# Multiprocessing assignments

**Mandatory**

**Section A – Process basics, System calls**

1. **What is the difference between a program and process?**

Ans) A **Program** is an executable file which contains a certain set of instructions written to complete the specific job or operation on your computer.

A **Process** is an execution of a specific program. It is an active entity that actions the purpose of the application. Multiple processes may be related to the same program.

1. **What are the attributes associated with program and process?**

Ans) A process has various attributes associated with it: -

1. Process Id- Every process will given an ID called process ID so that we can uniquely identify that process from other process.
2. Process state- Each and every process has some state associated with it at a particular instant of time. It can be new , ready or running
3. CPU Scheduling- There is number of processes going on but CPU respond to a particular process at a time for that it follows scheduling algorithm. There are different kind os scheduling algorithms
4. Priority- Process with the highest priority get the CPU first for its execution.
5. Process counter- Stores the address of the last instruction of the process on which the process was suspended.
6. I/O information-Each process needs some I/O devices for their execution.
7. **Which command is used for listing out the attributes of program and process?**

Ans) “ps “command is used to show some attributes of a process. This command reads through the kernel’s data structures and process tables to fetch the characteristics of a process. By default, ps command displays the processes owned by the user running the command.

1. **How do we uniquely identify processes in Unix/?**

Ans) Each process in the system has a unique PID.

By giving [ $> ps -ef] command we can get the PPID and PID of different processes.

1. **How do we create processes in UNIX environment? (Name the system calls used)**

Ans) There are three distinct phases in the creation of process:

1)Forking

2)Overlying and Execution

3)Waiting

After the fork () there will be two processes, one is parent process and other will be child process. Child process have new PID. Parents keep waiting for the child process to complete its task. By calling system call wait (). The second phase created by exec () function. At the last exit () terminates the child process. The parent awakens only when it receives a complete signal from the child ,after which it will be free to continue with its other function.

1. **Which command is used to view list of system calls used by a process?**

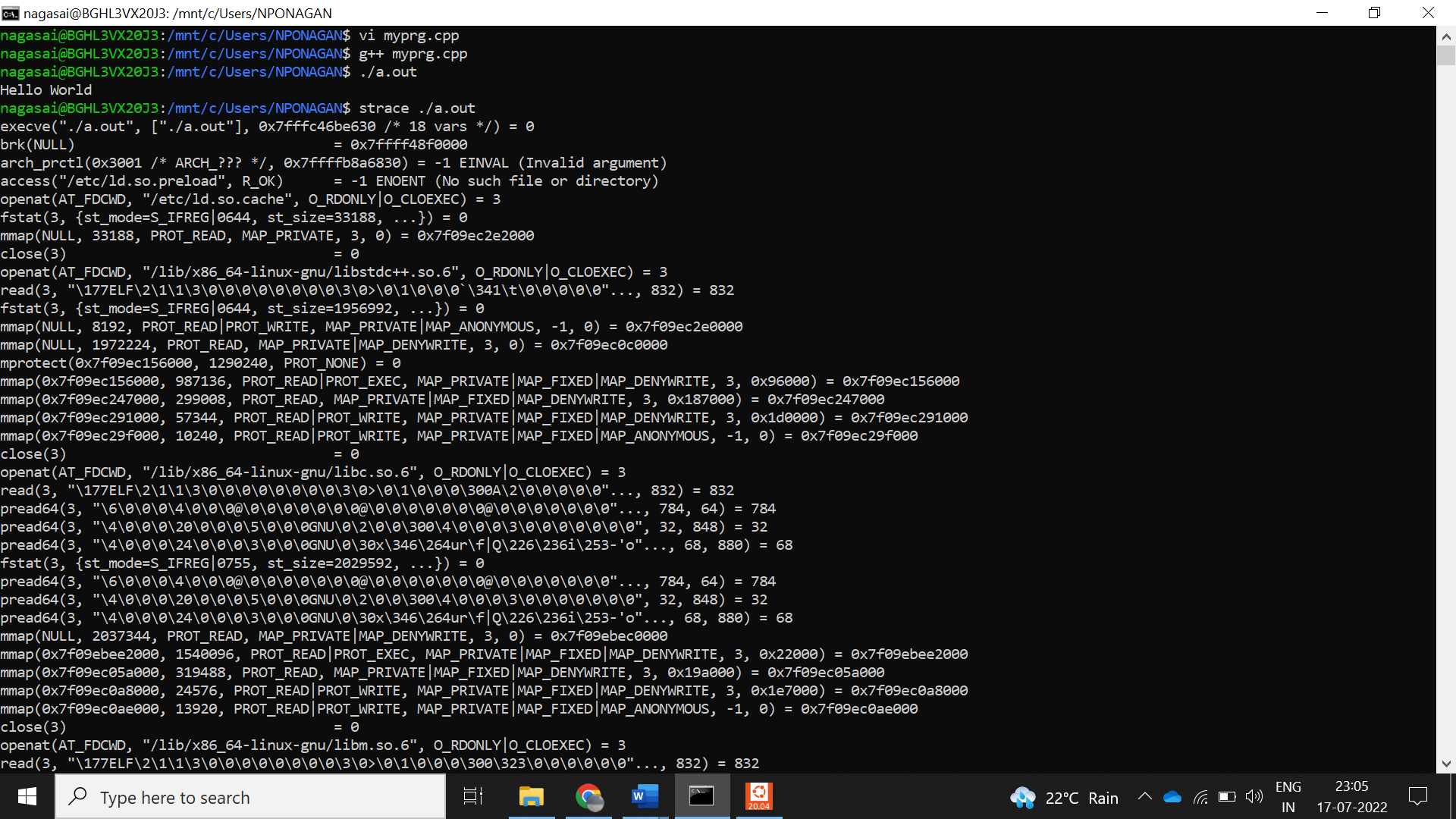
Ans) Strace command is used. It traces system calls and signals. It intercepts and record the system calls which are called by the process.

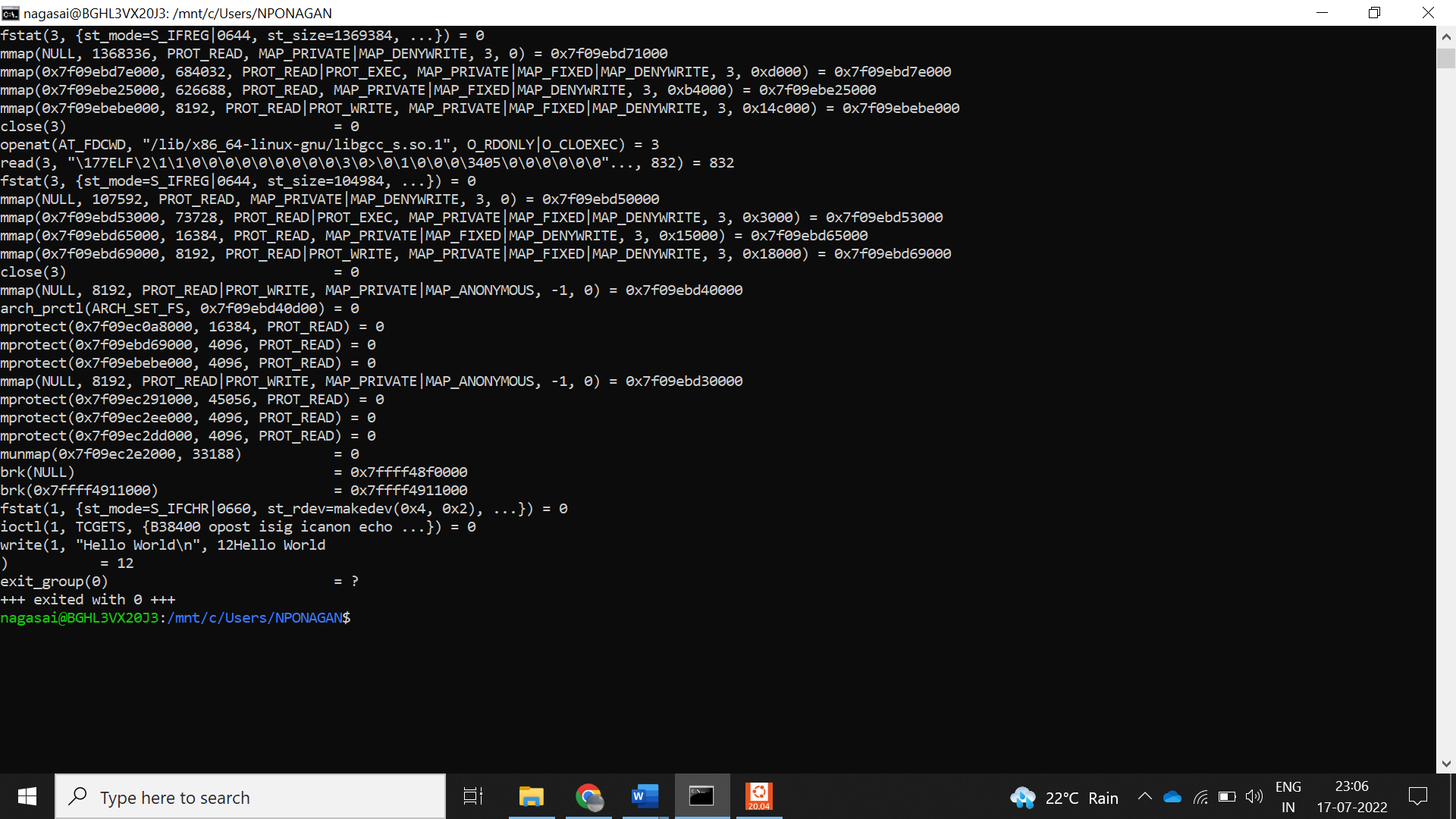
The name of each system call, its arguments and its return value are printed on standard error or the file specified with -o option.

1. **Choose any working file program created by you, rename as myprg, run the command below and view the output**

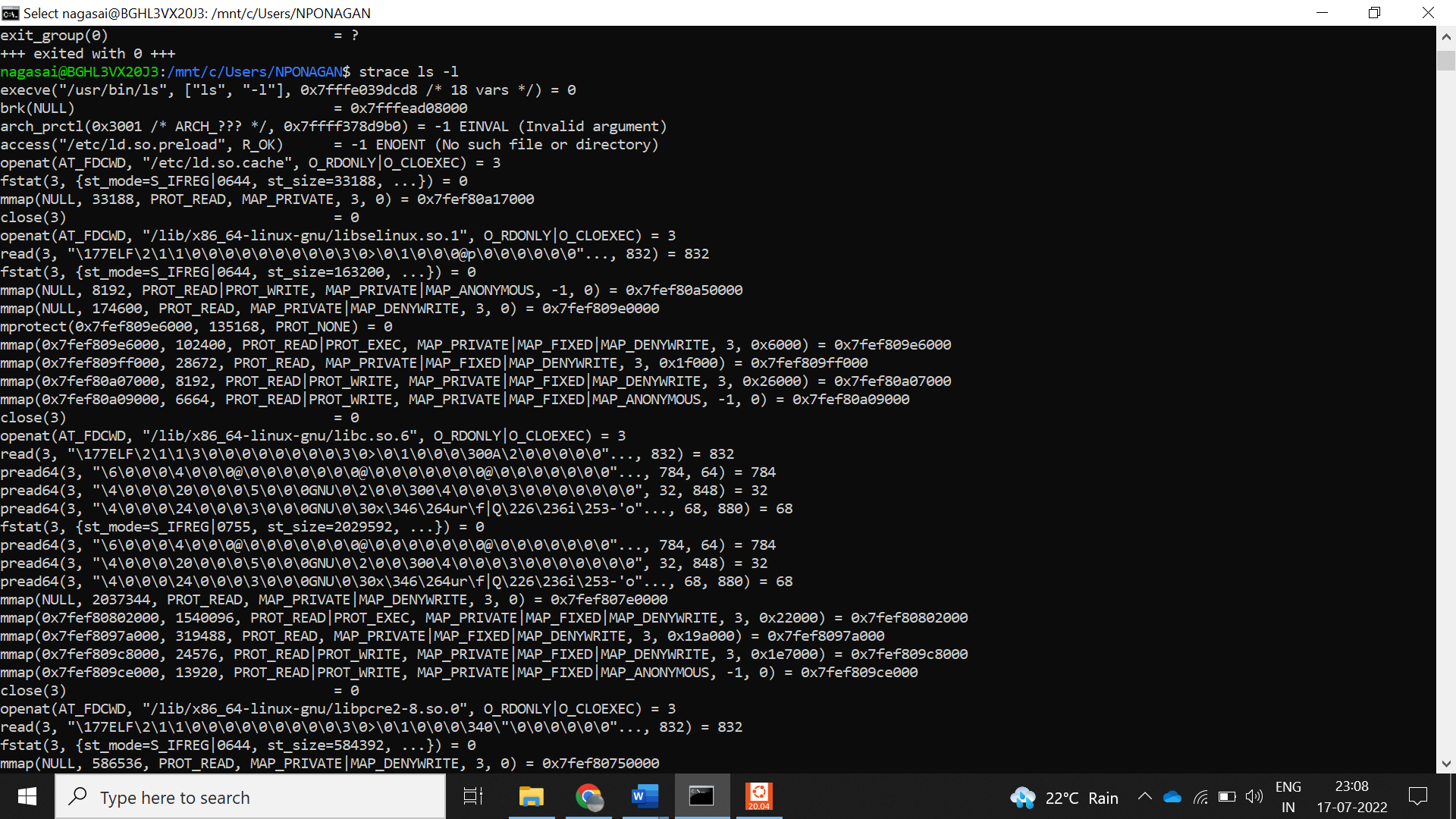
**$>strace ./myprg**

**Identify atleast 2 system calls used by your program**





1. **Run strace command on “ls -l” and identify atleast 4 system calls related to file**



1. **What is the first output line of command “strace ls -l”? What does it mean?**

Ans) execve("/usr/bin/ls", ["ls", "-l"], 0x7fffcda0f5d8 /\* 19 vars \*/) = 0

1. **Which system call is used to launch processes from command line?**
2. **Execute the command below, read and understand output of all functions in this family.**

**man exec**

**What is the difference between execlp and execvp?**

Ans) int execlp(const char \*file, const char \*arg, ………………………………….. /\* (char \*) NULL \*/);

int execvp(const char \*file, char \*const argv[]);

Difference:

l - execlp()

The const char \*arg and subsequent ellipses can be thought of as arg0, arg1, ..., argn. Together they describe a list of one or more pointers to null-terminated strings that represent the argument

list available to the executed program. The first argument, by convention, should point to the filename associated with the file being executed. The list of arguments must be terminated by a null

pointer, and, since these are variadic functions, this pointer must be cast (char \*) NULL.

By contrast with the 'l' functions, the 'v' functions (below) specify the command-line arguments of the executed program as a vector.

v - execvp()

The char \*const argv[] argument is an array of pointers to null-terminated strings that represent the argument list available to the new program. The first argument, by convention, should point to

the filename associated with the file being executed. The array of pointers must be terminated by a null pointer.

1. **What is the difference between \_exit and exit? When can we use \_exit?**

Ans) \_exit and \_Exit, which return to the kernel immediately, and exit, which performs certain cleanup processing and then returns to the kernel.

The key difference between exit() and \_exit() function lies in the fact that exit() function performs some cleanup tasks before terminating the program. These include clearing the buffer, terminating the connection, etc. On the other hand \_exit() does not perform any such cleaning operation. It simply terminates the program. \_Exit() function does not make any call to object destructors or the functions registered with atexit() or at\_quick\_exit() functions.

1. **What will happen to the child if a parent terminates before a child? Who will do the cleanup on child exit?**

Ans) When we kill parent process then child process will become Orphan and child process is adopt by init process

1. **What do you mean by COW? (Copy on write)? Which system call uses this concept?**

Ans) The idea behind a copy-on-write is that when a parent process creates a child process then both of these processes initially will share the same pages in memory and these shared pages will be marked as copy-on-write which means that if any of these processes will try to modify the shared pages then only a copy of these pages will be created and the modifications will be done on the copy of pages by that process and thus not affecting the other process.

Fork() system call uses this concept

1. **Run command below, view the output and answer queries below.**

$>ps -ef

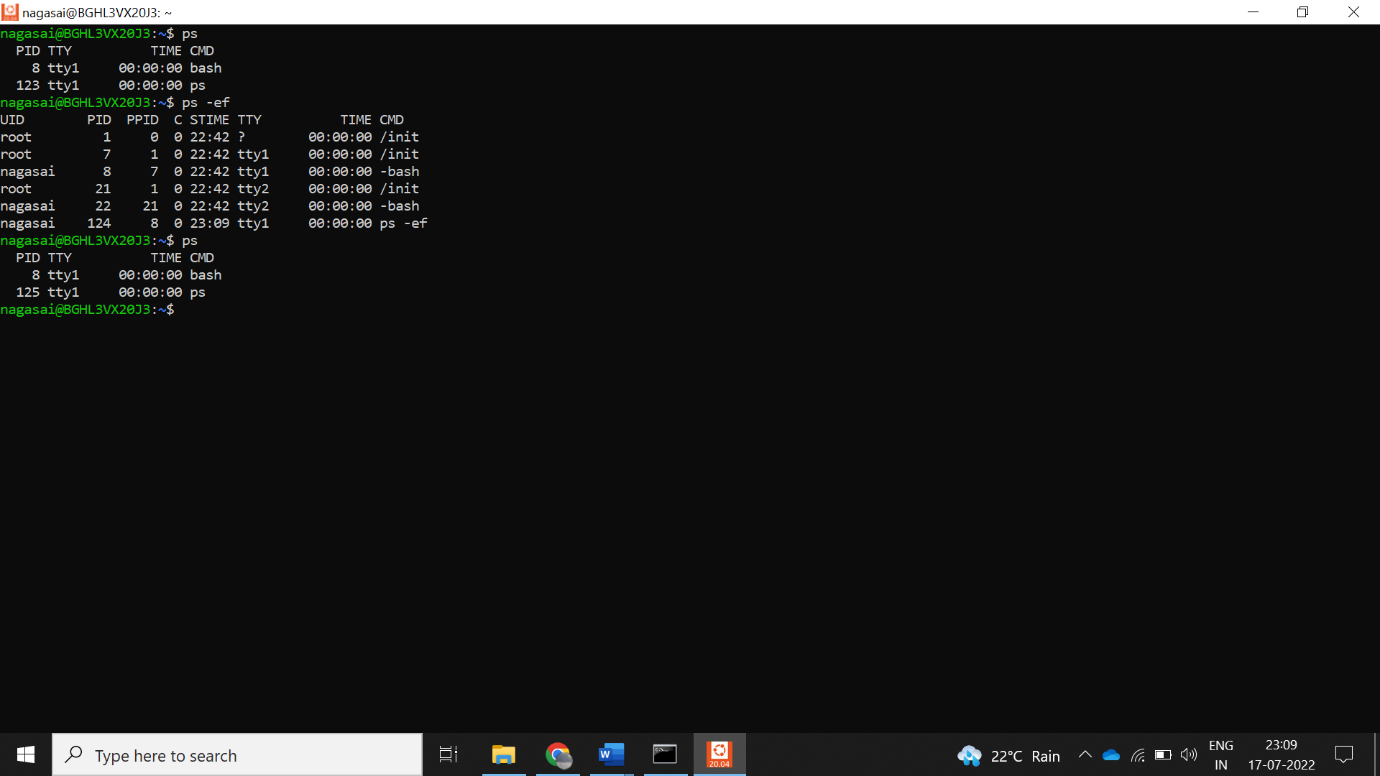
* 1. What is the pid and ppid of init?

Pid->8 ppid->1

* 1. Run command below, view tty

ps

Write the grep command to filter output from ps -ef with your tty content



1. Refer link below and find the commands to

<https://www.tecmint.com/ps-command-examples-for-linux-process-monitoring/>

* 1. View user process

1. $ ps -x

b.View process with a given pid

1. $ ps -fp 1178

c.View process tree

$ ps -e --forest

1. View pid, ppid of a process

$ ps -eo pid,ppid,user,cmd

e.Find pid of your bash shell

$ ps

**Section B- Program execution**

1. Run the program “fork\_exec.c” in process directory in socodery, modify the code to execute command below in child process
   * + 1. cat <userfilename>
       2. env

[env command is to display the environment variables in console]

Test your code changes