Command-line Arguments

Learn how to pass additional parameters to your program by the command-line arguments.

The example in the previous lesson also illustrates how to pass command-line arguments to your C programs.

> Command-line arguments are those which are not part of the written code, but instead, are received from the terminal in which you compile your code. These arguments can then be used in your code.

The `main()` function can be defined using two arguments:

1. `int argc`
2. `char *argv[]`

When your program is executed, the first argument `argc`, will contain the **number of command-line arguments passed to your program, plus one**. The first argument is always the name of your program. So in the example above, we check to see if `argc<2` because we want at least one extra argument passed to our program in addition to the automatic program name argument.

The second argument `argv` is a one-dimensional array of strings. As we will see later, the `(char *)` data type, which is actually a pointer to a character, is typically used in C to represent strings (as an array of characters).

In the example above, we use the `atoi()` function to convert **ascii** to **integer**, in order to convert the command-line argument (the second one, hence `argv[1]`) which is an ascii string, into an integer `n`. We then declare an array `grades[n]` to be of length `n`.

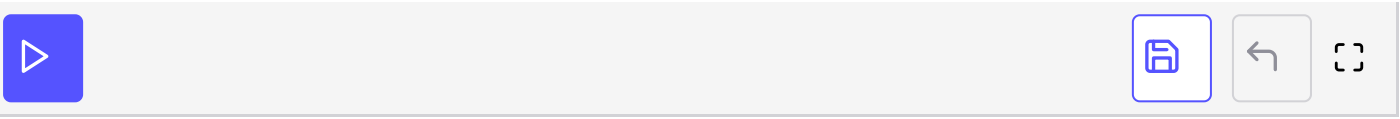
Here is some code that does nothing except output to the screen the number of input arguments, and their value:

```
#include <stdio.h>

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

```
  int i;
  printf("argc = %d\n", argc);
  for (i=0; i<argc; i++) {

    printf("argv[%d] = %s\n", i, argv[i]);
  }
  return 0;
}
```



You can find an illustration of your terminal below. Look closely at what is displayed by the code we just wrote:

```
plg:Desktop plg$ gcc -o dryrun dryrun.c
plg:Desktop plg$ ./dryrun //Command-line argument "./dryrun"
argc = 1 // Number of arguments
argv[0] = ./dryrun
plg:Desktop plg$
plg:Desktop plg$ ./dryrun hello // Two command-line arguments
argc = 2
argv[0] = ./dryrun
argv[1] = hello
plg:Desktop plg$
plg:Desktop plg$ ./dryrun 1 2 3 hi "the dude" 3.14159 // Multiple arguments
argc = 7
argv[0] = ./dryrun
argv[1] = 1
argv[2] = 2
argv[3] = 3
argv[4] = hi
argv[5] = the dude
argv[6] = 3.14159
```

Remember, all inputs are treated as null-terminated strings, so if you want to pass integers or floating point values in, you will have to convert them from strings in your code. Also remember that the first argument is always the name of the program, so if you expect to have to pass `n` input arguments to your program, then expect `argc` to be `n+1`.

We're pretty much done with arrays, but in C, there are other ways of grouping data together. One of these is called a **structure**. Find out more in the following lesson.