Birla Institute of Technology & Science, Pilani
Pilani Campus

I SEMESTER 2020-2021
Assignment-1

Course No.: IS F462 **Course Title:** Network Programming
Deadline: 4th Oct 2020 **Maximum Marks:** 48M (12%)

Note:

- Maximum of two students per group. Upload code in Canvas
- Name your file idno1_idno2_assignment1.tar .
- Group information to be entered here: https://docs.google.com/spreadsheets/d/1a3_OsjEIQQlbpoXD_vX2sZcLYawRpoYtJZQYbkYfmUo/edit?usp=sharing

---

P1. You are required to build a bash-like shell for the following requirements. Your program should not use temporary files, popen(), system() library calls. It should only use system-call wrappers from the library. It should not use *sh* or *bash* shells to execute a command.

a) Shell should wait for the user to enter a command. User can enter a command with multiple arguments. Program should parse these arguments and pass them to execv() call. For every command, shell should search for the file in PATH and print any error. Shell should also print the pid, status of the process before asking for another command.

b) shell should support any number of commands in the pipeline. e.g. `ls|wc|wc|wc`. Print details such as pipe fds, process pids and the steps. Redirection operators can be used in combination with pipes.

c) shell should support two new pipeline operators "||" and "|||". E.g.: `ls -l || grep ^-, grep ^d`. It means that output of ls -l command is passed as input to two other commands. Similarly, "|||" means, output of one command is passed as input to three other commands separated by ","...

d) shell should mask all signals except SIGQUIT and SIGINT. When SIGINT is received, it should print last 10 commands executed by the user, along with status of each. When SIGQUIT is pressed, it should ask user "Do you really want to exit?". If yes, it should exit

Deliverables:

- Brief Design Document (.pdf)
- shell.c

[20 M]

P2. Implement a client-server system within an operating systems using IPC mechanisms such as message queues. The requirements are as follows.

a) The system is file storage and data processing system. It has metadata server (M), data servers (D) and a client (C).

b) A client submits request to add a file at a given path to the M server. M server makes changes to file hierarchy and replies.

c) Then client divides the file into blocks of CHUNK_SIZE (taken as input). For each chunk, C will send ADD_CHUNK request to M server. M returns list of addresses of three D servers. C will send chunk of data directly to D. D will reply with status. D stores each chunk as a file within its directory.

d) To copy file from one path to another, C sends CP request to M. M will change metadata and reply. M marks the chunks to be copied. In the next communication to D servers, M instructs them where to copy the chunk.

e) To move file, C sends MV request. M changes file hierarchy and reply.

f) To delete file, C sends RM request. M changes file hierarchy and marks chunks to be deleted. In the next communication between M and D, M will instruct Ds to remove the chunks.

g) C can send a command to Ds to be executed on the chunks stored and send the result back to C. For e.g. C sends wc –w/m/l/c to D servers. D servers execute this command on their local chunks and send the results back. Other e.g. may be cat, od, ls, sort …

Deliverables:

- m_server.c, d_server.c, client.c
- pdf file explaining design decisions

[28 M]

--&--