National University of Computer and Emerging Sciences

Laboratory Manual # 03 Operating Systems Lab

Lab Instructor	Fatima Ali
Section	BCS-4A, BCS-4B, BCS-6A
Semester	Spring - 26

Instructions:

- Submit a word file containing screenshots of your outputs with question number.
- In case of any explanation, you can add a multiline comment or add details under the screenshot of the output.
- Submit your code files with question number and roll number.

Objectives:

- Practicing Fork & Exec system call
- File processing
- Dup2 system call

1. Exercise:

Write two programs in C:

a) even.c
   - A standalone program that prints first 10 even numbers.
   - This will later be executed by the second process using exec.

b) parent.c (parent program)
   - Creates a child process using fork().
   - In the child process, use an exec system call (execlp() or execv()) to run the compiled task program.
   - Pass the executable file name (./even) through command line arguments.

2. Exercise:

Create a source file (file_process.c).

- This program will handle both parent and child logic using fork().

Open/Create a file.

- Use fopen() or open() system calls.
- Choose a filename, e.g., output.txt.

Call fork() to create a child process. In the Child Process:

- Open the file in write mode (e.g., "w").
- Write some random text (e.g., "Hello from the child process!\nThis is some sample text.\n").
- Close the file.
- Exit successfully.

Parent Process:

- Wait for the child process to complete using wait().
- Open the file in read mode (e.g., "r").
- Read the contents of the file line by line or character by character.
- Print the contents to the console and close the file.

3. Exercise:

Write a program that uses multiple fork() calls to create the following process tree: P

```
        /     \
     C1         C2
    /    \
GC1   GC2
```

Each process should print its PID and PPID. Use wait() to ensure that parents wait for their children before exiting.

4. Exercise:

Write a C program that demonstrates how to use the dup2() system call to redirect the output of a command into a file.

The program should:

- Fork a child process.
    - In the child process:
        - Open (or create) a file named output.txt with write permissions.
        - Use dup2() to duplicate the file descriptor onto standard output (stdout).
        - Execute the ls -l command using execvp().
        - (This means the ls -l output should be written to output.txt instead of the terminal.)
    - In the parent process:
        - Wait for the child to finish.
        - Open output.txt in read mode and print its contents to the terminal.

5. Exercise:

Write a program that reads an input from a file named input.txt and sums all the digits found in the file. The program should then write the sum to a new file named output.txt. Instead of using the standard read or write system calls for file operations, employ the dup2 system call to redirect the input and output. Specifically, the program should open input.txt for reading and use dup2 to redirect standard input (stdin) to read from this file. Similarly, it should open output.txt for writing and use dup2 to redirect standard output (stdout) to write to this file. Ensure that the program includes proper error handling for all file operations and system calls. For example, if input.txt contains the digits 1 2 3 4 5, the program should calculate the sum as 15 and write this result to output.txt.