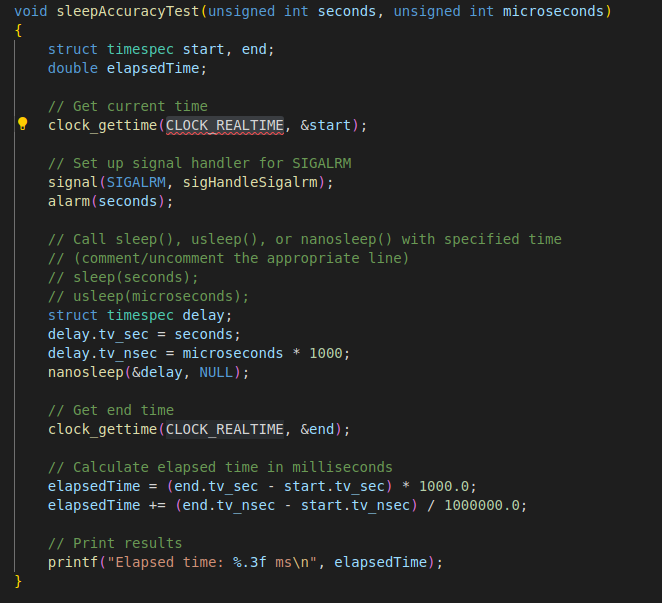
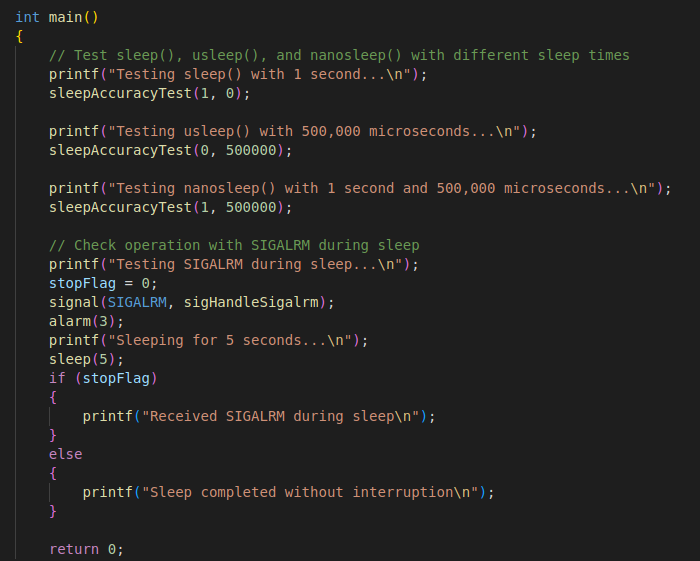
**7.2**

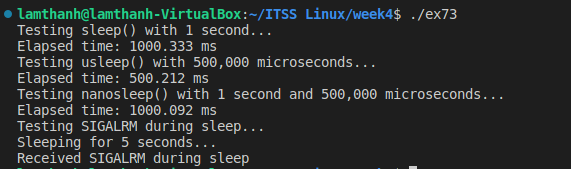
The sleepAccuracyTest() function is then defined, which takes two arguments - seconds and microseconds, representing the sleep time in seconds and microseconds, respectively. Inside this function, it uses the clock\_gettime() function with CLOCK\_REALTIME clock to get the current time before and after calling the sleep-related function (either sleep(), usleep(), or nanosleep()) with the specified sleep time. It then calculates the elapsed time in milliseconds by subtracting the start time from the end time, and prints the result.



In the main() function, the sleepAccuracyTest() function is called multiple times with different sleep times to test the accuracy of sleep-related functions. After that, it sets up a signal handler for SIGALRM, and sets an alarm to trigger SIGALRM after 3 seconds. It then calls sleep() function for 5 seconds. If the stopFlag is set to 1 after waking up from sleep, it indicates that the SIGALRM signal was received during sleep, otherwise, it indicates that the sleep completed without interruption.

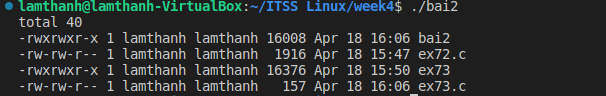


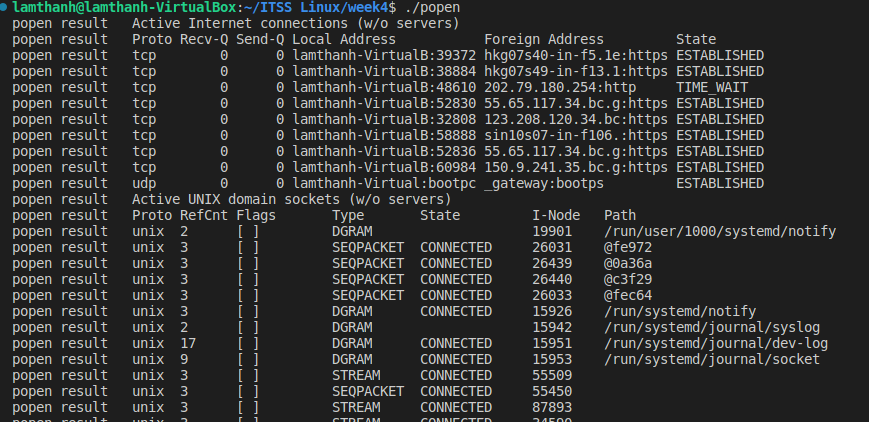
Result:



**7.3**

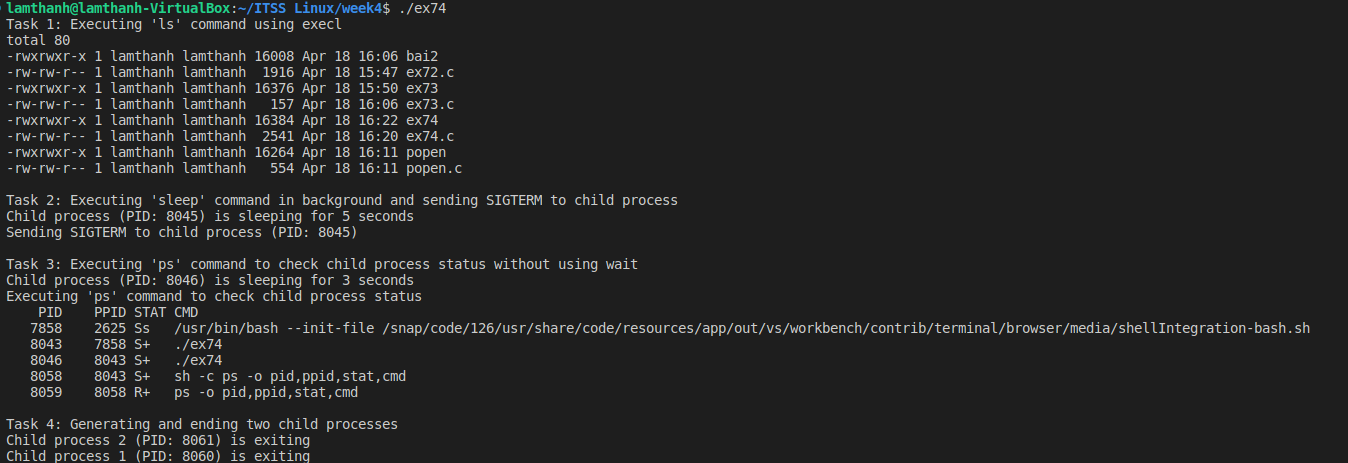
- Shell command:



* Popen stream:
* 

**7.4**

Result:



* Task 1: This task involves replacing the execv() function with other variants of the exec\* family of functions, such as execl(), execle(), execvp(), etc., and trying out different Linux commands to be executed by the child process.
* Task 2: A child process is created using fork(), and the child process sleeps for 5 seconds using the sleep() function. Meanwhile, the parent process sends a SIGTERM signal to the child process using the kill() function. The SIGTERM signal (15) is a termination signal that asks the process to terminate gracefully. The child process receives the signal and terminates itself by calling exit(). The parent process then waits for the child process to terminate using wait().
* Task 3: A child process is created using fork(), and the child process sleeps for 3 seconds using the sleep() function. However, the parent process does not wait for the child process using wait(). Instead, it directly executes the "ps" command using the system() function to check the status of all processes, including the child process. This demonstrates that the child process becomes a zombie process after it terminates, as the parent process does not wait for it to exit.
* Task 4: A loop is used to generate and end two child processes. The child processes are created using fork(), and each child process simply prints a message and terminates using exit(). The parent process waits for both child processes to terminate using wait() in a loop, ensuring that all child processes have exited before the program terminates.