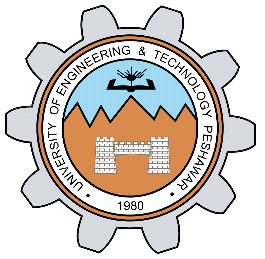
**LAB 3**



**fall 2023**

**Signal Prossacing**

**Submitted by**: **Muhammad Ilyas**

**Registration No**. :21PWCSE2055

**Class Section**: **B**

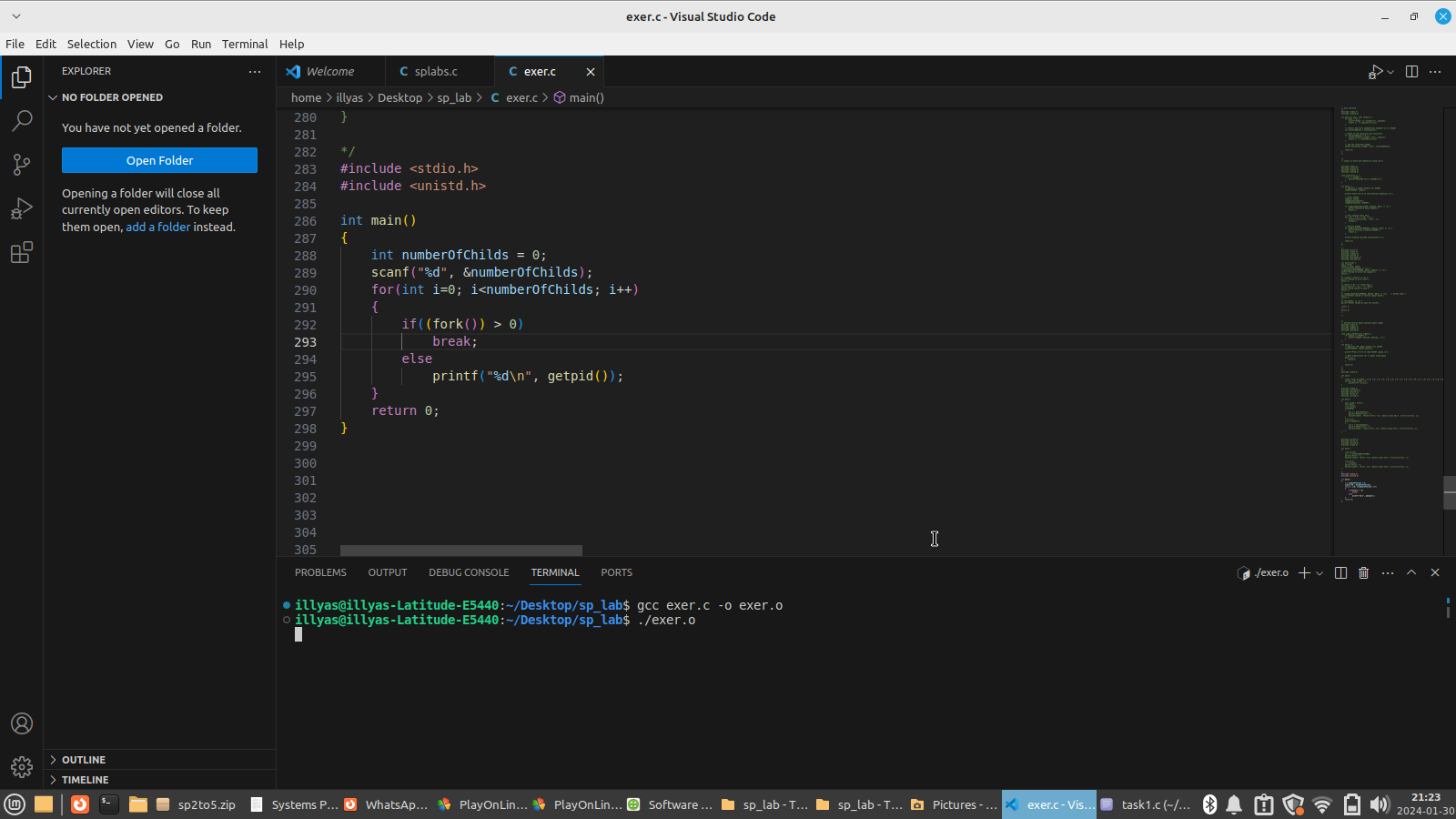
Submitted to:

**Engr. Sir ABDULLAH HAMID**

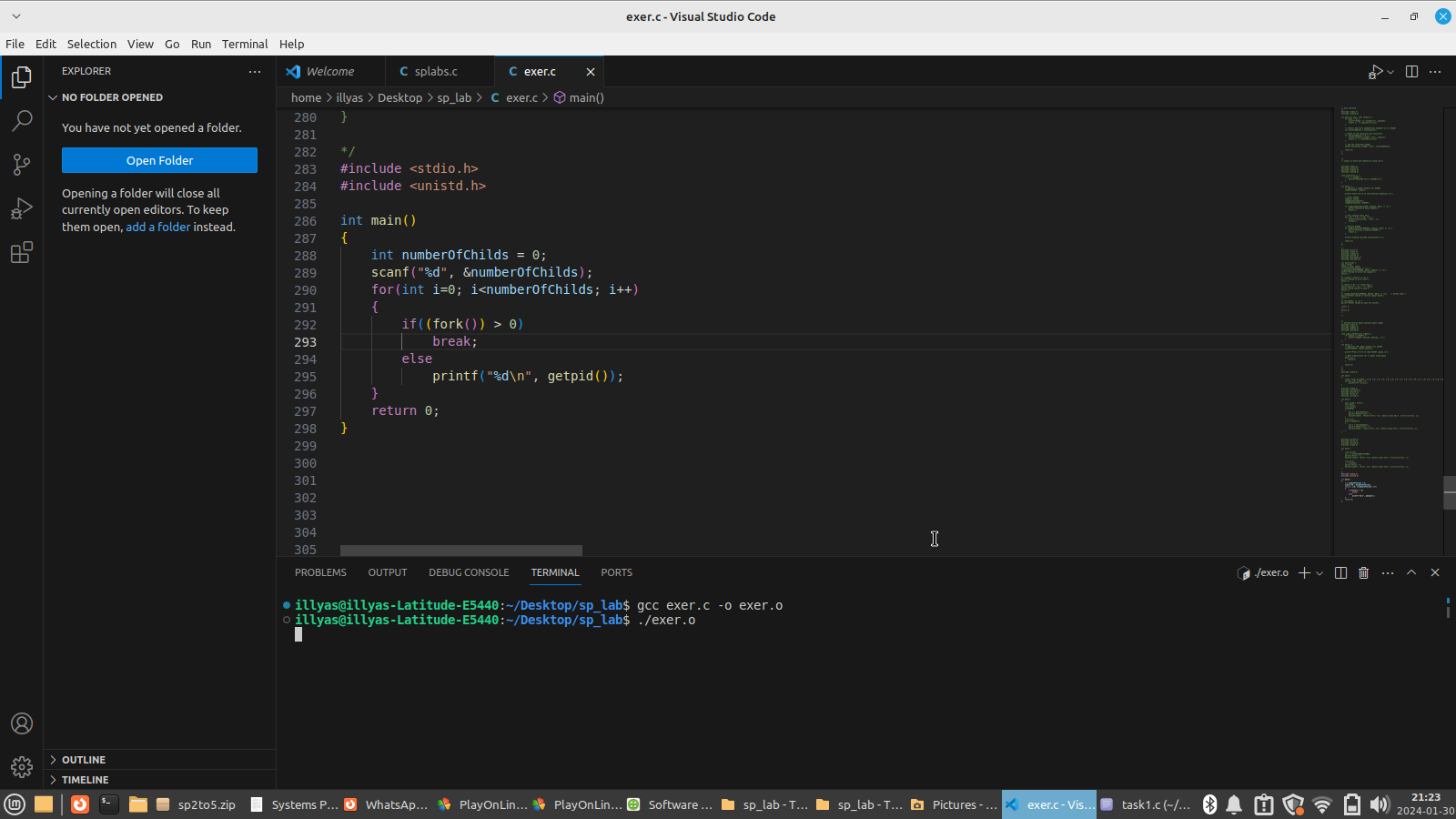
**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

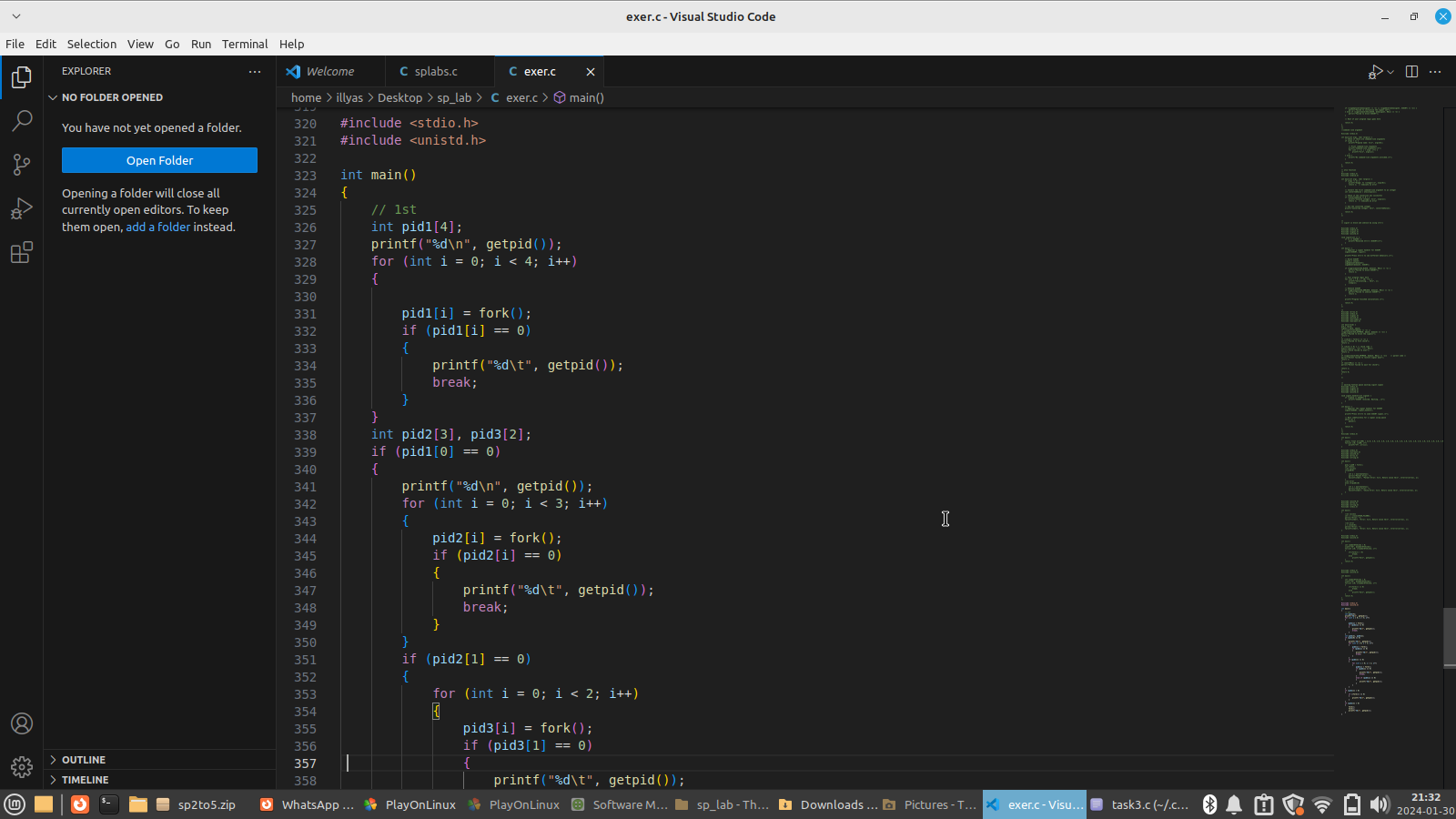
Task 1: Create process chain as shown in figure 3.1(b) and fill the figure 3.1 (b) with actual IDs. The program shall take a single command-line argument that specifies the number of processes to be created. Before exiting, each process shall output its i value (loop variable), its process ID (using getpid()), its parent process ID (getppid()) and the process ID of its child (return value of fork). The parent does not execute wait.

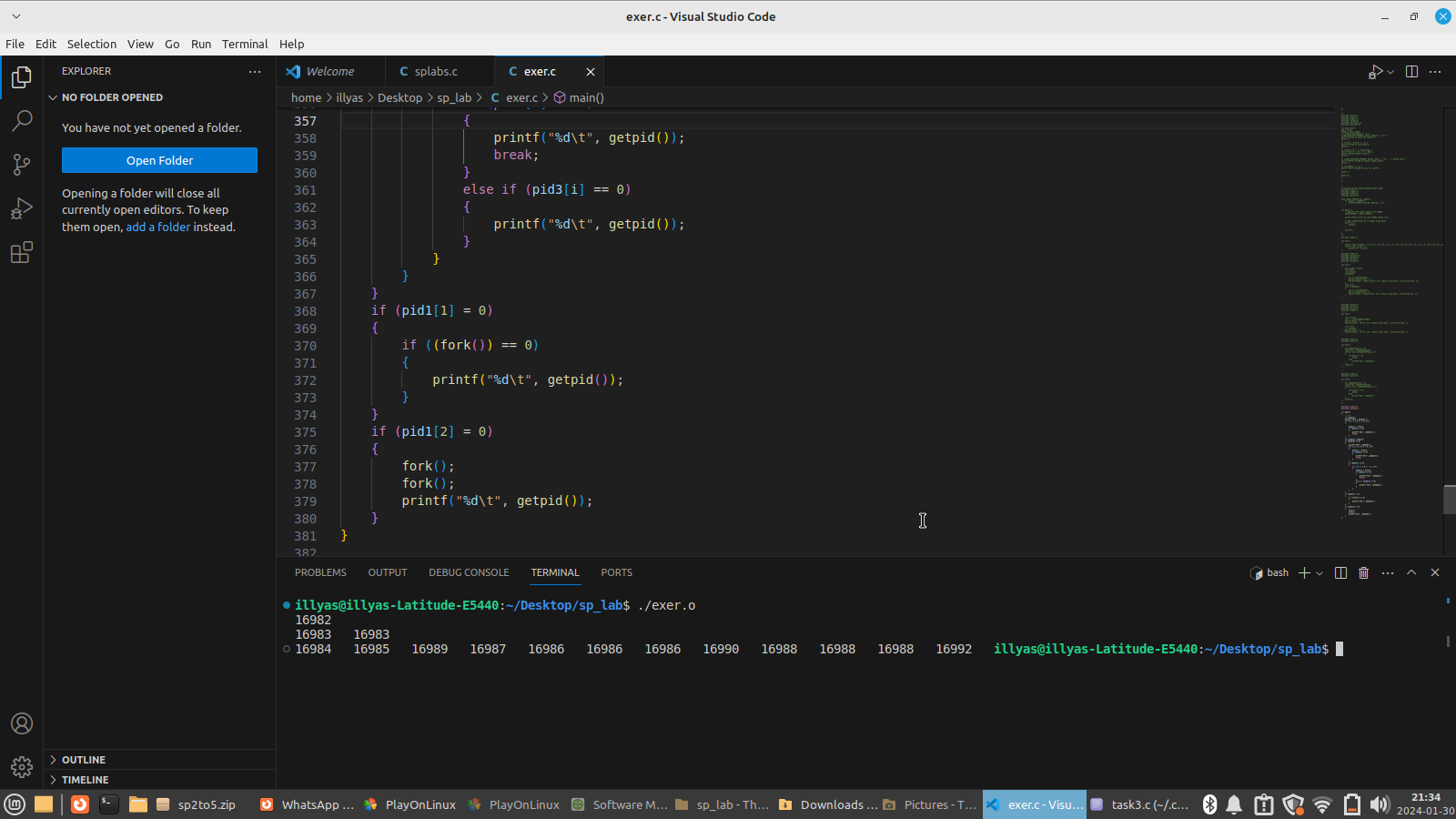


Task 2: Create process fan as shown in figure 3.1 (a) and fill the figure 3.1 (a) with actual IDs.

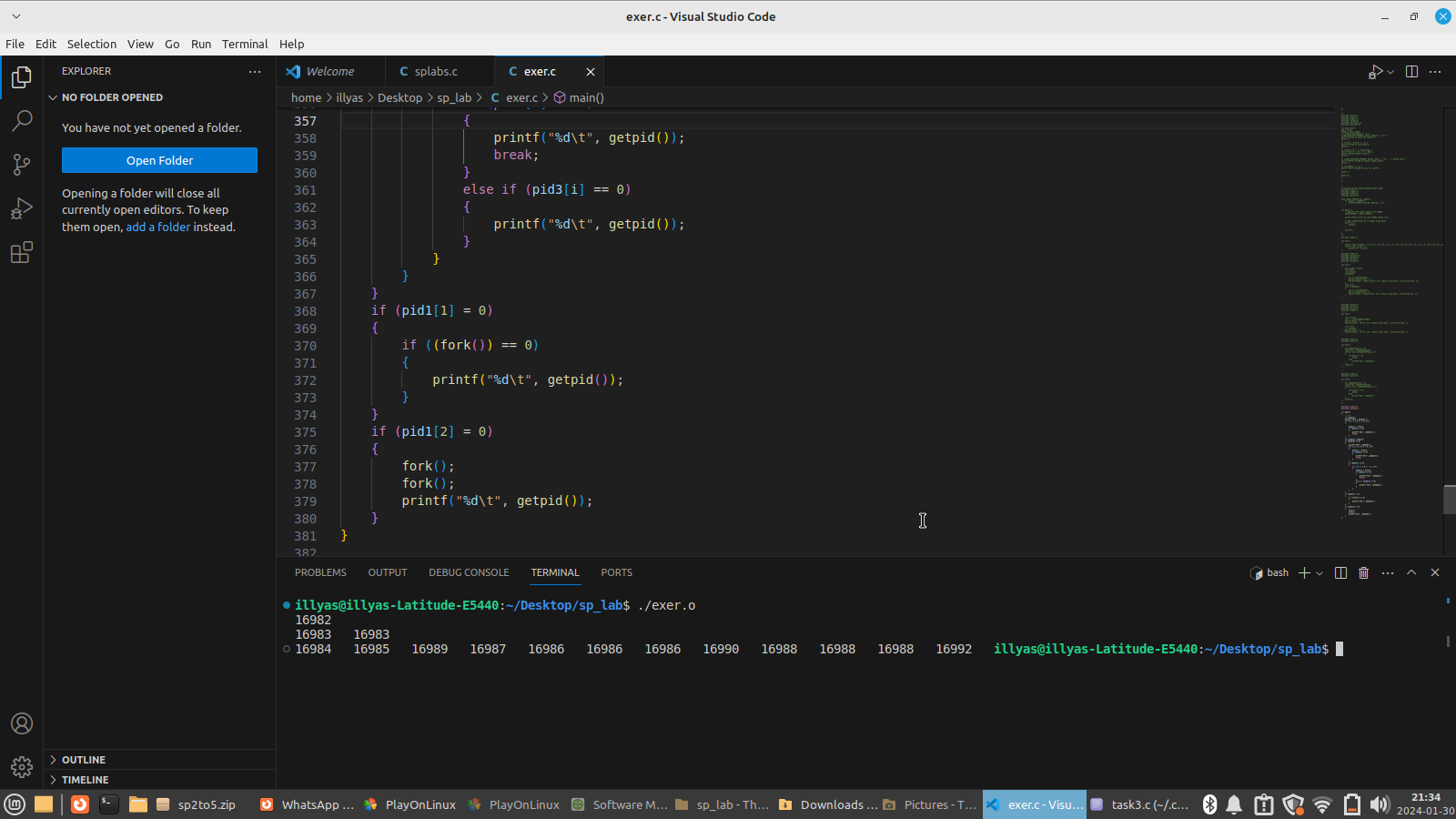


Task 3: Create process tree as shown in figure 3.2 and fill figure 3.2 with actual IDs.

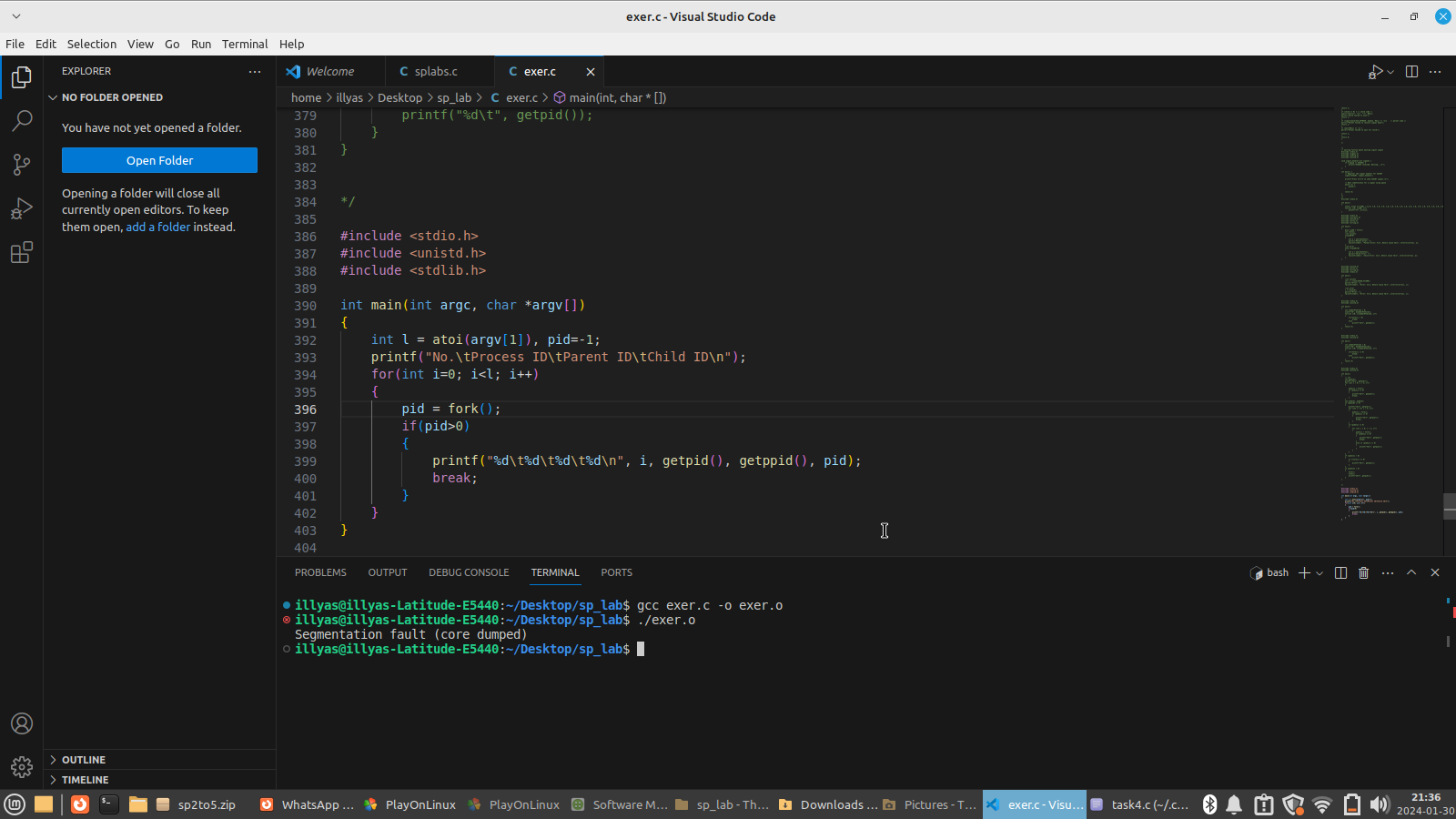


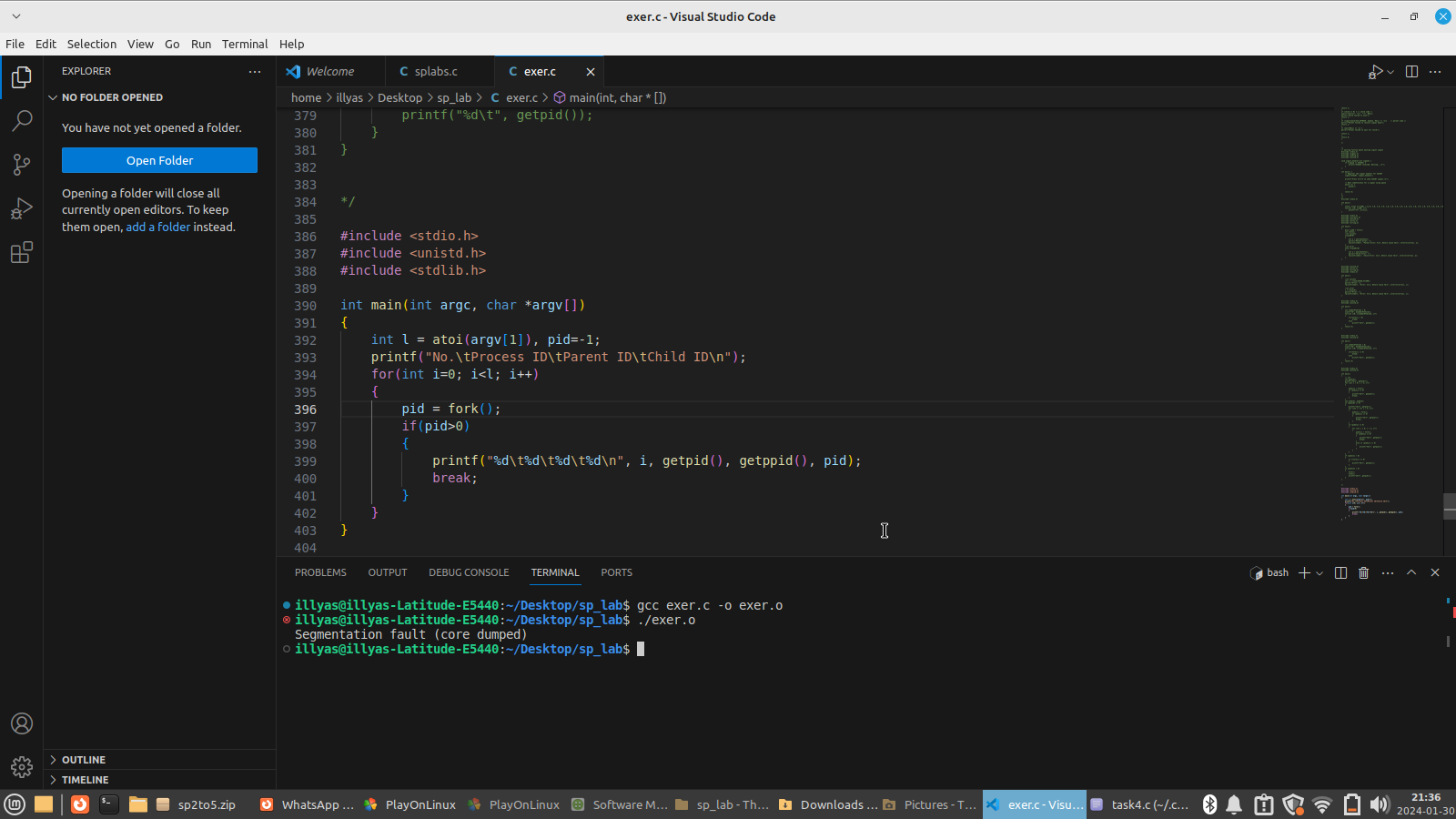


OUTPOT

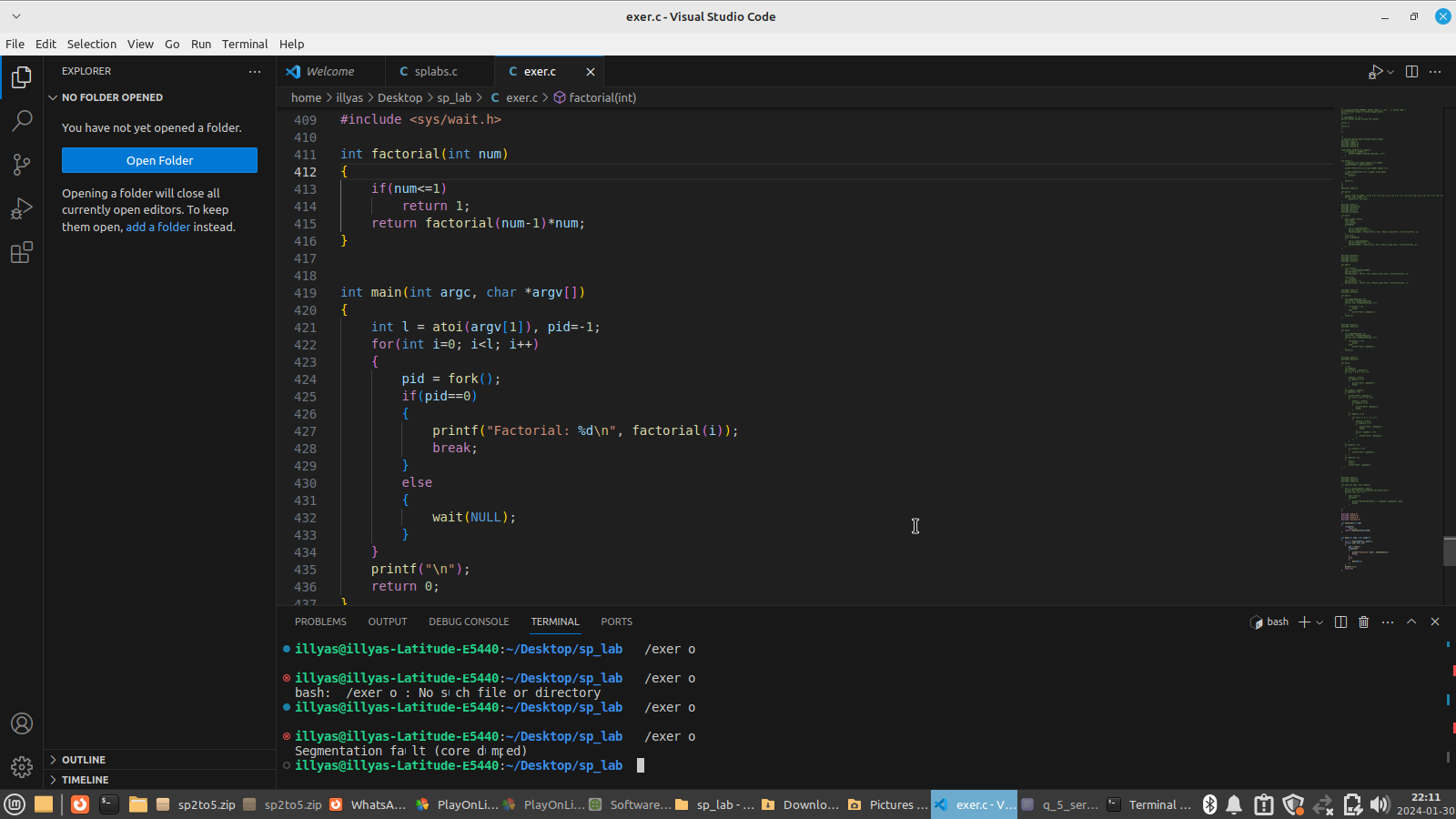


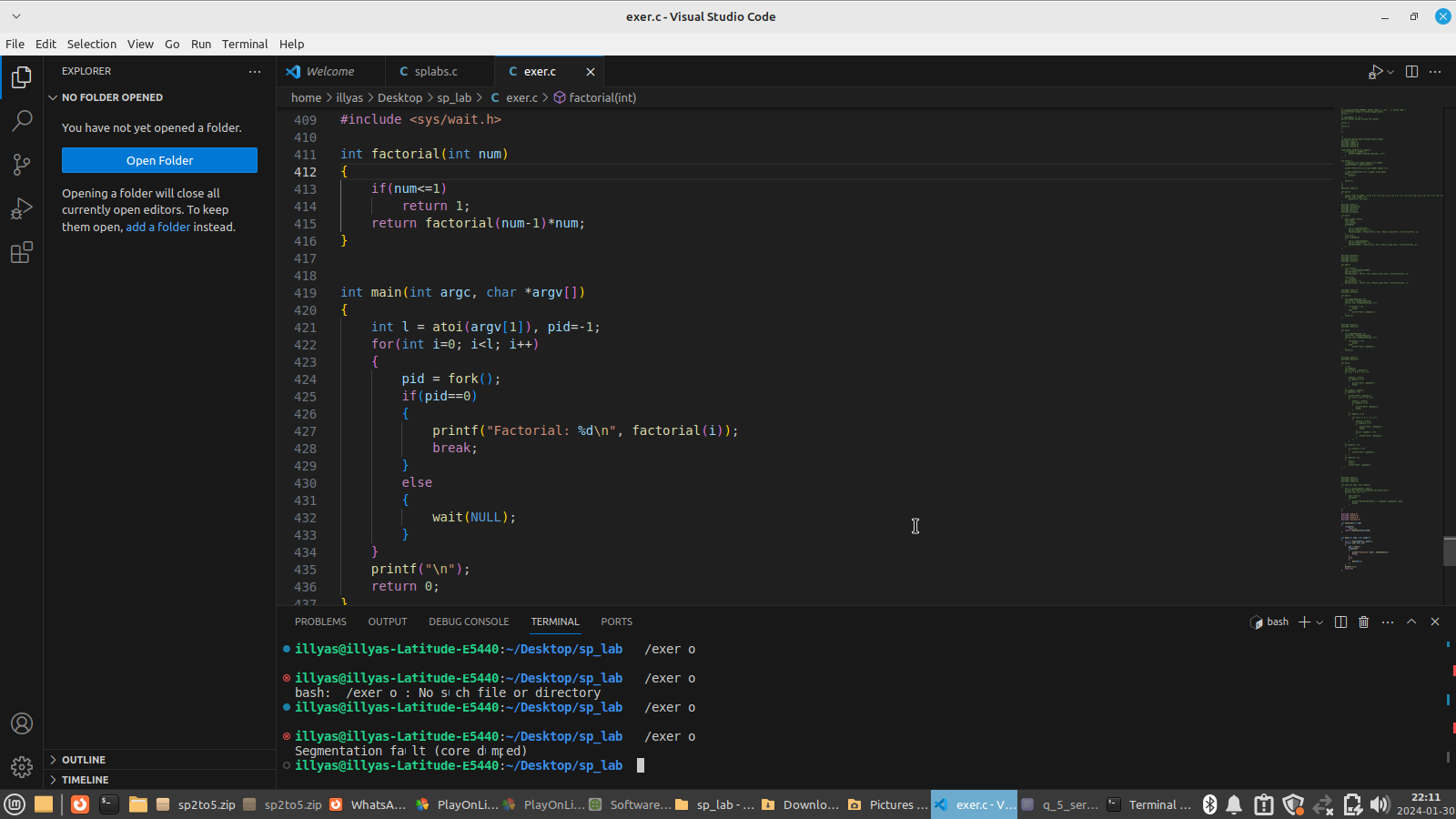
Task 4: creates a chain of processes. It takes a single command-line argument that specifies the number of processes to create. Before exiting, each process outputs its i value, its process ID, its parent process ID, and the process ID of its child. The parent does not execute wait. If the parent exits before the child, the child becomes an orphan. In this case, the child process is adopted by a special system process (which traditionally is a process, init, with process ID of 1). As a result, some of the processes may indicate a parent process ID of 1.





Task 5: Write a program that takes N number of integers as argument and displays the factorials of N integers (print 1 only if integers are not less than zero, 0 or 1). Create separate child process for each integer. Make sure no child is orphan/zombie.





Task 6: Write a program that creates an array of size 100. Initialize the array with random numbers. Create 10 child processes divide the array between them. Each child will add the portion and return their sum to parent process. Parent will add the results and display a final sum.

