Command line arguments in C/C++

Last Updated: 21-12-2018

The most important function of C/C++ is 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 and C++. Command-line arguments are given after the name of the program in command-line shell of Operating Systems.

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

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

or

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

- **argc (ARGument Count)** is int and stores number of command-line arguments passed by the user including the name of the program. So if we pass a value to a program, 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 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.

```
// Name of program mainreturn.cpp
#include <iostream>
using namespace std;

int main(int argc, char** argv)
{
    cout << "You have entered " << argc
         << " arguments:" << "\n";

    for (int i = 0; i < argc; ++i)
        cout << argv[i] << "\n";

    return 0;
}
```

Input:

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

Output:

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

Note : Other platform-dependent formats are also allowed by the C and 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:

1. They are passed to main() function.
2. They are parameters/arguments supplied to the program when it is invoked.
3. They are used to control program 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[n] points last argument.

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

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

int main(int argc,char* argv[])
{
    int counter;
    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(counter=0;counter<argc;counter++)
```

```
            printf("\nargv[%d]: %s",counter,argv[counter]);
    }
    return 0;
}
```

Output in different scenarios:

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

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

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

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

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

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