National University of Computer and Emerging Sciences

Laboratory Manual # 03
Operating Systems

Course Instructor	Mubashar Hussain
Lab Instructor	Muhammad Hashir Mohsineen
Section	BCS-4E
Date	11-Feb-2025
Semester	Spring 25

Instructions:

- Submit a world/LibreOffice file containing screenshots of terminal commands/ Output
- Submit your .c (Code files)
- In case of any explanation you can add a multiline comment.

Objectives:

- Working with Linux system calls
- Creating process using Fork system call

Reading material:
https://docs.google.com/document/d/1OvZZ-MAkXwX8xKyqZh4ay4uPhf_Q2yReEAwARfOOgAA/edit?usp=sharing

1. Exercise 1: [10]

A. Write a program that uses the fork() system call to create a child process. Print the PID of the parent and child processes.
B. Update the above program, and declare a global variable and modify it in both the parent and child processes. Print the value of the variable in both processes and explain the output.

2. Exercise 2: [10]

A. Write a program that creates 5 child processes using a loop. Each child should print "Child n", where n is the child's number. The parent should wait for all child processes to terminate before exiting.
B. Update above program so that the parent process creates multiple child processes, each of which runs for a short time and then exits with a specific exit code. The parent process should collect all exit statuses using waitpid() and print each child's exit status.

3. Exercise 3: [10]

Write a program that uses two processes. One is a parent and the other is a child. The child reads a text file and finds the total number of vowels (a, e, i, o , u) in the file.
While the child is processing, the Parent waits for the child to finish with code and prints 'Program completed' message.
Make the text file in the same directory as your code file. You are required to pass the file name through command line arguments.

4. Exercise 3: [10]

Write a program that creates a child process using the fork system call. The child process should replace its process image with a new program using the exec system call. You will create two source files: one for the parent process and one for the child process. The parent process will execute the child process using exec. You have to pass the file name (that will be passed in exec) through command line arguments.

Example:
task.c: print 1 to 10 numbers in a loop.
parent.c: it forks a child process and uses exec to replace the child process image with the task.c program.
You have to pass the executable file in exec i.e. ./task after gcc task.c -o task.
