Vim, serious?

~~How to exit the Vim editor?~~

Text Code Editor

- Shawn Wang 王榮祥
- 成大電通所

- EasyStack Distro (openstack on k8s)
- Canonical OEM / infra
- 廣達 NB BIOS enginneer
- KaLUG Kaohsiung linux user group
- slide / labs

integrated development environment

https://en.wikipedia.org/wiki/Integrated_development_environment

- Editor
- Code completion, search - LSP
- Debug Adapter Protocol - DAP
- Syntax highlighting - tree-sitter
- Version control - git
- lang package manager - npm / cargo / go
- ~~Visual programming (*)~~

- CI/CD
  - testing
    - unit test
    - integration test
  - release
    - package
    - container / Kubernetes
- dev flow
  - github flow
  - pair programming

code editor

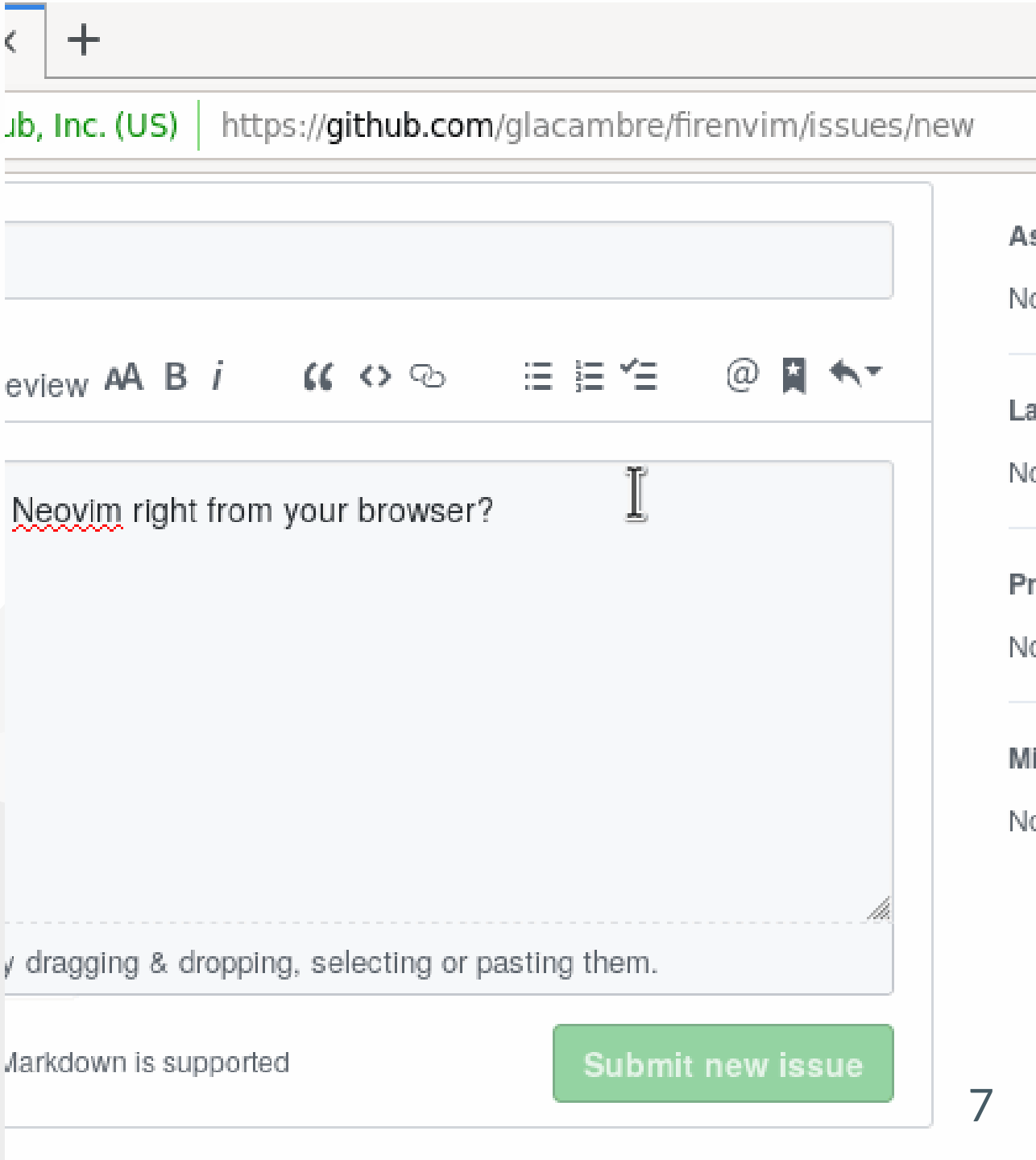
- text editor (console)
  - nano / vi / emacs(*) / helix
- gui editor
  - vscode / emacs
- web
  - eclipse theia - ex: killercoda
  - github.dev
  - vscode.dev

why text editor

- console
- lightweight
- remote edit (*)
- not just for coding
  - good for config
  - UNIX - everything is a file

vi family

- a screen-oriented text editor
- Vim Cheat Sheet
- vim (Vi iMproved) / gvim - c
  - vim7 2006
  - vim8 2016 - after neovim
  - vim9 - vim9script
- Others: Neovim - c + lua / Kakoune / helix - rist



vi modes:

- Normal / Command (:)
  - Normal mode - h (left), j (down), k (up), l (right)
- Insert (a,i) / Replace mode (r)
  - esc: exit insert mode
- visual mode (v, V, ctrl+v) (vim only)

Don't forget exit: esc + :q

range / num / sort

- :set nu
- :1,$s
- :1,10s
- :sort -
- :sort u -

Searching And Replacing

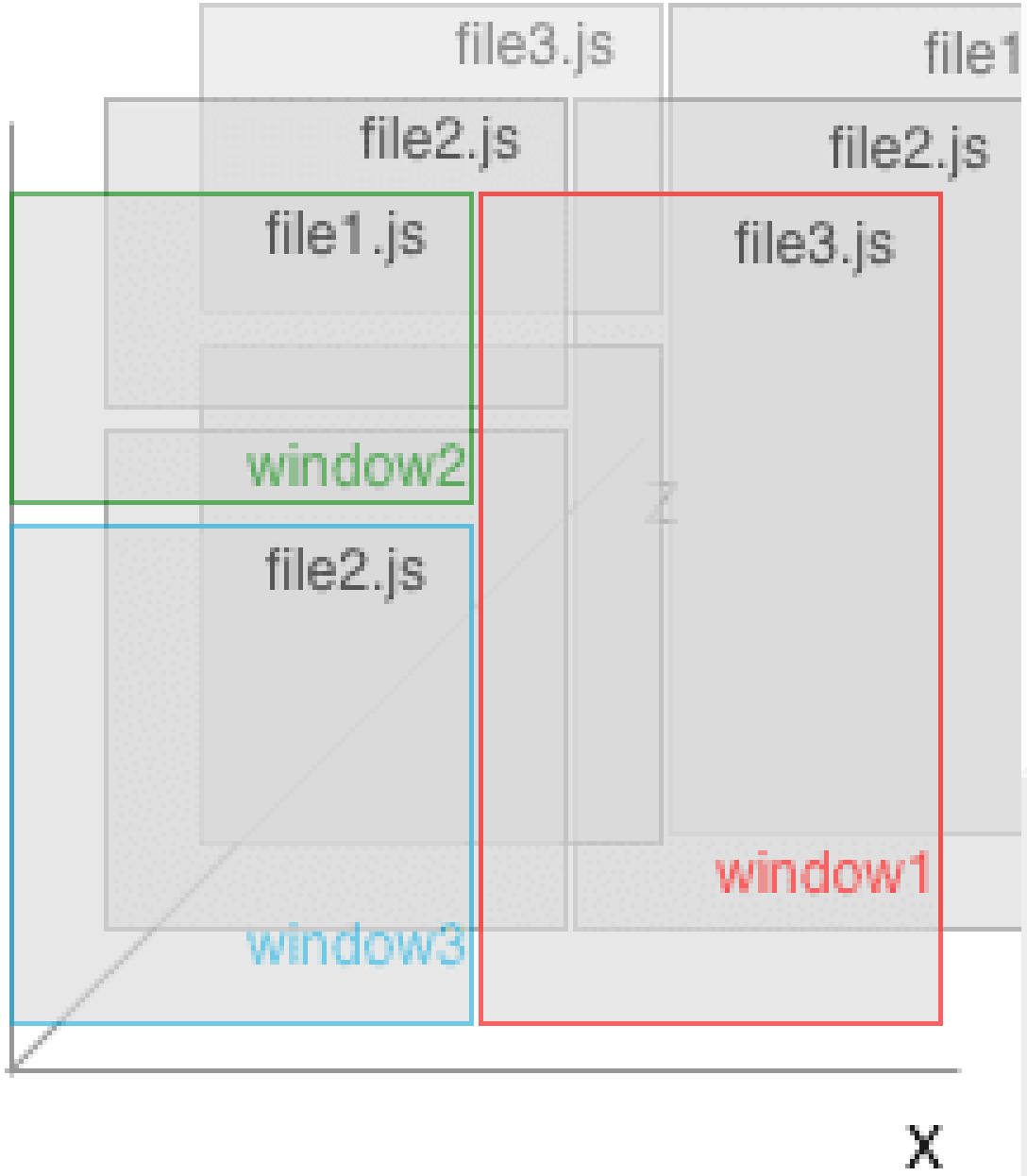
- /text search for text in the document, going forward.
- n move the cursor to the next instance of the text from the last search. This will wrap to the beginning of the document.
- N move the cursor to the previous instance of the text from the last search.
- ?text search for text in the document, going backwards.
- :%s/text/replacement text/g search through the entire document for text and replace it with replacement text.
- :%s/text/replacement text/gc search through the entire document and confirm before replacing text.

Copying And Pasting

- v highlight one character at a time.
- V highlight one line at a time.
- ctrl + v highlight by columns.
- p paste text after the current line.
- P paste text on the current line.
- y yank text into the copy buffer.

Saving And Quitting

- :w - write
- :wq - write and quit
- :w! - force write
- :o file - open file
- :q! - force quit



windows & tabs

- :tabnew file.txt # open file.txt in a new tab / :tabclose # Close current tab
- :tabnext / :tabprevious
- :tablast / :tabfirst
- split windows
  - ctrl + w + v
  - ctrl + w + s
- vimdiff a b
- :buffers

key binding

- ctrl + b back page
- ctrl + f forward page
- ctrl + v visual block
- ctrl + w split windows

vim plugins

- vim-plug
- Vundle
- https://vimawesome.com/

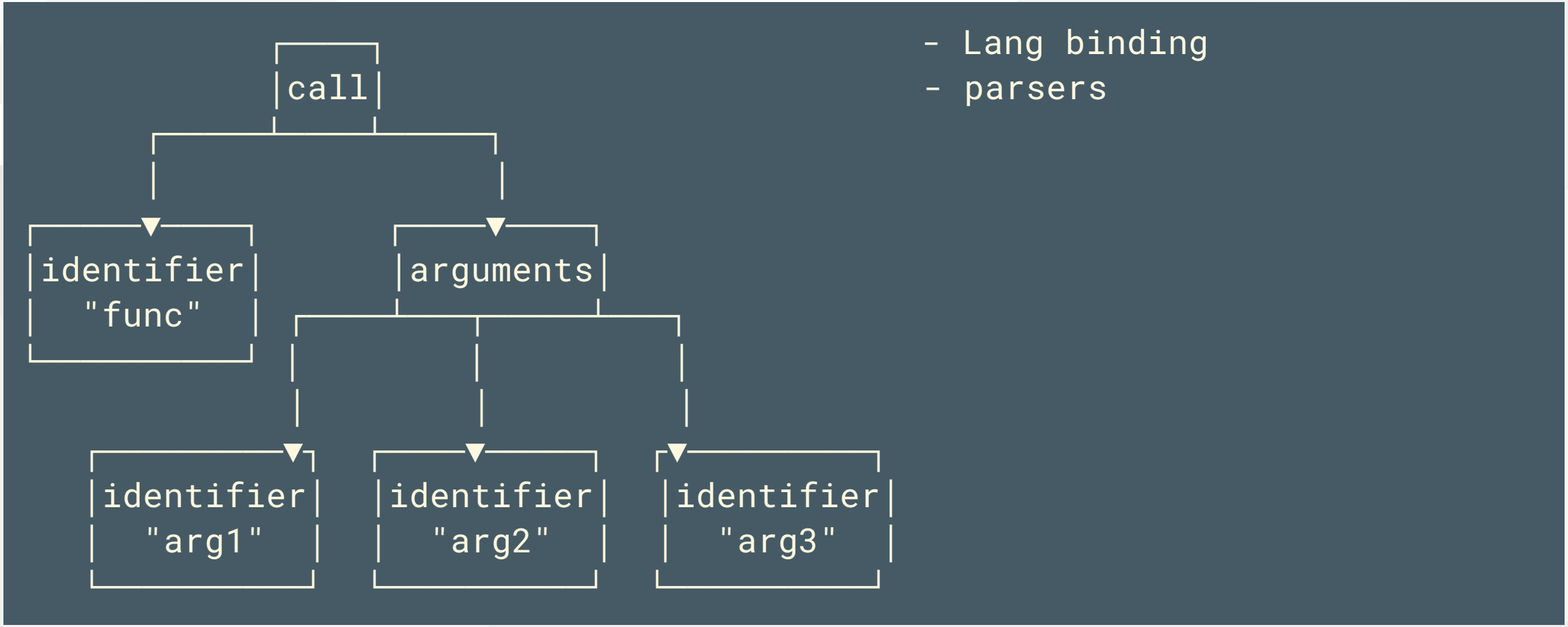
emacs - "Editor MACroS"

- emacs (vi mode- spaceemacs)
- editor war (vi/emacs)
  - simple vs complex
  - mode vs key binding
- LISP

helix

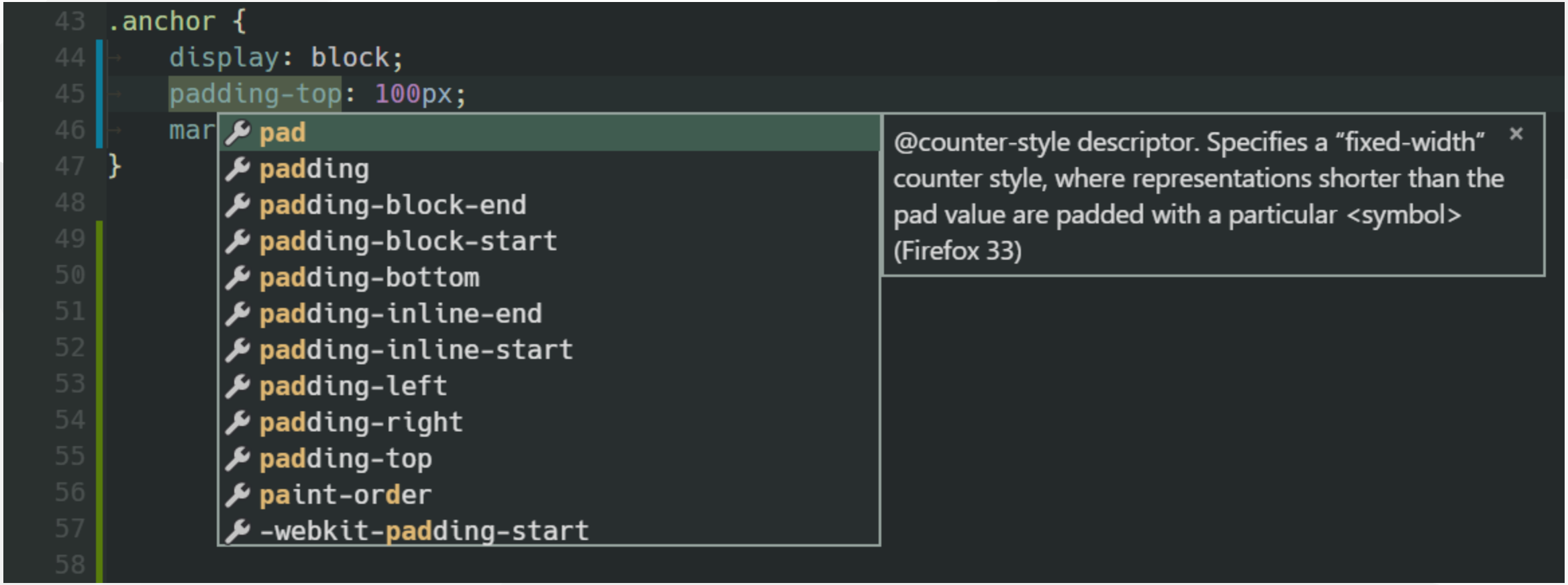
- Multiple selections
- Tree-sitter integration
- Powerful code manipulation
  - Navigate and select functions, classes, comments, etc and select syntax tree nodes instead of plain text.
- Language server support
- Modern builtin features
  - config-less
- Debug Adapter Protocol (WIP)

Tree-sitter



- Lang binding
- parsers

LSP


43 .anchor {
44 display: block;
45 padding-top: 100px;
46 mar
47 }
48
49
50
51
52
53
54
55
56
57
58
pad
padding
padding-block-end
padding-block-start
padding-bottom
padding-inline-end
padding-inline-start
padding-left
padding-right
padding-top
paint-order
-webkit-padding-start
@counter-style descriptor. Specifies a “fixed-width” counter style, where representations shorter than the pad value are padded with a particular <symbol> (Firefox 33)

select and action

- vi: verb (action) + none (target)
- helix: none (target) + verb (action)
  - delete vi (dw) / helix (wd)
- multi select
  - X
  - c - multi line
  - s -

Tree-sitter Textobject Based Navigation

- move to next function use]f
- move to previous class use [c

support status

- hx --heleath (c)

```
hx --health c
Configured language server: clangd
Binary for language server: Not found in $PATH
Configured debug adapter: lldb-vscode
Binary for debug adapter: Not found in $PATH
Highlight queries: ✓
Textobject queries: ✓
Indent queries: ✓
```

```
$ hx --health
```

space mode (helix)

- space + k
- space + s

ctags

- generates an index (or tag) file of language objects found in source files
- Vim 8 中 C/C++ 符号索引：GTags 篇
- universal-ctags

C/C++ LSP server

- clangd: part of the LLVM project.
- ccls: more features

https://emacs-china.org/t/topic/6428

GCC & GDB 簡介

聯庭智能 技術長

白勝文

白勝文

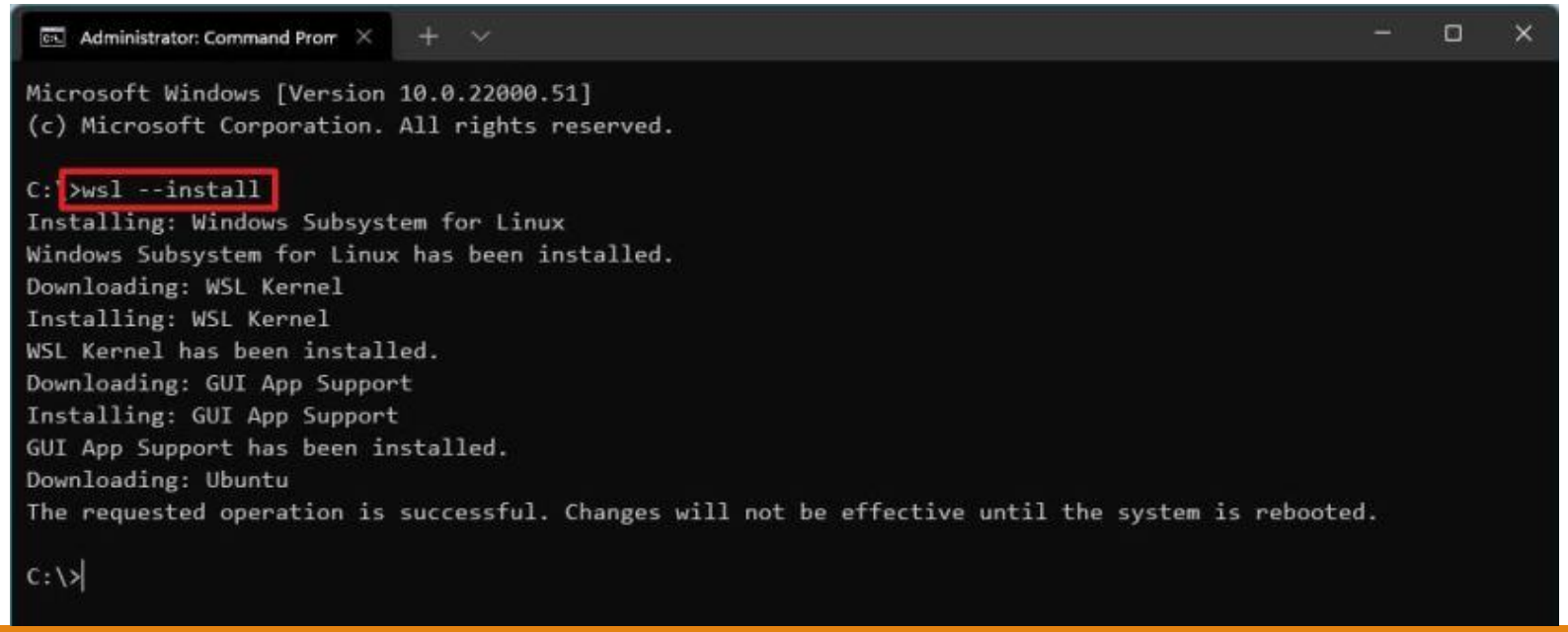
- ✓中山大學 資訊工程博士
- ✓Corel 數位科技 資深經理
- ✓鴻海精密 資深副理
- ✓慶奇科技 技術長
- ✓華譽國際電子 技術總監
- ✓聯庭智能 技術長

GCC

- 原名為 GNU C 語言編譯器 (GNU C Compiler)
- 最新定義為 GNU 編譯器套裝 (GNU Compiler Collection)
  - GNU 計劃製作的一種優化編譯器，支持各種編程語言、操作系統、計算機系統結構。該編譯器是以 GPL 及 LGPL 許可證所發行的自由軟體，也是 GNU 計劃的關鍵部分，還是 GNU 工具鏈的主要組成部份之一。GCC (特別是其中的 C 語言編譯器)也常被認為是跨平台編譯器的事實標準。
  - 支援 C、C++、Objective-C、Objective-C++、Fortran、Ada、D 以及 Go。

Windows 上使用 GCC

1. 使用支援 POSIX 的模擬環境
   - Cygwin, MinGW, MinGW64, MSYS2
2. 使用 WSL（Windows 子系統 Linux 版）



編譯與連結 C 程式

```c
// main1.c
#include <stdio.h>

int main() {
    printf("Hello world from main1.c!\n");
    return 0;
}
```

編譯指令

```
$ gcc main1.c ← 預設輸出執行檔 a.out
$ gcc main1.c -o main1 ← 指定執行檔名為 main1
```

編譯與連結 C++ 程式

```cpp
// main1.cpp
#include <iostream>
using namespace std;

int main() {
 cout << "Hello world from main1.cpp!" << endl;
 return 0;
}
```

編譯指令

```
$ g++ main1.cpp -o main1cpp
```

也可以使用 gcc，但要特別指定 c++ 版本的程式庫，建議直接用 g++ 編譯。

當編譯出現問題時

1. 程式出現語法**錯誤**時，編譯時直接顯示錯誤訊息
2. 程式出現語法**警告**時，預設不顯示警告訊息
   - 可以加上 -Wall 參數顯示所有編譯警告

※編譯器不一定能偵測到所有問題

```c
// main2-1.c
#include <stdio.h>

int main() {
  int sum;
  sum += 1;
  printf("sum: %d\n", sum);
  return 0;
}
```

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc main2-1.c -o main2-1
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc main2-1.c -o main2-1 -Wall
main2-1.c: In function ‘main’:
main2-1.c:5:7: warning: ‘sum’ is used uninitialized [-Wuninitialized]
    5 |   sum += 1;
      |   ~~~~^~~~
```

```c
// main2-2.c
#include <stdio.h>

int main() {
  int sum;

  for(int i = 0; i < 10; i++) {
    sum += i;
  }
  printf("sum: %d\n", sum);
  return 0;
}
```

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc main2-2.c -o main2-2 -Wall
sevenbai@LAPTOP-G0HPD2AP:~/pd2$
```

定義巨集（Macro）變數

1. 參數 -D[巨集變數]
   - ✓ 開發期間需要輸出詳細訊息，但正式版的程式又不需要
   - ✓ 根據不同的標的，設定不同的程式碼內容

```c
// main3.c
#include <stdio.h>

int main() {
  int sum;

  for(int i = 0; i < 10; i++) {
#ifdef DEBUG
    printf("i=%d, sum=%d\n",i, sum);
#endif
    sum += i;
  }
  printf("sum: %d\n", sum);
  return 0;
}
```

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc main3.c -o main3 -DDEBUG
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ ./main3
i=0, sum=-469098400
i=1, sum=-469098400
i=2, sum=-469098399
i=3, sum=-469098397
i=4, sum=-469098394
i=5, sum=-469098390
i=6, sum=-469098385
i=7, sum=-469098379
i=8, sum=-469098372
i=9, sum=-469098364
sum: -469098355
```

```c
// main4.c
#include <stdio.h>

int main() {

#ifdef LEO
    const char name[] = "Leo";
#elif defined(JOHN)
    const char name[] = "John";
#else
    const char name[] = "Nobody";
#endif

  printf("Hi %s, how are you?\n", name);
  return 0;
}
```

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc main4.c -o main4
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ ./main4
Hi Nobody, how are you?
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc main4.c -o main4 -DLEO
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ ./main4
Hi Leo, how are you?
```

多檔案編譯與連結

1. 參數 -c ←只編譯，不連結
   - ✓ 多個原始碼檔案分開編譯，然後一起連結（lib1.c lib2.c main5.c）
   - ✓ 方便開發、管理
   - ✓ 加快建置時間
   - ✓ 進階工具 → make

```c
// lib1.c
int add(int a, int b) {
    return (a+b);
}
```

```c
// lib2.c
int mul(int a, int b) {
    return (a*b);
}
```

```c
// main5.c
#include <stdio.h>

extern int add(int, int);
extern int mul(int, int);
int main() {
    int s = add(1, 2);
    int ans = mul(s, 3);
    printf("ans = %d\n", ans);
    return 0;
}
```

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc -c lib1.c
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc -c lib2.c
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc -c main5.c
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc lib1.o lib2.o main5.o -o main5
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ ./main5
ans = 9
```

最佳化編譯

1. 參數 -O[最佳化程度]
   - ✓ 編譯器自行判斷，在不影響執行結果的前提下調整機械碼以縮小執行檔或加快執行速度（main6.c）

```c
// main6.c
#include <stdio.h>

int add (int a, int b) {
    return (a+b);
}
int main() {
    int sum = 0;
    for (int i=1; i<=100000; i++) {
        for (int j=1; j<=10000; j++) {
            int a = i, b = j;
            sum = add(a, b);
        }
    }
    printf("sum=%d\n",sum);
    return 0;
}
```

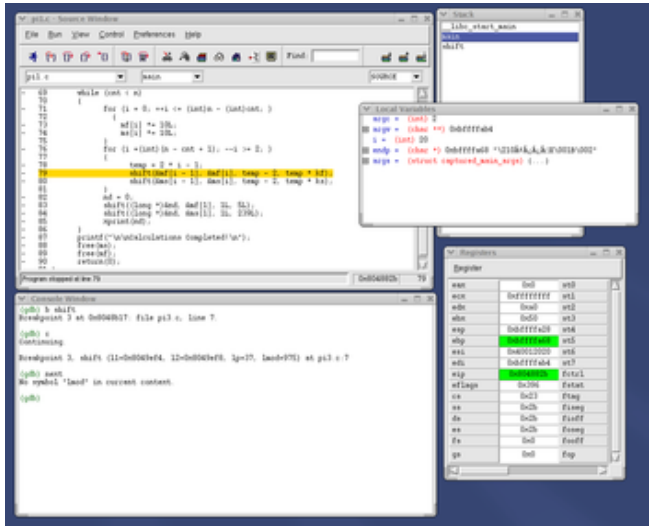
```
~/pd2$ gcc main6.c -o main6
 sum=110000

 real    0m3.035s

~/pd2$ gcc main6.c -o main601 -O1
 sum=110000

 real    0m0.272s


~/pd2$ gcc main6.c -o main602 -O2
 sum=110000

 real    0m0.002s

~/pd2$ gcc main6.c -o main603 -O3
 sum=110000

 real    0m0.002s
```

附加除錯訊息

1. 參數 -g
   - ✓ 編譯出來的執行檔內嵌除錯訊息，可以使用 GDB 除錯。

GDB

- GNU 偵錯器（GNU Debugger）
  - GNU 軟體系統中的標準偵錯器，也是個具有可攜性的偵錯器，經過可攜需求的調修與重新編譯，如今許多的類 UNIX 作業系統上都可以使用 GDB。
  - 支援 C、C++、Pascal 以及 FORTRAN。
  - 有部分圖形介面可以使用，但未形成標準，也不具備可攜性。



常見指令（1/2）

- help (h)：顯示指令簡短說明。例：help breakpoint
- start：開始執行，並停在第一行指令。
- run (r)：執行程式，或是從頭再執行程式。
- kill：中止程式的執行。
- backtrace (bt)：顯示程式呼叫的堆疊(stack)。
- print (p)：印出變數內容。例：p i，印出變數 i 的內容。
- list (l)：印出程式碼。可指定行號，例：l 10。
- breakpoint (b)：設定中斷點
- delete (d)：刪除某個 breakpoint。
- info breakpoint (i b)：查看已設定了哪些中斷點。

常見指令（2/2）

- watch：設定某個變數被改變時中斷。
- continue (c)：繼續執行。
- frame (f)：顯示正在執行的行數、副程式名稱、及其所傳送的參數等資訊。
- next (n)：單步執行，但遇到函式時會將呼叫的函式作為一個語句執行。
- step (s)：單步執行，但遇到函式時會進入呼叫的函式執行。
- quit (q)：離開 gdb。
- 按下 Enter：直接執行上個指令。

基本操作

```
$ gcc main2-2.c -o main2-2 -g ← 編譯時加上 -g 參數
$ gdb main2-2 ← 用 gdb 載入 main2-2
```

```
(gdb) l ← 顯示原始碼
```

```
(gdb) b 8 ← 在第 8 行設定中斷點
```

```
(gdb) r ← 執行程式，停在剛剛設定的第 8 行
```

```
(gdb) p i ← 顯示變數 i 的值
(gdb) p sum ← 顯示變數 sum 的值
```

```
(gdb) n ← 執行一行程式
```

```
(gdb) q ← 結束 GDB
```

GDB watch 指令

```c
#include <stdio.h>

int a, b;

void fun1() {
    int c = a + b;
}
void fun2() {
    int *iPtr = &a;
    ++*iPtr;
}
void fun3() {
    int c = a * b;
}
```

```c
int main() {
    a = 10;
    b = 20;
    printf("Before: a = %d; b = %d.\n", a, b);
    fun1();
    fun2();
    fun3();
    printf("After:  a = %d; b = %d.\n", a, b);
    return 0;
}
```

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ ./gdb1
Before: a = 10; b = 20.
After:  a = 11; b = 20.
```

Core Dump

1. 當程式在執行的過程中發生了異常的中止或是非法存取記憶體，作業系統會將當時的記憶體狀態記錄下來，這個狀態包括暫存器狀態，程式堆疊等等，然後把這一些資訊儲存成一個檔案，而這個檔案被稱為核心轉存檔案（core dump）。
2. Core dump 檔案雖然缺少許多除錯資訊，但仍然可以被 GDB 讀取解析，還原錯誤發生時的狀態。
3. 通常應用在已經 release 的執行檔，在客戶環境重現異常中止的狀況，產生 core dump 檔案，交由研發人員分析並修正錯誤。

Core Dump 範例

```c
#include <stdio.h>

int main(void)
{
    int *a;
    printf("Input a number: ");
    scanf("%d", a);
    return 0;
}
```

$ulimit -c unlimited ← 開啟 core dump

```
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ gcc gdb2.c -o gdb2
sevenbai@LAPTOP-G0HPD2AP:~/pd2$ ./gdb2
Input a number: 111
Segmentation fault (core dumped)
```

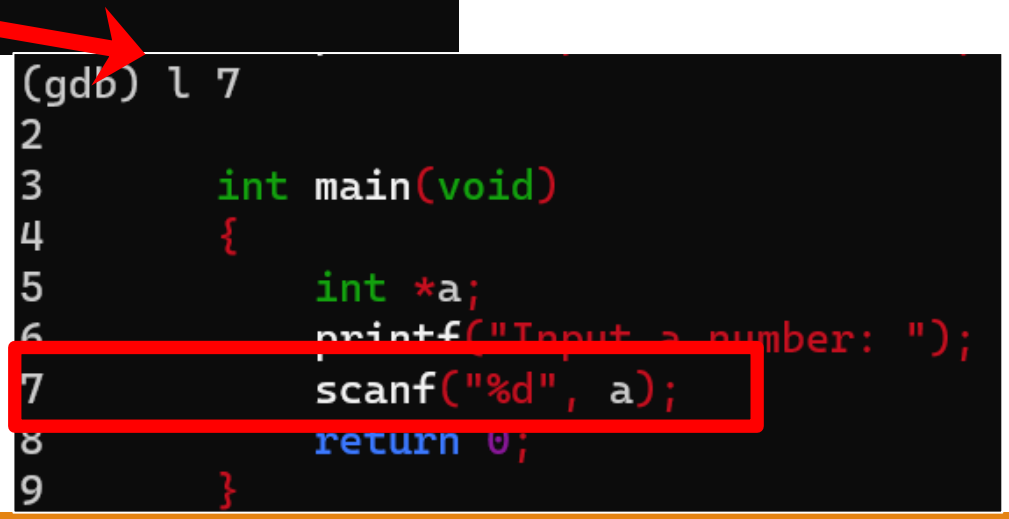
```
~/pd2$ gdb ./gdb2 core.gdb2
```
← 載入 core dump 檔案

```
Core was generated by `./gdb2'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  0x00007fa77b6911c9 in __vfscanf_internal (
    s=<optimized out>, format=<optimized out>,
    argptr=argptr@entry=0x7ffcf8ee9a90,
    mode_flags=mode_flags@entry=2)
    at ./stdio-common/vfscanf-internal.c:1896
1896    ./stdio-common/vfscanf-internal.c: No such file or direc
tory.
(gdb) bt
#0  0x00007fa77b6911c9 in __vfscanf_internal (
    s=<optimized out>, format=<optimized out>,
    argptr=argptr@entry=0x7ffcf8ee9a90,
    mode_flags=mode_flags@entry=2)
    at ./stdio-common/vfscanf-internal.c:1896
#1  0x00007fa77b68c1c2 in __isoc99_scanf (
    format=<optimized out>) at ./stdio-common/isoc99_scanf.c:30
#2  0x000055919c35e170 in main ()
(gdb)
```

```
~/pd2$ gcc gdb2.c -o gdb2.gdb -g
~/pd2$ strip -g gdb2.gdb -o gdb2
```
← 保留 gdb2.gdb，交付 gdb2

```
~/pd2$ gdb ./gdb2.gdb core.gdb2
```
← 用 gdb2.gdb 來 debug

```
(gdb) bt
#0  0x00007f0cf0fb71c9 in __vfscanf_internal (
    s=<optimized out>, format=<optimized out>,
    argptr=argptr@entry=0x7ffcacd4cb50,
    mode_flags=mode_flags@entry=2)
    at ./stdio-common/vfscanf-internal.c:1896
#1  0x00007f0cf0fb21c2 in __isoc99_scanf (
    format=<optimized out>) at ./stdio-common/isoc99_scanf.c:30
#2  0x000055f56f9ab1a4 in main () at gdb2.c:7
```



Q&A

GNU Toolchain : Make & Makefile

What is GNU make?

Lloyd Huang
KaLUG
February 21 2023

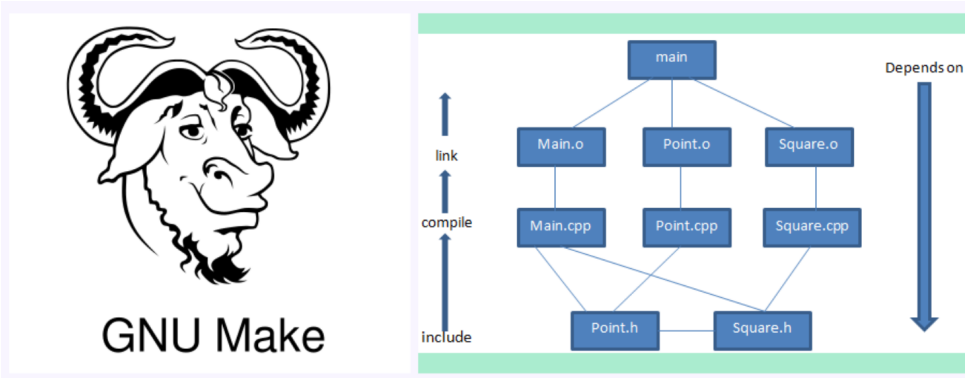
Outline

About Me:

Lloyd Huang a.k.a 黃宇新

- 自由軟體愛好者 - Open Source 亦師亦友的生態系
- COSCUP, PyCon, MOPCON, CLE, KaLUG 參與者
- ChromeOS, Debian, Gentoo, GUIX, Emacs, Elisp, Python
- 信任網路 BiiLabs 技術副總
- 鴻海科技集團創新數位系統事業群 (iDSBG) 經理
- 宏澧科技 Covia Global 副總經理、總工程師
- 網虎國際 Coventive 經理、資深工程師

GNU Make - What and Why?



- A Build & Install System.
- Rules of Dependence.
- Timestamp tracking for dependency management

Basic Syntax

Basic syntax - Comments, Targets: Prerequisites, @Commands

```
# Target1 Target2: Prerequisite1 Prerequisite2
#    Commands..
#    Commands..
1F:
    touch 1F
    echo -e "<<<hello world [1F]>>> - create 1F file\n"

2F: 1F
    @echo -e "<<<hello world [2F]>>> - don't create file\n"

3F: 1F 2F
    @echo -e "<<<hello world [3F]>>>"
    touch 2F 3F

clean:
    rm -f 1F 2F 3F
```

Start a Project - main.cpp

main.cpp

```
#include <iostream>
extern int lib_hello();
extern int square(int i);

int main() {
    std::cout << "Hello World!\n";
    lib_hello();
    std::cout << "ARI: square 5 * 5 = "
          << square(5) << "\n";
    return 0;
}
```

Start a Project - lib.cpp square.cpp

lib.cpp

```
#include <iostream>

int lib_hello() {
  std::cout << "LIB: Hello World!\n";
  return 0;
}
```

square.cpp

```
int square(int i) {
  return i*i;
}
```

Start a Project - Makefile

makefile

```
all: main

main: main.o lib.o square.o
    g++ main.o lib.o square.o -o main

main.o: main.cpp
    g++ -c main.cpp -o main.o

square.o: square.cpp
    g++ -c square.cpp -o square.o

lib.o: lib.cpp
    g++ -c lib.cpp -o lib.o

clean:
    rm -f *.o main

install:
    cp main /tmp
```

Variables and Pattern Rule

Variables and Pattern Rule

```
# Variable
CFLAGS  = -O2 -Wall -g
objects = main.o lib.o square.o

all: main

main: $(objects)
    g++ $(CFLAGS) $(objects) -o main

# Define a pattern rule that compiles every .cpp file into a .o file
# EX: %  = xxxx         :: %.o = xxxx.o,    %.cpp = xxxx.cpp
#     $@ = Target       :: %.o = xxxx.o
#     $< = Prerequisite :: ../start_project/%.cpp = ../start_project/xxx
%.o: ../start_project/%.cpp
    g++ $(CFLAGS) -c $< -o $@

clean:
    rm $(objects) main
```

FAQ

FAQ - ☑<tab> ✗<spaces>

```
# 常見錯誤, command 需以 <tab> 字符為開始
all:
    @echo "<<ERROR: command stat with 8 spaces>>"
    @echo "-------------------------------------"
    @echo "<<command stat with <tab> character>>"
```

FAQ - Commands in isolation.

```
# Commands in subshell, that isolate Working Directory and Variables.
MVER = "version 5.0"
all:
    @                       echo -n "1)[$(MVER)] [$$VER] "  ; pwd
    @cd /usr/local ;        echo -n "2)[$(MVER)] [$$VER] "  ; pwd
    @VER="version A.2.3";   echo -n "3)[$(MVER)] [$$VER] "  ; pwd
    @MVER="version A.2.3"; echo -n "4)[$(MVER)] [$$MVER] " ; pwd
```

Version Control - Git

聯庭智能 研發經理

陳泊亨

陳泊亨

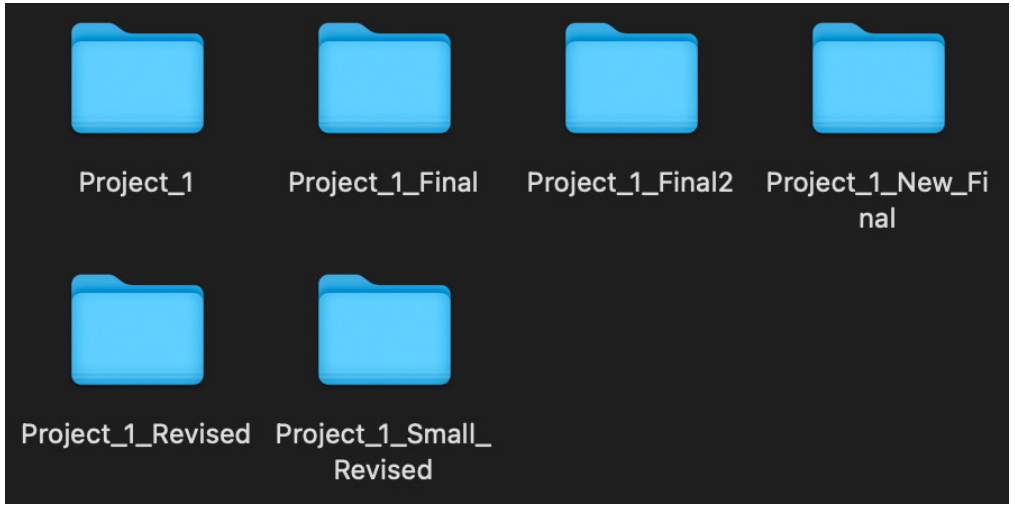
- 國立成功大學 資訊工程博士
- 國立成功大學 資訊工程學系 博士後研究員
- 聯庭智能 研發經理

情境一

情境一

一個月後...

情境一


Project_1
Project_1_Final
Project_1_Final2
Project_1_New_Final
Project_1_Revised
Project_1_Small_Revised


為了避免下次遇到一樣的狀況，每次修改的版本都存下來好了

情境二

再一個月後...

情境三

情境三

版本控制

- 版本控制（Version Control）
  - 維護工程藍圖的標準作法
  - 能追蹤工程藍圖從誕生一直到定案的過程
  - 軟體工程技巧，藉此能在軟體開發的過程中，確保由不同人所編輯的同一程式檔案都得到同步

版本控制 – 概述

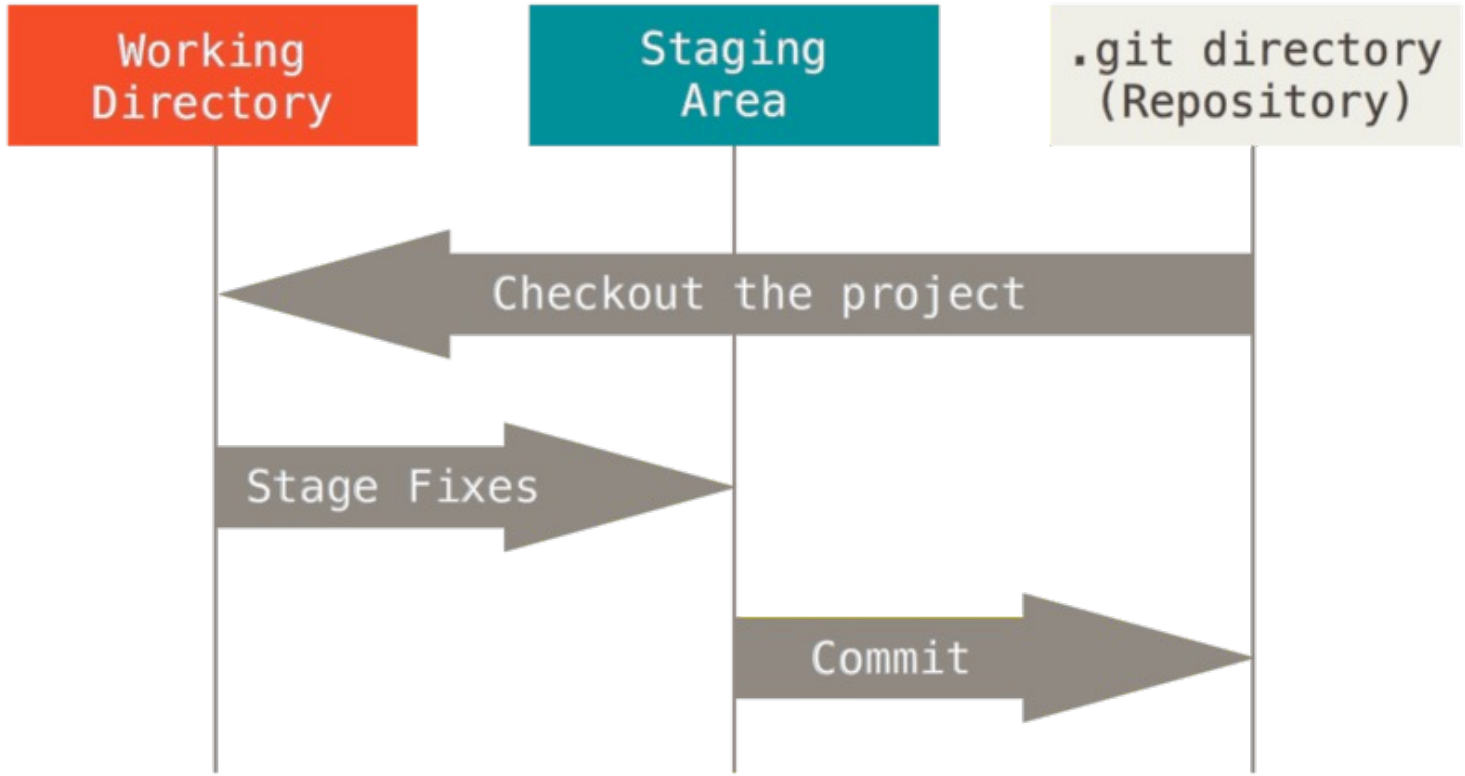
- 透過文件控制（documentation control），能記錄任何工程專案內各個模組的改動歷程，並為每次改動編上序號
- 一種簡單的版本控制形式如下：賦予初版一個版本等級「A」。做了第一次改變後，版本等級改為「B」，以此類推。
- 版本控制能提供專案的設計者，將設計回復到之前的任一狀態，這特性在設計過程中碰到完全無法運作的情況時顯得特別重要

Git

- Version control system
- Trace code changelog
- Work together

Git – Three States

- Working Directory
  - state: modified
- Staging Area
  - state: staged
- Repository
  - state: committed



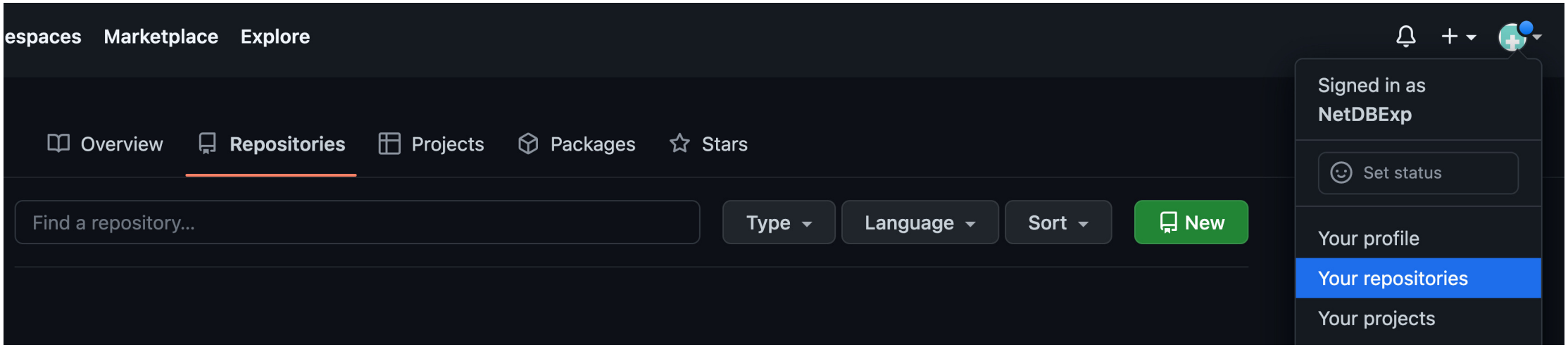
Git hosting platforms

- Github
- Gitlab
- Bitbucket
- …

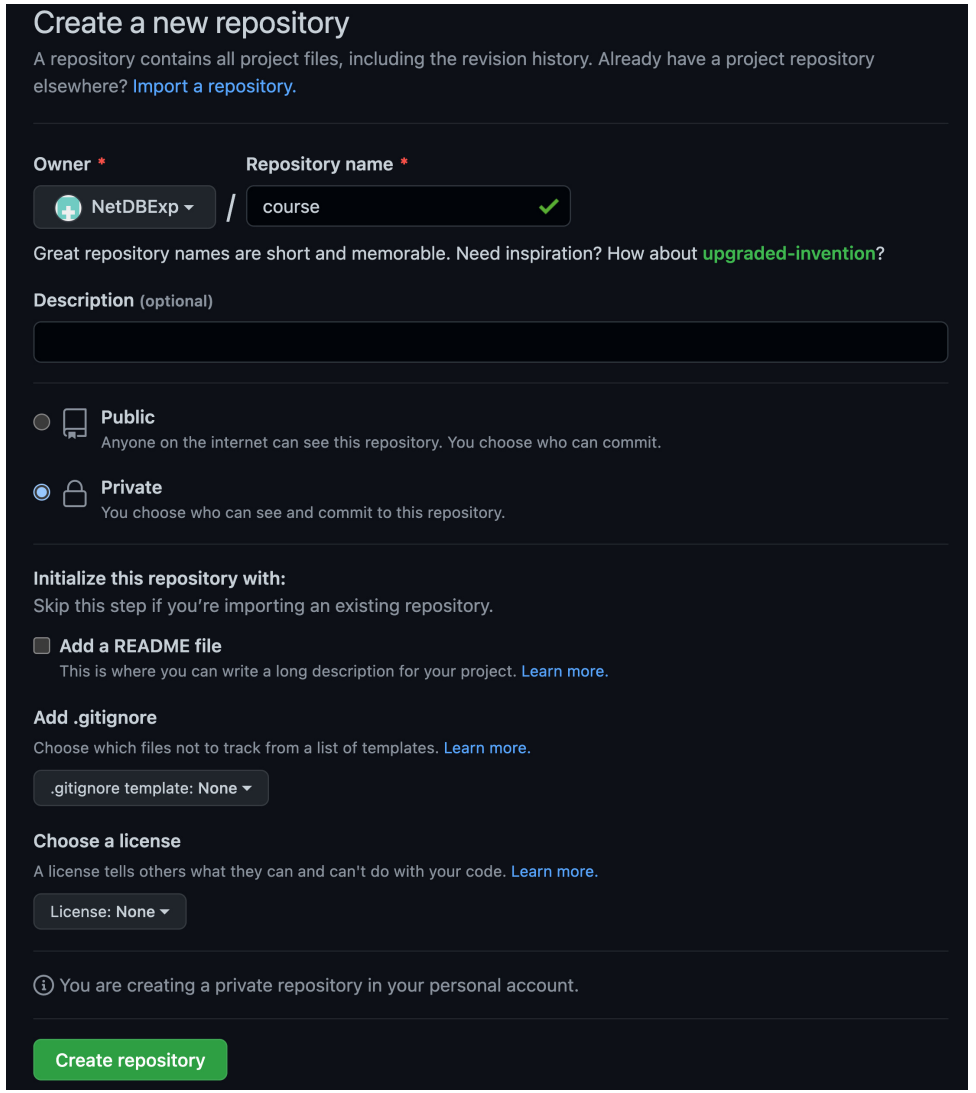
Github

- https://github.com/
- GitHub在2008年由GitHub公司開發上線，並在十年後被微軟以75億美元收購
- 截至2023年1月26日，已經有超過1億開發人員使用GitHub，是世界上最大的開源社群。
- 它不但是程式碼存放處，也是工程師們交流程式碼的主要社群。

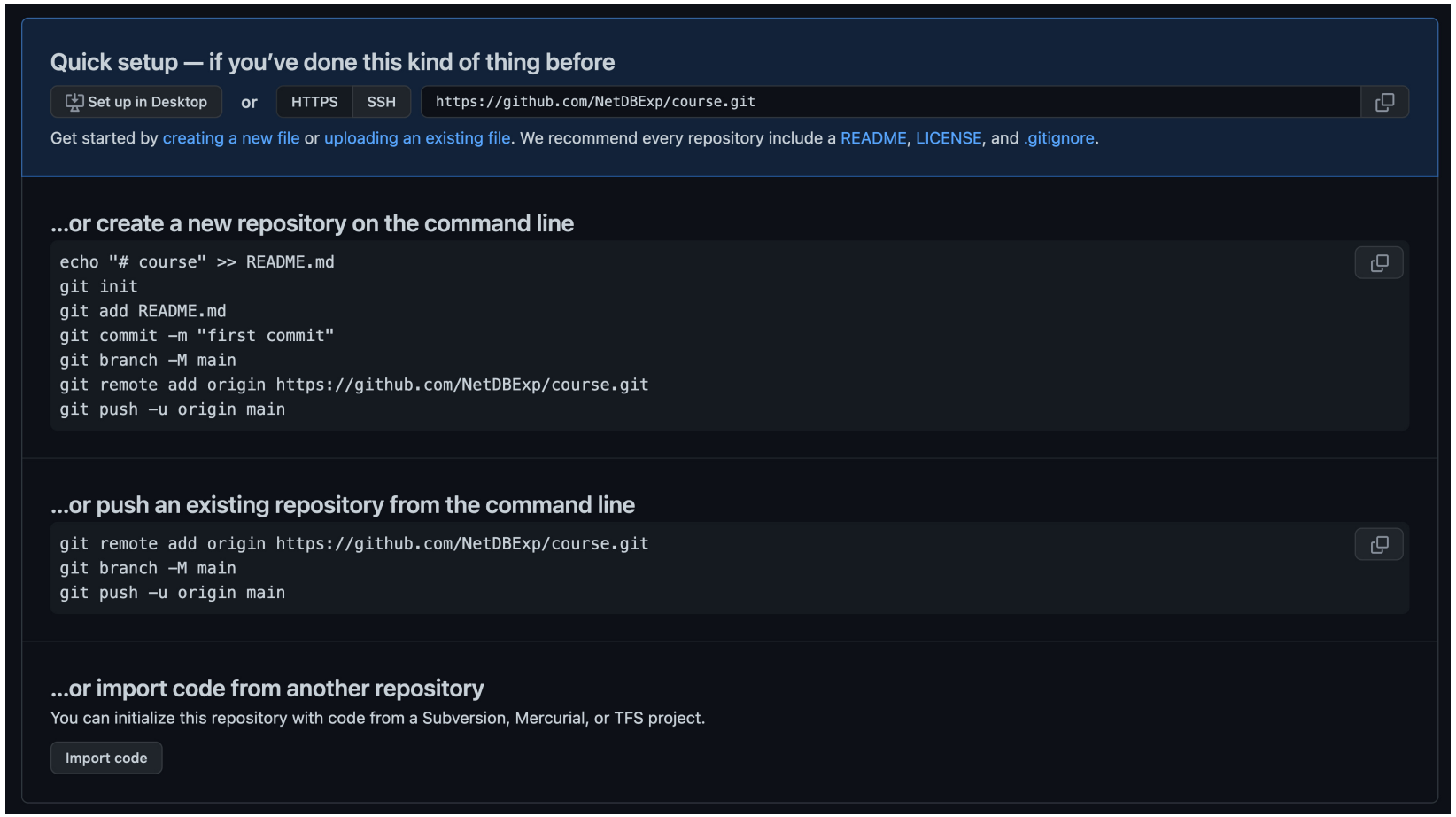
Github


espaces
Marketplace
Explore
Signed in as
NetDBExp
Set status
Your profile
Your repositories
Your projects
Overview
Repositories
Projects
Packages
Stars
Find a repository...
Type
Language
Sort
New

Github - Create a new repository



Github - Create a new repository



Git config

- 查看設定
  - git config -l
- 設定基本資訊
  - git config --global user.name "Your Name"
  - git config --global user.email "Your Email"
- 設定預設的branch名稱
  - git config --global init.defaultBranch main

```
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
```

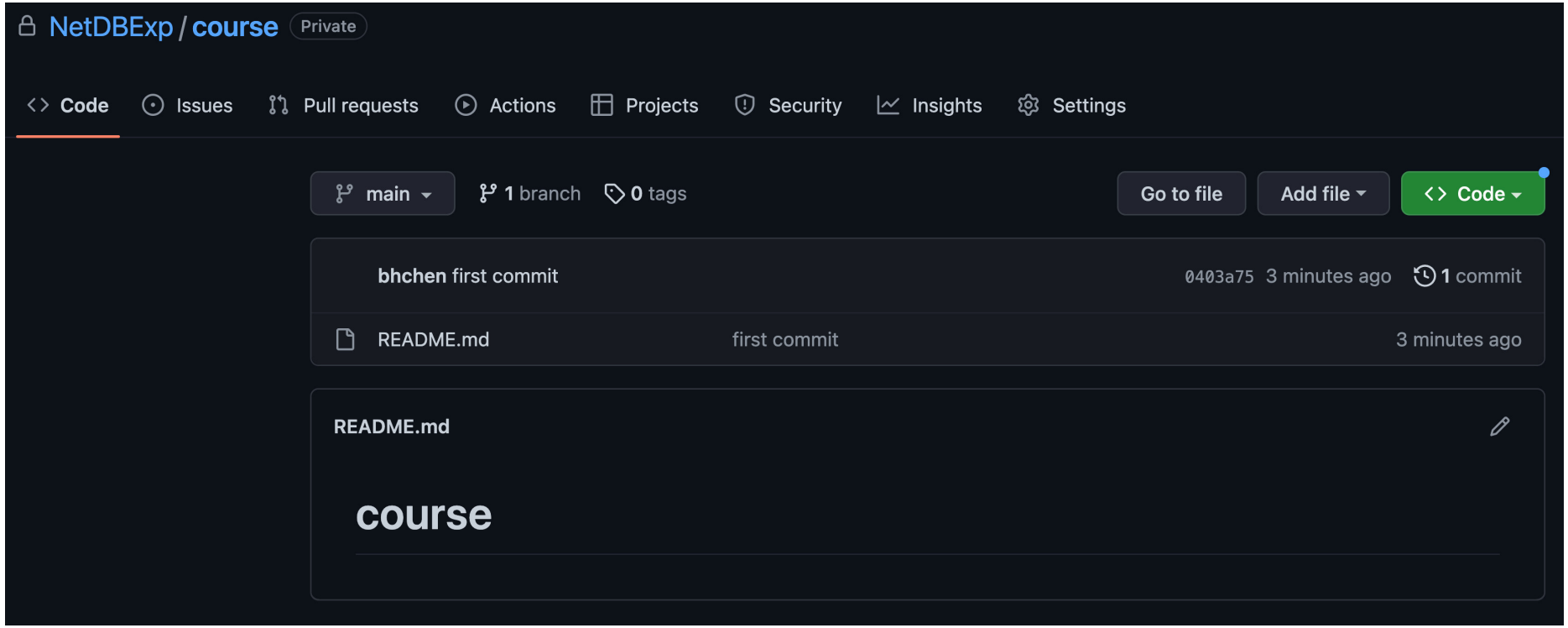
```
bhchen@2023pd2:/home/bhchen/course$ git config -l
bhchen@2023pd2:/home/bhchen/course$ git config --global user.name "bhchen"
bhchen@2023pd2:/home/bhchen/course$ git config --global user.email "ncku.pd2.ta@netdb.csie.ncku.edu.tw"
bhchen@2023pd2:/home/bhchen/course$ git config -l
user.name=bhchen
user.email=ncku.pd2.ta@netdb.csie.ncku.edu.tw
```

Create a new repository on the command line

- echo "# course" >> README.md
- git init
- git add README.md
- git commit -m "first commit"
- git branch -M main
- git remote add origin {remote_url}
- git push -u origin main

```
bhchen@2023pd2:/home/bhchen/course$ echo "# course" >> README.md
bhchen@2023pd2:/home/bhchen/course$ git init
Initialized empty Git repository in /home/bhchen/course/.git/
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)
bhchen@2023pd2:/home/bhchen/course$ git add README.md
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md

bhchen@2023pd2:/home/bhchen/course$ git commit -m "first commit"
[main (root-commit) 0403a75] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
nothing to commit, working tree clean
bhchen@2023pd2:/home/bhchen/course$ git remote add origin https://github.com/NetDBExp/course.git
bhchen@2023pd2:/home/bhchen/course$ git push -u origin main
Username for 'https://github.com': NetDBExp
Password for 'https://NetDBExp@github.com':
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 226 bytes | 226.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NetDBExp/course.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Create a new repository on the command line



Ignoring Files

- .gitignore

```
# ignore all .a files
*.a

# but do track lib.a, even though you're ignoring .a files above
!lib.a

# only ignore the TODO file in the current directory, not subdir/TODO
/TODO

# ignore all files in any directory named build
build/

# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt

# ignore all .pdf files in the doc/ directory and any of its subdirectories
doc/**/*.pdf
```

```
bhchen@2023pd2:/home/bhchen/course$ echo "test" > test.a
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.a

nothing added to commit but untracked files present (use "git add" to track)
bhchen@2023pd2:/home/bhchen/course$ echo "*.a" >> .gitignore
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

Viewing Your Staged and Unstaged Changes

- git diff by itself doesn't show all changes made since your last commit — only changes that are still unstaged

```
bhchen@2023pd2:/home/bhchen/course$ echo "123" >> README.md
bhchen@2023pd2:/home/bhchen/course$ git diff
diff --git a/README.md b/README.md
index 31e1755..5b54231 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,2 @@
 # course
+123
```

Removing Files

```
bhchen@2023pd2:/home/bhchen/course$ ls
README.md  test.a  test.b
bhchen@2023pd2:/home/bhchen/course$ rm test.b
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    test.b

no changes added to commit (use "git add" and/or "git commit -a")
bhchen@2023pd2:/home/bhchen/course$ git rm test.b
rm 'test.b'
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    test.b
```

Viewing the Commit History

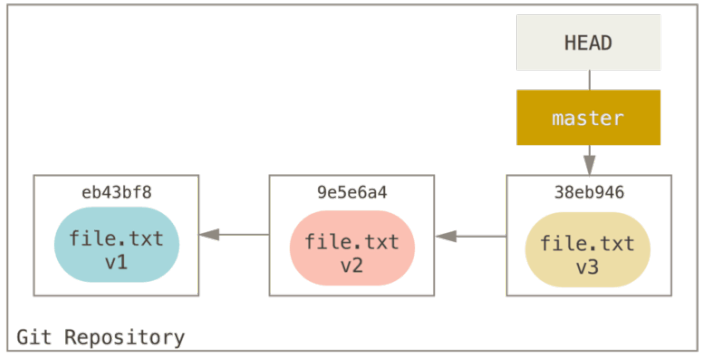
```
bhchen@2023pd2:/home/bhchen/course$ git log
commit b85ed51cfa876ca7f6bd0a4ec6f74845cfcb0615 (HEAD -> main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:30:01 2023 +0000

    rm test.c

commit c878482e47cec98f24987ac08e0de5c4fdeddbb6
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:27:46 2023 +0000

    add test.c

commit 7b3b82a6e49dad56ad5fa954c0756394c1a85f10
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:18:18 2023 +0000

    add test.b

commit 0403a75ecd3f4b73b4e058a046d62aeb5dfa5e6f (origin/main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 16:27:49 2023 +0000

    first commit
```

Undoing Things

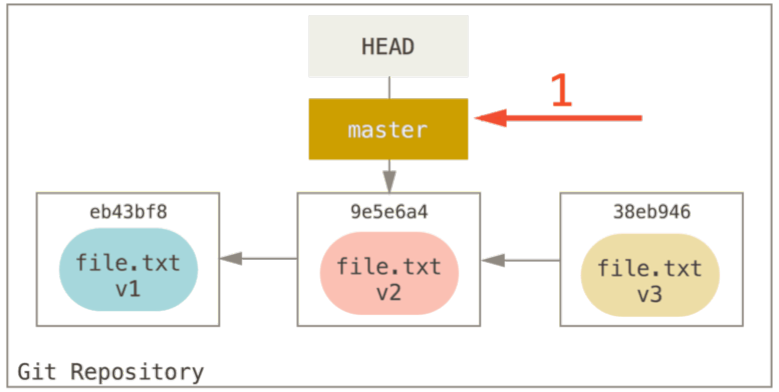
- Unstage the file
  - git restore --staged <file>
- Unmodifying a modified file
  - git checkout -- <file>

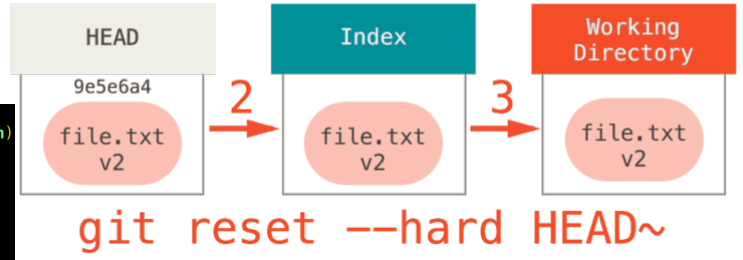
```
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.b
        new file:   test.d

bhchen@2023pd2:/home/bhchen/course$ git restore --staged test.b
bhchen@2023pd2:/home/bhchen/course$ git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test.d

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   test.b

bhchen@2023pd2:/home/bhchen/course$ git diff
diff --git a/test.b b/test.b
index 9daeafb..357bfd8 100644
--- a/test.b
+++ b/test.b
@@ -1 +1,2 @@
 test
+123
bhchen@2023pd2:/home/bhchen/course$ cat test.b
test
123
bhchen@2023pd2:/home/bhchen/course$ git checkout -- test.b
bhchen@2023pd2:/home/bhchen/course$ cat test.b
test
bhchen@2023pd2:/home/bhchen/course$
```

Reset Demystified - hard




```
bhchen@2023pd2:/home/bhchen/course$ git log
commit 0cc91ee8f738174e7306d19242c14e7eb4088434 (HEAD -> main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 18:13:38 2023 +0000

    add test.d

commit b85ed51cfa876ca7f6bd0a4ec6f74845cfcb0615
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:30:01 2023 +0000

    rm test.c

commit c878482e47cec98f24987ac08e0de5c4fdeddbb6
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:27:46 2023 +0000

    add test.c

commit 7b3b82a6e49dad56ad5fa954c0756394c1a85f10
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:18:18 2023 +0000

    add test.b

commit 0403a75ecd3f4b73b4e058a046d62aeb5dfa5e6f (origin/main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 16:27:49 2023 +0000

    first commit
```





git reset --hard HEAD~

```
bhchen@2023pd2:/home/bhchen/course$ ls
README.md  test.a  test.b  test.d
bhchen@2023pd2:/home/bhchen/course$ git reset --hard HEAD~
HEAD is now at b85ed51 rm test.c
bhchen@2023pd2:/home/bhchen/course$ git log
commit b85ed51cfa876ca7f6bd0a4ec6f74845cfcb0615 (HEAD -> main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:30:01 2023 +0000

    rm test.c

commit c878482e47cec98f24987ac08e0de5c4fdeddbb6
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:27:46 2023 +0000

    add test.c

commit 7b3b82a6e49dad56ad5fa954c0756394c1a85f10
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 17:18:18 2023 +0000

    add test.b

commit 0403a75ecd3f4b73b4e058a046d62aeb5dfa5e6f (origin/main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 16:27:49 2023 +0000

    first commit
bhchen@2023pd2:/home/bhchen/course$ ls
README.md  test.a  test.b
```

Undo git reset --hard

- Reference logs
  - git reflog

```
bhchen@2023pd2:/home/bhchen/course$ git reflog
b85ed51 (HEAD -> main) HEAD@{0}: reset: moving to HEAD~
8a9b37c HEAD@{1}: commit: add test.d
b85ed51 (HEAD -> main) HEAD@{2}: reset: moving to HEAD~
a1aeba9 HEAD@{3}: commit: add test.d
b85ed51 (HEAD -> main) HEAD@{4}: reset: moving to HEAD~
0cc91ee HEAD@{5}: commit: add test.d
b85ed51 (HEAD -> main) HEAD@{6}: commit: rm test.c
c878482 HEAD@{7}: commit: add test.c
7b3b82a HEAD@{8}: commit: add test.b
0403a75 (origin/main) HEAD@{9}: commit (initial): first commit
bhchen@2023pd2:/home/bhchen/course$ ls
README.md  test.a  test.b
bhchen@2023pd2:/home/bhchen/course$ git reset a1aeba9
Unstaged changes after reset:
D       test.d
bhchen@2023pd2:/home/bhchen/course$ git reset a1aeba9 --hard
HEAD is now at a1aeba9 add test.d
```

Basic Branching

```
bhchen@2023pd2:/home/bhchen/course$ git branch b1
bhchen@2023pd2:/home/bhchen/course$ git checkout b1
Switched to branch 'b1'
bhchen@2023pd2:/home/bhchen/course$ git branch
* b1
  main
```

```
bhchen@2023pd2:/home/bhchen/course$ git branch
* b1
  main
bhchen@2023pd2:/home/bhchen/course$ cat test.b
test
bhchen@2023pd2:/home/bhchen/course$ echo "b1_add" >> test.b
bhchen@2023pd2:/home/bhchen/course$ git diff
diff --git a/test.b b/test.b
index 9daeafb..48a5a1d 100644
--- a/test.b
+++ b/test.b
@@ -1 +1,2 @@
 test
+b1_add
bhchen@2023pd2:/home/bhchen/course$ git add .
bhchen@2023pd2:/home/bhchen/course$ git commit -m "add text from b1"
[b1 a9d30db] add text from b1
 1 file changed, 1 insertion(+)
bhchen@2023pd2:/home/bhchen/course$ git log
commit a9d30dbfaf7cdf96c30b92061f9f397e907558eb (HEAD -> b1)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 18:46:04 2023 +0000

    add text from b1

commit a1aeba929b91f9b02a85c7a458c85de927426a97 (main)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 18:19:22 2023 +0000

    add test.d
```

Basic Merging

```
bhchen@2023pd2:/home/bhchen/course$ git branch
* b1
  main
bhchen@2023pd2:/home/bhchen/course$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 4 commits.
  (use "git push" to publish your local commits)
bhchen@2023pd2:/home/bhchen/course$ cat test.b
test
bhchen@2023pd2:/home/bhchen/course$ git merge b1
Updating a1aeba9..a9d30db
Fast-forward
 test.b | 1 +
 1 file changed, 1 insertion(+)
bhchen@2023pd2:/home/bhchen/course$ cat test.b
test
b1_add
bhchen@2023pd2:/home/bhchen/course$ git log
commit a9d30dbfaf7cdf96c30b92061f9f397e907558eb (HEAD -> main, b1)
Author: bhchen <ncku.pd2.ta@netdb.csie.ncku.edu.tw>
Date:   Mon Feb 20 18:46:04 2023 +0000

    add text from b1
```

Reference

- https://en.wikipedia.org/wiki/Version_control
- https://git-scm.com/book/en/v2