Content 53

Command Line Arguments In C

#### Definition:

A command-line utility is a way of giving operating system**instructions** using lines of texts. Command-line programs operate via **command line or PowerShell**. It will interact with a command-line script.

#### Advantages of using command-line utility:

Coming towards the part of why we should use the command-line utility in our program. We can call a command line program in python or any other language into a different language program quickly as each program has calling support in it for calling the command lines program. So in cases where we are writing a program in some other language, but we want to perform a task in C and call it in our program, then the command line can help us do that.

For creating a Command Line Utility In C, we should first have some knowledge about argc and argv.

#### argc:

If we divide the keyword into two parts, the first will be 'agr', a short form for arguments. The second one will be 'c,' which stands for **count**. Some argc as a whole stands for argument count, which means that it stores the total number of arguments passed to the utility. The first argument count is reserved for the executable program's name, and the next ones are for the arguments passed to the program.

#### argv:

Same as in argc, the arg stands for argument, but he ‘v’ stands for **vector**. Vector can be said as a **one-dimensional array** in this case because argv stores the pointer to the arguments passed to the program in an array of strings.

Note: argv does not store the actual argument, but the pointer to that argument.

**Examples:**

Suppose that we have a .exe executable, named xyz. We are passing two arguments to xyz, i.e., a and b.

xyz.exe a b

Now the value of argc will be**thre**e as there are two arguments.

And argv will store **pointers** xyz, a, and b.

**Code For Explaining the argc and argv in details:**

#include<stdio.h>

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

{

    printf("The Value of argc is:~ %d", argc);

    return 0;

}

**Output:**

The Value of argc is:~ 1

It gives 1 because it is its first member is string which the name of program.

**Now if I write in terminal(infront of my .exe file) like**

PS C:\Users\Aamaan Satvilkar\Desktop\C cource> .\CmdLineCnt53.exe the value is

The Value of argc is:~ 4

Here it gives 4 because I added “the value is” infront of my .exe file so it readed it in string format and gives the value.

“Here We wrote the **command Line** which simply prints the number of Arguments”

#include<stdio.h>

// here argc gives count while argv gives value.

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

{

    printf("The Value of argc is:~ %d\n", argc);

    for (int  i = 0; i < argc; i++)

    {

        printf("he value of arument at index number %d has the value of::: %s  \n",i, argv[i]);

    }

    return 0;

}

Output:-

The Value of argc is:~ 1

The value of arument at index number 0 has the value of::: C:\Users\Aamaan Satvilkar\Desktop\C cource\CmdLineCnt53.exe

It shows that at index number 0 there is full path of .exe file.

**But if I add in terminal like:**

PS C:\Users\Aamaan Satvilkar\Desktop\C cource> .\CmdLineCnt53.exe the value is

**Output:**

The Value of argc is:~ 4

The value of arument at index number 0 has the value of::: C:\Users\Aamaan Satvilkar\Desktop\C cource\CmdLineCnt53.exe

The value of arument at index number 1 has the value of::: the

The value of arument at index number 2 has the value of::: value

The value of arument at index number 3 has the value of::: is