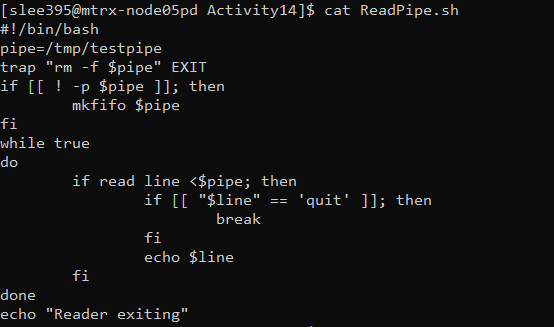
**UNX510 – Act14**

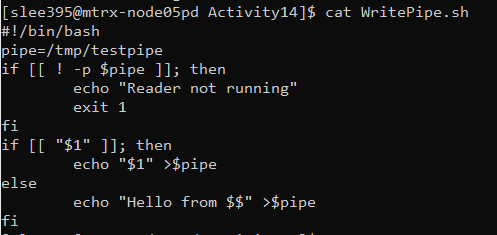
**Sanghyuk Lee(129405171)**

**Question 1)** Complete the following steps:

**Step 1:** Create ReadPipe.sh as follow:



**Step 2**: Create WritePipe.sh as follow:



**Step 3**: Run the Scripts as follow:

$ readpipe.sh &

[3] 23842

$ writepipe.sh

Hello from 23846

$ writepipe.sh

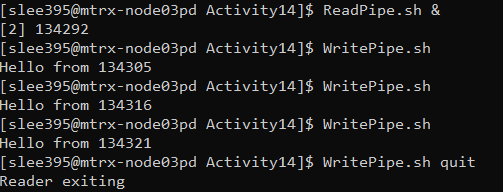
Hello from 23847

$ writepipe.sh

Hello from 23848

$ writepipe.sh quit

Reader exiting



**Step 4**: Based on the observation, explain how the two scripts communicate together/work.

Also answer the following questions:

ReadPipe.sh simply read the value from $pipe and WritePipe.sh displays the value.

Once you do ReadPipe.sh & then it goes into background and wait until the new process comes in to do action. And then you type WritePipe.sh and redirect the output to $pipe so that ReadPipe.sh can do action based on argument of WritePipe.sh

The reason why WritePipe.sh gets different PID is whenever you execute command on shell, it creates a new child process.

In detail, Inside the WritePipe.sh, it checks if the variable pipe, which is variable assigned PATH, is pipe or not. If it is pipe then it checks if there is a argument or not and based on the argument it sends the content into pipe.

Inside the ReadPipe.sh, it checks whether given content is same as the ‘quit’ and if it is it finish the process and it if is not, then it displays what it got from the pipe.

1. What is the use of “trap” command in ReadPipe.sh?

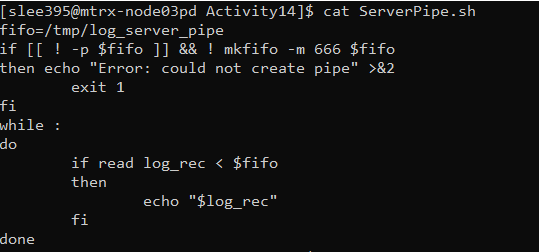
– trap specifies action to take when a signal is detected. In this case, when script it exiting, remove the $pipe, which is /tmp/testpipe.

2) What do “$1” and “$$” refer to (in WritePipe.sh)?

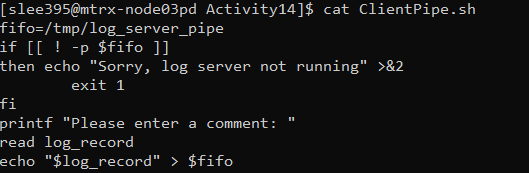
- $1 stands for first argument and $$ stands for process id.

**Question 2**) Complete the following steps:

**Step 1**: Create ServerPipe.sh as follow:



**Step 2**: Create ClientPipe.sh as follow:



**Step 3**: Demonstrate/Explain how the above Server/Client scripts work.

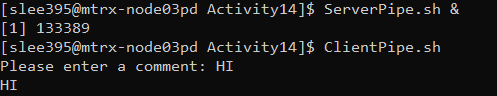
Client script sends a comment to a server and Server scripts wait a content.

In other words, Client pass the input to a Server and then Server is going to display output.

In detail, inside the ClientPipe.sh, it checks whether the variable fifo is pipe or not and if it is not then it accepts the input from the user. And it redirects the output the the variable fifo which is pipe.

In ServerPipe.sh, it checks if fifo is pipe and permission MODE is set to 666. And if it is it accepts the value from fifo into a new variable log\_rec and echo it.

**Execution)**

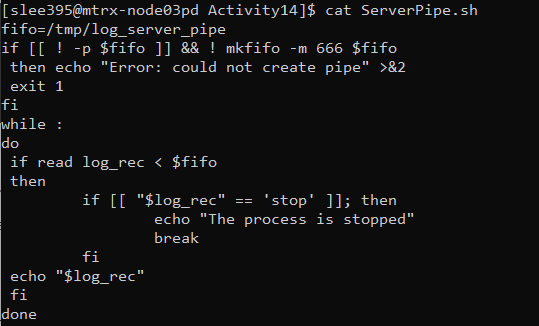


**Step4**: What does mkfifo -m 666 $fifo do? Explain your answer

mkfifo -m does set file permission bits to MODE, not a=rw – unmask. So ‘mkfifo -m 666 $fifo’ set a permission MODE to 666.

**Step 5**: How to stop the server\_pipe? Modify the script to stop the script if user enter a comment “stop”.

**Code)** To execute required action, I added a one if statement below when it reads and redirect the content into a variable log\_rec. And checks if the user input is stop then it echoes “The process is stopped” and break the if statement.



**Execution)**

