Testing Text Editors & IDEs for the D Language

OMG How Many Text Editors & IDEs are There

Being a long time daily reader of the Dlang forums, I have noticed that new people often ask what is the best editor to use. The usual answers are Visual Studio Code, some ones other favorite editor, or check out the text editor pages at on Dlang's web site. I am going to attempt answering this question, hoping to supply a more complete answer and benefit new D language users.

Detailed Description

- What is expected from a text editor for the D language
- Text editors & IDEs that work on Windows and/or Linux
- Will cover all the text editors & IDEs on the Dlang web site
- Ease or difficulty of configuration or plugins required
- Whether the user can edit C & C++ files also
- Whether the user can edit multiple files at the same time
- Whether code completion and call tips are available
- Whether debugging or internal terminal is available
- Whether git or other source control integration is possible
- Whether it is possible to run the compiled executable from the menu
- Whether it is possible to compile the current file from the menu
- Whether it is possible to build a multi file project from the menu

Who am I

- I am not a programmer, system analysis or software engineer
- BA degree in Biology concentrating in microbiology, chemistry minor
- Worked 20 years as a Clinical Lab Technologist & research assistant
- Took programming classes at Junior College after retiring the first time
- First programming class was C, then some different flavors of assembly and on to Java, C++ and Visual Basic ending with a associate degree in computer science with certification in OOP and Visual Basic
- I found the Digital Mars web site while looking for C compilers online
- I still have the Windows only version 1.030 D compiler and the Dcode text editor on my Windows 11 computer
- Now, I am basically a retired hobby programmer who uses mainly D and a little C

What to Expect from Programming Editors

Text Editors with built in line number & syntax highlighting

Code folding

Run command from menu item

Code Snippets

Open a Terminal in the working folder from menu item

Project or workspace folder

Compile the current file from menu item

Build project from menu item preferably with dub for D editors

Symbol | function | outline | tag list generation

Code completion and function call tips

Integrated debugging

Integrated Development Editor (IDE)

No Longer Usable or Available

Atom

- It all its repositories have been archived 12/15/22
- The last available version was 1.60
- Downloaded it two months ago, Dlang plugin was not available
- It still supports many other languages with its builtin packages, but no updates
- Can’t view community package info, install community packages or themes

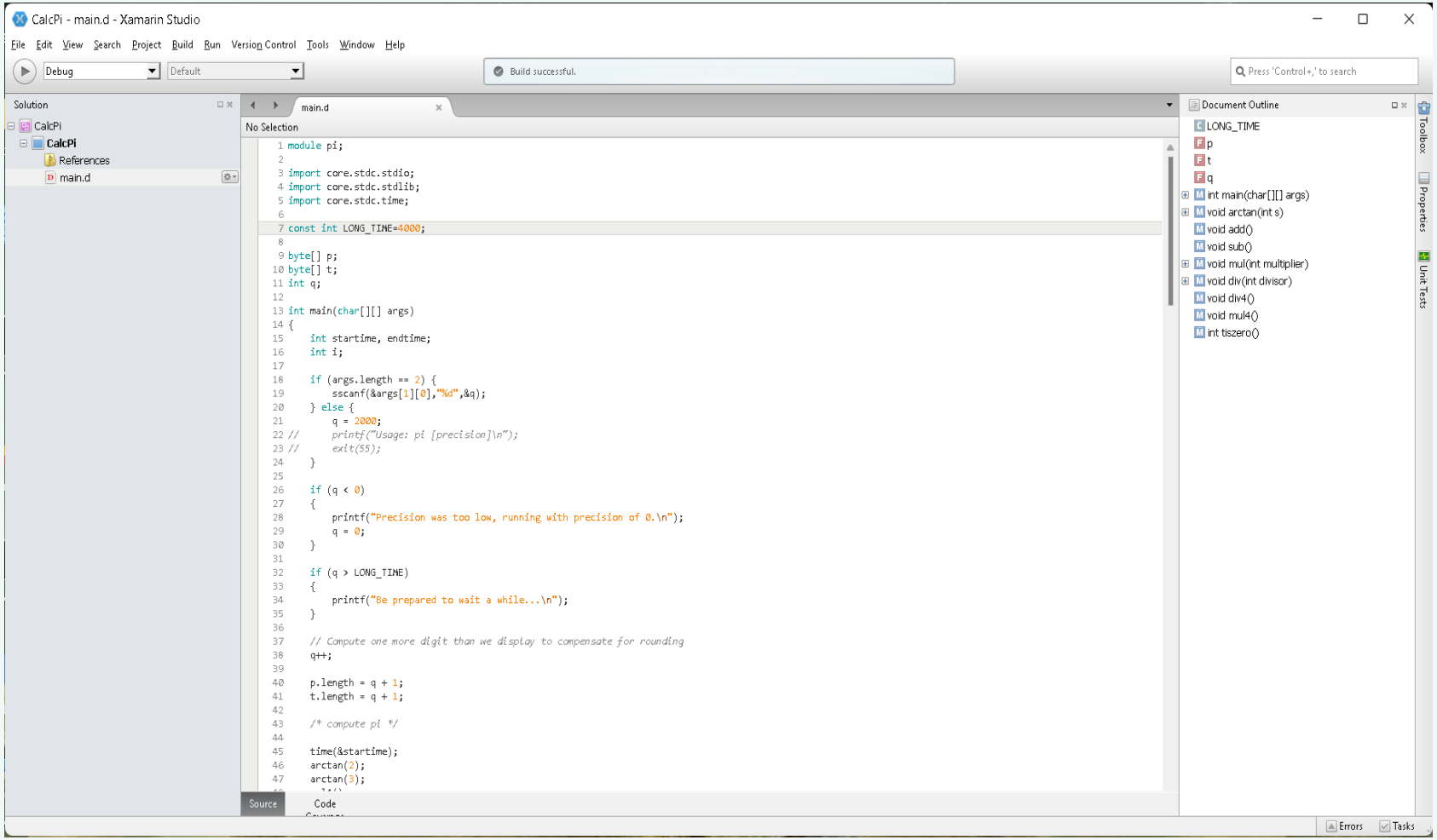
Dhee

- It can still be downloaded, but it was not usable on my Linux desktop which no longer supports python 2
- It needs pyqt4 which is not supported, but can be compiled
- pyqt4 needs sip version 4 to compile, however it is also out of date I had sip version 6, pyqt5 and pyqt6 installed on my Linux desktop

Monodevelop

- No active development after 2020
- the source code for all versions can be downloaded
- If all dependencies are available it can be compiled on Linux and Windows.
- The D plugin Mono-d is not being actively developed
- If can still be downloaded, but only usable in Monodevelop v5.8 to the last v5.10
- In Linux distros that use the deb or rpm install files can install the latest v5.10
- When starting up a warning appears about missing ”GnomePlatform,5”

Xamarin Studio on Windows



Editors Built only for D Programming
Legacy Editors Only Usable on Windows

Dcode

- The first editor made strictly for the D language
- By Christopher E. Miller who also developed Entice Designer
- Had syntax highlighting, code folding and edited single file
- Had project menu item to create a project with a default run command, which could run any compiler, dub or OS command like bash or cmd.exe

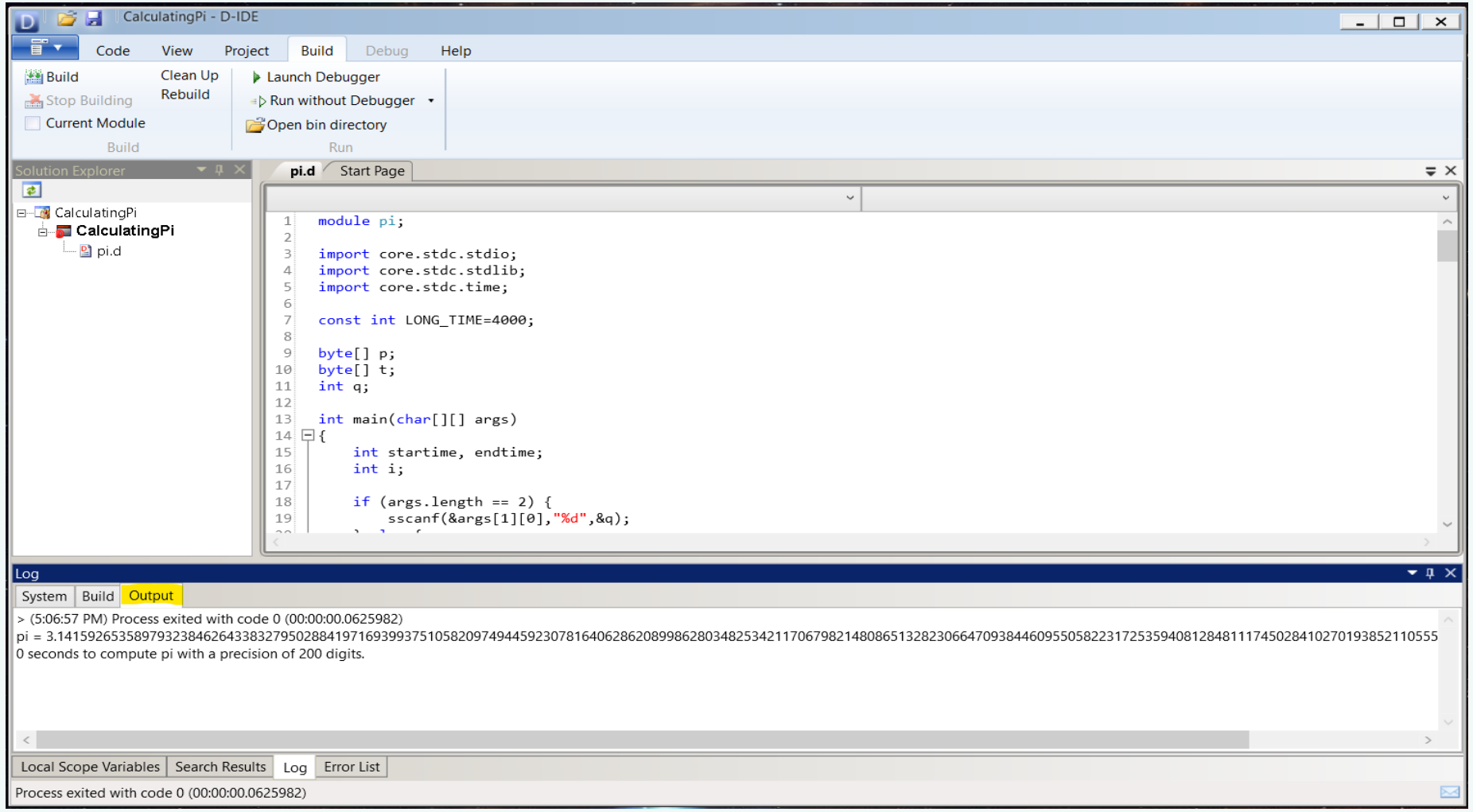
Entice Designer

- The first form designer strictly for the D language.
- It had syntax highlighting and code folding
- It had a project dialog and run command menu item to input commands to run the compiler, the executable or any OS command like cmd.exe
- However, it could also edit multiple files in a project in tabs in the same window.

D-IDE

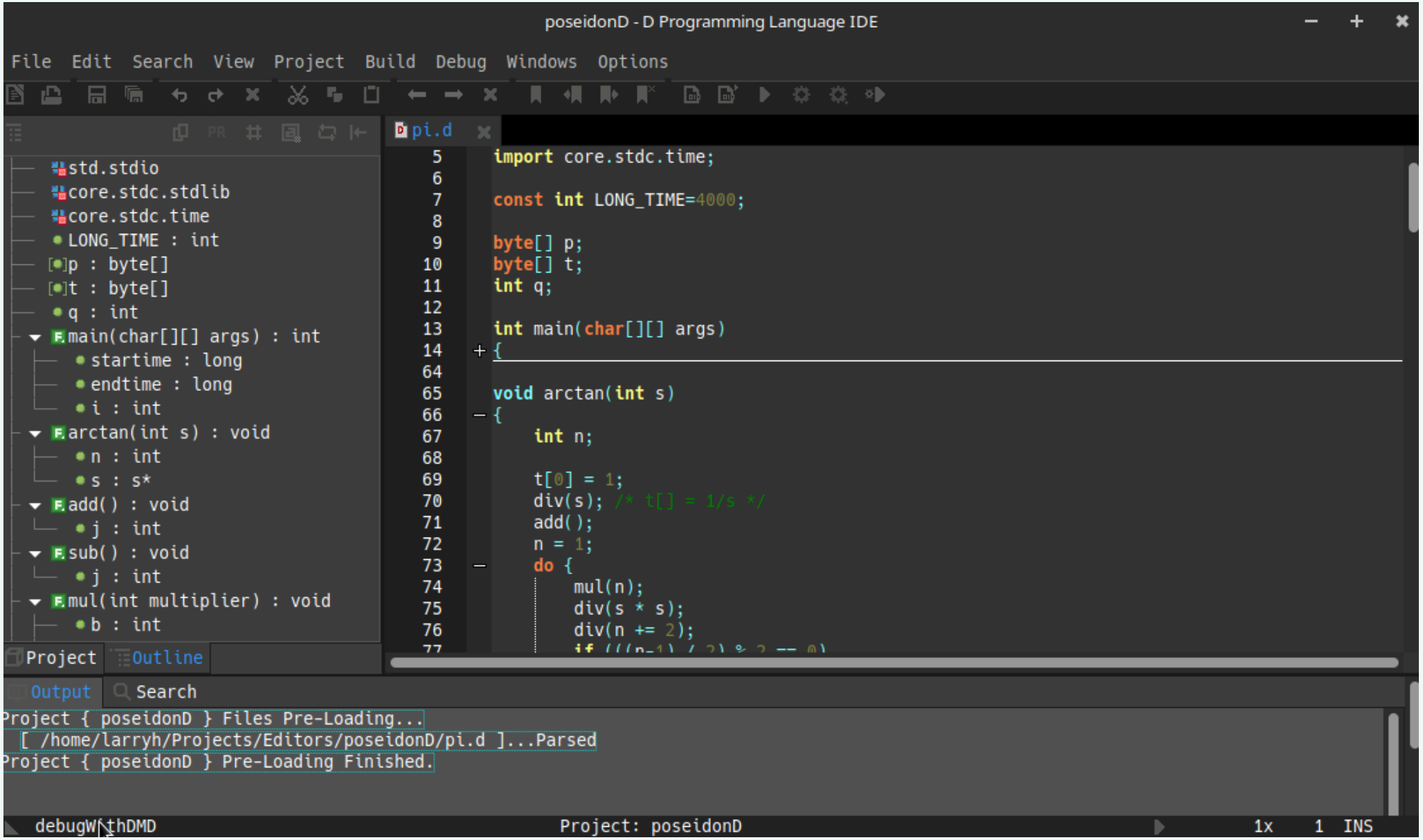
- Developed by Alexander Bothe who later went on to create the Mono-D plugin for Monodevelop
- I was not able to test the ability to debug programs, because the default debug parameters included the -gc argument
- The release build and run w/o debugging both run okay
- Of course it also had syntax highlighting and code folding

D-IDE on Windows



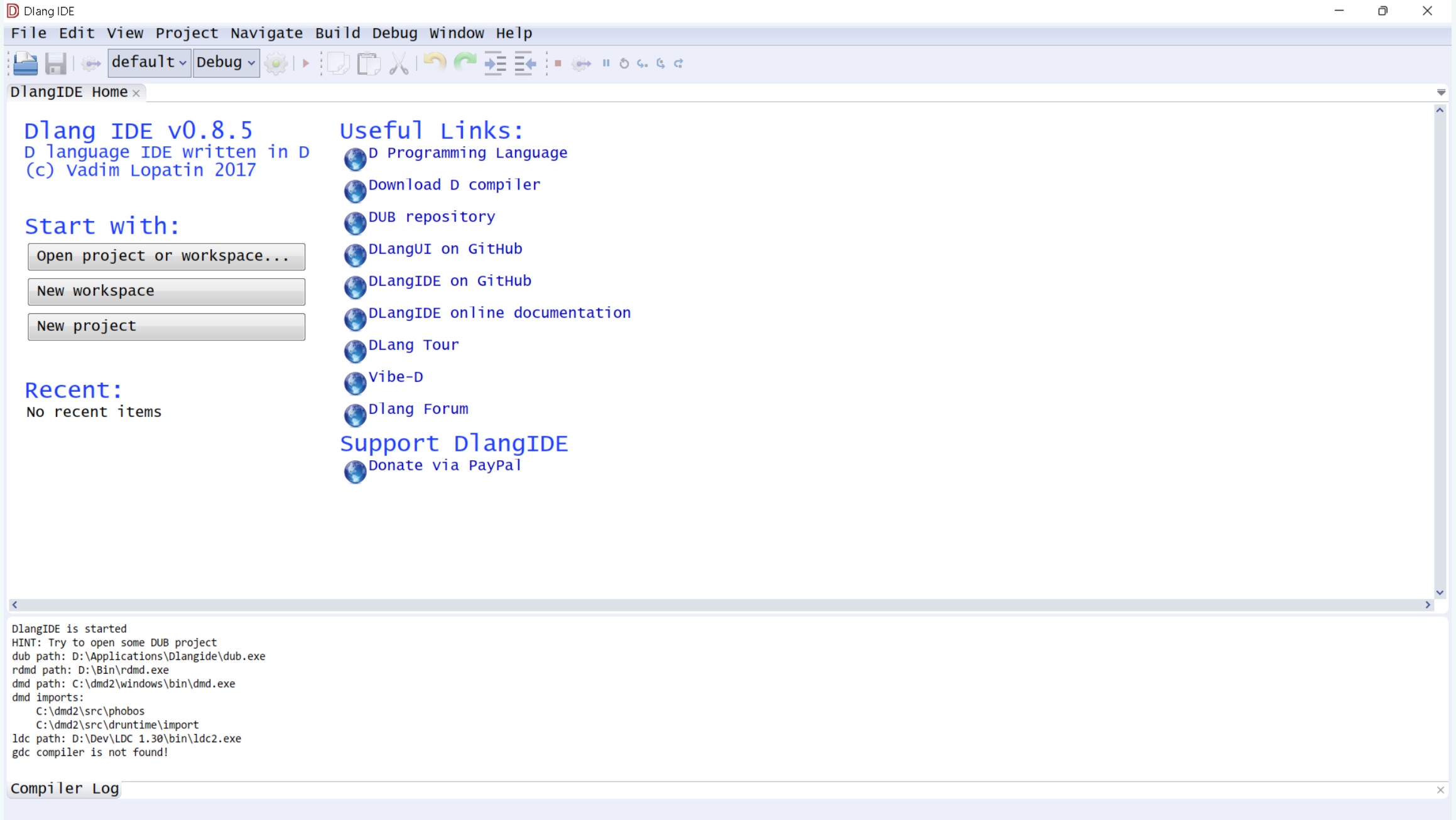
Built for D - poseidonD

- The poseidonD IDE is current & updated along with poseidonFB because the old FreeBasic IDE was not current & not maintained by the original developer. So both are now maintained by the same persons and are updated concurrently.
- The newest version has runtime errors on Windows, but the older version I have still works on Windows and the new version does work on Linux
- There is no provision for using dub and no menu item that opens a terminal. Does however launch a terminal to run the executable, but the terminal will not stay open unless the program keeps it open with the getchar() function.
- Can not add any arguments to the executable before running the program from the menu
- Has syntax highlighting, code folding, compile file. run the executable, code completion, call tips, goto definition and symbol generation
- The default compile option is just -g, but custom compile configurations can be created for each of the three compilers for different debug or release options
- The build menu item compiles every d file in the folder separately to object file and links them into an executable with the name of the project (with no dub support) like people use make to compile a project
- There is the ability to create custom tools to add to the tool menu, but when adding a terminal it would not open in the project folder. Instead the terminal would always open in the folder that poseidonD executable was in
- Debugging would not work on Windows or Linux and could not figure out why
- Written in D using the iup graphics library


poseidonD - D Programming Language IDE
File Edit Search View Project Build Debug Windows Options
std.stdio
core.stdc.stdlib
core.stdc.time
LONG_TIME : int
p : byte[]
t : byte[]
q : int
main(char[][] args) : int
startime : long
endtime : long
i : int
arctan(int s) : void
n : int
s : s*
add() : void
j : int
sub() : void
j : int
mul(int multiplier) : void
b : int
Project
Outline
pi.d
5 import core.stdc.time;
6
7 const int LONG_TIME=4000;
8
9 byte[] p;
10 byte[] t;
11 int q;
12
13 int main(char[][] args)
14 {
64
65 void arctan(int s)
66 {
67 int n;
68
69 t[0] = 1;
70 div(s); /* t[] = 1/s */
71 add();
72 n = 1;
73 do {
74 mul(n);
75 div(s * s);
76 div(n += 2);
Output
Search
Project { poseidonD } Files Pre-Loading...
[/home/larryh/Projects/Editors/poseidonD/pi.d]...Parsed
Project { poseidonD } Pre-Loading Finished.
debugWithDMD
Project: poseidonD
1x 1 INS

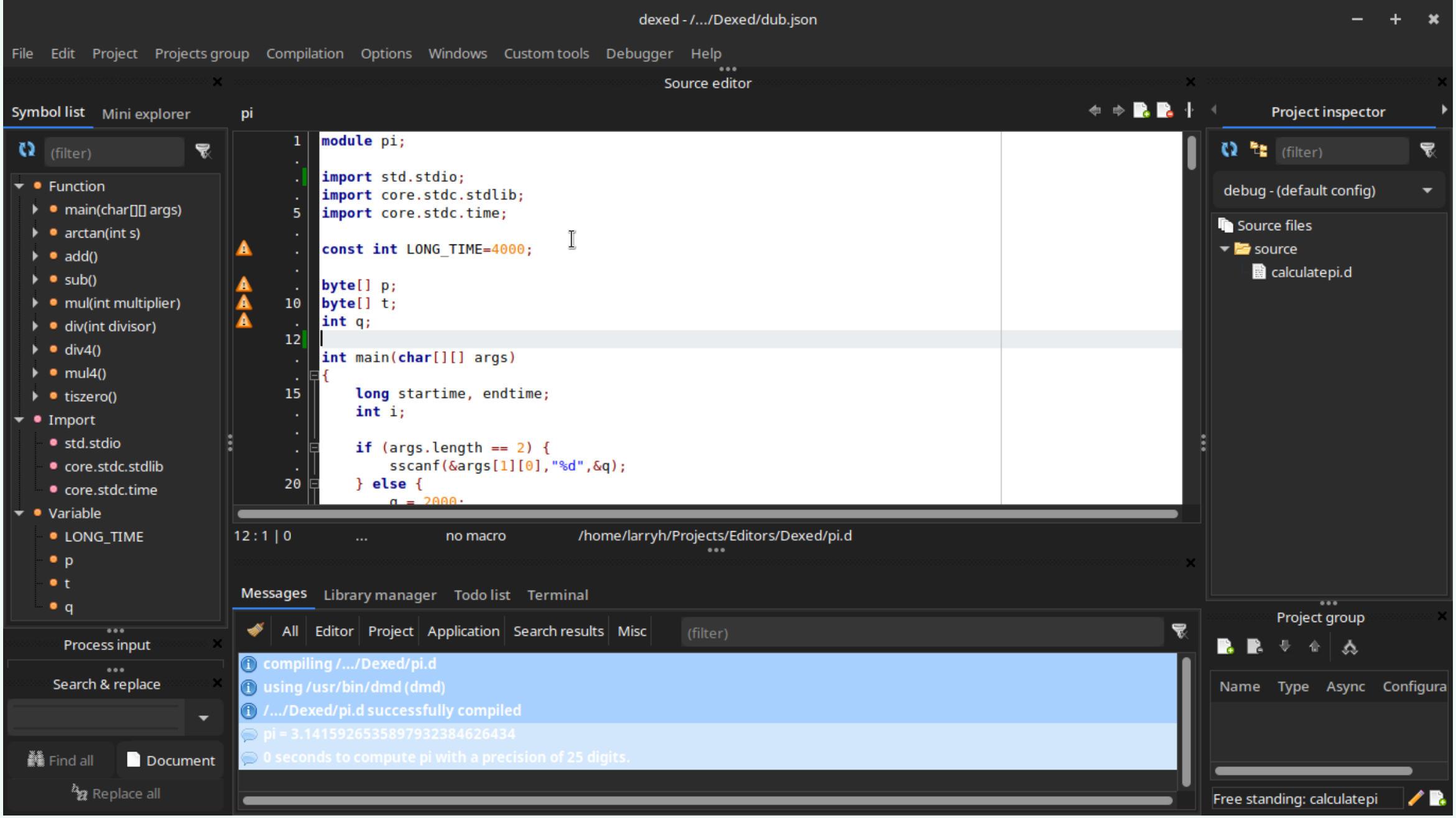
Built for D - Dlangide

- Written in D for the D programming language using dlangiu (d binding to the IUP graphic library)
- Can download from github release page the latest Windows only version 0.8.5 compiled on 9/2/2017
- On Linux you can build from a git clone, just follow the instruction on the readme doc and use dub. If possible use the ”-d” compile argument to silence all the many depreciations in the libraries used.
- The Linux program has a bug that can usually be fixed by setting the dpi to 96 in the options
- There is syntax highlighting, and can run the executable using rdmd but no code folding
- New project and workspace dialog uses dub to build and then run the executable in an external terminal
- There is automatic code completion as you type. Goto definitions and call tips work via shortcut key, although the menu item for these appears to not work
- Debugging did not work compiling with dmd or ldc; got the error “breakpoint binding error, no symbols have been loaded”. The compiler switches -gf for dmd and --gdwarf for ldc did not work since they caused dub to error during the build when adding them to the compiler options dialog
- Did not try adding them to dflags options in the dub.json configuration file (which may work)


Dlang IDE
File Edit View Project Navigate Build Debug Window Help
default
Debug
DlangIDE Home
Dlang IDE v0.8.5
D language IDE written in D
(c) Vadim Lopatin 2017
Start with:
Open project or workspace...
New workspace
New project
Recent:
No recent items
Useful Links:
D Programming Language
Download D compiler
DUB repository
DLangUI on GitHub
DLangIDE on GitHub
DLangIDE online documentation
DLang Tour
Vibe-D
Dlang Forum
Support DlangIDE
Donate via PayPal
DlangIDE is started
HINT: Try to open some DUB project
dub path: D:\Applications\Dlangide\dub.exe
rdmd path: D:\Bin\rdmd.exe
dmd path: C:\dmd2\windows\bin\dmd.exe
dmd imports:
C:\dmd2\src\phobos
C:\dmd2\src\druntime\import
ldc path: D:\Dev\LDC 1.30\bin\ldc2.exe
gdc compiler is not found!
Compiler Log

Built for D - Dexed

- Setup, deb, rpm and binary packages for Linux distros, but have to build your own for Window or download it from the gitlab appveyor within one month of a new release. (anxiously awaiting a new release, hint, hint)
- The terminal does not work or open from the menu item, probably because the Linux distro used does not have vte2
- When building a dub project, the project would not build and got a dub error exit status 127, this may be due to what looked like dub was trying to build dub.json.
- However, dub worked okay when executed from an external terminal opened from the project folder outside the IDE
- When trying to debug the debugger runs immediately to the end of the program without stopping at any of the three breakpoints set. This has happened in other IDEs, the -g dmd compile switch is used instead of the -gf switch and there was no way to change the default debug settings in the IDE that I could see.
- It has multi file editing in multiple tabs, highlighting, code folding, run from menu item, source control, compiling single files, symbol generation and code completion, which all works very well.
- Did not try editing any C/C++ files to check the highlighting, although the readme file claims it works as a nice addition


dexed - /.../Dexed/dub.json
File Edit Project Projects group Compilation Options Windows Custom tools Debugger Help
Source editor
Symbol list
Mini explorer
pi
Project inspector
(filter)
Function
main(char[][] args)
arctan(int s)
add()
sub()
mul(int multiplier)
div(int divisor)
div4()
mul4()
tiszero()
Import
std.stdio
core.stdc.stdlib
core.stdc.time
Variable
LONG_TIME
p
t
q
1 module pi;
5 import std.stdio;
import core.stdc.stdlib;
import core.stdc.time;
const int LONG_TIME=4000;
byte[] p;
10 byte[] t;
int q;
12
int main(char[][] args)
{
15 long startime, endtime;
int i;
if (args.length == 2) {
sscanf(&args[1][0],"%d",&q);
20 } else {
12 : 1 | 0 ... no macro /home/larryh/Projects/Editors/Dexed/pi.d
debug - (default config)
Source files
source
calculatepi.d
Process input
Search & replace
Find all
Document
Replace all
Messages
Library manager
Todo list
Terminal
All Editor Project Application Search results Misc
(filter)
compiling /.../Dexed/pi.d
using /usr/bin/dmd (dmd)
/.../Dexed/pi.d successfully compiled
pi = 3.14159265358979323846264338
0 seconds to compute pi with a precision of 25 digits.
Project group
Name Type Async Configura
Free standing: calculatepi

Code Editors with Built In D Support

Kwrite

- Syntax highlighting and code folding built in for D and many other programming languages
- Can only edit a single file per window, no tabs or split window
- Need to open a external terminal in the project folder to use dub or any compiler or run the executable
- This is just a basic text editor that is barely a code editor
- Linux only

Xed

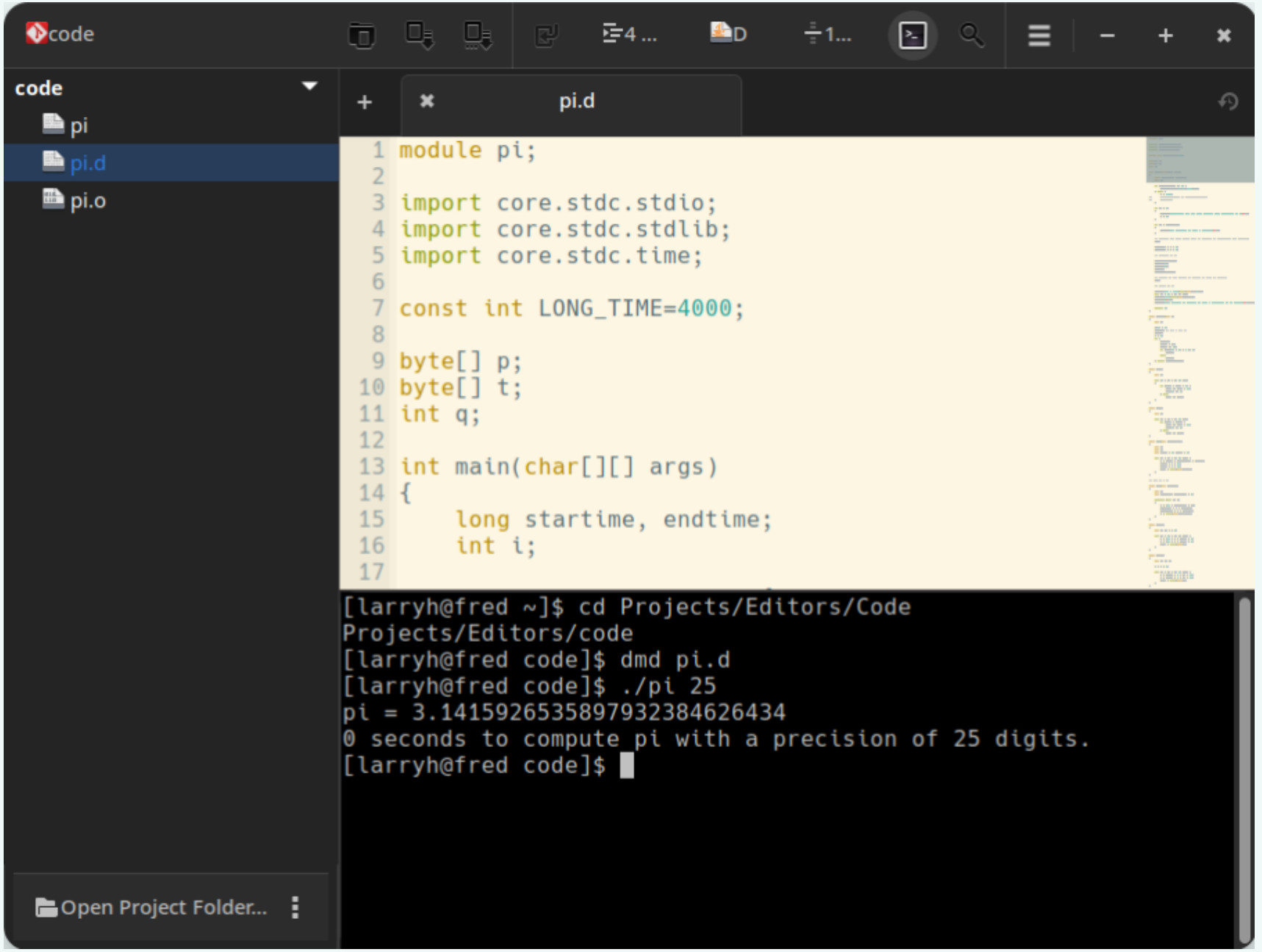
- Syntax highlighting built in for many programming languages including D, automatic selection by extension
- No terminal and no external tools can be configured
- To use dub, any compiler or run any shell script or run any executable must open OS terminal in the project folder
- The ctags plugin to generate symbols only work for C files
- Toggle comments is for C style only.
- Linux only

Code

- I simple code editor with built in syntax highlighting for D and many other programming languages
- No code folding or code completion, has word completion
- Internal terminal opens in the current files project folder
- Current document is auto saved after any file change
- Never have to press control + S and the only save on the file menu is “save as...” menu item
- Linux only and maybe only on Arch distros

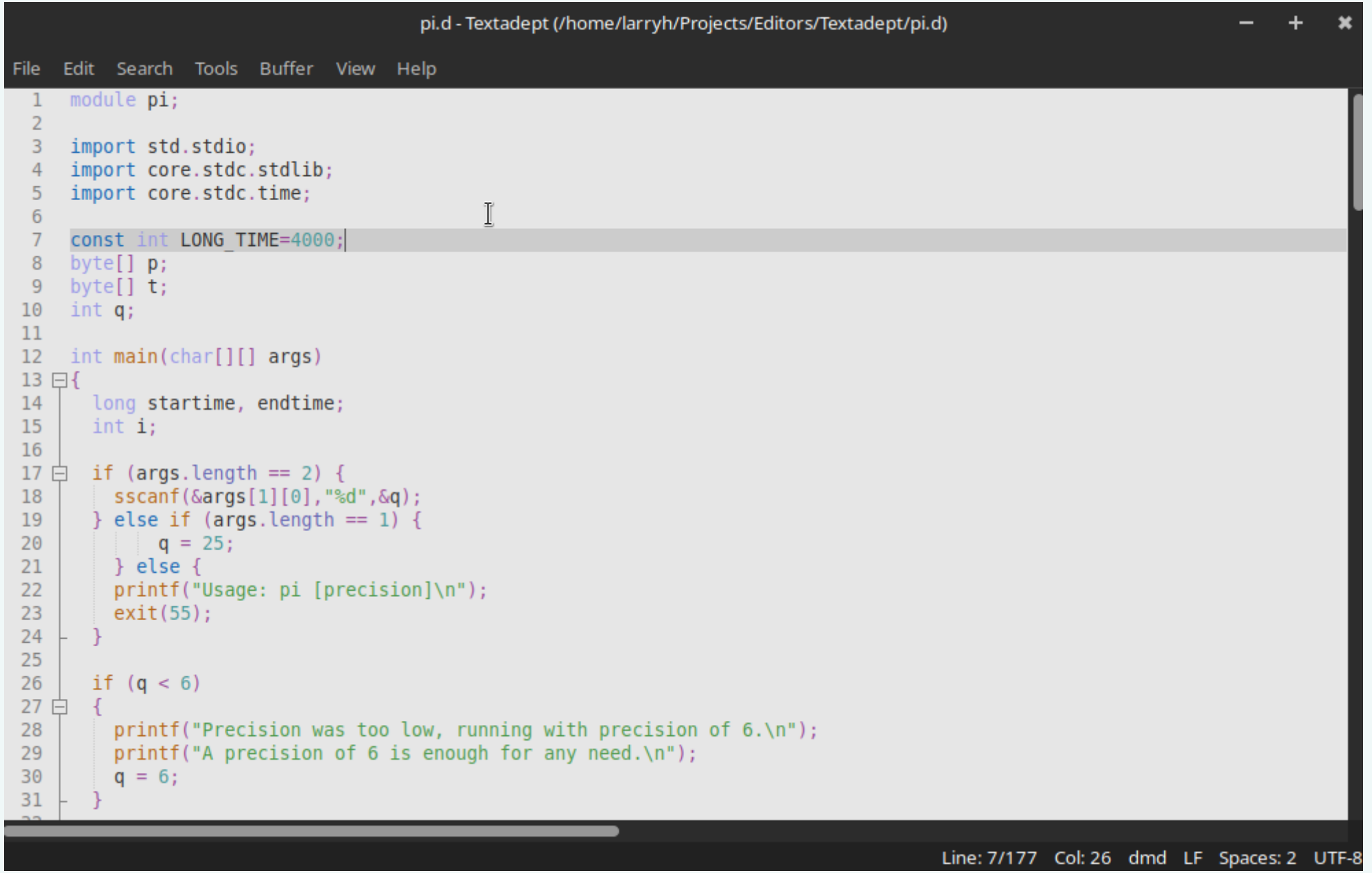
Scite

- Has D syntax highlighting and code folding, not by default
- The d properties file is present, but excluded in the list at the end of the global.properties file, just delete the d in the exclude list
- Some of the defaults in d.properties file needs updating
- Fortunately any properties file can be opened from the option menu and edited as a text file
- The default build uses make and the compile is optimized
- Usable on Windows and Linux


code
code
pi
pi.d
pi.o
Open Project Folder...
pi.d
1 module pi;
2
3 import core.stdc.stdio;
4 import core.stdc.stdlib;
5 import core.stdc.time;
6
7 const int LONG_TIME=4000;
8
9 byte[] p;
10 byte[] t;
11 int q;
12
13 int main(char[][] args)
14 {
15 long startime, endtime;
16 int i;
17
[larryh@fred ~]$ cd Projects/Editors/Code
Projects/Editors/code
[larryh@fred code]$ dmd pi.d
[larryh@fred code]$ ./pi 25
pi = 3.14159265358979323846264338
0 seconds to compute pi with a precision of 25 digits.
[larryh@fred code]$

Code Editors with only one D Plugin

VisualD Plugin for Visual Studio	Textadept with texadept.d plugin
• Can use Visual Studio not only program in D, but also C, C++, C#, F# and Visual Basic • It has almost everything one would need in an IDE, code completion, symbol generation, build, compile, run menu items, source control and even debugging. • The only menu item to run the compiled app is to run without debugging which closes the terminal when the program ends unless the program keeps it open with the getchar() hack. • It has a menu item to compile & run the results that runs the files unittest, which can be configured differently • Unfortunately, there is no menu item to open an internal or external command prompt to run OS commands or external programs from in the project directory • Of course it also has highlighting and code folding • It has a good debugging interface and the capability to debug the compiled files	• If one clicks on the preferences menu item they get a blank default init.lua file, where default commands can be entered to the users preference if they know lua • The textadept.d plugin is easily installed and written in lua so it is very easy to modify • The plugin uses DCD for code completion, call tips, symbol generation and goto definitions • It uses Dscanner for syntax and style checks when you save the file and uses Dfmt to format the file by with a menu click or keyboard shortcut • It was possible to edit textadept.d's init.lua file to change the run_compile command to a debug compile command with the usual debug arguments • I tried creating a build command using dub, which would not run even after running dub init from a cmd line in the project folder • Then tried using dmd -i ... but it would not run either


pi.d - Textadept (/home/larryh/Projects/Editors/Textadept/pi.d)
File Edit Search Tools Buffer View Help
1 module pi;
2
3 import std.stdio;
4 import core.stdc.stdlib;
5 import core.stdc.time;
6
7 const int LONG_TIME=4000;
8 byte[] p;
9 byte[] t;
10 int q;
11
12 int main(char[][] args)
13 {
14 long startime, endtime;
15 int i;
16
17 if (args.length == 2) {
18 sscanf(&args[1][0],"%d",&q);
19 } else if (args.length == 1) {
20 q = 25;
21 } else {
22 printf("Usage: pi [precision]\n");
23 exit(55);
24 }
25
26 if (q < 6)
27 {
28 printf("Precision was too low, running with precision of 6.\n");
29 printf("A precision of 6 is enough for any need.\n");
30 q = 6;
31 }
Line: 7/177 Col: 26 dmd LF Spaces: 2 UTF-8

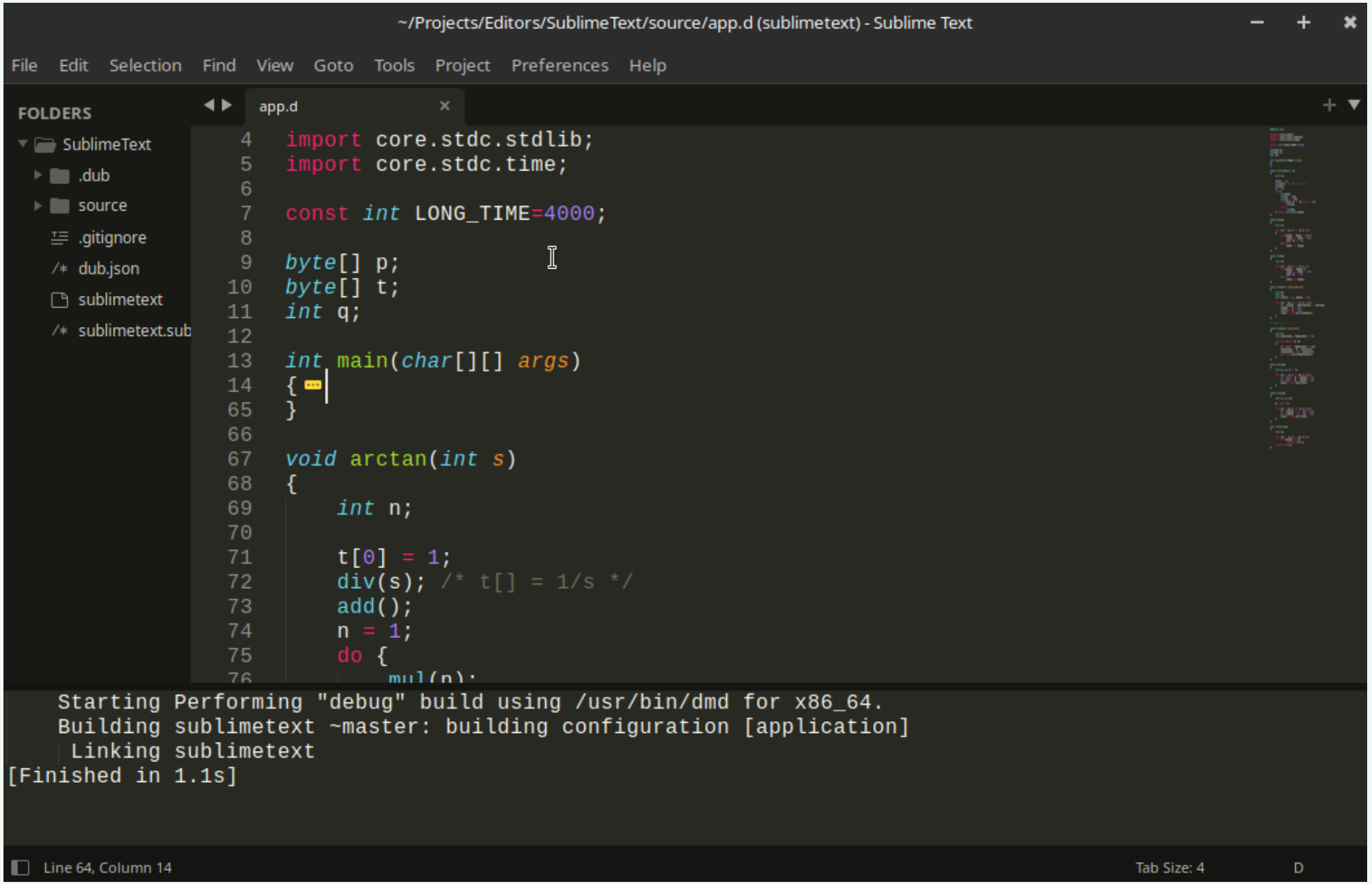
Code Editors with only one D Plugin

Sublime Text with Dkit

- There is a code-d by webfreak for sublime text, but I have used dkit for years and was already familiar with it
- It has highlighting, code folding, code completion and call tips which are automatic as you type most of the time
- Can go to definition of function under cursor by pressing F12 or from the menu
- It includes snippets and build commands already configured using rdmd, dmd or dub for debug configuration
- However, there is no release build commands on the build menu, they are easily added fortunately
- Build menu also includes includes single file builds using a run script header to identify it as such
- If one opens the command palette using ctl+shf+p shortcut and type in "build" you get a complete list of them
- Type "dkit" in the command palette to see the many dkit commands including "create dub package" and "create project from dub package file" which have no menu items

VS Code with Code-d

- Applies to Visual Studio Code, VScodium and Code OSS
- Uses the plugin developed by webfreak which is used in a few other editors
- If you what debugging one can use the other plugin as well by webfreak
- Provides d language support using dub, DCD, Dscanner and Dfmt for projects, code completion, call tips, syntax and style checking, symbol generation and code formatting
- Use "dub init" with a terminal in the project folder, then open the project folder as a workspace will auto configure the dub task for the user
- If not using dub, just open the folder as a workspace and configure the default task in the task.json file
- With the other plugin the debugging with code-d works okay if one uses the "-gf" debug switch
- With the "-g" debug switch debugging does not work very well, because the debugger runs through breakpoints to the end of the program


~/Projects/Editors/SublimeText/source/app.d (sublimetext) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
SublimeText
.dub
source
.gitignore
dub.json
sublimetext
sublimetext.sub
app.d
4 import core.stdc.stdlib;
5 import core.stdc.time;
6
7 const int LONG_TIME=4000;
8
9 byte[] p;
10 byte[] t;
11 int q;
12
13 int main(char[][] args)
14 { ...
65 }
66
67 void arctan(int s)
68 {
69 int n;
70
71 t[0] = 1;
72 div(s); /* t[] = 1/s */
73 add();
74 n = 1;
75 do {
Starting Performing "debug" build using /usr/bin/dmd for x86_64.
Building sublimetext ~master: building configuration [application]
Linking sublimetext
[Finished in 1.1s]
Line 64, Column 14
Tab Size: 4
D

Code Editors with only One D plugin

Itellij idea

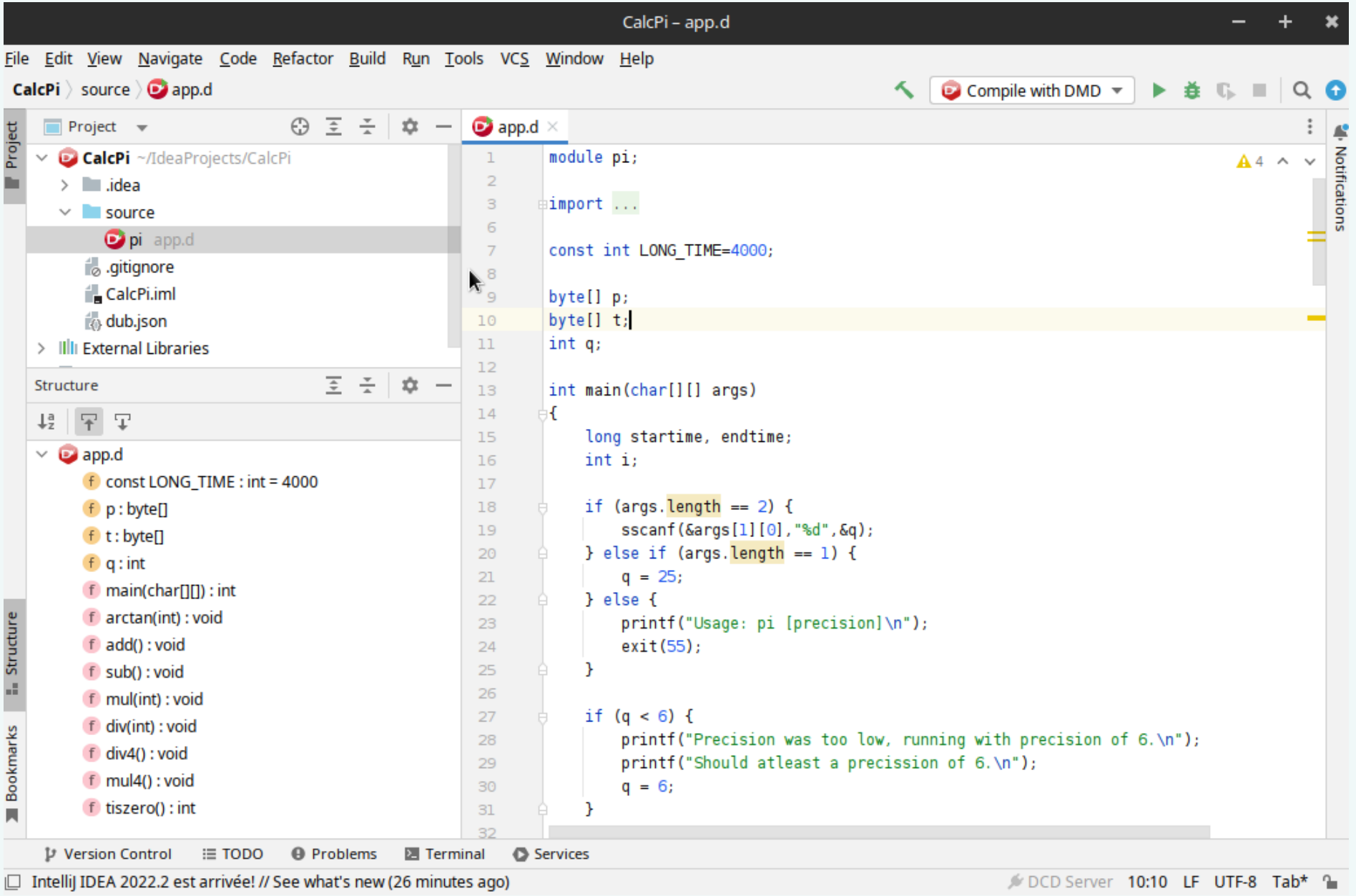
- Install the Dlang plugin and use dub
- If you what other programming languages other than D and Java one has to install other plugins
- Unfortunately, it appears that the user has to use only the reference dmd compiler maybe the developer will add the other D compilers someday
- The user can configure dub projects and everything works, like symbol generation, source control, code completion and the rest
- The compile and build menu items need a little configuration
- Debugging works also if the ”-gf” debug argument is used, does not work with the ”-g” debug argument

Synwrite

- Windows only and comes with D lexer plugin which needs to be installed
- If has syntax highlighting, code folding and multi file edits
- Need to configure custom tools to call an external cmd prompt, run the current compiled file and to compile the current d file
- Also can configure custom tools to do syntax and style checks with Dscanner
- Need to configure custom project tools to build dub projects, but you need to save the project in order to keep them in the menu (learned the hard way)
- There is no code completion, call tips or debugging available

Emacs

- I must admit that this started out being the last editor on my list
- It has always and still baffles me even with a cheat sheet
- In my defense, I did not start out using Unix or Linux or Mac
- Install d-mode plugin to add d support
- This is the only one you need to use D in the emacs editor, since by default it loads many other plugins.
- There is a git plugin, which I didn’t try
- There is a debugging plugin, which I tried out but gave up on after trying it for awhile
- The compile menu item in the D menu runs make to build the project


CalcPi – app.d
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
CalcPi source app.d
Compile with DMD
Project
CalcPi ~/IdeaProjects/CalcPi
.idea
source
pi app.d
.gitignore
CalcPi.iml
dub.json
External Libraries
Structure
app.d
const LONG_TIME : int = 4000
p : byte[]
t : byte[]
q : int
main(char[][]) : int
arctan(int) : void
add() : void
sub() : void
mul(int) : void
div(int) : void
div4() : void
mul4() : void
tiszero() : int
app.d
1 module pi;
3 import ...
7 const int LONG_TIME=4000;
9 byte[] p;
10 byte[] t;
11 int q;
13 int main(char[][] args)
14 {
15 long startime, endtime;
16 int i;
18 if (args.length == 2) {
19 sscanf(&args[1][0],"%d",&q);
20 } else if (args.length == 1) {
21 q = 25;
22 } else {
23 printf("Usage: pi [precision]\n");
24 exit(55);
25 }
27 if (q < 6) {
28 printf("Precision was too low, running with precision of 6.\n");
29 printf("Should atleast a precission of 6.\n");
30 q = 6;
31 }
Notifications
Bookmarks
Version Control
TODO
Problems
Terminal
Services
IntelliJ IDEA 2022.2 est arrivée! // See what's new (26 minutes ago)
DCD Server
10:10
LF
UTF-8
Tab*

Code Editors with Simple Configuration and or One Not D plugin

Gedit

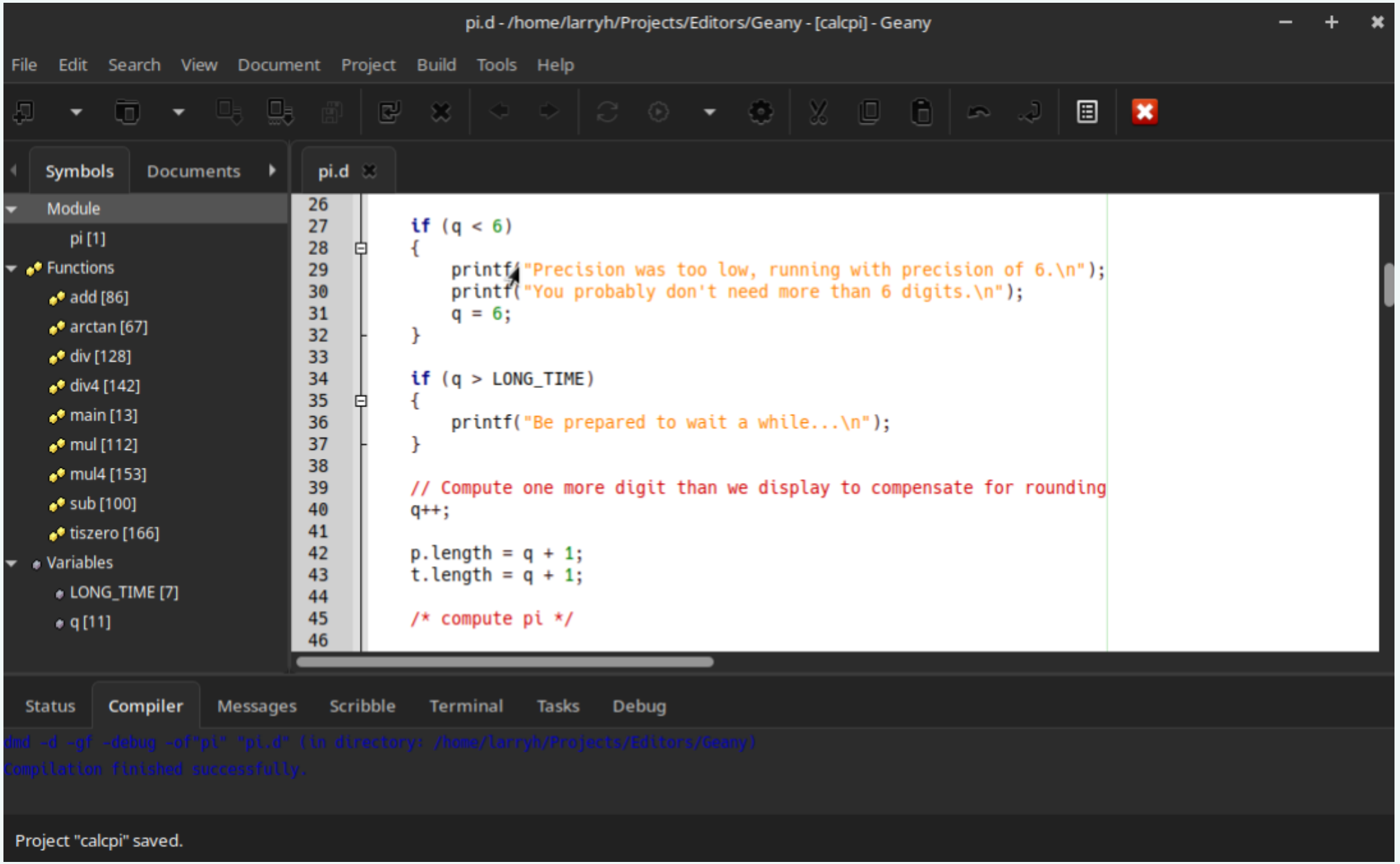
- Highlighting mode auto detected by extension, no code folding
- Menu item on tools menu to open a terminal in the current files folder
- Tools menu item to create external tools to add extension specific or global tools for all extensions
- Can create tools to run the compiled program, compile & run the current file with rdmd, debug & release compiles of the current file, and debug & release builds for the project using dub
- Can add plugins for building extension specific or global snippets plus many other non d specific plugins

Geany

- Comes configured for the D language and has almost everything a developer needs
- Even debugging work if a custom compile is configured with the ”-gf” compiler argument
- The default debug build commands use ”-g” compiler argument, which causes gdb to skip the breakpoints and run to the end of the program
- The default build command uses make, one can configure custom build commands using dub and custom compile commands using any of the three D compilers
- Need to install a plugin for source control like git

Notepad++

- On Linux the same editor is called Notepadpp for some reason
- Has syntax highlighting and code folding built in for D and many other programming languages
- Can run any executable from the run menu including an external and/or internal terminal
- Handles multi file editing and word completion, but no code completion
- There is no compile, build or debugging integrated into the editor
- Needs a plugin for source control
- This was the favorite editor for a lot of fellow students in at least one junior college, because of its versatility


pi.d - /home/larryh/Projects/Editors/Geany - [calcpi] - Geany
File Edit Search View Document Project Build Tools Help
Symbols Documents
pi.d
Module
pi [1]
Functions
add [86]
arctan [67]
div [128]
div4 [142]
main [13]
mul [112]
mul4 [153]
sub [100]
tiszero [166]
Variables
LONG_TIME [7]
q [11]
26
27 if (q < 6)
28 {
29 printf("Precision was too low, running with precision of 6.\n");
30 printf("You probably don't need more than 6 digits.\n");
31 q = 6;
32 }
33
34 if (q > LONG_TIME)
35 {
36 printf("Be prepared to wait a while...\n");
37 }
38
39 // Compute one more digit than we display to compensate for rounding
40 q++;
41
42 p.length = q + 1;
43 t.length = q + 1;
44
45 /* compute pi */
46
Status Compiler Messages Scribble Terminal Tasks Debug
dmd -d -gf -debug -of"pi" "pi.d" (in directory: /home/larryh/Projects/Editors/Geany)
Compilation finished successfully.
Project "calcpi" saved.

Code Editors with Simple Configuration and or One Not D plugin

Code::Blocks

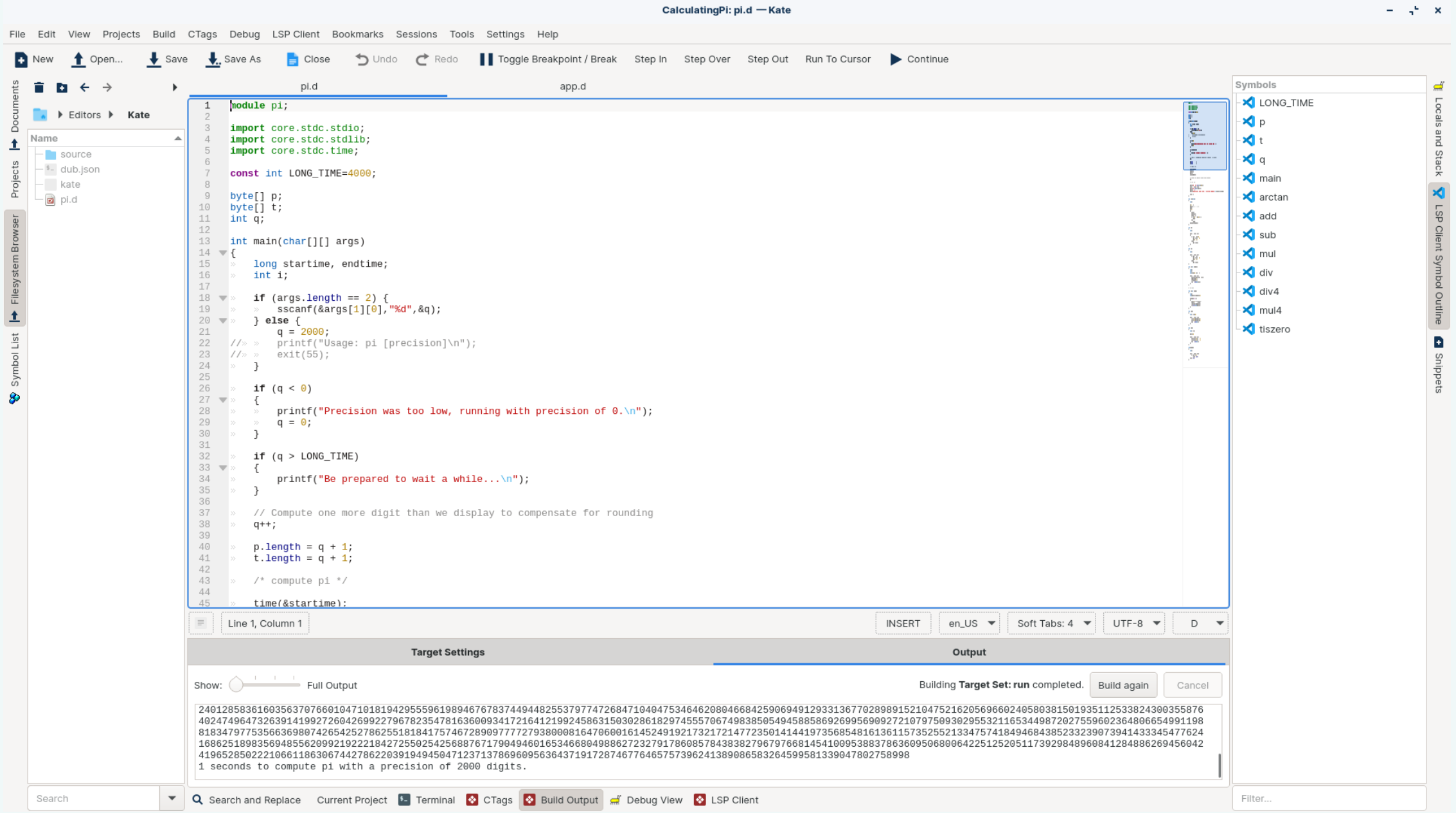
- This code editor has been around for a long time and recently has been updated and has installers for Linux and Windows
- It has built in configuration for D and many other programming languages
- However, the D configuration does not work since can not create a D project on Linux
- One can create D projects on Windows, but got link errors on building the project
- Can create custom tools to compile and run dub projects, compile D files with dmd and compile & run with rdmd

Kate

- Syntax highlighting and code folding with built in configuration for D
- Can create a Kate project file in the project folder with custom build cmds
- Internal terminal open in project folder
- Can run "dub init" from this terminal
- Can configure custom build commands
- Uses DLS instead of serve-d
- Symbol generation, code completion and git for source control works well, even with the outdated D Language Server
- Can create snippets for the D programming language
- Debugging even works with the correct debug arguments for gdb

Jedit

- Checking in the plugin manager could not find a d specific plugin
- There are many other plugins available, one for debugging (gdb plugin) and one for source control (git plugin) and one to run the console with compile and run commands, but not any build menu items
- No code completion (just word completion), no call tips and no symbol generation
- Just multiple file editing in tabs, syntax highlighting and code folding is built into editor


CalculatingPi: pi.d — Kate
File Edit View Projects Build CTags Debug LSP Client Bookmarks Sessions Tools Settings Help
New Open... Save Save As Close Undo Redo Toggle Breakpoint / Break Step In Step Over Step Out Run To Cursor Continue
Documents
Projects
Filesystem Browser
Symbol List
Editors Kate
Name
source
dub.json
kate
pi.d
pi.d
app.d
module pi;
import core.stdc.stdio;
import core.stdc.stdlib;
import core.stdc.time;
const int LONG_TIME=4000;
byte[] p;
byte[] t;
int q;
int main(char[][] args)
{
long startime, endtime;
int i;
if (args.length == 2) {
sscanf(&args[1][0],"%d",&q);
} else {
q = 2000;
//printf("Usage: pi [precision]\n");
//exit(55);
}
if (q < 0)
{
printf("Precision was too low, running with precision of 0.\n");
q = 0;
}
if (q > LONG_TIME)
{
printf("Be prepared to wait a while...\n");
}
// Compute one more digit than we display to compensate for rounding
q++;
p.length = q + 1;
t.length = q + 1;
/* compute pi */
time(&startime);
Symbols
LONG_TIME
p
t
q
main
arctan
add
sub
mul
div
div4
mul4
tiszero
Locals and Stack
LSP Client Symbol Outline
Snippets
Line 1, Column 1
INSERT
en_US
Soft Tabs: 4
UTF-8
D
Target Settings
Output
Show:
Full Output
Building Target Set: run completed.
Build again
Cancel
2401285836160356370766010471018194295559619894676783744944825537977472684710404753464620804668425906949129331367702898915210475216205696602405803815019351125338243003558764024749647326391419927260426992279678235478163600934172164121992458631503028618297455570674983850549458858692699569092721079750930295532116534498720275596023648066549911988183479775356636980742654252786255181841757467289097777279380008164706001614524919217321721477235014144197356854816136115735255213347574184946843852323290739414333454776241686251898356948556209921922218427255025425688767179049460165346680498862723279178608578438382796797668145410095388378636095068006422512520511739298489608412848862694560424196528502221066118630674427862203919494504712371378696095636437191728746776465757396241389086583264599581339047802758998
1 seconds to compute pi with a precision of 2000 digits.
Search
Search and Replace
Current Project
Terminal
CTags
Build Output
Debug View
LSP Client
Filter...

Code Editors That Require Multiple Plugins, Multiple Configuration Changes

Neovim

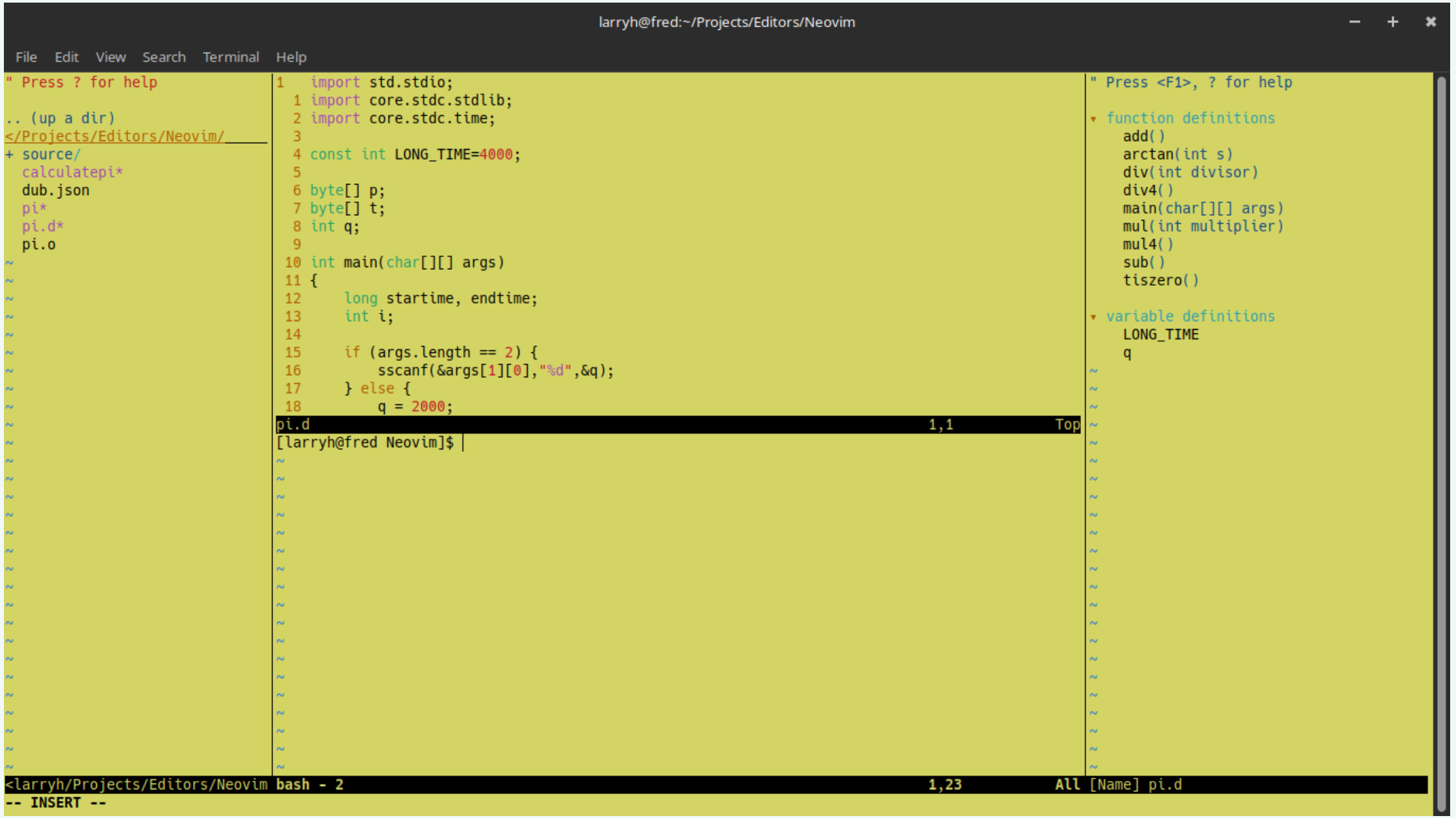
- All three of these are based on Vi, a editor that runs in a console
- They require multiple plugins, eight of them and require configuration by writing an init file in vi script
- Highlighting is auto detected, but there is no code folding (maybe there is a plugin for it, I couldn't find it)
- There is a plugin for an internal terminal and a tag bar for symbol generation
- No compile or build except in the internal terminal
- Code completion, call tips and goto definition works sometimes
- The debugger didn't work in neovim

Vim

- The plugins and vi script for Vim is similar but also work different than the did in Neovim
- Vim used the vundle plugin manager suggested on the "vim for d" wiki page while Neovim used the plug.vim standard vim plugin manager
- Vim used the dutyl.vim code completion plugin while neovim used ncm2-d.
- Since they used slightly different plugins the init file vi scripts differed
- Vim & Neovim used the same debugger, which was painful but worked in Vim.
- Both code completion plugins though different used DCD, Dscanner, etc

Gvim

- This version of vim is just a gui interface around vim in a console with a menu and tool bar to make life easier for the gui inclined
- It works in Windows as well as in Linux which the other do not
- They were all tested in Linux for this
- It used the same init script .vimrc and plugins as installed for vim
- However, nerd tree (directory & files) work better than in vim and the terminal in Gvim opened, but nothing could be enter after the prompt so it did not really work
- Did not test the debugger in gvim
- The menu & tool bar has a make command, but I did not try it out


larryh@fred:~/Projects/Editors/Neovim
File Edit View Search Terminal Help
" Press ? for help
.. (up a dir)
</Projects/Editors/Neovim/
+ source/
calculatepi*
dub.json
pi*
pi.d*
pi.o
1 import std.stdio;
1 import core.stdc.stdlib;
2 import core.stdc.time;
3
4 const int LONG_TIME=4000;
5
6 byte[] p;
7 byte[] t;
8 int q;
9
10 int main(char[][] args)
11 {
12 long startime, endtime;
13 int i;
14
15 if (args.length == 2) {
16 sscanf(&args[1][0],"%d",&q);
17 } else {
18 q = 2000;
pi.d 1,1 Top
[larryh@fred Neovim]$
" Press <F1>, ? for help
function definitions
add()
arctan(int s)
div(int divisor)
div4()
main(char[][] args)
mul(int multiplier)
mul4()
sub()
tiszero()
variable definitions
LONG_TIME
q
<larryh/Projects/Editors/Neovim bash - 2 1,23 All [Name] pi.d
-- INSERT --

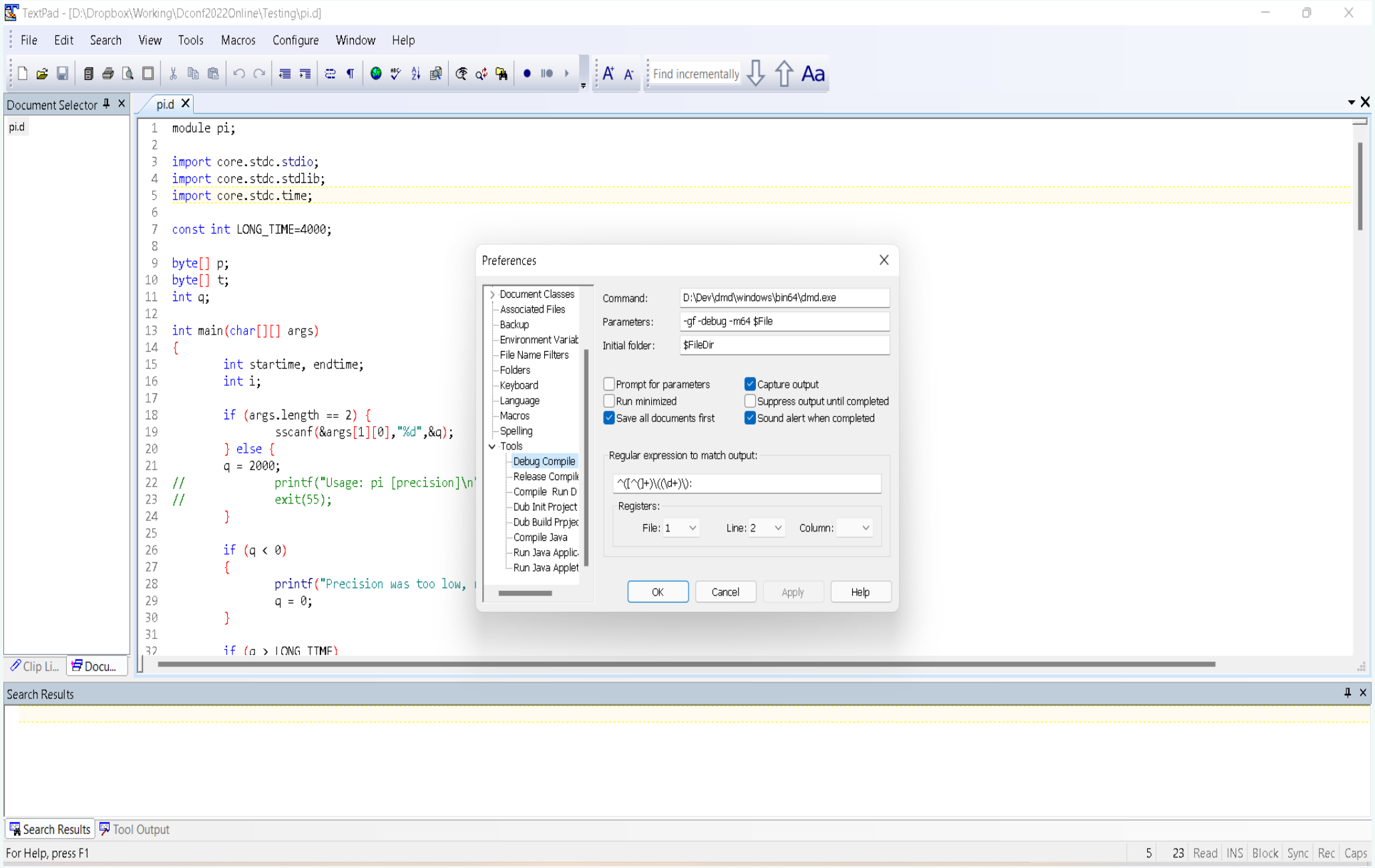
Editors That Have Multiple Things to do to Get Them Configured

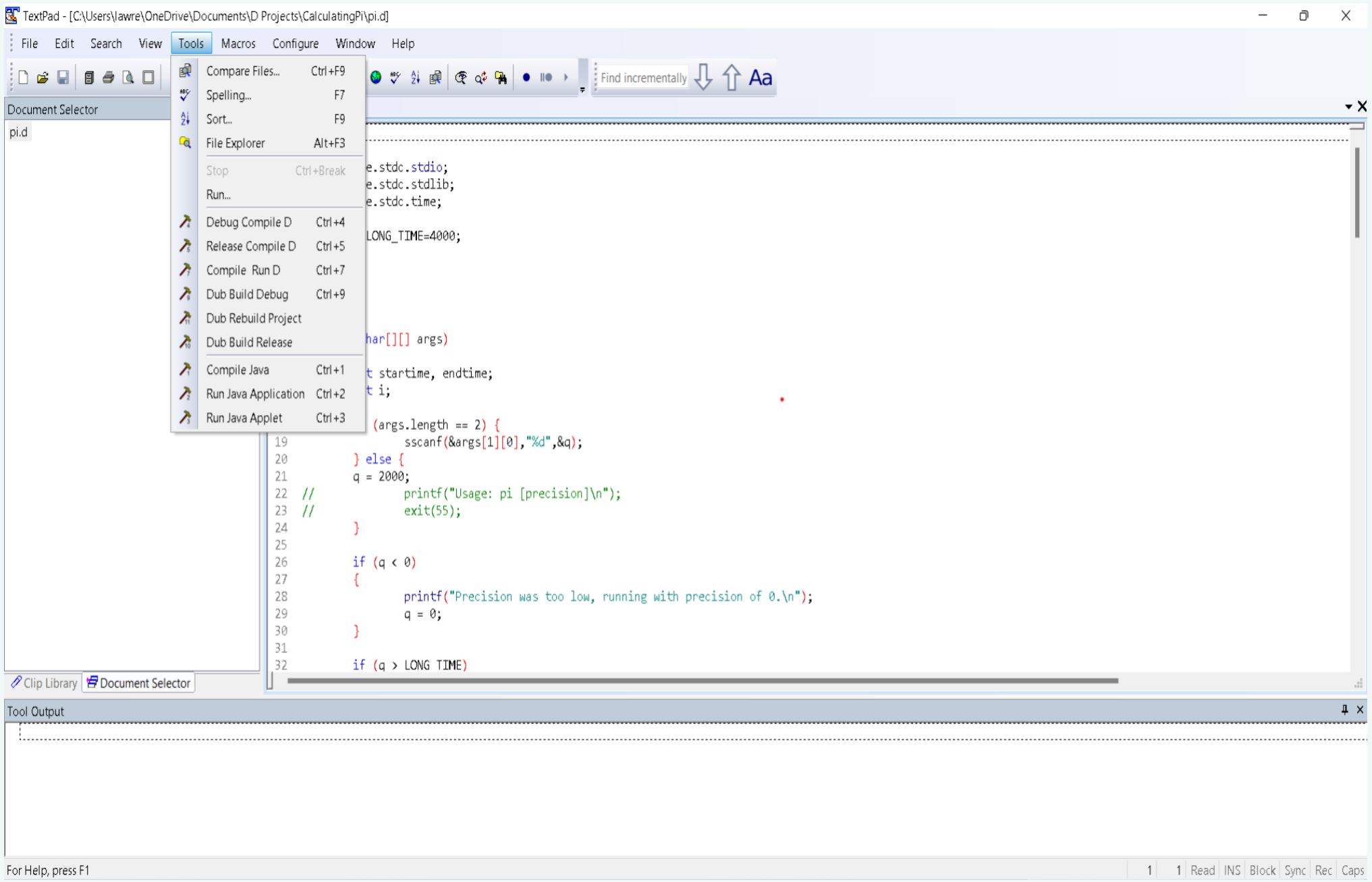
Textpad

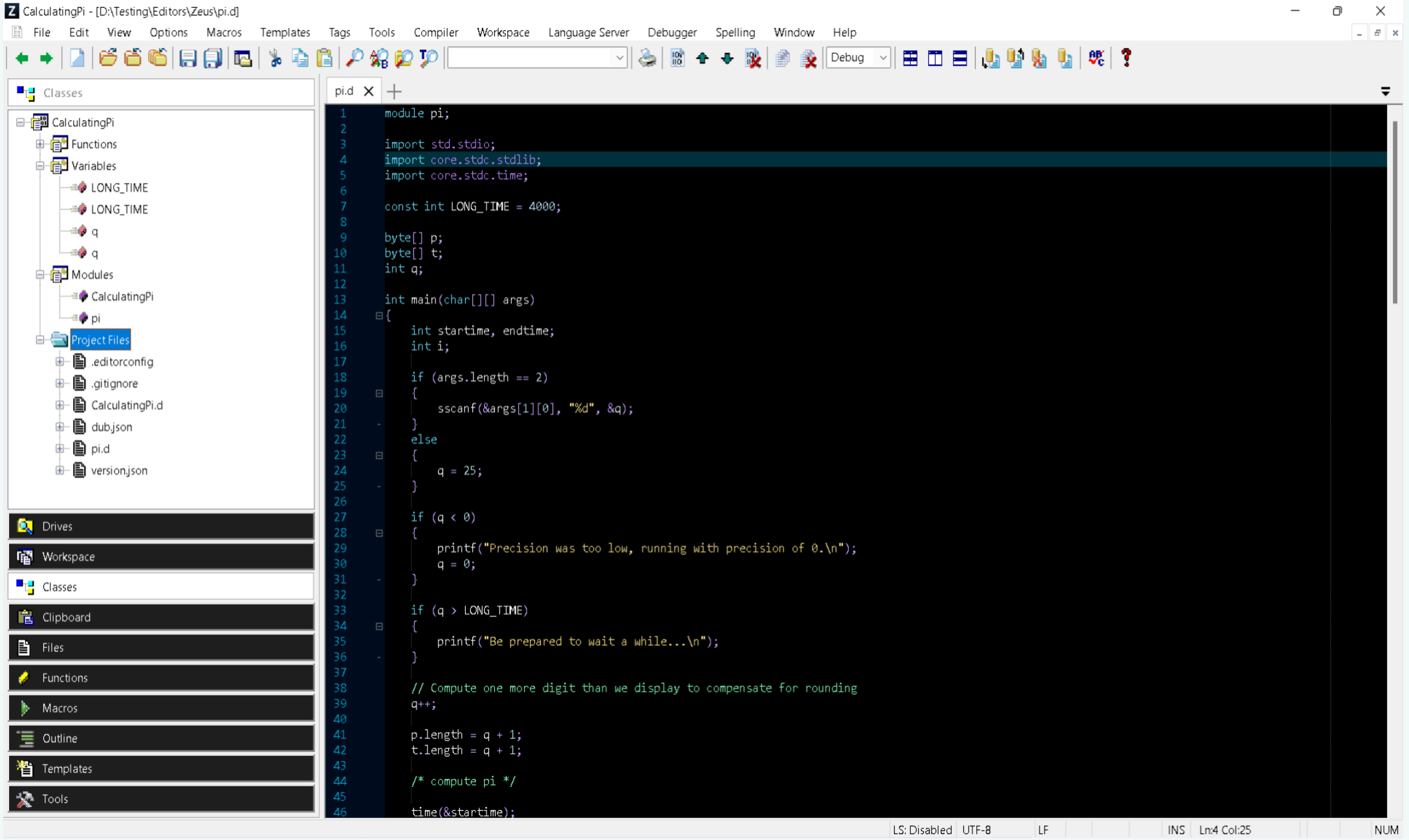
- This is shareware, but is fully function after trial period
- It does not come configured for the D language
- One can download the d.zip file from textpad addons web page and extract the d.syn file to the samples folder in the programs folder were it keep its config data
- After extracting the d syntax file open the "configure menu", select the "new document item" and fill out the dialog box, when prompted select the d.syn file
- Setup the compiler tools in "configure" menu, select "preferences", click on "tools" menu item
- It comes configured with the "compile Java" and "run Java file" already set up
- One can just follow the Java example or check out the help menu "help topic" for info on how to run program
- In my first programming class, which was for C, the instruction gave out instruction sheet to download Textpad and Borland command line compiler.....

Zeus IDE

- Zeus IDE for windows is propriety and will stop working after the trial period, which is 55 days according to the person who developed the software
- The paid version has built in partial support for D
- The rest has to be configured, which is a little complicated and has full instructions on there web site using D1
- It uses the same programs every one else uses namely DCD, Dscanner, Dfmt and can use dub instead of make
- In the past it used the D language server, but now uses python scripts to trigger the utilities action.
- One can even install the D2.chm help file into the quick help menu item in the help menu
- The developer updated some of the python scripts to python 3 from python 2 and disabled the DLS which solved some of the problems
- We did get debugging to work with ldc and the -gdwarf argument as suggested in the D forums


TextPad - [D:\Dropbox\Working\Dconf2022Online\Testing\pi.d]
File Edit Search View Tools Macros Configure Window Help
Find incrementally
Document Selector
pi.d
module pi;
import core.stdc.stdio;
import core.stdc.stdlib;
import core.stdc.time;
const int LONG_TIME=4000;
byte[] p;
byte[] t;
int q;
int main(char[][] args)
{
int startime, endtime;
int i;
if (args.length == 2) {
sscanf(&args[1][0],"%d",&q);
} else {
q = 2000;
// printf("Usage: pi [precision]\n"
// exit(55);
}
if (q < 0)
{
printf("Precision was too low,
q = 0;
}
Preferences
Document Classes
Associated Files
Backup
Environment Variab
File Name Filters
Folders
Keyboard
Language
Macros
Spelling
Tools
Debug Compile
Release Compil
Compile Run D
Dub Init Project
Dub Build Prpjec
Compile Java
Run Java Applic
Run Java Applet
Command: D:\Dev\dmd\windows\bin64\dmd.exe
Parameters: -gf -debug -m64 $File
Initial folder: $FileDir
Prompt for parameters
Run minimized
Save all documents first
Capture output
Suppress output until completed
Sound alert when completed
Regular expression to match output:
^([^(]+)\((\d+)\):
Registers:
File: 1
Line: 2
Column:
OK
Cancel
Apply
Help
Search Results
Search Results
Tool Output
For Help, press F1
5
23
Read
INS
Block
Sync
Rec
Caps


TextPad - [C:\Users\lawre\OneDrive\Documents\D Projects\CalculatingPi\pi.d]
File Edit Search View Tools Macros Configure Window Help
Compare Files... Ctrl+F9
Spelling... F7
Sort... F9
File Explorer Alt+F3
Stop Ctrl+Break
Run...
Debug Compile D Ctrl+4
Release Compile D Ctrl+5
Compile Run D Ctrl+7
Dub Build Debug Ctrl+9
Dub Rebuild Project
Dub Build Release
Compile Java Ctrl+1
Run Java Application Ctrl+2
Run Java Applet Ctrl+3
Find incrementally
Document Selector
pi.d
e.stdc.stdio;
e.stdc.stdlib;
e.stdc.time;
LONG_TIME=4000;
har[][] args)
t startime, endtime;
t i;
(args.length == 2) {
sscanf(&args[1][0],"%d",&q);
} else {
q = 2000;
// printf("Usage: pi [precision]\n");
// exit(55);
}
if (q < 0)
{
printf("Precision was too low, running with precision of 0.\n");
q = 0;
}
if (q > LONG TIME)
Clip Library
Document Selector
Tool Output
For Help, press F1
1 1 Read INS Block Sync Rec Caps


CalculatingPi - [D:\Testing\Editors\Zeus\pi.d]
File Edit View Options Macros Templates Tags Tools Compiler Workspace Language Server Debugger Spelling Window Help
Debug
Classes
CalculatingPi
Functions
Variables
LONG_TIME
LONG_TIME
q
q
Modules
CalculatingPi
pi
Project Files
.editorconfig
.gitignore
CalculatingPi.d
dub.json
pi.d
version.json
Drives
Workspace
Classes
Clipboard
Files
Functions
Macros
Outline
Templates
Tools
pi.d
module pi;
import std.stdio;
import core.stdc.stdlib;
import core.stdc.time;
const int LONG_TIME = 4000;
byte[] p;
byte[] t;
int q;
int main(char[][] args)
{
int startime, endtime;
int i;
if (args.length == 2)
{
sscanf(&args[1][0], "%d", &q);
}
else
{
q = 25;
}
if (q < 0)
{
printf("Precision was too low, running with precision of 0.\n");
q = 0;
}
if (q > LONG_TIME)
{
printf("Be prepared to wait a while...\n");
}
// Compute one more digit than we display to compensate for rounding
q++;
p.length = q + 1;
t.length = q + 1;
/* compute pi */
time(&startime);
LS: Disabled UTF-8 LF INS Ln:4 Col:25 NUM

Editor\Ide	Win	Nix	D	C, C++	HL	CF	RN	IT	ET	MF	SC	CP	BD	SG	CC	DB
Textpad	Yes	No	Config	Config	Download	No	Yes	No	Yes	Yes	No	Config	Config	No	No	No
Zeus	Yes	No	Config	Default	Yes	Yes	Yes	Yes	Yes	Yes	No Git	Yes	Config	No	Yes	NWVM
Notepad++	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Plugin	No	No	Partial	No	No
Synwrite	Yes	No	Plugin	Yes	Yes	Yes	Config	No	Yes	Yes	No	Yes	Yes	Partial	Partial	No
VisualD	Yes	No	Yes	MSVS	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Kate	No	Yes	Yes	Yes	Yes	Yes	Config	Yes	Yes	Yes	Yes	Config	Config	Yes	Yes	Yes
Kwrite	No	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	No	No
Xed	No	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No	No	No	No	No	No
Gedit	No	Yes	Yes	Yes	Yes	No	Config	Yes	Yes	Yes	No	Yes	Yes	No	No	No
Code	No	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No	No	No	No	No
Neovim	No	Yes	Plugins	Plugins	Plugins	No	No	Plugins	No	Yes	No	No	No	Plugins	No	Plugins
Vim	No	Yes	Plugins	Plugins	No	No	No	Plugins	No	Yes	No	No	No	Plugins	No	Plugins
VS Code	Yes	Yes	Plugin	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Config	Config	Yes	Yes	Config
Code::Blocks	Yes	Yes	Yes	Yes	Yes	Yes	Config	No	config	Yes	No	Config	Config	No	No	NWVM
Itellij IDEA	Yes	Yes	Plugin	Plugins	Yes	Yes	Yes	Yes	No	Yes	Yes	Config	Config	Yes	Yes	Yes
Geany	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Plugin	Yes	Yes	Yes	Yes	Yes
Gvim	Yes	Yes	Plugin	Plugins	Plugins	No	No	Plugins	No	Yes	No	No	Make	Yes	No	Plugins
Sublime Text	Yes	Yes	Dkit	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No	No	No
Emacs	Yes	Yes	Plugin	Yes	Yes	No	No	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Jedit	Yes	Yes	Yes	Yes	Yes	Yes	Plugin	No	No	Yes	Plugin	Plugin	No	No	No	Plugins
Textadapt	Yes	Yes	Plugin	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	No	No	Yes	No
Scite	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No	No	No
Dlangide	Yes	BYO	Yes	No	Yes	No	Yes	No	Yes	Yes	No	No	Yes	No	Yes	NWFM
Dexed	BYO	Yes	Yes	No	Yes	Yes	Yes	NWFM	No	Yes	Yes	Yes	NWFM	Yes	Yes	NWFM
PoseidonD	NVDW	Yes	Yes	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	Yes	Yes	NWFM
Dcode	Yes	No	Yes	No	Yes	Yes	Yes	No	No	No	No	Yes	No	No	No	No
Entice	Yes	No	Yes	No	Yes	Yes	Yes	No	No	Yes	No	Yes	No	No	No	No
D-IDE	Yes	No	Yes	No	Yes	Yes	Yes	No	No	Yes	No	Yes	Yes	No	No	No
Atom	No longer available for d programming since no default plugin for it															
Dhee	No longer able to use, needs pyqt4 which is pass its end of life															
Mono-d	This is still available, but you have to build you own monodeverlop version 5.8 to 5.10 usable only															
KEY																
HL	Syntax Highlighting available															
CF	Code Folding available															
RN	Run Executables by menu item															
IT	Internal Terminal opened by menu item															
ET	External Terminal opened by menu item															
MF	Multi File Projects and multi file editing															
SC	Source Control available by menu item															
CP	Compile Current File by menu item															
BD	Build Project with dub or make															
SG	Symbol Generation available															
CC	Code Completion available															
DB	Debugging available in editor															
NWFM	Not Working For Me															
NVDW	New Version Not Working															
BYO	Build Your Own															
MSVS	Microsoft Visual Studio															

Larry Hemsley

Dconf 2022 Online