



**Department of Computer Science and Engineering,
PES University, Bangalore, India**

**Lecture Notes
Problem Solving With C
UE24CS151B**

***Lecture #6
Command Line Arguments***

**By,
Prof. Sindhu R Pai,
Theory Anchor, Feb-May, 2025
Assistant Professor
Dept. of CSE, PESU**

**Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Center, PES University)
Prof. Nitin V Poojari (Dean, Internal Quality Assurance Cell, PES University)**

Unit #: 3**Unit Name: Text Processing and User-Defined Types****Topic: Command Line Arguments**

Course objectives: The objective(s) of this course is to make students

- Acquire knowledge on how to solve relevant and logical problems using computing Machine.
- Map algorithmic solutions to relevant features of C programming language constructs.
- Gain knowledge about C constructs and its associated ecosystem.
- Appreciate and gain knowledge about the issues with C Standards and it's respective behaviours.

Course outcomes: At the end of the course, the student will be able to:

- Understand and Apply algorithmic solutions to counting problems using appropriate C Constructs.
- Understand, Analyze and Apply sorting and Searching techniques.
- Understand, Analyze and Apply text processing and string manipulation methods using Arrays, Pointers and functions.
- Understand user defined type creation and implement the same using C structures, unions and other ways by reading and storing the data in secondary systems which are portable.

Sindhu R Pai

Theory Anchor, Feb - May, 2025

Dept. of CSE,

PES University

Introduction

Command line arguments are values or inputs passed **to a program directly from the terminal or command prompt when the program is run**. They allow users to customize the program's behavior without changing the code or without waiting for user input during execution. Hence, the program input can be dynamic and flexible, accepting data without hard coding it. These are **widely used in scripting, automation, and software tools** where inputs need to be customized without modifying the code.

Structure of main() with CLA

The arguments are provided to the program after the name of the executable in command-line shell of Operating Systems and handled using two special parameters in the main() function:

int main (int argc, char *argv[])

argc (Argument Count) tells how many arguments were passed. It always includes the program name as the first argument. So, argc is **at least 1**.

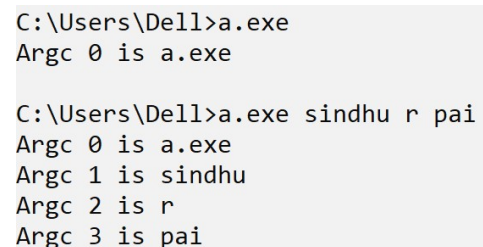
argv (Argument Vector) is an array of character pointers listing all arguments, starting with the program name itself.

- argv[0] is the name of the program.
- argv[1], argv[2], ..., argv[argc-1] are the actual command line arguments.

NOTE: All the arguments which are passed in command line are accessed as **string inside the program and must be converted to desired type using different functions.**

Coding Example_1: Display all the command line arguments passed

```
#include<stdio.h>
int main (int argc, char *argv[])
{
    int i;
    for (i = 0; i < argc; i++)
        printf ("Argc %d is %s\n", i, argv[i]);
    return 0;
}
```



```
C:\Users\Dell>a.exe
Argc 0 is a.exe

C:\Users\Dell>a.exe sindhu r pai
Argc 0 is a.exe
Argc 1 is sindhu
Argc 2 is r
Argc 3 is pai
```

Coding Example_2: Program to add three integers passed in the command line

```
#include<stdio.h>
#include<string.h>
int main(int argc,char *argv[]) //conventionally argc and argv, Can use any variable
{
    printf("count of arguments are %d\n",argc); // argc =4
    printf("argv[0] is %s\n",argv[0]);
    printf("argv[1] is %s\n",argv[1]);
    printf("argv[2] is %s\n",argv[2]);
    printf("argv[3] is %s\n",argv[3]);
    printf("Sum is is %d\n",atoi(argv[1])+ atoi(argv[2])+ atoi(argv[3]));
    // The C library function atoi(str) converts the string argument str to an integer
    //(type int). If no valid conversion could be performed, it returns 0
    return 0;
}
```

Coding Example_3: Find the sum of all numbers provided in the command line.

```
#include<stdio.h>
#include<stdlib.h>
int main(int argc,char *argv[])
{
    int i;
    int sum = 0;
    for(i = 0; i < argc; i++)
        sum = sum + atoi(argv[i]);
    printf("Sum of command line arguments are %d\n",sum);
    return 0;
}
```

```
C:\Users\Dell>a
Sum of command line arguments are 0

C:\Users\Dell>a 3 4
Sum of command line arguments are 7

C:\Users\Dell>a 3 4 9
Sum of command line arguments are 16
```

Real-world use cases of Command Line Arguments (CLA)

Command Line Arguments in C are widely used in real-world applications like file processors, data converters, and automation scripts, enabling users to pass inputs dynamically at runtime without modifying the code.

- Programs that read from or write to files often take file names as command line arguments.

```
./filecopy input.txt output.txt // Executable name is filecopy.
```

- Perform arithmetic based on operands and operators passed via CLA.

```
./calc 12 + 8
```

- Convert data or files from one format to another.

```
./convert file.csv file.json
```

- Search for a keyword or pattern in a file.

```
./search "error" log.txt
```

- Automate tasks like backups or process monitoring using file paths.

```
./backup /home/user/docs/
```

- Accept runtime parameters like speed, size, or time to control simulation.

```
./simulate 60 300 //Runs simulation with speed = 60 and duration = 300 units.
```

- Sort data in files by a specific field or apply filters.

```
./sort students.csv age
```

- Accept credentials or access keys as arguments (often in secure environments)

```
./login admin mypass123
```

- Pass file names and operation types (compress/decompress).

```
./zipper compress data.txt
```

- Customize game levels or character selection at startup.

```
./game start level3 warrior
```

- Input a date and get formatted output or day of the week.

```
./calendar 2025-04-11
```

Happy coding with Command Line Arguments!