Operating Systems
Assignment-2

Submission Date
19/04/2022 07 AM (Morning)

Instructions:

- This is an individual Assignment.
- All parties involved in any kind of plagiarism/cheating (even in a single line of code) will be given zero marks in all the assignments.
- Assignment deadline won't be extended
- **Late submissions will be discarded so submit your assignment on-time**
- You must follow below submission guidelines, otherwise your submission won't be accepted
    - create a new directory
    - change its name to the following format
      YOURSECTION_ROLLNUMBER_NAME.
      E.g. A_20I-0012_Muhammad Ahamd
    - put all your files (.c / .cpp) into this newly created directory
    - Compress the directory into a compressed .zip file
    - Submit it on google classroom
- Use good programming practices (well commented and indented code; meaningful variable names, readable code etc.).

Question No. 1 [10 marks]

ls -l / **|** grep a **|** sort -r **|** wc **>** count.txt

Write a C/C++ program for the above commands using **exec**, **unnamed pipe** and **dup**

Question No. 2 [10 marks]

Write four C/C++ programs that are connected using named pipes. The first program take input from user and pass the data to other programs. The other programs calculate the following things respectively:

- Vowels
- Reverse
- Capitals

Question No. 3 [10 marks]

Write a C/C++ program in which a parent process represents an owner and it has ten employees. The owner wants to arrange a meeting with all employees, and he requires input from all employees. The owner suggests a time to all employees. A meeting will be schedule after the confirmation of all the employees' using pipes.

Question No. 4 [15 marks]

Write a C/C++ program to create the following process Tree.

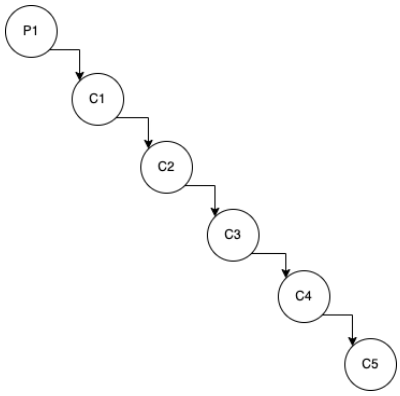

Figure 1 Process Tree

The process will connect each other using unnamed pipes (shown as white arrows) and form a virtual chain/ring of communication as shown below:

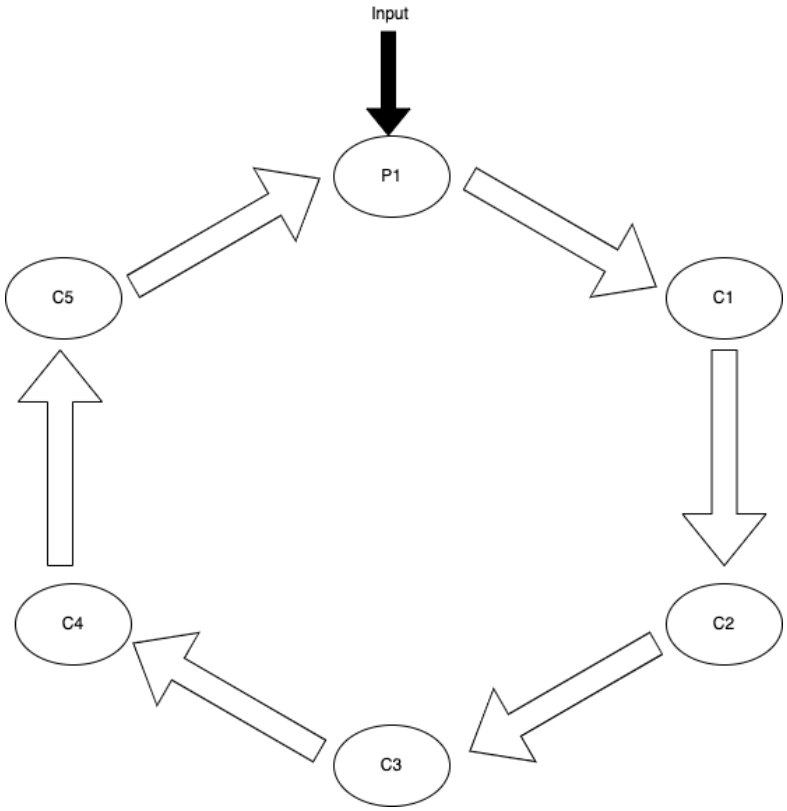

Figure 2 The interprocess communication between the processes using unnamed pipes

Consider P1 gets Pid as 7003, C1 as 7004 and so on. The first process P1 will get an input from the user (e.g. “Hello”) and pass the message to the next process (i.e. C1), by appending its own process id (in character form). Before sending/passing the message to the next process, It also writes the current output to the screen. The next process will read the message from its incoming pipe, append ':' and its own process id, and write the resulting message to the output screen and then its outgoing pipe. Eventually the message should get back to P1 which should read the message and write it to output screen. The output will eventually look like this for the 6 processes (an example output):

Hello:7003
Hello:7003:7004
Hello:7003:7004:7005
Hello:7003:7004:7005:7006
Hello:7003:7004:7005:7006:7007
Hello:7003:7004:7005:7006:7007:7008

The following conditions are mandatory for the above program:

1. The unnamed pipes needs to be visible only to the concerned process (where possible) i.e. No other process should preferably know about the existence of pipe between the two process.
1. You can pass data between processes using **stdin/stdout** system calls only. You cannot use the usual read/write system calls to read/write to/from the pipes.
2. The program quits when user types "Quit" as input.

Hint: You have to use **fork, pipe and dup/dup2** system calls.

Question No. 5 [15 marks]

Consider the same process tree as above. The virtual ring/chain between the processes is also same like the last question except for one difference i.e. there is a two-way communication between all the processes in the ring:

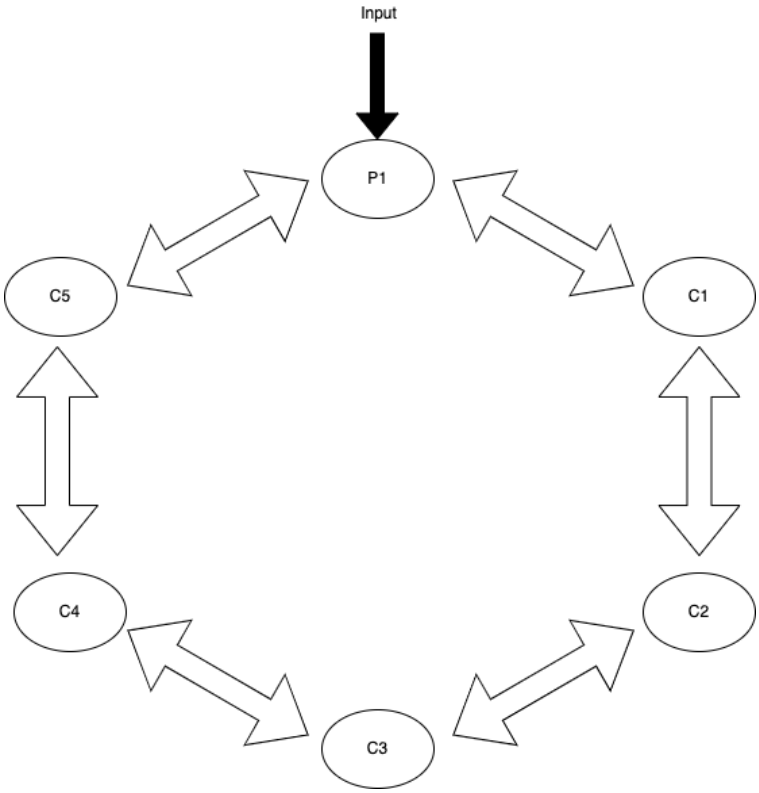

Figure 3 The interprocess communication between the processes using unnamed pipes

Consider P1 gets Pid as 7003, C1 as 7004 and so on. The first process P1 will get an input from the user (e.g. "Hello") and pass the message to the next process (i.e. C1), by appending its own process id (in character form). Before sending/passing the message to the next process, It also writes the current output to the screen. The next process will read the message from its incoming pipe, append

':' and its own process id, and write the resulting message to the output screen and then its outgoing pipe. Eventually the message should get back to P1 which should read the message and write it to output screen. The output will eventually look like this for the 6 processes (an example output):

Hello:7003
Hello:7003:7004
Hello:7003:7004:7005
Hello:7003:7004:7005:7006
Hello:7003:7004:7005:7006:7007
Hello:7003:7004:7005:7006:7007:7008

For the next input (e.g. World), the messages will pass in anti clock wise direction i.e.

World:7003
World:7003:7008
World:7003:7008:7007
World:7003:7008:7007:7006
World:7003:7008:7007:7006:7005
World:7003:7008:7007:7006:7005:7004

For the next input the program will pass messages clockwise and so on.

Like The above program following conditions are mandatory for the this program:

2. The unnamed pipes needs to be visible only to the concerned process (where possible) i.e. No other process should preferably know about the existence of pipe between the two process.
3. You can pass data between processes using **stdin/stdout** system calls only. You cannot use the usual read/write system calls to read/write to/from the pipes.
4. The program quits when user types “Quit” as input.

Hint: You have to use **fork, pipe and dup/dup2** system calls.