Quiz#2
SE-5B

Time: **9 +1 mins**

Q#1.

a. What command would you use to view the complete list of processes running on your system? Explain the purpose of at least three columns from the output. (2 mins)

Command: `ps au`

- **User**: The user who owns the process.
- **PID**: The process ID, which is a unique identifier for each process.
- **STAT**: The current state of the process (e.g., R for running, S for sleeping, Z for zombie).

b. Explain the difference between the ***fork()*** and ***exec()*** system calls. How do they complement each other in process management? (2 mins)

- **fork():** Creates a new process by duplicating the current process. The new process is a child that runs the same code as the parent.
- **exec():** Replaces the current process image with a new program. Once exec() is called, the current process is completely replaced by the new program.
- Together, they are used to first create a new process (fork()), and then run a new program in that process (exec()).

Q#2.

a. How would you use the ***getpid()*** and ***getppid()*** system calls in a C program to display the process ID and parent process ID? Provide an example. (3 mins)

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main() {
   printf("Process PID: %d, PPID: %d\n", getpid(), getppid());
   return 0;
}
```

- In this program, `getpid()` returns the process ID of the current process, and `getppid()` returns the parent process ID.

b. In a system call ***fork()***, what is the significance of the return value, and how can you differentiate between the parent and child process based on it? (2 mins)

- The fork() system call returns:
  - 0 to the child process.
  - A positive non-zero value (the PID of the child) to the parent process.
- You can differentiate between the parent and child by checking the return value of fork(). If it returns 0, the process is the child; otherwise, it’s the parent.