**Child Processes and Filesystem Manipulation with Fork and Exec**

**TASKS**

a. To make three new processes from the main process, use two fork system calls.

b. Use execl() to make Child 2 (C2) of the main process differentiate by running "ls -1 /home/username". C2 is one of the immediate descendants of the main process that does not create another process. Child 1 (C1) is the other immediate descendant of the main process that creates a grandchild (GC) of the main process.

c. The grandchild process must change its working directory to /home/username/lab4 and create a file named sample.txt with permissions 0777. Assume that /home/username/lab4 already exists.

d. The process C1 must wait for the grandchild process (GC) to complete execution, and only upon normal exit of the grandchild process, use execv() to differentiate and print the contents of /home/username/lab4 from its present working directory.

SOLUTION APPROACH

C program that demonstrates the use of the fork(), execv(), and execl() system calls to create child processes and execute commands in them.

In the program, the main process forks two child processes, Child 1 and Child 2. Child 1 forks a grandchild process (GC), changes its working directory to /home/username/lab4, creates a file named sample.txt with permissions 0777, waits for GC to complete execution, and uses execv() to run the "ls -1 /home/username/lab4" command to differentiate and print the contents of the /home/username/lab4 directory. Child 2 differentiates itself by running the "ls -1 /Users/jaskaransingh" command using execl(). Finally, the main process waits for Child 1 and Child 2 to complete their execution before printing a message.

The program also handles errors that may occur during execution and prints appropriate error messages.