Command-line arguments in the C language

R.C. Maher EE475 Fall 2004

The C language provides a method to pass parameters to the `main()` function. This is typically accomplished by specifying arguments on the operating system command line (console).

The prototype for `main()` looks like:

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

There are two parameters passed to `main()`. The first parameter is the number of items on the command line (`int argc`). Each argument on the command line is separated by one or more spaces, and the operating system places each argument directly into its own null-terminated string. The second parameter passed to `main()` is an array of pointers to the character strings containing each argument (`char *argv[]`).

For example, at the command prompt:

```
test_prog 1 apple orange 4096.0
```

There are 5 items on the command line, so the operating system will set `argc`=5. The parameter `argv` is a pointer to an array of pointers to strings of characters, such that:

```
argv[0] is a pointer to the string “test_prog”
argv[1] is a pointer to the string “1”
argv[2] is a pointer to the string “apple”
argv[3] is a pointer to the string “orange”
```

and

```
argv[4] is a pointer to the string “4096.0”
```

Notes

- The `main()` routine can check `argc` to see how many arguments the user specified.
- The minimum count for `argc` is 1: the command line just contained the name of the invoked program with no arguments.
- The program can find out its own name as it was invoked: it is stored in the `argv[0]` string! Some operating systems don't provide this feature, however.
- The arguments from the command line are *not* automatically converted: the characters are just copied into the `argv` strings.

- If an argument on the command line is to be interpreted as a numerical constant, such as `argv[1]` and `argv[4]` in this example, it can be converted using a string conversion.

```
int int_val; float float_val;

sscanf(argv[1],”%d”,&int_val);
sscanf(argv[4],”%f”,&float_val);
```

and

```
printf(“The 3rd and 4th items on the command line are %s and
%s\n”, argv[2], argv[3]);
```

results in:

```
The 3rd and 4th items on the command line are apple and orange
```

The string functions `atoi()`, `atol()`, `atof()`, etc., will also work.