Birla Institute of Technology & Science, Pilani
Pilani Campus

II SEMESTER 2019-2020
Assignment-1

Course No.: IS F462 **Course Title:** Network Programming
Deadline: 15th Mar 2020 **Maximum Marks:** 48M (12%)

Note:

- Maximum of two students per group.
- Upload code in https://nalanda.bits-pilani.ac.in Name your file idno1_idno2_assignment1.tar .
- Group information to be entered here:

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 a command `shmcreate <size>` that creates a shared memory segment. `shmls` should list all shared memory segments created with some id. When a command such as `ls > <shm_id>` is executed, the output should go to corresponding shared memory segment. Same can be done with input redirection like `wc < <shm_id>`. When >> is given, data should be appended. In all cases, if no space left, it should give error message.

d) Shell should support a command `tcpcreate <ip> <port>`. This should output an identifier. `tcpls` should list all such ids with ip and port. When `ls > <tcp_id>` is entered, the output of ls should be sent to the server. Similarly, >>, < carry the same semantics as described above.

Deliverables:

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

[20 M]

P2. Write C program webclient.c that does the following.

Client takes server ip, port, N and URL of a file as command line arguments. Client is supposed to download the file (of huge size) using multiple connections established by a pool of N child processes. Each child process establishes a connection and sends the following request (given

below). URL (after GET) will be the url given in command line. This requests bytes from 500 to 999. This range will be different in each child process. This can be indicated to a child process from parent.

```
GET /index.pdf HTTP/1.1
Host: xyz.com
Range: bytes=500-999
```

The response will be of the format:

```
HTTP/1.1 200 OK
Date: Mon, 25 Jan 1999 18:55:17 GMT
Server: Apache/1.2.6 Red Hat
Last-Modified: Sun, 18 Oct 1998 22:45:13 GMT
Content-Length: 18    //length of content that arrives after headers
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Type: octet-stream/pdf
                      //blank line to indicate end of headers
<data follows>
```

If the code returned is 200, child sends the data to parent. If the code returned is something else, child sends the error code and the error message to the parent. In error case, parent should fork another child and repeat the same. Finally, parent assembles the parts and stores as file, displays a success message and exits.

Deliverables:

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

[28 M]

--&--