txt
server: -rw-r--r--@ 1 ray  staff  57 19 Oct 20:15 commands.txt
client: wc commands.txt
server:        5       9      57 commands.txt
client: echo FIT2100
server: FIT2100
client: date
server: Thu 19 Oct 2017 21:04:08 AEDT
client: pwd
server: /Users/ray/Desktop/Monash/Sem 2 2017/FIT2100/26445549_A2B/task1
client: ls -l commands.txt
server: -rw-r--r--@ 1 ray  staff  57 19 Oct 20:15 commands.txt
client: wc commands.txt
server:        5       9      57 commands.txt
Press enter to exit the program...
```

Passing multiple input files via input redirection

```
haha:task1 ray$ ./main.sh < commands.txt commands.txt


client: echo FIT2100
server: FIT2100
client: date
server: Thu 19 Oct 2017 21:09:10 AEDT
client: pwd
server: /Users/ray/Desktop/Monash/Sem 2 2017/FIT2100/26445549_A2B/task1
client: ls -l commands.txt
server: -rw-r--r--@ 1 ray  staff  57 19 Oct 20:15 commands.txt
client: wc commands.txt
server:        5       9      57 commands.txt
Press enter to exit the program...
```

Sample input file

```
haha:task1 ray$ cat commands.txt
echo FIT2100
date
pwd
ls -l commands.txt
wc commands.txt
```

Sample output

```
client: echo FIT2100
server: FIT2100
client: date
server: Thu 19 Oct 2017 21:19:52 AEDT
client: pwd
server: /Users/ray/Desktop/Monash/Sem 2 2017/FIT2100/26445549_A2B/task2
client: ls -l commands.txt
server: -rw-r--r--@ 1 ray  staff  57 19 Oct 21:11 commands.txt
client: wc commands.txt
server:        5       9      57 commands.txt
Press enter to exit the program...
```

```
client: echo FIT2100
server: FIT2100
client: date
haha:task2 ray$ server: Thu 19 Oct 2017 21:11:45 AEDT
client: pwd
server: /Users/ray/Desktop/Monash/Sem 2 2017/FIT2100/26445549_A2B/task2
client: ls -l commands.txt
server: -rw-r--r--@ 1 ray  staff  57 19 Oct 21:11 commands.txt
client: wc commands.txt
server:        5       9      57 commands.txt
Press enter to exit the program...
```

Note:

- Occasionally, the client program finishes its execution while the server program is still executing the input file. As a result, as indicated by the red rectangle, the output of the program includes a prompt. Therefore, in order to eliminate this issue, it is recommended that the user runs the IPC mechanism program by running the main.sh shell script.

3. **Press the “enter” key to exit the program**

```
client: echo FIT2100
server: FIT2100
client: date
server: Thu 19 Oct 2017 21:23:43 AEDT
client: pwd
server: /Users/ray/Desktop/Monash/Sem 2 2017/FIT2100/26445549_A2B/task1
client: ls -l commands.txt
server: -rw-r--r--@ 1 ray  staff  57 19 Oct 20:15 commands.txt
client: wc commands.txt
server:        5       9      57 commands.txt
Press enter to exit the program...
[1]+  Done                    ./26445549_pipeserver
haha:task1 ray$
```