Quick Start With D

Release 0.0.2

Ilya Yaroshenko

April 27, 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:

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

void main()
{
    double mean = 0;
    auto sample = File("10numbers.txt")
        .byLine
        .takeExactly(10)
        .map!(to!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:

```d
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.

2.1 Hellow Wolrd!

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[1]. Use `import core.stdc.MODULE;` construction to import `MODULE` from the C Standard library[2].

```
import core.stdc.stdio;

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

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

[1]http://dlang.org/pretod.html

[2]http://www.cplusplus.com/reference/clibrary/

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[3] 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[4]. However most of them can be implemented in C style.

2.2 Simple project with dub

DUB[5] 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.

2.3 Plotting with matplotlib (python)

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

- Plotcli[6] 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[7] 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.

[3] http://dlang.org/phobos/
[4] http://dlang.org/ctod.html
[5] http://code.dlang.org/getting_started
[6] https://github.com/BlackEdder/plotd
[7] http://plplot.sourceforge.net

matplotlib[8] 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[9] 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",
    },
}
```

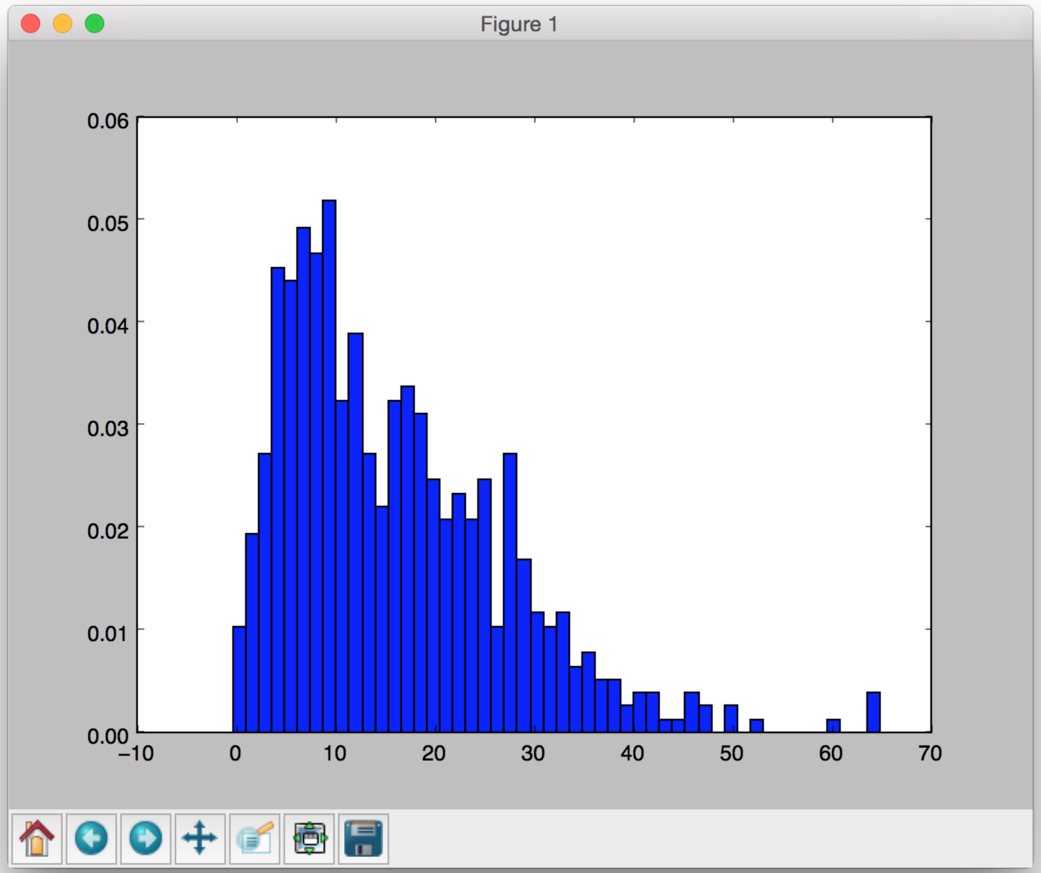
Note: The python should be installed[10]. PyD searches the version of the python that is noted in the sub-configuration (`"pyd": "python34"` in this example). For more information, see the PyD's dub configuration file[11].

The following program[12] reads data from a file and shows the histogram.

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

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

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

```
}

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("view/data.txt"));
    ---------
    +/
    pythonContext.sample = "data/data.txt".readData.toNumpyArray;
    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.
}

immutable script = `
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

num_bins = 50
ax = plt.subplot()
n, bins, patches = ax.hist(sample, num_bins, normed=1)
plt.show()
`;
```



2.4 Web Application

Web application is a pretty good example of the last chapters of the 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 complexity used in this library.

And this example is no exception. Then why do people who want to learn this language very quickly have a need in it? 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.

The article "Creating a simple restful web app with node.js, Express, and MongoDB[13]" by Christopher Buecheler is taken as a basis for this example.

2.4.1 Initialization

To create a skeleton web application, run:

[13] http://cwbuecheler.com/web/tutorials/2014/restful-web-app-node-express-mongodb/

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

This will make in directory `vibed-mongo` with a minimal HTTP server based on vibe.d.

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

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

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

The project has the following structure:

After installing MongoDB run Mongo servers

```
$ mongod
```

In another console run Mongo console

```
$ 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 script above will create 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',
            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 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 MongoDB.

```
module service;

import std.conv;
import vibe.d;

class MongoService
{
    private MongoCollection collection;
    const string title;

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

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

    void postAdduser(
        string username,
        string email,
        string fullname,
        int age,
        string location,
        string gender,
        HTTPServerResponse res,
    )
    {
        logInfo(text("MongoService: POST /adduser : ", username));

        import vibe.utils.validation;
        auto bson = Json.emptyObject;
        bson.username = validateUserName(username);
        bson.email = email.validateEmail;
        bson.fullname = fullname;
        enforce(age < 200 && age >= 0, "wrong age");
        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("");
    }
}
```

2.4.4 App

Following static constructor connects `vebid` MongoDB, creates vibe.d server and implements an error handler.

```
import vibe.d;
import service;

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

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

    logInfo("Querying DB...");
    //Bson query = Bson(["name" : Bson("hans")]);
    auto result = collection.find();

    logInfo("Iterating results...");
    foreach (i, doc; result)
        logInfo("Item %d: %s", i, doc.toJson().toString());

    auto mongoService = new MongoService(collection, title);

    auto mongoServiceSettings = new WebInterfaceSettings;
    mongoServiceSettings.urlPrefix = "/users";

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

    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",
                        error.code,
                        error.message,
                        error.exception ? error.exception.msg : ""));
            };
    }
    listenHTTP(settings, router);
}
```

CHAPTER 3

Integration with other languages

NOT READY!

3.1 C and friends

D that has full support for C ABI (application binary interface) had recently been significantly improved from C++ ABI [1] (however it's worth noting that there is no support for 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 library matplotlib (python). Since most of them have a C API [2], their integration with D can be performed without problems.

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

It is also worth mentioning a popular computer games scripting language Lua. Unlike many other libraries built on the Lua C API, LuaD 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] Application Binary Interface
[2] Application Programming Interface

CHAPTER 4

Links

NOT READY!

4.1 General

4.2 Libraries and Frameworks

4.3 Articles

4.4 Compilers

4.5 Build systems

4.6 Development Environments

4.7 Documentation generators

4.8 Online tools