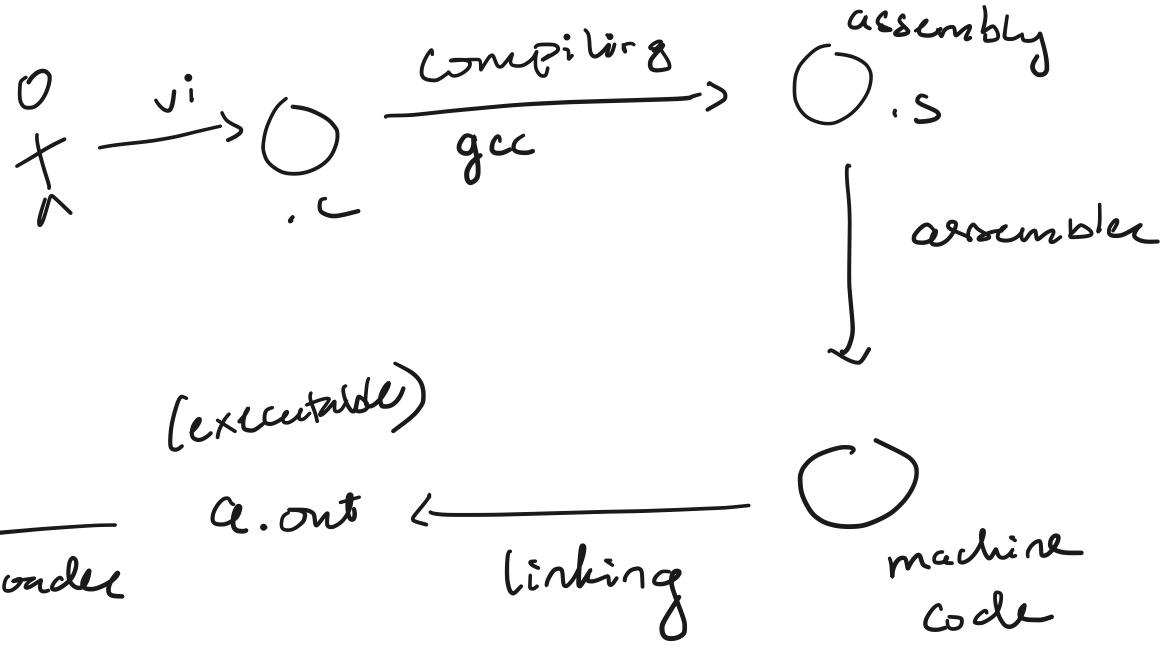
CS202 - 001 : OS

Review Session 1

- [x] 0. Record + Attendance
- [x] 1. Introductions
- [x] 2. Logistics
- [x] 3. Motivation + tips
- [ ] 4. Compilation Process
  Makefile
- [ ] 5. Git Introduction
- [ ] 6. Lab Overview + C Review
- [ ] 7. Debugging

gcc:



Makefile:

Special file w/ shell commands
"Makefile"

Goal:

- Efficient in compiling code. Avoid mixing up commands.
- Only compile files that change

4. Git (Good)

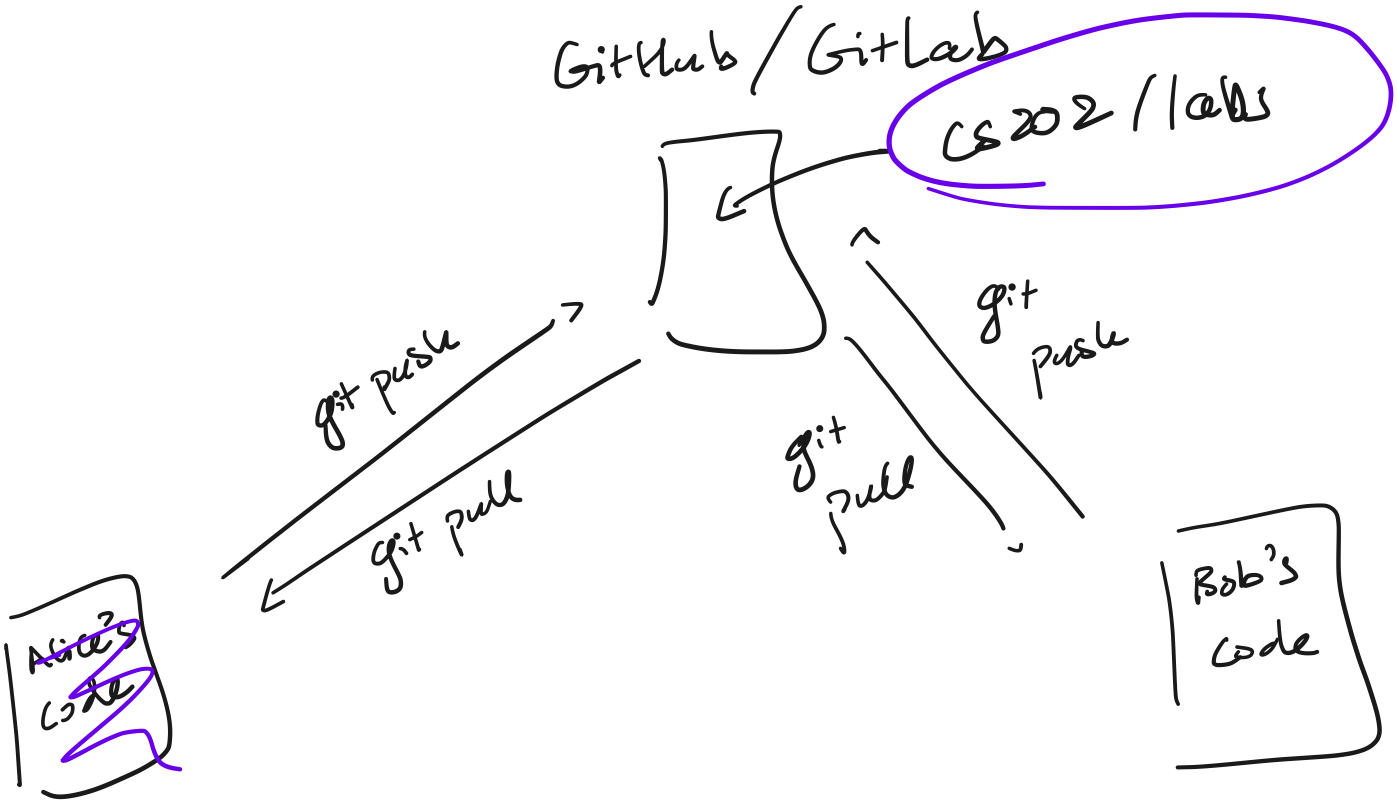
1> built by linus

git add — — —

git commit

git push

git pull === git fetch & git merge origin/main



Git != GitHub

DOCS

Goggle

Facebook

GitLab

internal implementation

Be familiar w/ Unix / terminal

---

5. Lab Overview + C Review

- Declaration & Initialization

```
int x;   --> undefined value
x = &;
```

- Pointers:
  - An address within memory
  - variable whose value is an memory address.

E.g: `int * p1;`

type * var_name

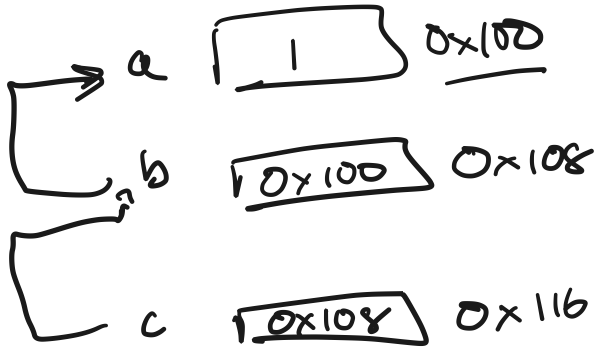
Dereferencing / Access the pointers: *p1

& : get me the address in memory of what follows.

```
int x = 5;
int * y = &x;
```
← y has the address of x

- Mental model

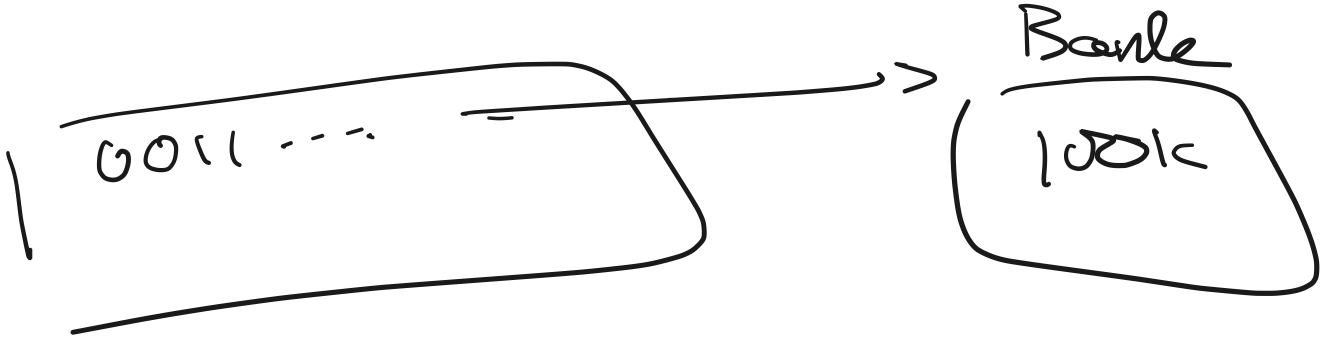
```
int a = 1;
int *b = &a;
int **c = &b;
```



double pointer

void fn (int * p)

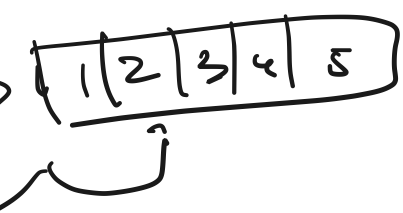
```
int *x = *c;
int z = *(*c);
```



- Array & String

type var-name

int arr[5] = {1, 2, 3, 4, 5};



arr[1] = 2;

int *ptr = arr

ptr

Equivalent

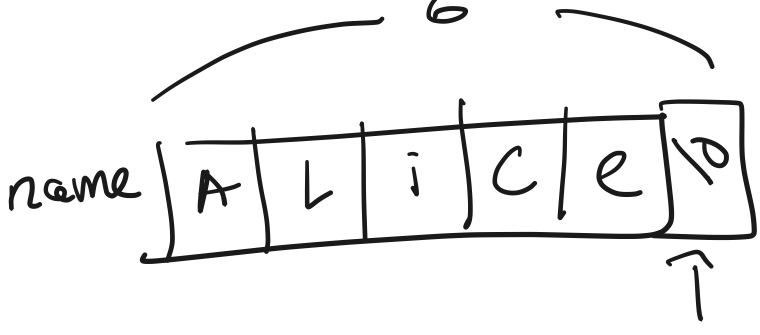
*(ptr + 1) == *(ptr + sizeof(int)*1) == access 2nd element

char name[] = "Alice" == 6

char *name = "Alice"

name[1] = 'B'

String



-- read only region.

constants

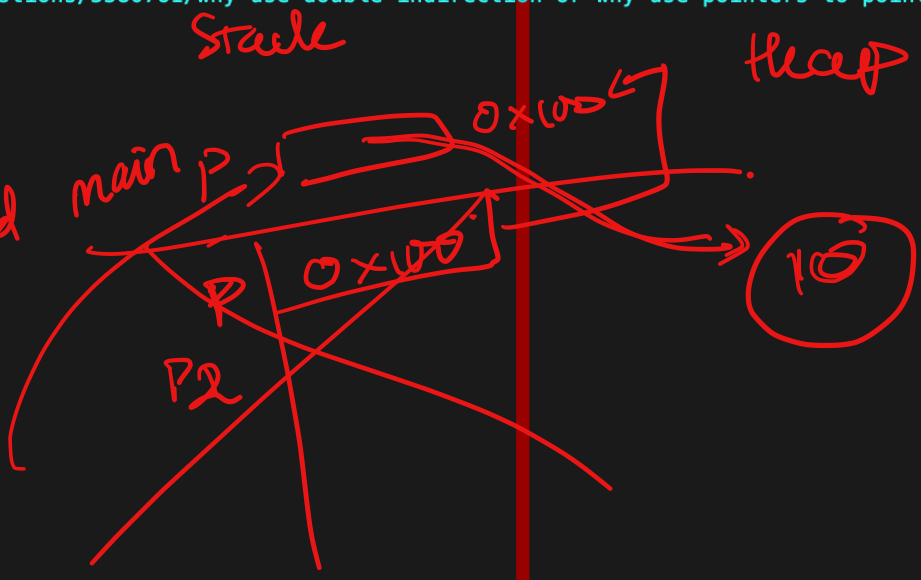
---

6. Debugging

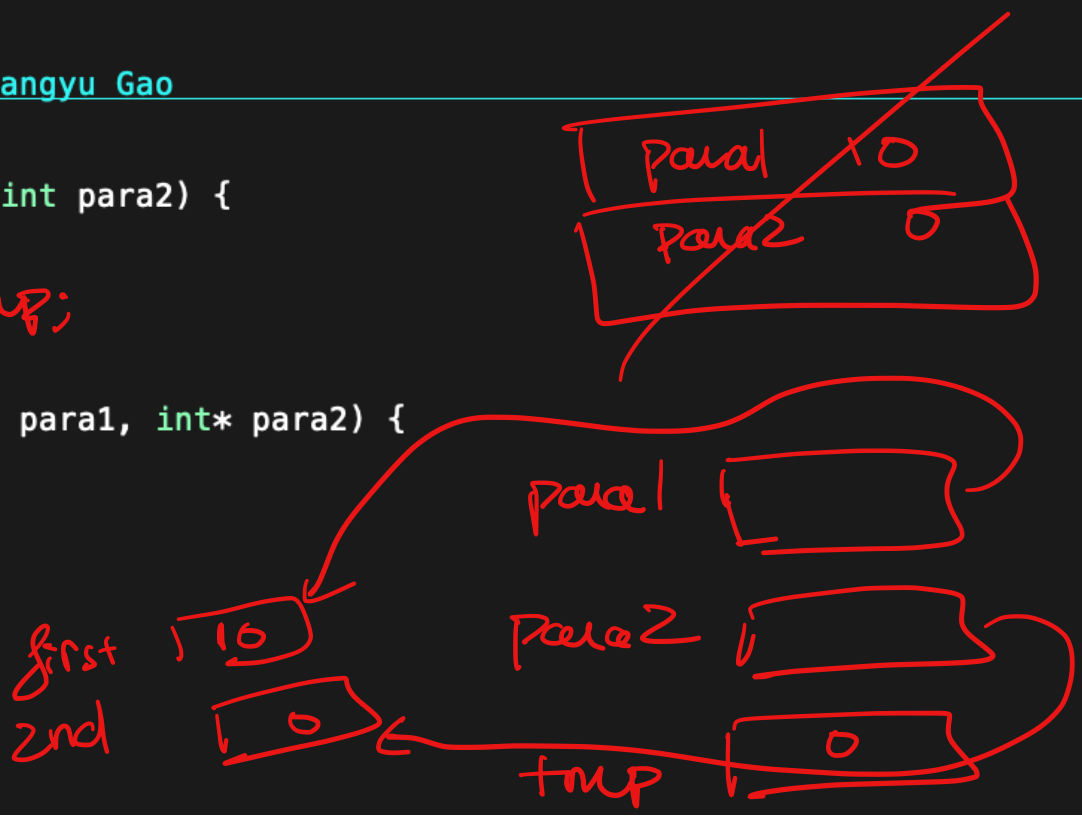
---

7. Q & A.

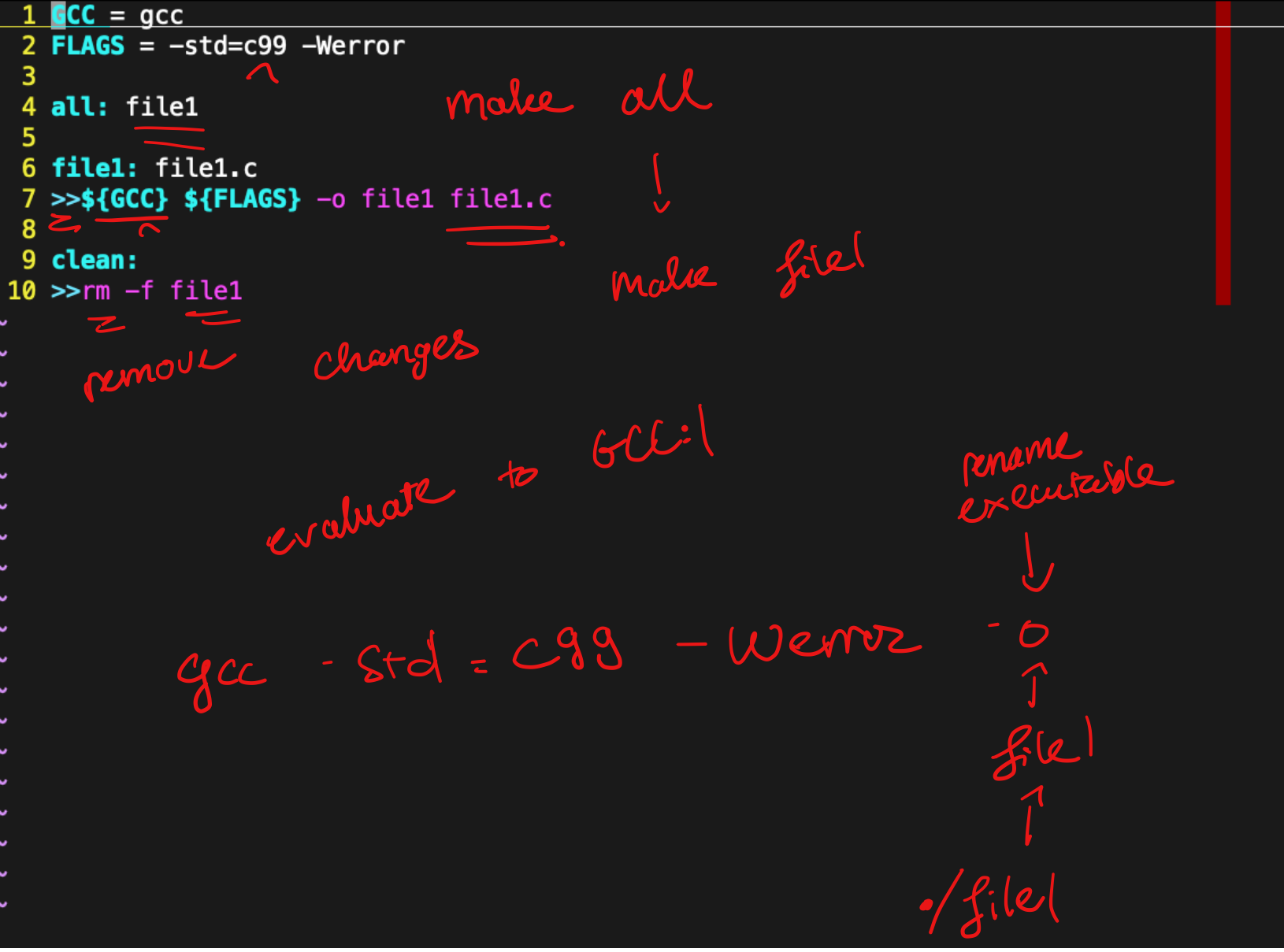
```c
#include <stdio.h>
#include <stdlib.h>

// Source: https://stackoverflow.com/questions/5580761/why-use-double-indirection-or-why-use-pointers-to-pointers

void p1(int* p) {
    p = (int *) malloc(sizeof(int));
    *p = 10;
}

void p2(int** p) {
    *p = (int *) malloc(sizeof(int));
    **p = 10;
}

int main()
{
    int* p = NULL;
    p1(p);
    printf("%d\n", *p);
....
    p2(&p);
    printf("%d\n", *p);
....
    free(p);

    return 0;
}
```



```c
#include <stdio.h>
#include <stdlib.h>

/*
Source: Example by Xiangyu Gao
*/

void swap(int para1, int para2) {
  int tmp = para1;
  para1 = para2;
  para2 = para1; tmp;
}

void second_swap(int* para1, int* para2) {
  int tmp = *para1;
  *para1 = *para2;
  *para2 = tmp;
}

int main() {
  int first = 0;
  int second = 10;
}
```



second.swap(&first, &second);


GCC = gcc
FLAGS = -std=c99 -Werror
all: file1
file1: file1.c
>>${GCC} ${FLAGS} -o file1 file1.c
clean:
>>rm -f file1
make all
make file1
remove changes
evaluate to GCC:1
gcc -std=c99 -Werror
rename executable
-o
file1
./file1