



# Introduction to Programming Utilities

Lecture 2: Text Editors and Compilers



# Course Syllabus

1. **Linux** and **The Command Line**
2. **Text editor** and **Compiler** (*This Lecture*)
3. Basics of **Git** and **GitLab**
4. Integrated Development Environment (**IDE**)
5. **Advanced Git** for Group Projects



# This Lecture: **Text Editor** and **Compiler**

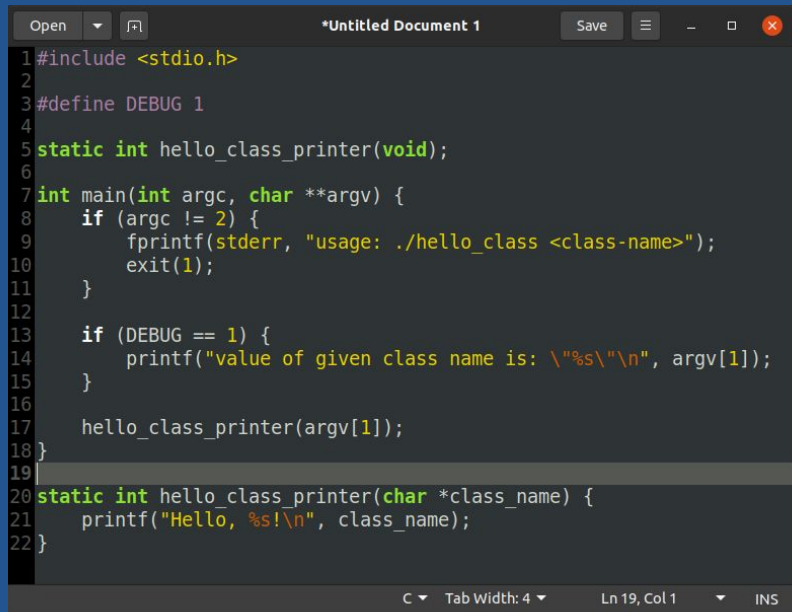
- What is a text editor?
- How do I choose which text editor to use?
- How to set up a text editor on your own computer
  - Demo using Atom
- What is a compiler and why do we need it?
- How do we install and use a compiler?
  - Demo using GHC/GHCI

# Text Editors





# What is a Text Editor?



```
1#include <stdio.h>
2
3#define DEBUG 1
4
5static int hello_class_printer(void);
6
7int main(int argc, char **argv) {
8    if (argc != 2) {
9        fprintf(stderr, "usage: ./hello_class <class-name>");
10        exit(1);
11    }
12
13    if (DEBUG == 1) {
14        printf("value of given class name is: \"%s\"\n", argv[1]);
15    }
16
17    hello_class_printer(argv[1]);
18}
19
20static int hello_class_printer(char *class_name) {
21    printf("Hello, %s!\n", class_name);
22}
```

- “Computer program that edits plain text documents”
- Source code is also just a plain text file!
  - You **can** technically use any text editor
- But there are some text editors that are better than others

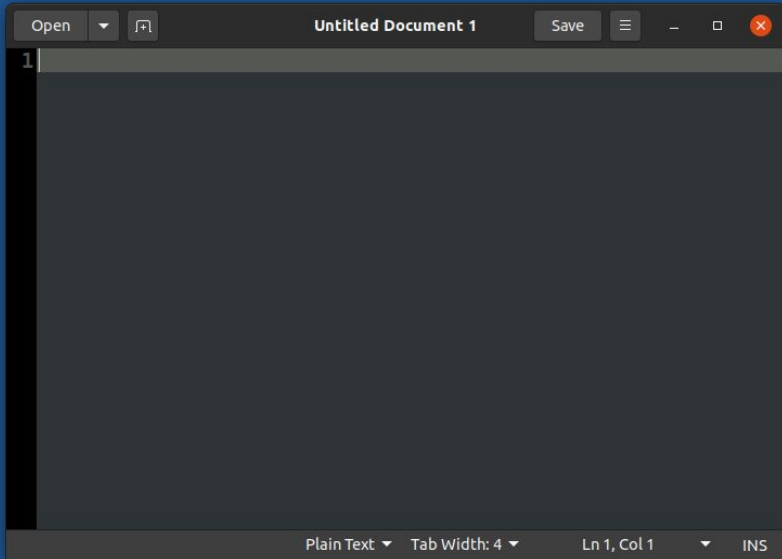


# Some preferable features for programming

- Syntax highlighting
- Line numbering
- Auto-close brackets and quotes
- Automatic indentation
- Automatically convert tabs to spaces (“soft tabs”)
- Multi-cursor
- Vertical rulers for maximum line length
- Customisable user interface
- Keyboard shortcuts
- Some refactoring features
  - Search and replace
  - Renaming identifiers of variables, functions, etc.
  - Linter support
  - Comment multiple lines w/ shortcut
- Version control (i.e. Git) integration
- Plugin support

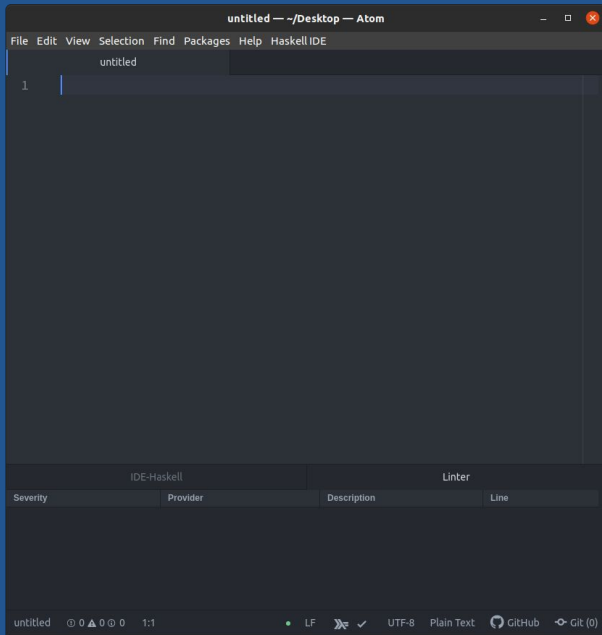


# gedit



- Default text editor for Ubuntu
- Not that sophisticated
- Some basic features for programming
  - Syntax highlighting
  - Line numbering
  - Vertical ruler
  - Soft tabs
- Not great for any serious programming

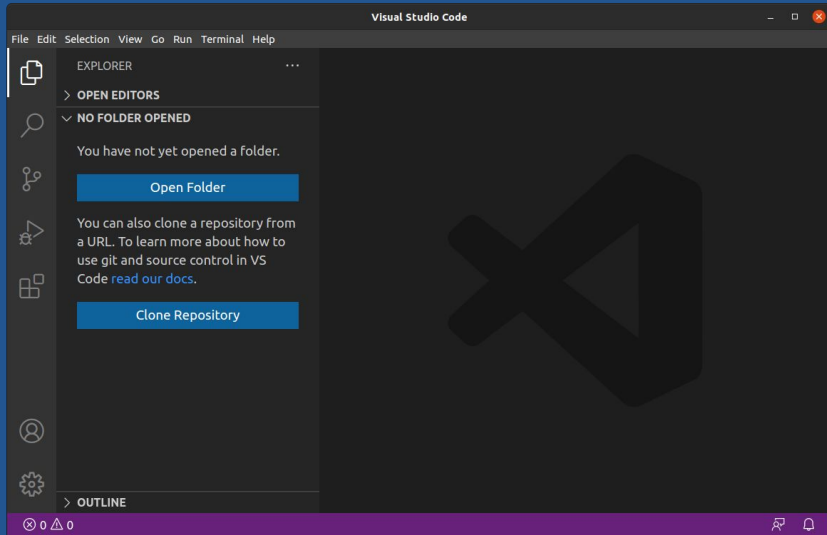
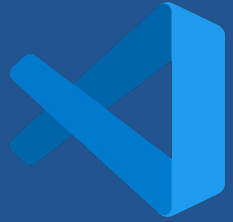
# Atom



- Developed by GitHub
  - “Hackable Text Editor for the 21st century”
- Has essentially all features preferable for programming
- Many plugins you can install extend its feature set
- Also highly customisable
- Installed by default on all DoC machines
  - Reason why I recommend Atom for Haskell
- Configuring it for Haskell can get complicated
  - We will go through it in a demo later
- Relatively slow and resource-intensive



# Visual Studio Code (VSCode)



- Developed by Microsoft
  - Free, open-source source code editor
  - Stripped-down version of Visual Studio
- Has all features for programming
- Extensive list of plugins
- One of the best source code editors, especially for web development
- Also doesn't support Haskell by default

# Vim



```
File Edit View Search Terminal Help
82
83 " Colorscheme
84 set t_Co=256
85 colorscheme molokai
86
87 set colorcolumn=80,100
88
89 highlight ColorColumn ctermBg=238 guibg=#232728
90
91 " Tab sanity
92 set expandtab
93 set tabstop=4
94 set shiftwidth=4
95
96 " Show hidden characters, tabs, trailing whitespace
97 set list
98 set listchars=tab:~\,trail:~.nbsp:~
99
100 " Different tab/pace steps
101 autocmd FileType yaml setlocal tabstop=2 shiftwidth=2 softtabstop=2 expandtab
102 autocmd FileType html setlocal tabstop=2 shiftwidth=2 softtabstop=2 expandtab
103 autocmd FileType css setlocal tabstop=2 shiftwidth=2 softtabstop=2 expandtab
104 autocmd FileType scss setlocal tabstop=2 shiftwidth=2 softtabstop=2 expandtab
105 autocmd FileType json setlocal tabstop=2 shiftwidth=2 softtabstop=2 expandtab
106 autocmd FileType javascript setlocal tabstop=2 shiftwidth=2 softtabstop=2 expandtab
107 autocmd FileType make setlocal noexpandtab
108
109 "
110 " Plugin Configuration
111 "
112 "
113 " ALE Configuration
114 let g:ale_fixers = [
115 \   '[remove_trailing_lines', 'trim_whitespace'],
116 \   'css': ['prettier'],
117 \   'javascript': ['prettier', 'eslint'],
118 \   'json': ['prettier'],
119 \   'less': ['prettier'],
120 \   'python': ['black', 'isort'],
121 \   'scss': ['prettier'],
122 \   'yaml': ['prettier'],
123 \ ]
124 let g:ale_linters = [
125 \   'css': ['prettier'],
126 \   'javascript': ['prettier', 'eslint'],
127 \   'json': ['prettier'],
128 \   'less': ['prettier'],
129 \   'python': ['flake8'],
130 \   'scss': ['prettier'],
131 \   'yaml': ['prettier'],
132 \ ]
[Default] 0:vim 132.1 98%
```

- Command-line text editor
- Very steep learning curve at first
- But also very customizable
  - Increase workflow speed significantly
- Not approachable for beginners
- Not useful for Haskell



```

file Edit Options Buffers Tools c Help
struct module "Module"
int ter_device;
int ter_subdevice;
char id[64];
char name[60];
unsigned int flags;
int running; /* running instances */
unsigned long ticks; /* running ticks */
void *private_data;
void (*private_free) (struct and_timer *timer);
struct and_timer hardware bus;
spinlock_t lock;
struct list_head device_list;
struct list_head open_list_head;
struct list_head active_list_head;
struct list_head ack_list_head;
struct list_head ack_list_head; /* slow ack list head */
struct tasklet_struct task_queue;
};

struct and_timer_instance {
struct and_timer *timer;
char *owner;
unsigned int flags;
void *private_data;
void (*private_free) (struct and_timer_instance *ti);
void (*callback) (struct and_timer_instance *timer);
unsigned long ticks, unsigned long resolution;
void (*callback) (struct and_timer_instance * timer,
int event,
struct timespec * tstamp,
unsigned long resolution);
void (*disconnect) (struct and_timer_instance *timer);
void *callback_data;
unsigned long ticks; /* auto-load ticks when expired */
unsigned long eticks; /* current ticks */
unsigned long piticks /* accumulates ticks by callback */
unsigned long resolution; /* current resolution for tasklet */
unsigned long lost; /* lost ticks */
int slave_client;
unsigned int slave_id;
};


2013-11-11 11:55:01.555 of 5.66 (101.56) (C/I View #log Abbrev)

2017-09-19 09:02:19PM use EDT
src/56c1/linenr:4,9,9,3-common/include/sound $

[00-11-11-11 "csheli" All of 112 (4.54) [Esheli #0 wrap]
[00-11-11-11 "csheli" All of 112 (4.54) [Esheli #0 wrap]

```

- Command-line text editor
- *Very* steep learning curve at first
- But also *very* customizable
  - Increase workflow speed significantly
- Not approachable for beginners
- Not useful for Haskell



# So which text editor should I choose for Haskell (at Imperial)?



- Installed by default on all DoC machines
- What the department endorses for Haskell
- Easy (enough) to use
- But very hard to set up without guidance
  - Not enough documentation on Atom for Haskell

# Reference sheet - Installing Atom w/ Haskell

- 1) Install **Atom**: <https://atom.io/> - download .deb package
- 2) Install **Haskell Platform** (for Haskell support) - <https://www.haskell.org/platform/>
- 3) Install **Stack** (for installing Haskell binary dependencies of Atom plugins) by running the following commands on terminal:

```
wget -qO- https://get.haskellstack.org/ | sh
```

```
echo "export PATH=$HOME/.local/bin:$PATH" >> ~/.bashrc
```

(refer [https://docs.haskellstack.org/en/stable/install\\_and\\_upgrade/](https://docs.haskellstack.org/en/stable/install_and_upgrade/) for details)

- 4) Install **binary dependencies** by running the following commands on terminal

```
stack install stylish-haskell
```

```
stack --resolver lts-9 install ghc-mod
```

(refer <https://atom-haskell.github.io/installation/installing-binary-dependencies/> for details)

Then get the executable's path (e.g. "/home/<your-username>/.local/bin") and save it

- 5) Get Atom **packages** (or plugins)

Find atom-haskell package OR run `apm install atom-haskell`

- 6) Go to haskell-ghc-mod settings and **add the executable's path** to "additional path directories"

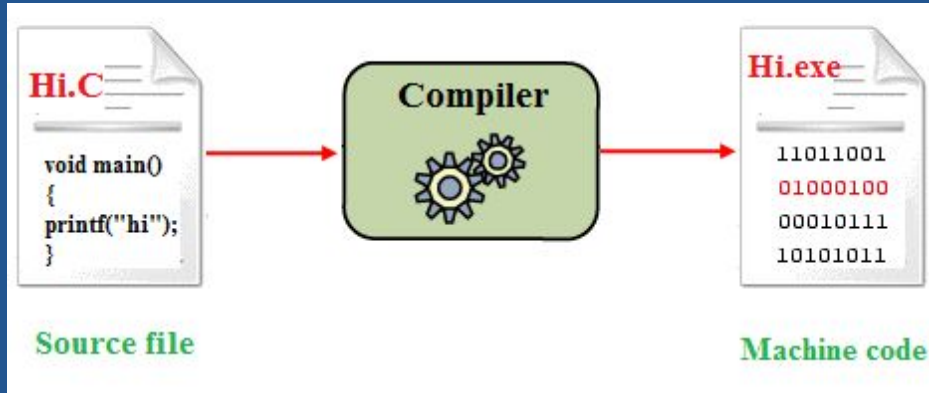
# Demo 1: Text Editors



# Compilers



# How source code is converted into a runnable program



- **Compiler** : converts source code into an executable program
- Source code is just a text file
- Computers cannot understand text
  - “Computers don’t speak English”
  - They only read 0s and 1s (Machine Code)

```
print("Hello, world!")
```

=>

```
00010010011101011011000.....
```



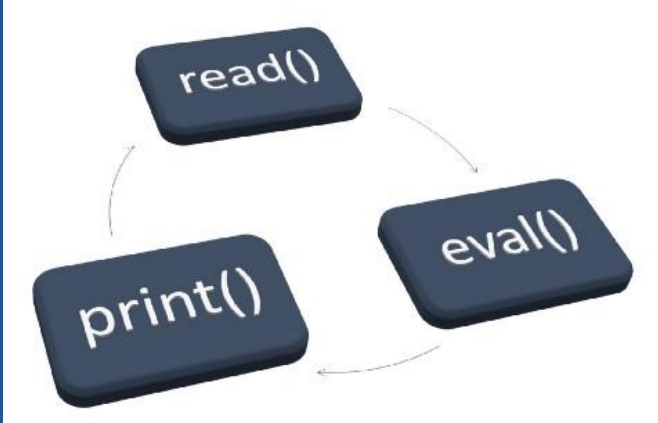


# GHC (Glasgow Haskell Compiler)



- Compiler for Haskell source code (.hs files)
- Gets the “main” function in a file
- Does the action of whatever the main function does
- Realistically never used for this course
  - You never really create an executable program for Tutorial exercises and LEXIS Tests
- Instead create program “modules” that you load into an interactive REPL
  - Module: a collection of data and functions wrapped up in a single source code file/package
  - *REPL???*

# The REPL (Read-Evaluate-Print Loop)



- Repeatedly reads code, evaluates it, and prints the result
  - Read : get user input (code statement)
    - e.g. calling a function
  - Evaluate : evaluates the given code and run it
  - Print : print the result of what was evaluated



# GHCI (GHC interactive)

```
hk619@hk619-ThinkPad-X1C7: ~/Docume...
hk619@hk619-ThinkPad-X1C7:~/Documents/lecture-2/calculator-demo$ ghci
GHCi, version 8.6.5: http://www.haskell.org/ghc/
/? for help
Prelude> :l
Calculator.hs Tests.hs      IC
Prelude> :l Calculator.hs
[1 of 1] Compiling Calculator      ( Calculator
r.hs, interpreted )
Ok, one module loaded.
*Calculator> add 1 2
3
*Calculator> 
```

- **ghci**
- Haskell's REPL Interactive environment
- Load modules you created into GHCI
- “Prelude” module is loaded by default
  - Contains all basic functions you need
- Call functions
- See the printed result



## Some useful commands for GHCi

- **:l** *<module/file>*
  - load a file/module into GHCi to use
- **:r**
  - Reload all modules loaded on the current environment
- **:browse** *<module>*
  - List all functions with its types in the module
  - Very useful for LEXIS tests
- **:q**
  - Quit the current GHCi session

## Demo 2: Compilers





# Department of Computing Society

Imperial College London

[docsoc.co.uk](http://docsoc.co.uk) | [fb.com/icdocsoc](https://fb.com/icdocsoc) | [twitter.com/icdocsoc](https://twitter.com/icdocsoc)