



Command Line Arguments in C

Last Updated : 09 Jan, 2024

The most important function of C is the `main()` function. It is mostly defined with a return type of `int` and without parameters.

```
int main() {  
    ...  
}
```

We can also give command-line arguments in C. Command-line arguments are the values given after the name of the program in the command-line shell of Operating Systems. Command-line arguments are handled by the `main()` function of a C program.

To pass command-line arguments, we typically define `main()` with two arguments: the first argument is the **number of command-line arguments** and the second is a **list of command-line arguments**.

Syntax

```
int main(int argc, char *argv[]) { /* ... */ }  
or  
int main(int argc, char **argv) { /* ... */ }
```

Here,

- **argc (ARGument Count)** is an integer variable that stores the number of command-line arguments passed by the user including the name of the program. So if we pass a value to a program, the value of `argc` would be 2 (one for argument and one for program name)
- The value of `argc` should be non-negative.
- **argv (ARGument Vector)** is an array of character pointers listing all the arguments.

- If argc is greater than zero, the array elements from argv[0] to argv[argc-1] will contain pointers to strings.
- argv[0] is the name of the program , After that till argv[argc-1] every element is command -line arguments.

For better understanding run this code on your Linux machine.

Example

The below example illustrates the printing of command line arguments.

C

```
// C program named mainreturn.c to demonstrate the working
// of command line argument
#include <stdio.h>

// defining main with arguments
int main(int argc, char* argv[])
{
    printf("You have entered %d arguments:\n", argc);

    for (int i = 0; i < argc; i++) {
        printf("%s\n", argv[i]);
    }
    return 0;
}
```

Output

```
You have entered 4 arguments:
./main
geeks
for
geeks
```

for Terminal Input

```
$ g++ mainreturn.cpp -o main $ ./main geeks for geeks
```

Note: Other platform-dependent formats are also allowed by the C standards; for example, Unix (though not POSIX.1) and Microsoft Visual C++ have a third argument giving the program's environment, otherwise accessible through `getenv` in `stdlib.h`. Refer [C program to print environment variables](#) for details.

Properties of Command Line Arguments in C

1. They are passed to the `main()` function.
2. They are parameters/arguments supplied to the program when it is invoked.
3. They are used to control programs from outside instead of hard coding those values inside the code.
4. `argv[argc]` is a NULL pointer.
5. `argv[0]` holds the name of the program.
6. `argv[1]` points to the first command line argument and `argv[argc-1]` points to the last argument.

Note: You pass all the command line arguments separated by a space, but if the argument itself has a space, then you can pass such arguments by putting them inside double quotes "" or single quotes ' '.

Example

The below program demonstrates the working of command line arguments.

C

```
// C program to illustrate
// command line arguments
#include <stdio.h>

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

```
printf("Program name is: %s", argv[0]);

if (argc == 1)
    printf("\nNo Extra Command Line Argument Passed "
           "Other Than Program Name");

if (argc >= 2) {
    printf("\nNumber Of Arguments Passed: %d", argc);
    printf("\n---Following Are The Command Line "
           "Arguments Passed---");
    for (int i = 0; i < argc; i++)
        printf("\nargv[%d]: %s", i, argv[i]);
}
return 0;
}
```

Output in different scenarios:

1. Without argument: When the above code is compiled and executed without passing any argument, it produces the following output.

Terminal Input

```
$ ./a.out
```

Output

```
Program Name Is: ./a.out
No Extra Command Line Argument Passed Other Than Program Name
```

2. Three arguments: When the above code is compiled and executed with three arguments, it produces the following output.

Terminal Input

```
$ ./a.out First Second Third
```

Output

```
Program Name Is: ./a.out
Number Of Arguments Passed: 4
---Following Are The Command Line Arguments Passed---
```

```
argv[0]: ./a.out
argv[1]: First
argv[2]: Second
argv[3]: Third
```

3. Single Argument: When the above code is compiled and executed with a single argument separated by space but inside double quotes, it produces the following output.

Terminal Input

```
$ ./a.out "First Second Third"
```

Output

```
Program Name Is: ./a.out
Number Of Arguments Passed: 2
----Following Are The Command Line Arguments Passed----
argv[0]: ./a.out
argv[1]: First Second Third
```

4. A single argument in quotes separated by space: When the above code is compiled and executed with a single argument separated by space but inside single quotes, it produces the following output.

Terminal Input

```
$ ./a.out 'First Second Third'
```

Output

```
Program Name Is: ./a.out
Number Of Arguments Passed: 2
----Following Are The Command Line Arguments Passed----
argv[0]: ./a.out
argv[1]: First Second Third
```

Similar Reads

C++ | Function Overloading and Default Arguments | Question 5

Which of the following in Object Oriented Programming is supported by Function overloading and default arguments features of C++. (A) Inheritance (...)

1 min read

C++ | Function Overloading and Default Arguments | Question 2

Output? `#include<iostream> using namespace std; int fun(int x = 0, int y = 0, int z) { return (x + y + z); } int main() { cout <<< fun(10); return 0; }` (A) 10...

1 min read

C++ | Function Overloading and Default Arguments | Question 3

Which of the following overloaded functions are NOT allowed in C++? 1) Function declarations that differ only in the return type `int fun(int x, int y); void...`

1 min read

C++ | Function Overloading and Default Arguments | Question 4

Predict the output of following C++ program. `include<iostream> using namespace std; class Test { protected: int x; public: Test (int i):x(i) { } void fun()...`

1 min read

C++ | Function Overloading and Default Arguments | Question 5

Output of following program? `#include <iostream> using namespace std; int fun(int=0, int = 0); int main() { cout <<< fun(5); return 0; }` int fun(int x, int...

1 min read

[View More Articles](#)

Article Tags :

[TCS](#)

[C Language](#)

Practice Tags : [TCS](#)



A-143, 9th Floor, Sovereign Corporate
Tower, Sector-136, Noida, Uttar Pradesh -
201305



Company

- About Us
- Legal
- In Media
- Contact Us
- Advertise with us
- GFG Corporate Solution
- Placement Training Program
- GeeksforGeeks Community

DSA

- Data Structures
- Algorithms
- DSA for Beginners
- Basic DSA Problems
- DSA Roadmap

Languages

- Python
- Java
- C++
- PHP
- GoLang
- SQL
- R Language
- Android Tutorial
- Tutorials Archive

Data Science & ML

- Data Science With Python
- Data Science For Beginner
- Machine Learning Tutorial
- ML Maths
- Data Visualisation Tutorial

Top 100 DSA Interview Problems

DSA Roadmap by Sandeep Jain

All Cheat Sheets

Web Technologies

HTML

CSS

JavaScript

TypeScript

ReactJS

NextJS

Bootstrap

Web Design

Computer Science

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Software Development

Software Testing

System Design

High Level Design

Low Level Design

UML Diagrams

Interview Guide

Design Patterns

OOAD

System Design Bootcamp

Interview Questions

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

World GK

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Deep Learning Tutorial

Python Tutorial

Python Programming Examples

Python Projects

Python Tkinter

Web Scraping

OpenCV Tutorial

Python Interview Question

Django

DevOps

Git

Linux

AWS

Docker

Kubernetes

Azure

GCP

DevOps Roadmap

Interview Preparation

Competitive Programming

Top DS or Algo for CP

Company-Wise Recruitment Process

Company-Wise Preparation

Aptitude Preparation

Puzzles

GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects