OS LAB 4 TASKS

Write a program which uses fork () system-call to create a child process. The child process prints the contents of the current directory, and the parent process waits for the child process to terminate.

Write a program which prints its PID and uses fork () system call to create a child process. After fork () system call, both parent and child processes print what kind of process they are and their PID. Also, the parent process prints its child's PID, and the child process prints its parent's PID.

Develop a program that copies the contents of one file into another file using system calls. The program should accept two file paths as command-line arguments: the source file to be copied from and the destination file to be copied to. Ensure proper error handling for file opening, reading, and writing operations.

Write a program that lists all files and directories in the current directory using system calls. The program should traverse the directory structure recursively and print the names of all files and directories found, along with their respective types (file or directory).

Write a C program to create a backup of a file. The program takes two filenames as input (source and destination), where the parent process opens both files, forks a child process to read from the source file and write to the destination file, and ensures proper error handling if the file does not exist.

Write a C program to count the number of lines in a file. The parent process opens the file, forks a child process that reads character by character, counts the newline (\n) characters, and prints the total line count after the child process finishes execution.

Write a C program to display the contents of a file on the terminal. The parent process opens the file and forks a child process, which reads the file using read() and writes the content to the terminal using write(), while handling errors if the file does not exist.

Write a C program creates a child process using fork(). The child process redirects its output to a file and executes the who command using execlp(). The parent process waits for the child to finish and informs the user that the output has been successfully stored.