D Study Note

Refer to Programming in D by Ali Cehreli

Tae Geun Kim

2018-07-26

Hello, World!

1. Compile & Build

1) Compilers

D has three major compilers

- `dmd` : Digital Mars Compiler
- `gdc` : D Compiler of GCC
- `ldc` : D Compiler that targets the LLVM compiler infrastructure

In this note, I will use `dmd`.

2) How to compile?

```
1 $ dmd hello.d
```

3) Compile options

- `-de` : show use of deprecated features as errors
- `-unittest` : compile in unit tests
- `-w` : warnings as errors
- `-run` : Compiles source code and execute
- `-of=` : Output file

For example,

```
1 $ dmd hello.d -of=hello -de -w -unittest -run
```

2. `writeln` and `write`

In D, you can print any values with `writeln` or `write`. But, if you want to use these two functions then you should import `std.stdio`.

Here are some examples.

```d
// Hello, World!
import std.stdio;

void main() {
    writeln("Hello, World!", "Hello, HEP!");
}
```

```d
// Hello, World again!
import std.stdio;

void main() {
    write("Hello, ");
    write("World!");
    writeln();
}
```

Next examples are also allowed (explain next time)

```d
import std.stdio : writeln;

void main() {
    writeln("Hello, World!");
}
```

```d
void main() {
    import std.stdio : write, writeln;

    write("Hello, World!");
    writeln();
}
```

3. Compilation

Basic Concepts :

- Coding : Telling the CPU what to do
- Machine Code : Language of computer - So difficult

Then how to tell CPU? - **Programming Language!**

But, although human can understand programming languages, CPU can't understand. We need translator!

There are two kinds of translator

- Interpreter
- Compiler

Interpreter is a tool that reads the instructions from source code and executes them directly. The interpreter must read and understand the instructions every time the program is executed. For that reson, running a program with an interpreter is usually slower than running the compiled version of the same program. Python, R, Perl, Ruby and Javascript have been used with an interpreter.

Compiler is another tool that reads the instructions of a program from source code. Different from an interpreter, it does not execute the code; rather, it produces a program written in another language (usually machine code). Unlike an interpreter, the compiler reads and understands the source code only once, during compilation. For that reason and in general, a compiled program runs faster compared to executing that program with an interpreter. Ada, C, C++, Go, Rust and D have been used with an compiler.

Example

- Python : write `.py` file → interpreter → execute
- D : write `.d` file → compiler → create binary file → execute binary file

```
1 # Execute Python
2 $ python hello.py
3
4 # Compile D
5 $ dmd -o hello hello.d
6 # Execute D
7 $ ./hello
```

4. Fundamental Types

Type	Definition	Initial Value
`bool`	Boolean type	`false`
`byte`	signed 8 bits	`0`
`ubyte`	unsigned 8 bits	`0`
`short`	signed 16 bits	`0`
`ushort`	unsigned 16 bits	`0`
`int`	signed 32 bits	`0`
`uint`	unsigned 32 bits	`0`
`long`	signed 64 bits	`0L`
`ulong`	unsigned 64 bits	`0L`
`float`	32-bit floating point	`float.nan`
`double`	64-bit floating point	`double.nan`
`real`	the largest floating point type that the hardware support	`real.nan`
`ifloat`	imaginary value type of float	`float.nan * 1.0i`
`idouble`	imaginary value type of double	`double.nan * 1.0i`
`ireal`	imaginary value type of real	`real.nan * 1.0i`
`cfloat`	complex number type made of two floats	`float.nan + float.nan * 1.0i`
`cdouble`	complex number type made of two doubles	`double.nan + double.nan * 1.0i`
`creal`	complex number type made of two reals	`real.nan + real.nan * 1.0i`
`char`	UTF-8 code unit	`0xFF`
`wchar`	UTF-16 code unit	`0xFFFF`
`dchar`	UTF-32 code unit and Unicode code point	`0x0000FFFF`

1) Properties of types

D types have *properties*.

- `.stringof` : name of the type
- `.sizeof` : length of the type in terms of bytes
- `.min` : minimum value
- `.max` : maximum value
- `.init` : initial value

Example :

```
// code/prop.d
import std.stdio : writeln;

void main() {
    writeln("Type            : ", int.stringof);
    writeln("Length in bytes : ", int.sizeof);
    writeln("Minimum value   : ", int.min);
    writeln("Maximum value   : ", int.max);
    writeln("Initial value   : ", int.init);
}
```

5. Assignment and Order of Evaluation

1) The assignment operation

You can assign value for variable with = operator :

```
a = 10;
b = 20;
```

2) Order of evaluation

The operations of a program are applied step by step in the order that they appear in the program.

```
a = 10;
b = 20;
a = b;
```

Final result is : `a = 20, b = 20`

6. Variables

In previous chapter, we learned how to assign values to variables. But, in real, we can't. Next code occur error.

```
import std.stdio : writeln;

void main() {
    a = 10; // Error!
    a.writeln; // You can also write this way
}
```

We should declare variable's definition before assign values.

```
import std.stdio : writeln;

void main() {
    int a; // Declare integer a
    a.writeln; // Print initialize value of a
    a = 10; // Assign after declared. -> Okay!
    a.writeln; // Now 10
}
```

Or can do both declare and assign simultaneously.

```
import std.stdio : writeln;

void main() {
    int a = 10; // Declare and Assign together!
    a.writeln;
}
```

8. Reading from the Standard Input

Reading is also easy.

```d
// code/age.d
import std.stdio : readf, write, writeln;

void main() {
    write("How old are you?");

    int age;
    readf("%s", &age);
    writeln("Got it: Your age is ", age);
}
```

But there is a caveat. For some multiple read, you should insert space before `%s`.

```d
write("What's your age? ");
int age;
readf("%s", &age);

write("What's your favorite number? ");
int num;
readf("%s", &num); // Runtime Exception! (Not compile error)
```

Thus, you should modify this code to :

```d
// code/profile.d
import std.stdio;

void main() {
    write("What's your age? ");
    int age;
    readf(" %s", &age);

    write("What's your favorite number? ");
    int num;
    readf(" %s", &num);

    writeln("Your age is ", age, " and your favorite number is ", num);
}
```

2) Additional Information

You can write comment as follows :

```
// Single line of comment

/*
    Multiple lines of comment
*/

/+
    It also
+/

/++
    General documentation information comment
+/
```