Command-Line Options and the std.getopt Module

In this lesson, you will learn about command-line options and the std.getopt module.

WE'LL COVER THE FOLLOWING

- Command-line options
- `std.getopt` module

In the previous lesson, you learned all there is to know about the parameters and the return value of `main()`. However, parsing the arguments is a repetitive task. The `std.getopt` module is designed to help with parsing the *command line options* of programs.

Command-line options

Some parameters like "world" and "hello" are purely data for the program to use. Other kinds of parameters are called command-line options and are used to change the behaviors of programs. An example of a command-line option is the `-l` option.

Command-line options make programs more useful by removing the need for a human user to interact with the program to make it behave in a certain way. With command-line options, programs can be started from script programs and their behaviors can be specified through command-line options. Although the syntax and meanings of command-line arguments of every program is specific to that program, their format is somewhat standard. For example, in POSIX, command-line options start with `--` followed by the name of the option, and values come after `=` characters:

```
# ./deneme --an-option=17
```

`std.getopt` module

The `std.getopt` module simplifies parsing such options. It has more capabilities than what is covered in this section.

Let’s design a program that displays random numbers. Let the program take the minimum, maximum and total count of these numbers as program arguments. We specify the following syntax to get these values from the command line:

```
# ./deneme --count=7 --minimum=10 --maximum=15
```

The `getopt()` function parses and assigns those values to variables. As we saw with `readf()`, the addresses of variables must be specified by the `&` operator:

```
import std.stdio;
import std.getopt;
import std.random;

void main(string[] args) {
    int count;
    int minimum;
    int maximum;

    getopt(args,
           "count", &count,
           "minimum", &minimum,
          "maximum", &maximum);

    foreach (i; 0 .. count) {
        write(uniform(minimum, maximum + 1), ' ');
    }

    writeln();
}
```

getopt( ) function

```
# ./deneme --count=7 --minimum=10 --maximum=15
11 11 13 11 14 15 10
```

Many command line options of most programs have a shorter syntax as well. For example, `-c` may have the same meaning as `--count`. Alternative syntax for each option is specified in `getopt()` after a `|` character. There may be more than one shortcut for each option:

```
getopt(args,
       "count|c", &count,

       "minimum|n", &minimum,
       "maximum|x", &maximum);
```

It is common to use a single dash for the short versions, and the `=` character is usually either omitted or substituted by a space:

```
# ./deneme -c7 -n10 -x15
11 13 10 15 14 15 14
# ./deneme -c 7 -n 10 -x 15
11 13 10 15 14 15 14
```

`getopt()` converts the arguments from string to the type of each variable. For example, since count above is an `int`, `getopt()` converts the value specified for the `--count` argument to an `int`. When needed, such conversions may also be performed explicitly by `to`.

So far, we have used `std.conv.to` only when converting to string. The function `to` can convert from any type to any type as long as that conversion is possible. For example, the following program takes advantage of `to` when converting its argument to `size_t`.

```
import std.stdio;
import std.conv;

void main(string[] args) {
    // The default count is 10
    size_t count = 10;
    if (args.length > 1) {
        // There is an argument
        count = to!size_t(args[1]);
    }

    foreach (i; 0 .. count) {
        write(i * 2, ' ');
    }

    writeln();
}
```

Use of size_t

The program produces 10 numbers when no argument is specified:

```
# ./deneme
0 2 4 6 8 10 12 14 16 18
# ./deneme 3
0 2 4
```

In the next lesson, we will learn about environment variables and how a program can start another program.