BITS Pilani, Hyderabad Campus
CS F372 Operating Systems
Tutorial Sheet 2

In this tutorial, you will work with process forking in a Linux environment. Your task is to create a program that demonstrates the concept of process forking and prints information from both the parent and child processes.

Question 1:

1. Write a C program that **forks** the main process and prints the **process id (PID)**.
2. Create a **child process** and in each process, print which process it is (**Child or Parent**) along with its respective PID.
3. Extend the previous program from Subpart 2. In this case, the processes should also print the process ID of its respective parent processes.

Note: After forking, the child process will have its own copy of the variables and code, independent of the parent process.

Question 2:

On running the programs in question 1, it can be noticed that the order of statements printed by the child and the parent is random. This is because there is no restriction or specification that controls this order, and depends simply on which process is serviced first by the OS. This question deals with the **wait** command, which allows control of this order.

1. Create a child process. Print the process type (child, parent, etc.) of each process. Ensure through the use of **wait** statements that the parent executes its statements only after the child has finished its execution.
2. Create a child process. In the child process, create another child process (grandchild of the parent process). Print the process type (child, parent, etc.) and its process ID in each process. The child process should now wait for its child process (grandchild) to finish its execution. And the parent process should wait for the child process before executing. When does the **waitpid()** function come into play?

Question 3:

1. Write a C program that takes a filename as a command line argument and uses **execlp** to execute the 'cat' command on the provided filename. This should display the content of the specified file.
2. Modify the C program to create a child process. In the child process, use **execlp** to execute the 'cat' command on the same file as in the previous subpart. In the parent process, use execlp to execute the 'ls' command to list the names of files in the current directory.

Question 4:

Create a child process. In the child process, copy the contents of the file “input.txt” to “output.txt”. From the parent process, print the content of the file “output.txt”.