**Command Line Arguments**

**LAB # 07**



**Spring 2023**

**CSE-204L Operating Systems Lab**

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**Engr. Madiha Sher**

Date:

**19th May 2023**

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objectives:**

* To understand the concept of command-line arguments and their usage in C programming.
* To learn how to access and process command-line arguments in a C program.
* To explore process creation and execution using command-line arguments.
* To create multiple child processes and execute different commands in each child process.

**COMMAND LINE ARGUMENTS:**

Command-line arguments are essential for providing inputs and additional information to a program during runtime. In C programming, the main() function is used to access and process command-line arguments. The argc parameter represents the count of command-line arguments, and the argv parameter is an array of strings that holds the arguments themselves.

When executing a program from the command line, arguments can be passed after the program name. These arguments are separated by spaces. For example, executing a program named "myprogram" with three arguments would look like: ./myprogram arg1 arg2 arg3.

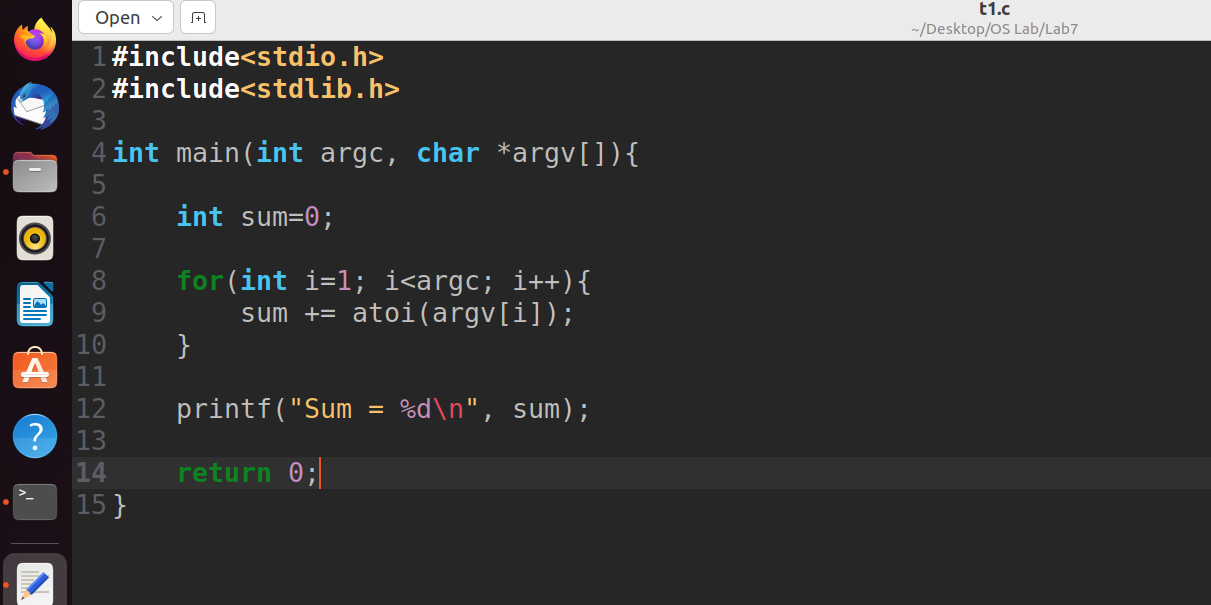
The argc parameter in the main() function will have a value of 4, indicating the program name plus three additional arguments. The argv parameter is an array where argv[0] holds the program name, argv[1] holds "arg1", argv[2] holds "arg2", and argv[3] holds "arg3".

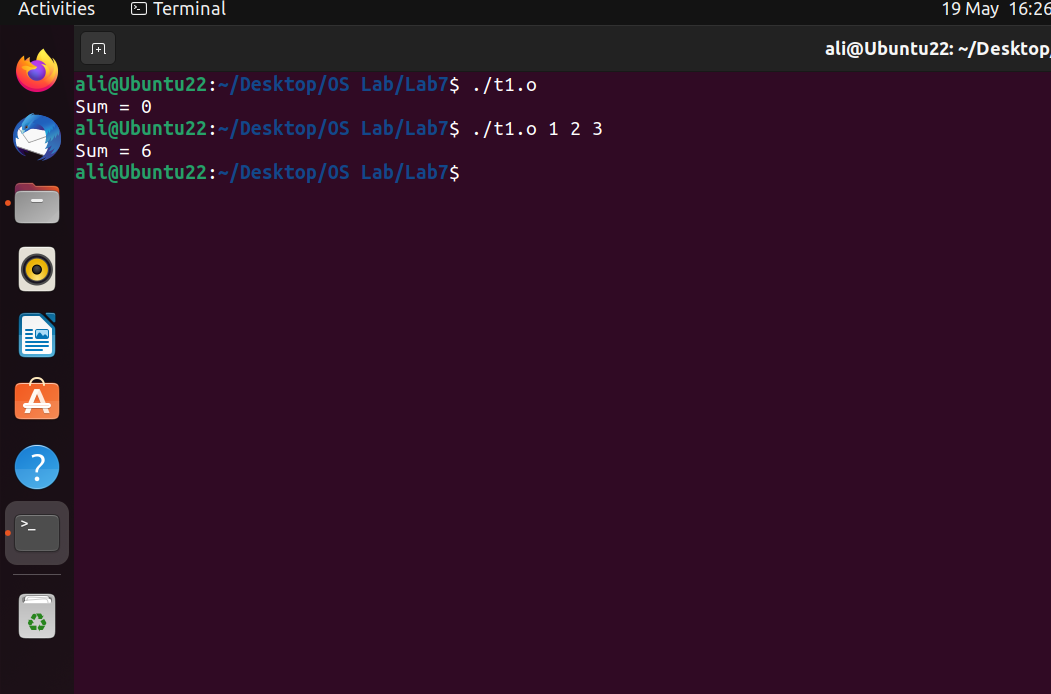
By accessing the argv array, a C program can process and utilize the command-line arguments as needed. This allows for dynamic behavior and customization of the program based on user inputs.

Overall, command-line arguments provide flexibility and interactivity to programs, allowing users to provide inputs and customize program behavior without modifying the source code.

**Task1:**

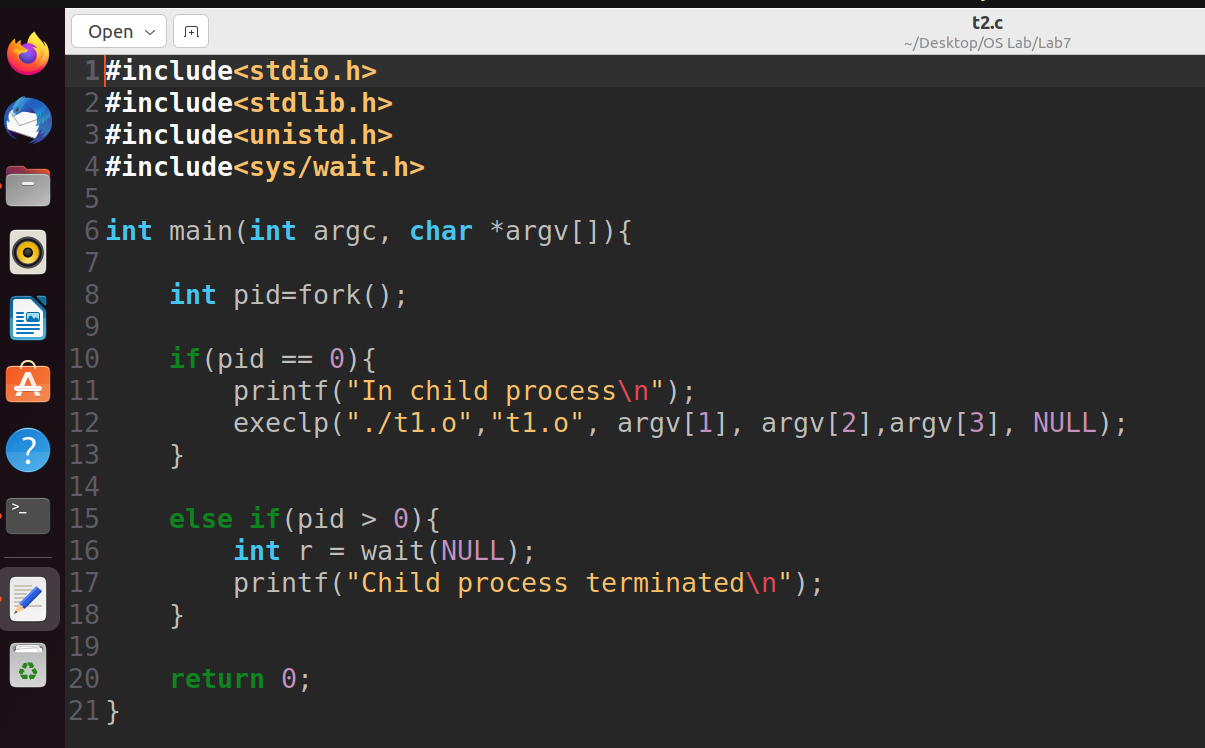
Write a C program that finds the sum of all CLA’s.





**Task2:**

Write a C program that creates a child process & execute task 1 in child process using execlp() system call. Parent process shall wait for the child process.

****

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Task3:**

Write a C program that takes built-in command on CLA’s and create separate child process for each command & execute these commands in child process. Parent shall wait for the child processes.

A screen shot of a computer program

Description automatically generated with low confidence

**A screenshot of a computer program

Description automatically generated with medium confidence**