Accepting Command Line Arguments – Argc & Argv

In C it is possible to accept command line arguments. To do so, you must first understand the full definition of **int main()**. It accepts two arguments, one is number of command line arguments, the other is a listing of the command line arguments.

It looks like this:

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

The integer, **argc** is the **arg**ument **c**ount. It is the number of arguments passed into the program from the command line, including the name of the program. The array of character pointers **argv** is the list of all the arguments passed through command line. **argv[0]** is the name of the program, or an empty string if the name is not available. After that, every element number less than argc is a command line argument. You can use each argv element just like a string, or use argv as a two dimensional array. **argv[argc]** is a null pointer.

How can this be used? Almost any program that wants its parameters to be set when it is executed would use this. One common use is to write a function that copies files, like the UNIX ***cp*** program.

```
/*Skeleton program for copying one file to another.*/
#include <stdlib.h>
#include <stdio.h>
-------------------
void copyfile(const char* file1, const char* file 2)
{
  /* Your actual code resides here */
}

int main ( int argc, char *argv[] )
{
  if ( argc != 3 ){ // argc should be 3 for correct execution
    printf("Usage:program_name file1 and file2");
    exit(1);
  }
  else { copyfile(argv[1],argv[2]); }
  /* your code here */
}
```

```
$ ./program_name file1 file2.
```

This program is fairly simple. The program takes two arguments (excluding the program name). The argc = 3 as we have the program name, *file1*, and *file2*. **argv[0]** has a program name, **argv[1]** has the name of *file1* and **argv[2]** has the name of *file2*.

For more description see the following reference.
http://publications.gbdirect.co.uk/c_book/chapter10/arguments_to_main.html