Quick Start With D

Release 0.0.2

Ilya Yaroshenko

September 16, 2015

Contents

CHAPTER 1

Introduction

It was mentioned that students can quickly master the D programming language without a detailed study using mostly its subset, which is close to the C PL.

Consider a simple program that reads from a file of 10 lines, each containing a single number and prints to the standard output at the same number, but shifted to the mathematician expectation.

Whereas idiomatic D code looks pretty unusual:

```
import std.algorithm.iteration : map, each;
import std.array : array;
import std.conv : parse;
import std.range : takeExactly, tee;
import std.stdio;

void main()
{
    double mean = 0;
    auto sample = File("10numbers.txt")
         .byLine
         .takeExactly(10)
         .map!(line => line.parse!double)
         .tee!((x){mean += x;})
         .array;
    mean /= sample.length;
    // prints one element per line
    sample.map!(x => x - mean).each!writeln;
}
```

for many unfamiliar with the language D the same program can be implemented even as more complex, but at the same time more understandable way:

```
import std.stdio;

void main()
{
    File fin = File("10numbers.txt");
    double[] sample;
```

```
    sample.length = 10;
    double mean = 0;
    for(int i = 0; i < sample.length; i++)
    {
        fin.readf("%s", &sample[i]);
        if(!fin.eof)
            fin.readln();
        mean += sample[i];
    }
    mean /= sample.length;
    // prints one element per line
    for(int i = 0; i < sample.length; i++)
    {
        writeln(sample[i] - mean);
    }
}
```

The present documentation is submitted to the rapid introduction to D for those who are already somehow familiar with the C language and for some reasons do not want to waste time on a consistent study of the D language and related tools.

If you decide to use the D language in your daily work, you should start immediately with the study of the official page[1] and of the book "The D Programming Language"[2] by Andrei Alexandrescu.

Probably D is the most powerful of the present system programming languages[3].

D is a dragon [4]. *Have a nice flight!*

[1] http://dlang.org
[2] http://erdani.com/index.php/books/tdpl/
[3] http://en.wikipedia.org/wiki/System_programming_language
[4] D is a dragon, or why D matters for Bioinformatics (http://thebird.nl/blog/D_Dragon.html) by Pjotr Prins.

CHAPTER 2

Examples

D is a complex multi-paradigm programming language. At the same time, if you know C programming language and you want to start using D then you just need to look through some examples.

Tip: All examples available on GitHub[1].

2.1 Hello, World!

C programs can be easily translated to D. The following program prints "Hello, World!" to the standard output.

```
#include <stdio.h>

const char* const nullTerminatedStrPtr = "Hello, World!";

int main(void)
{
    puts(nullTerminatedStrPtr);
    return 0;
}
```

D doesn't have a preprocessor[2]. Use `import core.stdc.MODULE;` construction to import `MODULE` from the C Standard library[3].

```
import core.stdc.stdio;

// terminates with a null character
immutable char[] nullTerminatedStr = "Hello, World!\0";

int main()
{
```

[1] http://github.com/andralex/thenextafterc
[2] http://dlang.org/pretod.html
[3] http://www.cplusplus.com/reference/clibrary/

```
    // calls external C function
    puts(nullTerminatedStr.ptr);
    return 0;
}
```

Module `core.stdc.stdio` contains the `puts` prototype:

```
extern(C) @system nothrow @nogc int puts(in char* s);
```

Common D "Hello, World!" program which is based on Phobos looks more simple:

```
/++
Deduces the type of a declared variable from its initialization
expression.
+/
immutable str = "Hello, World!";

void main()
{
    // Scoped and selective imports can be used.
    import std.stdio : writeln;
    writeln(str);
}
```

Phobos[4] is the standard runtime library that comes with the D language compiler.

See also:

To find a collection of common C techniques, and to find out how to do the corresponding task in D click here[5]. However most of them can be implemented in C style.

2.2 Simple project with dub

DUB[6] is a build tool for D projects with support for automatically retrieving dependencies and integrating them in the build process. The design emphasis is on maximum simplicity for simple projects, while providing the opportunity to customize things when needed.

To create the initial project with name `component`, run `dub init component`.

Remove automatically created `component/source/app.d` file and create the following file structure

```
dub.json
component/
    source/
        component/
            mod.d
            package.d
```

[4] http://dlang.org/phobos/
[5] http://dlang.org/ctod.html
[6] http://code.dlang.org/getting_started

where `component/package.d` is the main module `component`

```
/++
Pacakge component
+/
module component;

public import component.mod : removeSingleLineComments;
```

and `component/mod.d` is the inner module `component.mod`

```
/++
Module mod;
+/
module component.mod;

import std.algorithm, std.ascii, std.range, std.string, std.functional;

/++
Reads forward range `ir` and removes single line comments.
The result is stored in output range `or`.

Params:
    or = output range
    ir = input range
    cmt = comment prefix (like // in C or # in Python)
+/
void removeSingleLineComments
    (OutputRange, Range1, Range2)            // template parameters
    (OutputRange or, Range1 ir, Range2 cmt)  // function parameters
{
    foreach(line; lineSplitter(ir))
    {
        if(line.save.find!(not!isWhite).startsWith(cmt))
            continue;             //skips line
        put(or, line.until(cmt)); //skips comment
        put(or, "\n");
    }
}

/// Unittests with comment appears in documentation.
unittest
{
    auto app = appender!string;

    // A string that contains a code with C-like block syntax
    // can be framed with `q{` and `}`.
    immutable textBefore = q{
// main function
int main()
{
    // return statement
```

```
    return 0; //returns 0
}
};

    immutable textAfter = q{
int main()
{
    return 0;
}
}; // Note: "return 0; " ends with a space character.

    removeSingleLineComments(app, textBefore, "//");

    debug
    {
        import std.stdio;
        writeln("text:", app.data);
    }

    assert(app.data == textAfter);
}
```

To test this module, run `dub test` from package's folder.

`removeSingleLineComments` can be imported with `import component;` or `import component.mod;`. To use *component* package, put the following dependency into your project's dub.json into the dependencies section:

```
{
    ...
    "dependencies": {
        "component": "~master"
    }
}
```

2.3 Plotting with matplotlib (python)

These are two projects that can be used with the D programming language:

- Plotcli[7] is a command line application written in D that can create plots from text/csv files and from piped data, making it useful during data analysis.
- PLplot[8] is a cross-platform software package written in C for creating scientific plots. It includes low-level D bindings.

But these two are not so convenient to use, in comparison with matplotlib.

[7] https://github.com/BlackEdder/plotd
[8] http://plplot.sourceforge.net

matplotlib[9] is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell, web application servers, and different graphical user interface toolkits. To integrate with python the PyD package can be used.

PyD[10] is a library that provides seamless interoperability between the D programming language and Python. The minimal configuration file for this example is

```
{
    "name": "plotting_example",
    "dependencies": {
        "pyd": "~>0.9.4",
    },
    "subConfigurations": {
        "pyd": "python34",
    },
}
```

Note: The python[11] and matplotlib should be installed. PyD searches the version of the python that is noted in the sub-configuration (`"pyd": "python34"` in this example). For more information, see the PyD's dub configuration file[12].

The following program[13] reads data from a file and runs `show_histogram.py`.

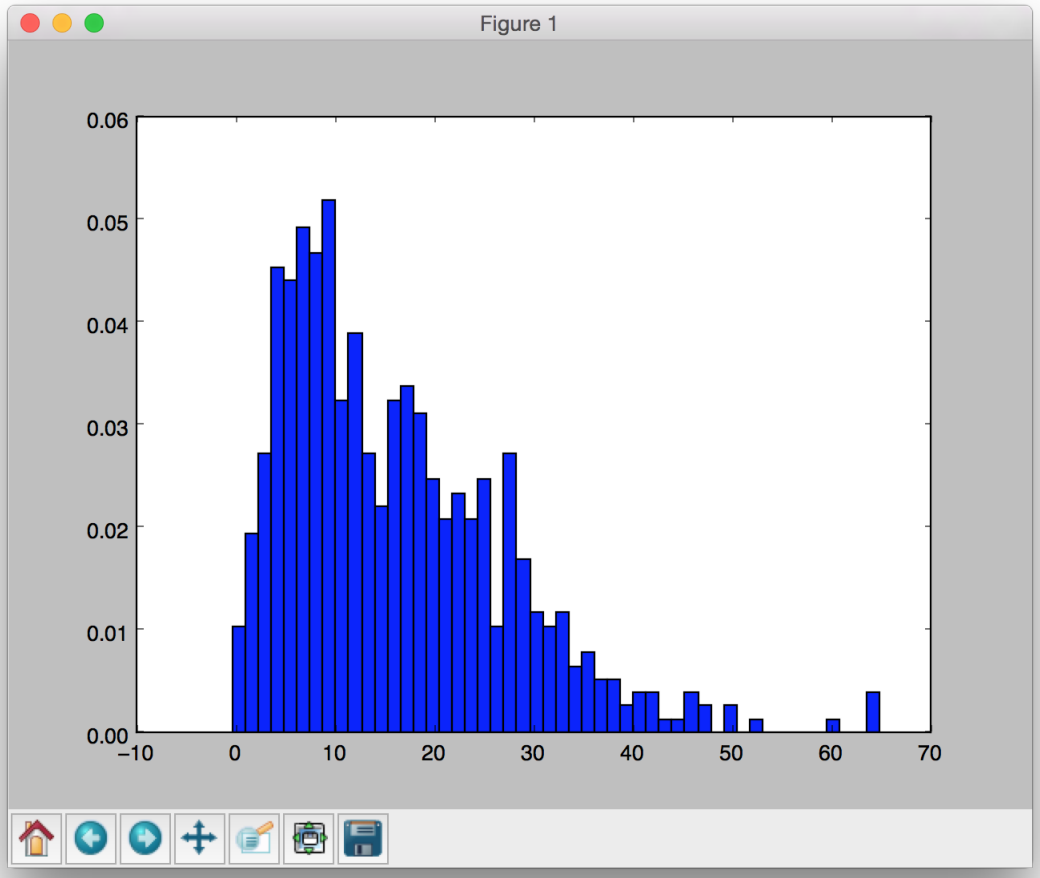
```
import pyd.pyd;
import pyd.embedded;
import pyd.extra;

/++
`srcipt` is a string that contains the python code to execute.

Alternatively, you can put your python code here:

--------
immutable script = `
YOUR = "PYTHON"; CODE = "HERE"
print(YOUR, CODE)
`;
--------

where string is framed with backtick character.
+/
immutable script = import("show_histogram.py");

/++
`d_to_python_numpy_ndarray` converts a D array to numpy.ndarray.
```

[9] http://matplotlib.org
[10] http://pyd.readthedocs.org
[11] https://www.python.org/downloads/
[12] https://github.com/ariovistus/pyd/blob/master/dub.json
[13] https://github.com/andralex/thenextafterc/tree/master/examples/matplotlib

```
`toNumpyArray` is only an alias.
+/
alias toNumpyArray = d_to_python_numpy_ndarray;

/++
A static constructor is a function that performs initializations of
thread local data before the `main()` function gets control for the
main thread.

Shared static constructors are executed before any static
constructors, and are intended for initializing any shared global
data.
+/
shared static this() {
    //initializes PyD package.
    py_init();
}

void main()
{
    auto pythonContext = new InterpContext();
    /+
    Uniform Function Call Syntax (UFCS)
    is used in the following line of code.

    Equivalent code would be just:
    --------
    pythonContext.sample = toNumpyArray(readData("data/data.txt"));
    --------
    +/
    pythonContext.sample = "data/data.txt".readData.toNumpyArray;
    pythonContext.num_bins = 50;
    pythonContext.py_stmts(script);
}

double[] readData(string file)
{
    import std.algorithm.iteration : map, splitter;
    import std.array : array;
    import std.conv : to;
    import std.file : readText;

    return file
        .readText   //Reads the contents of a text file into a string.
        .splitter        //Lazily splits words.
        .map!(to!double) //Lazily converts words to doubles.
        .array;          //Creates an array.
}
```

`show_histogram.py` is located in `views/` folder that is used by dub as a default string import [14] folder.

```
# variables `sample` and `num_bins` should be defined in context
import matplotlib.pyplot as plt
n, bins, patches = plt.hist(sample, num_bins, normed=1)
plt.show()
```



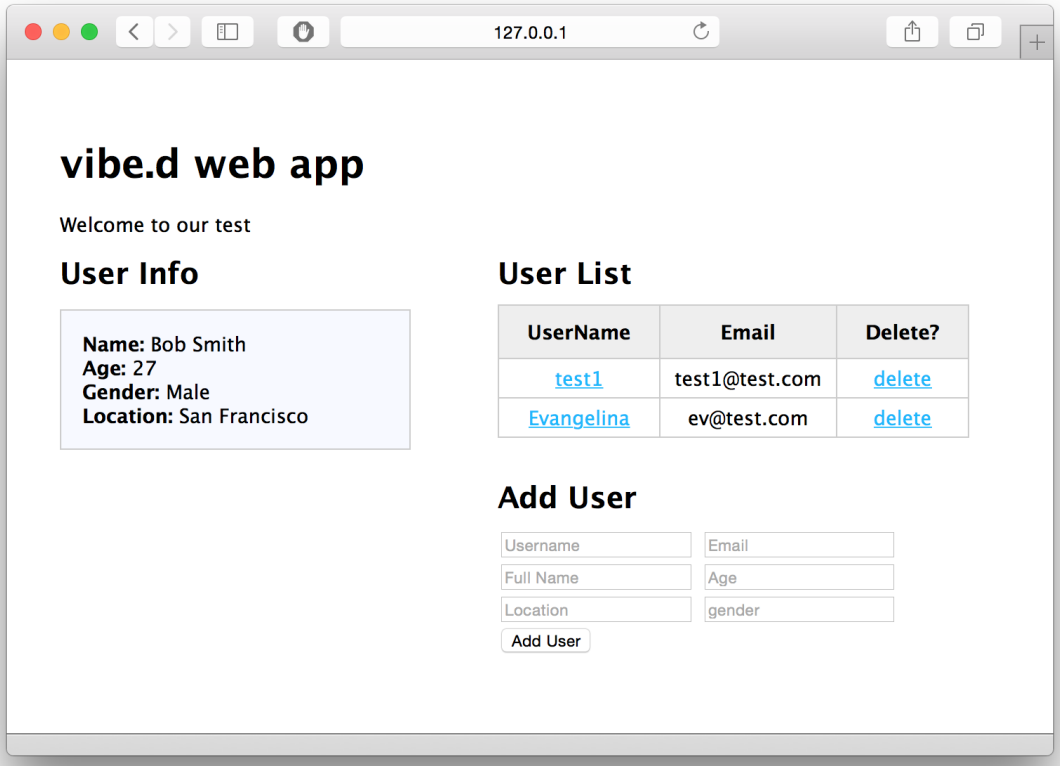
2.4 Web Application

Web application is a pretty good example of the last chapters of any book, where the reader is suggested to make use of the means of expression in the language. As a rule, web application is a complex product, both in terms of knowledge of the language and in terms of code complexity of the used libraries.

And this example is no exception. Then why do people who want to learn D language very quickly still need example of web app? Many of them have a reason and it is that they need to integrate quickly programs written in D with other services, programming languages and databases.

[14] Strings can be imported at compile time.

The article "Creating a simple restful web app with node.js, Express, and MongoDB[15]" by Christopher Buecheler is taken as a basis for this example.



2.4.1 Initialization

To create a skeleton web application, run:

```
$ dub init vibed-mongo vibe.d
```

This will make the directory `vibed-mongo` with a minimal HTTP server based on vibe.d[16].

The configuration file `dub.json` will look something like this:

```
{
    "name": "vibed-mongo",
    "dependencies": {
        "vibe-d": "~>0.7.23"
    },
    "versions": ["VibeDefaultMain"],
    "authors": ["Christopher Buecheler", "Ilya Yaroshenko"]
}
```

The version `"VibeDefaultMain"` includes the main function defined by default.

[15] http://cwbuecheler.com/web/tutorials/2014/restful-web-app-node-express-mongodb/
[16] http://vibed.org

The project has the following structure:

```
dub.json             - package information
source/              - D source code
    app.d
    service.d
views/               - Diet templates
    index.dt
    layout.dt
public/              - static directories
    javascripts/
        global.js
    styles
        style.css
    favicon.ico
```

After installing MongoDB[17], run the server

```
$ mongod
```

In another console run the Mongo interpreter

```
$ mongo
> use vibed
switched to db vibed
> db.createCollection("userlist", {autoIndexID : true})
{ "ok" : 1 }
> db.userlist.insert({
    'username' : 'test1',
    'email' : 'test1@test.com',
    'fullname' : 'Bob Smith',
    'age' : 27,
    'location' : 'San Francisco',
    'gender' : 'male'
    })
WriteResult({ "nInserted" : 1 })
> exit
bye
```

The above script creates a `vibed` database with a `userlist` collection, which will contain one record.

2.4.2 Patches

Comparing with the original article `global.js` was slightly changed:

```
        $.ajax({
            type: 'POST',
            data: newUser,
            url: '/users/adduser',
```

[17] https://www.mongodb.org

```
            success: function(data){
                $('#addUser fieldset input').val('');
                populateTable();
            },
            error: function(xhr, textStatus, error){
                alert(xhr.responseText);
            }
        });
```

```
        $.ajax({
            type: 'DELETE',
            url: '/users/deleteuser/' + $(this).attr('rel'),
            success: function(data){
                populateTable();
            },
            error: function(xhr, textStatus, error){
                alert(xhr.responseText);
            }
        });
```

2.4.3 Diet Templates

Diet templates[18] are HTML templates which are statically compiled down to native D code. Diet templates syntax equals that of Jade templates with the exception of some of the advanced syntax features.

First lines of `index.dt`:

```
extends layout

block content
    - /++
    - The main difference is that you write D expressions and
    - statements instead of JavaScript.
    - +/
    - import std.format;
    - auto fullTitle = format("%s web app", title);

    h1= fullTitle
    p Welcome to our test
```

2.4.4 Service

`vibe.d` is a good example of the use of declarative programming with D. Service performs an *insert*, *select* and *remove* operations for user entries at a mongo collection.

[18] http://vibed.org/templates/diet

```
module service;

import std.conv;
import vibe.d;

class MongoService
{
    private MongoCollection collection;
    const string title;

    this(MongoCollection collection, string title = "")
    {
        this.collection = collection;
        this.title = title;
    }

    void index()
    {
        logInfo("MongoService: GET /");
        render!("index.dt", title);
    }

    void postAdduser(
        string username,
        string email,
        string fullname,
        uint age,
        string location,
        string gender,
        HTTPServerResponse res,
    )
    {
        import vibe.utils.validation;

        logInfo(text("MongoService: POST /adduser : ", username));
        enforce(age < 200, "wrong age");

        auto bson     = Bson.emptyObject;
        bson.username = validateUserName(username);
        bson.email    = validateEmail(email);
        bson.fullname = fullname;
        bson.age      = age;
        bson.location = location;
        bson.gender   = gender.toLower;

        collection.insert(bson);
        res.writeBody("");
    }

    Json getUserlist()
    {
```

```
        logInfo("MongoService: GET /userlist");
        return Json(collection.find!Json.array);
    }

    @path("deleteuser/:id")
    @method(HTTPMethod.DELETE)
    void pullOutUser(BsonObjectID _id, HTTPServerResponse res)
    {
        logInfo(text("MongoService: GET /deleteuser/", _id));
        collection.remove(["_id": _id]);
        res.writeBody("");
    }
}
```

2.4.5 App

The following static constructor connects `vibed` database, creates the HTTP server and implements the error handler.

```
import vibe.d;
import service;

shared static this()
{
    immutable string title = "vibe.d";

    logInfo("Connecting to DB...");
    auto db = connectMongoDB("localhost").getDatabase("vibed");
    auto collection = db["userlist"];

    logInfo("Creating service...");
    auto mongoService = new MongoService(collection, title);
    auto mongoServiceSettings = new WebInterfaceSettings;
    mongoServiceSettings.urlPrefix = "/users";


    logInfo("Setup router...");
    auto router = new URLRouter;
    router.registerWebInterface(mongoService, mongoServiceSettings);
    router
        .get("/", (req, res)
            { res.redirect("/users"); } )
        .get("*", serveStaticFiles("public/"));

    logInfo("Setup HTTP server...");
    auto settings = new HTTPServerSettings;
    with(settings)
    {
        bindAddresses = ["127.0.0.1"];
```

```
        port = 8080;
        errorPageHandler =
            (req, res, error)
            {
                with(error) res.writeBody(
                    format("Code: %s\n Message: %s\n Exception: %s",
                        code,
                        message,
                        exception ? exception.msg : ""));
            };
    }

    //Listening http://127.0.0.1:8080
    listenHTTP(settings, router);
}
```

CHAPTER 3

Integration with other languages

3.1 C and friends

D that has[1] full support for C ABI [2] had recently been significantly improved for interfacing with C++[3] (however there is no support for C++ exceptions). Jacob Carlborg did a great job of integrating with Objective-C, which is still waiting to be no less grandiose review by Walter Bright.

3.2 Scripting languages

You are already somehow familiar with the integration of scripting languages on the example of the use of the matplotlib library and PyD[4]. Since most of them have a C API [5], their integration with D can be performed without problems.

There is a realization[6] of the ECMA 262 (Javascript) programming language written by Walter Bright and updated by Dmitry Olshansky.

It is also worth mentioning the Lua programming language. Unlike many other libraries built on the Lua C API, LuaD[7] does not expose the Lua stack - instead, it has wrappers for references to Lua objects, and supports seamlessly and directly converting any D type into a Lua type and vice versa.

[1] http://dlang.org/interfaceToC.html
[2] Application Binary Interface
[3] http://dlang.org/cpp_interface.html
[4] https://github.com/ariovistus/pyd
[5] Application Programming Interface
[6] https://github.com/DigitalMars/DMDScript
[7] https://github.com/JakobOvrum/LuaD

CHAPTER 4

Links

General:

- http://dlang.org - The D Programming Language
- http://wiki.dlang.org - The D Wiki
- http://code.dlang.org - The D Package Registry
- http://forum.dlang.org - Discussion
- http://dconf.org - The Conference
- http://ddocs.org - Documentation for all packages published at the D Package Registry (temporary unavailable)

Books[1]**:**

- **The D Programming Language**, Andrei Alexandrescu, June 12, 2010.
  - Amazon[2]
  - Read chapter 1 online - "D"iving In[3]
  - Read chapter 13 online - Concurrency[4]
  - Errata[5]
- **Programming in D**, Ali Çehreli - The online book[6]
- **D Cookbook**, Adam D. Ruppe, May 26, 2014
  - Publisher's page[7]
  - Amazon[8]
- **D Templates: A Tutorial**, Philippe Sigaud - Free book[9]

[2] http://www.amazon.com/D-Programming-Language-Andrei-Alexandrescu/dp/0321635361
[3] http://www.informit.com/articles/article.aspx?p=1381876
[4] http://www.informit.com/articles/article.aspx?p=1609144
[5] http://erdani.com/tdpl/errata/
[6] http://ddili.org/ders/d.en/index.html
[7] http://www.packtpub.com/discover-advantages-of-programming-in-d-cookbook/book
[8] http://www.amazon.com/dp/1783287217/?tag=packtpubli-20
[9] https://github.com/PhilippeSigaud/D-templates-tutorial

- **Pragmatic D Tutorial**, Andreas Zwinkau - Website[10]
- **Developing with compile time in mind**, Richard Cattermole, February 17, 2015 - Website[11]
- **D programming**, January 1, 2015 - Website[12]

Compilers[13]:

- DMD[14] - The reference D compiler
- LDC[15] - LLVM D Compiler
- GDC[16] - GCC D Compiler

Development Environments[17]:

- Visual D[18] - integration into Visual Studio
- Mono-D[19] - D support to the cross-platform XamarinStudio/MonoDevelop IDE
- DKit[20] - a package to aid developing D programs using Sublime Text 3
- DDT[21] - an Eclipse IDE for the D programming language

[10] http://qznc.github.io/d-tut/index.html
[11] https://leanpub.com/ctfe
[12] http://www.tutorialspoint.com/d_programming/
[14] http://dlang.org/download.html
[15] https://github.com/ldc-developers/ldc
[16] http://gdcproject.org/downloads
[18] https://github.com/D-Programming-Language/visuald/releases
[19] http://wiki.dlang.org/Mono-D
[20] https://github.com/yazd/DKit
[21] http://ddt-ide.github.io