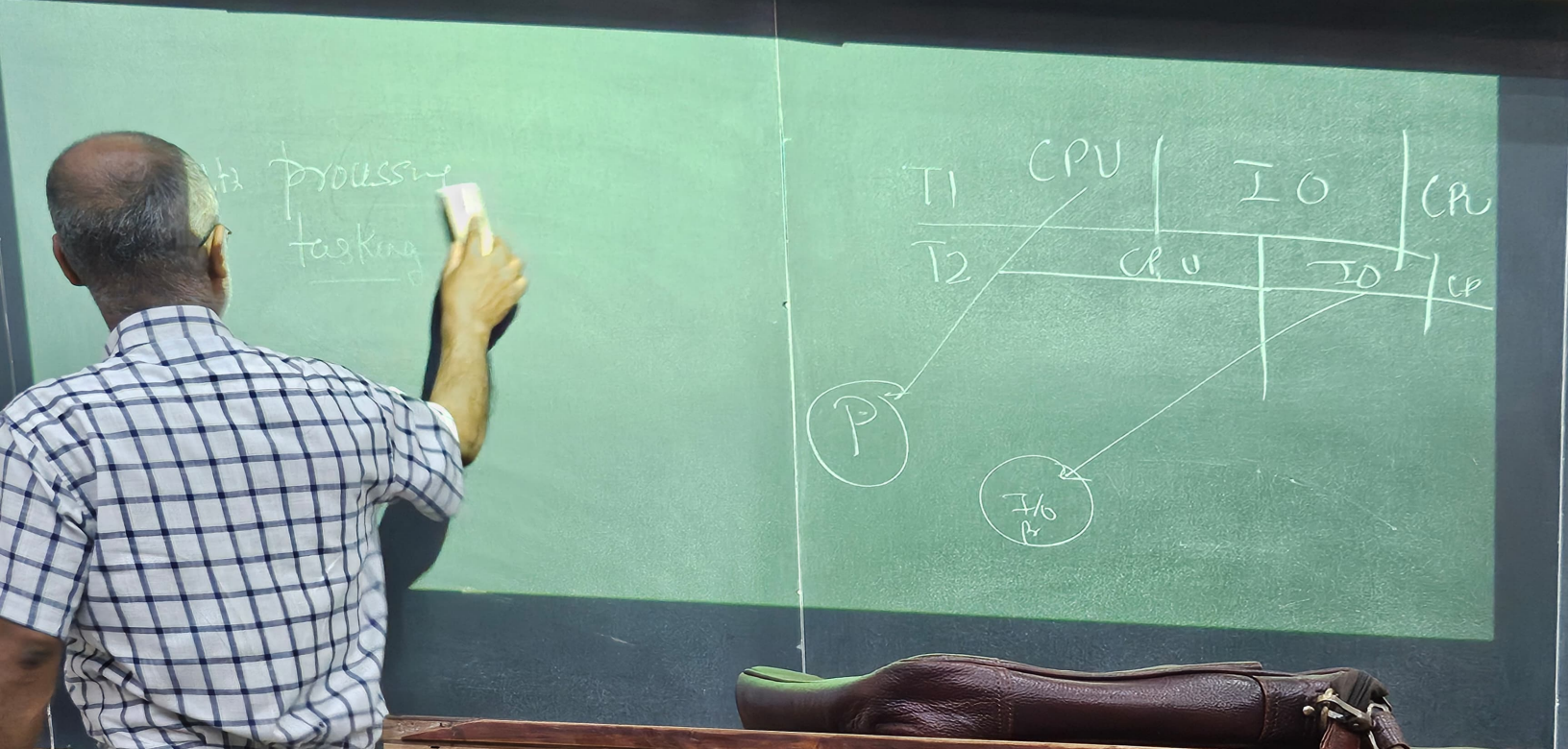
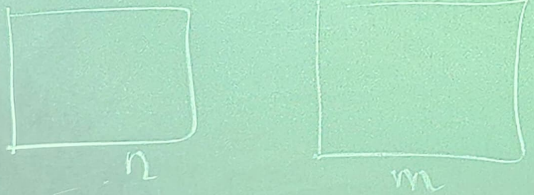
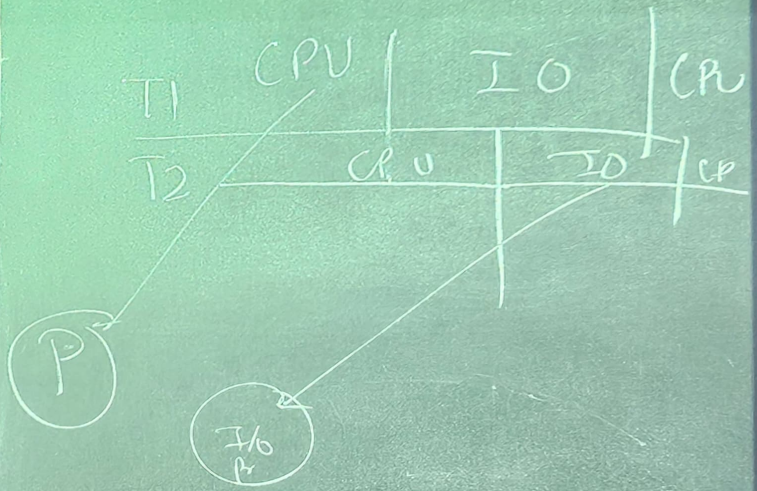

processing
tasking
T1
CPU
I O
T2
CPU
I O
P
I/O

JOIN





JOIN

$O\left(n^2/p\right)$

T1 CPU I O CPU

T2 C IO CP

int count = 0

```
Inc ( ) {
  for (i — 10^6
    count++
}

main ( ) {
    Inc( );
}
```

Parallel

Process

Thread

Pass Join

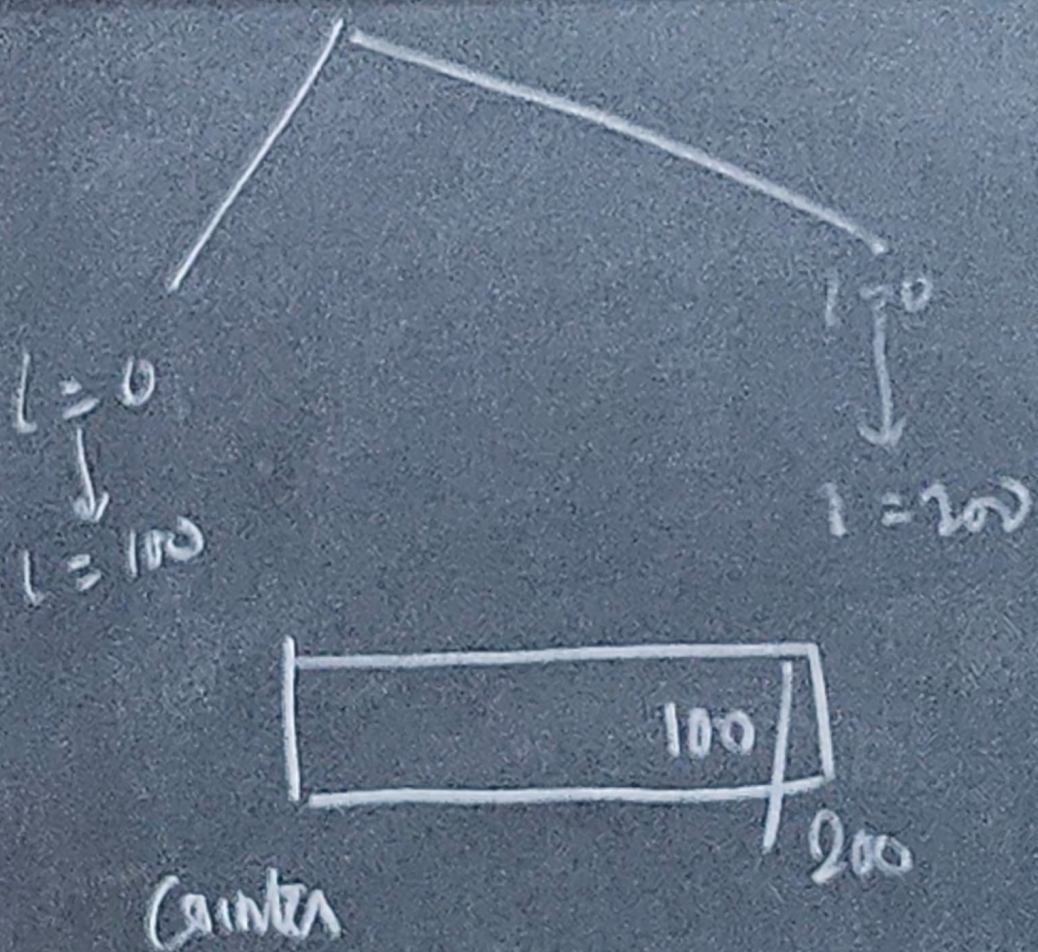
Concurrent

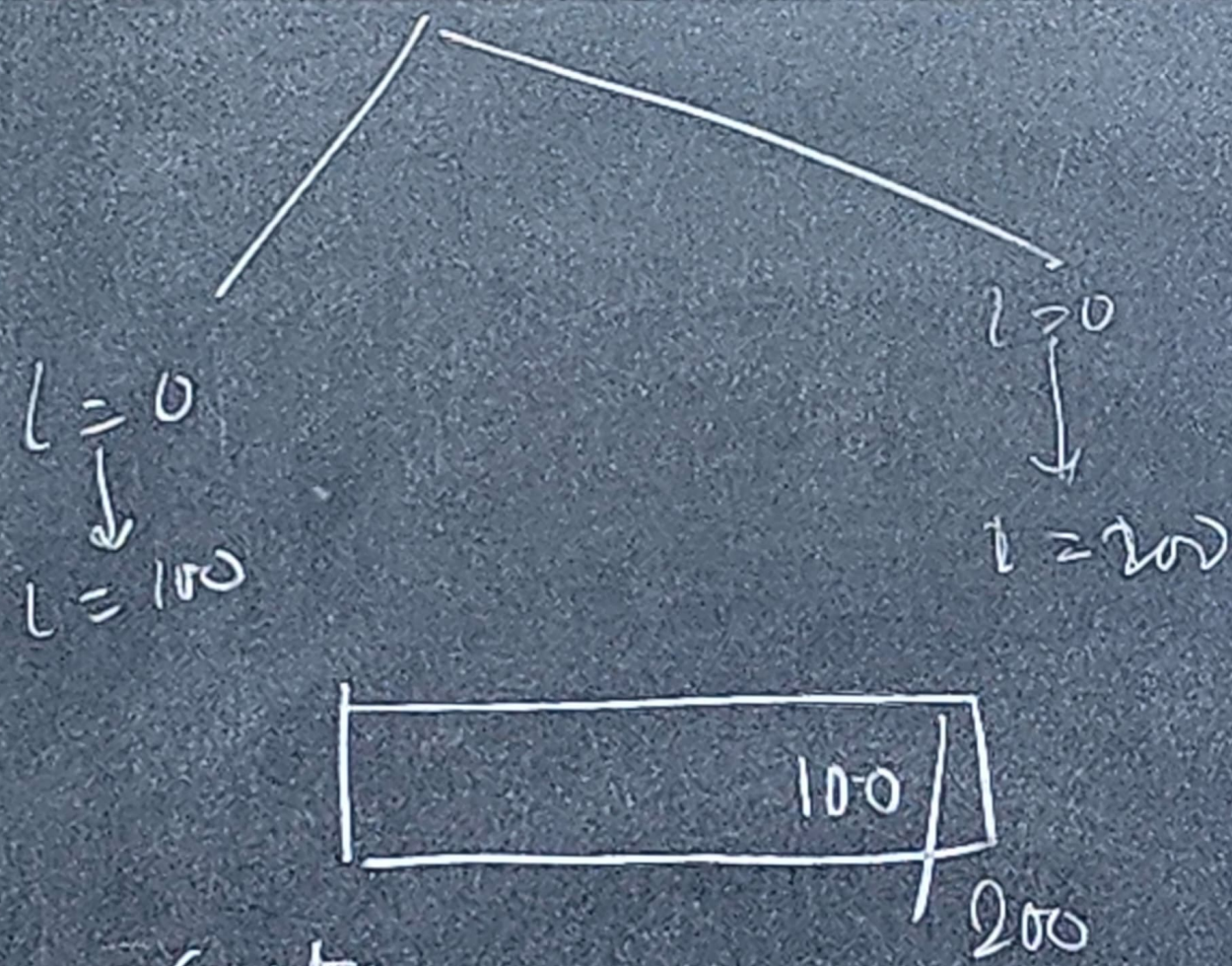
kolin@mosaic ~/col7001

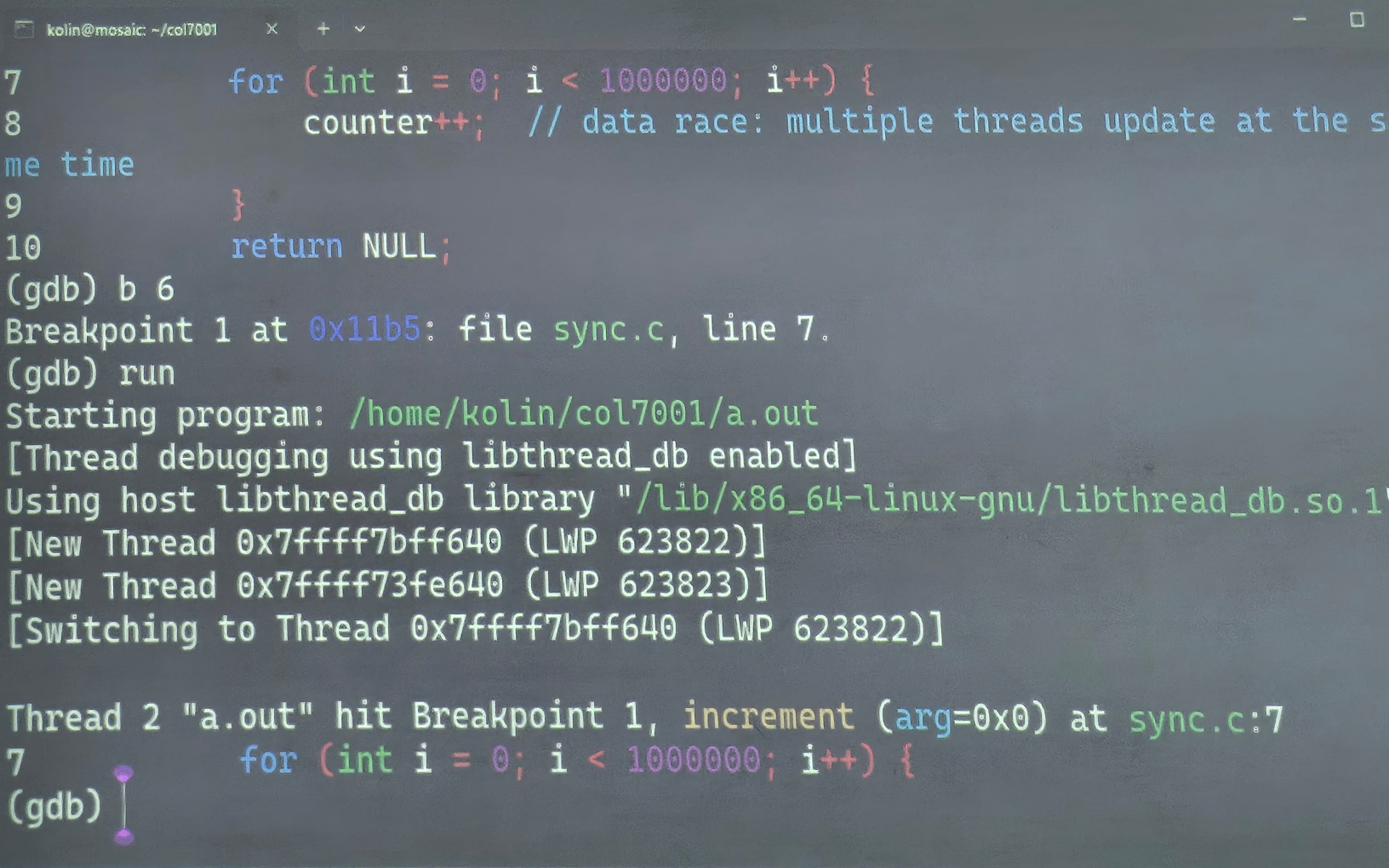
```c
int counter = 0;  // shared variable

void* increment(void* arg) {
    for (int i = 0; i < 1000000; i++) {
        counter++;  // data race: multiple threads update at the same time
    }
    return NULL;
}

int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("Final counter = %d \n", counter);
    return 0;
```

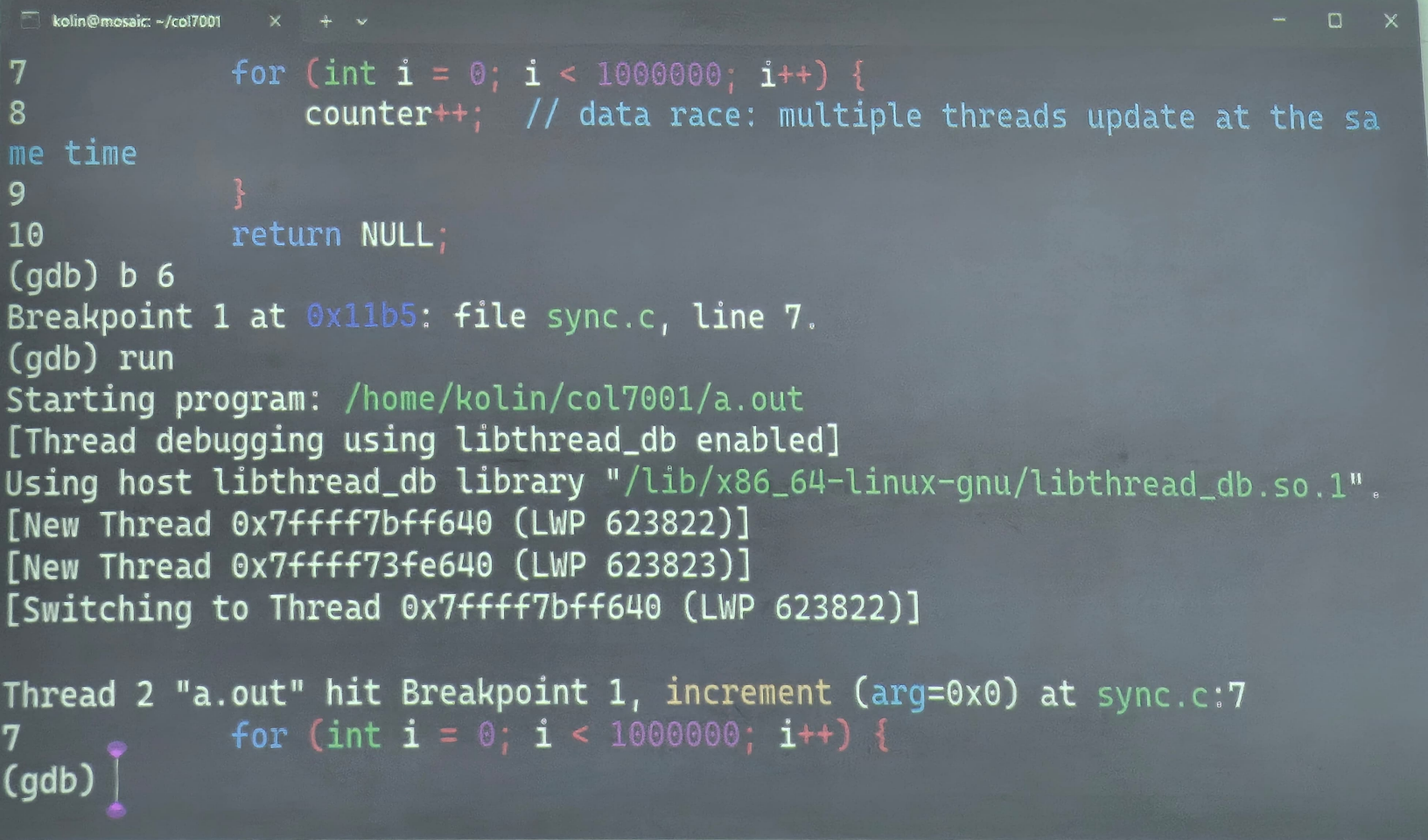
kolin@mosaic: ~/col7001

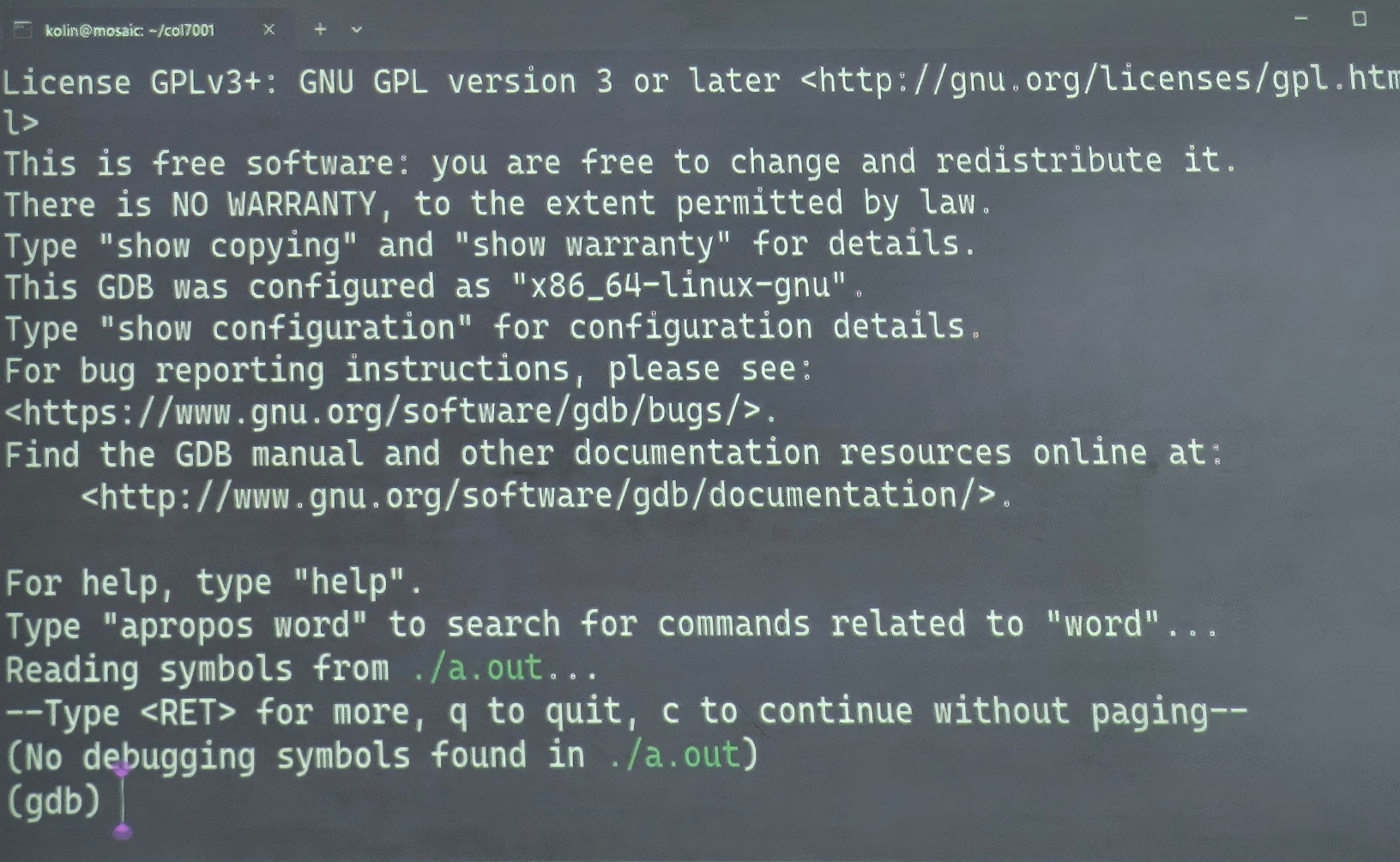
```
int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("Final counter = %d \n", counter);
    return 0;
}

(base) kolin@mosaic:~/col7001$
(base) kolin@mosaic:~/col7001$ gcc sync.c
(base) kolin@mosaic:~/col7001$ ./a.out
Final counter = 1019388
(base) kolin@mosaic:~/col7001$ ./a.out
Final counter = 1049159
(base) kolin@mosaic:~/col7001$
```

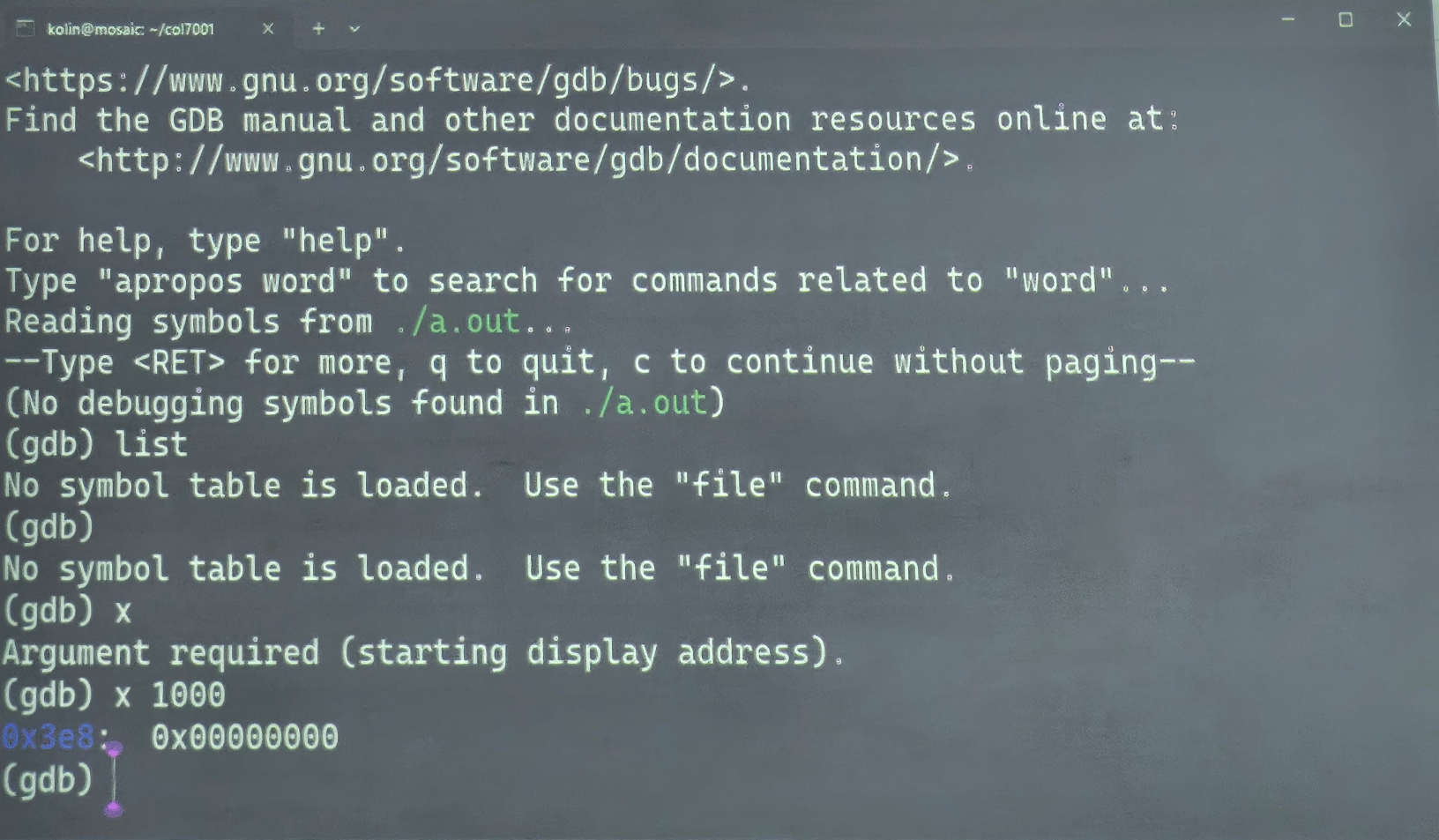

L = 0
L = 100
L = 0
L = 200
100
200
CAMERA


i = 0
i = 100
i = 0
i = 200
100
200
Counter
Count++
load R0, Count
add R0, R0, 1
store R0, Count
bne


kolin@mosaic: ~/col7001
7 for (int i = 0; i < 1000000; i++) {
8 counter++; // data race: multiple threads update at the s
me time
9 }
10 return NULL;
(gdb) b 6
Breakpoint 1 at 0x11b5: file sync.c, line 7.
(gdb) run
Starting program: /home/kolin/col7001/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1
[New Thread 0x7ffff7bff640 (LWP 623822)]
[New Thread 0x7ffff73fe640 (LWP 623823)]
[Switching to Thread 0x7ffff7bff640 (LWP 623822)]
Thread 2 "a.out" hit Breakpoint 1, increment (arg=0x0) at sync.c:7
7 for (int i = 0; i < 1000000; i++) {
(gdb)

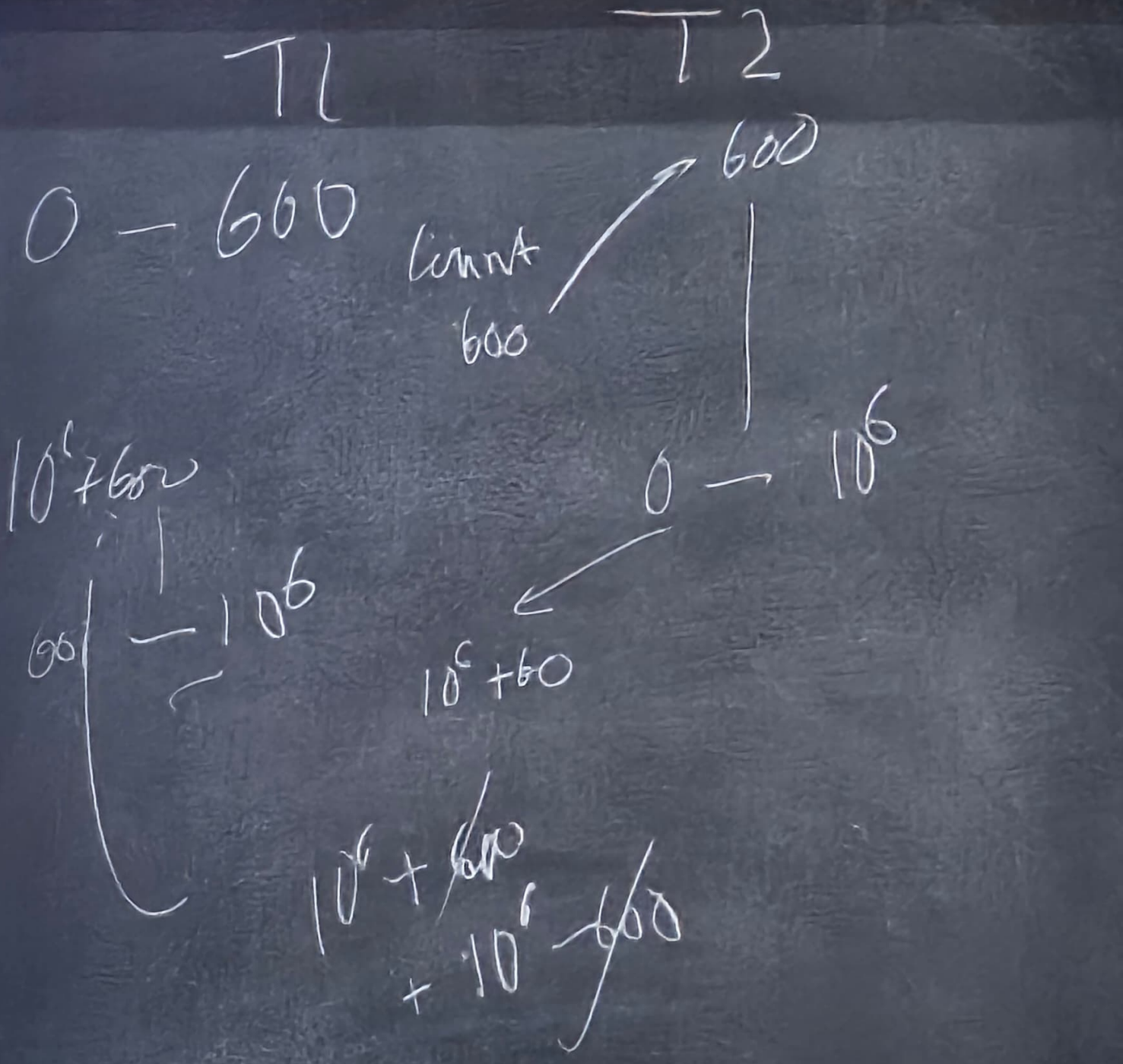

kolin@mosaic: ~/col7001
7 for (int i = 0; i < 1000000; i++) {
8 counter++; // data race: multiple threads update at the sa
me time
9 }
10 return NULL;
(gdb) b 6
Breakpoint 1 at 0x11b5: file sync.c, line 7.
(gdb) run
Starting program: /home/kolin/col7001/a.out
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x7ffff7bff640 (LWP 623822)]
[New Thread 0x7ffff73fe640 (LWP 623823)]
[Switching to Thread 0x7ffff7bff640 (LWP 623822)]
Thread 2 "a.out" hit Breakpoint 1, increment (arg=0x0) at sync.c:7
7 for (int i = 0; i < 1000000; i++) {
(gdb)


kolin@mosaic: ~/col7001
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.htm
l>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
--Type <RET> for more, q to quit, c to continue without paging--
(No debugging symbols found in ./a.out)
(gdb)


kolin@mosaic: ~/col7001
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
--Type <RET> for more, q to quit, c to continue without paging--
(No debugging symbols found in ./a.out)
(gdb) list
No symbol table is loaded. Use the "file" command.
(gdb)
No symbol table is loaded. Use the "file" command.
(gdb) x
Argument required (starting display address).
(gdb) x 1000
0x3e8: 0x00000000
(gdb)

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

int counter = 0;  // shared variable

void* increment(void* arg) {
    for (int i = 0; i < 1000000; i++) {
        counter++;  // data race: multiple threads update at the same time
    }
    return NULL;
}

int main() {
    pthread_t t1, t2;
    pthread_create(&t1, NULL, increment, NULL);
    pthread_create(&t2, NULL, increment, NULL);
    pthread_join(t1, NULL);
```

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

int counter = 0;  // shared variable
pthread_mutex_t lock;

void* increment(void* arg) {
    for (int i = 0; i < 1000000; i++) {
            pthread_mutex_lock(&lock);
            counter++;  // data race: multiple threads update at the same t
ime
            pthread_mutex_unlock(&lock);

    }
    return NULL;
}
```


T1
T2
0 — 600
600
Limit
600
$10^6 + 600$
10^6
0 — 10^6
$10^6 + 60$
$10^6 + 600$
$+ 10^6 - 600$

```
(gdb) ds
Undefined command: "ds".  Try "help".
(gdb) disassemble
No frame selected.
(gdb) disassemble quit
No symbol table is loaded.  Use the "file" command.
(gdb)
quit
(base) kolin@mosaic:~/col7001$ vi sync.c c
2 files to edit
(base) kolin@mosaic:~/col7001$ vi syncL.c
(base) kolin@mosaic:~/col7001$ gcc syncL.c
(base) kolin@mosaic:~/col7001$ ./a.out ^C
(base) kolin@mosaic:~/col7001$ less syncL.c
(base) kolin@mosaic:~/col7001$ gcc syncL.c
(base) kolin@mosaic:~/col7001$ ./a.out
Final counter = 2000000
(base) kolin@mosaic:~/col7001$
```

```
}

int main() {
    pthread_t t1, t2;
    pthread_mutex_init(&lock, NULL);

    pthread_create(&t1, NULL, bad_increment, NULL);
    pthread_create(&t2, NULL, bad_increment, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);

    printf("Final counter = %d\n", counter);
    pthread_mutex_destroy(&lock);
g   return 0;
g
```

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

int counter = 0;
pthread_mutex_t lock;

void* bad_increment(void* arg) {
    pthread_mutex_lock(&lock);
    for (int i = 0; i < 1000000; i++) {
        counter++;
        if (i == 500000) {
            // Forgot to unlock and return here
            return NULL;  // BUG: lock never released → deadlock
        }
    }
    pthread_mutex_unlock(&lock);
    return NULL;
```

```
(gdb)                                    is loaded.  Use the "file" command.
quit
(base) kolin@mosaic:~/col7001$ vi sync.c c
2 files to edit
(base) kolin@mosaic:~/col7001$ vi syncL.c
(base) kolin@mosaic:~/col7001$ gcc syncL.c
(base) kolin@mosaic:~/col7001$ ./a.out ^C
(base) kolin@mosaic:~/col7001$ less syncL.c
(base) kolin@mosaic:~/col7001$ gcc syncL.c
(base) kolin@mosaic:~/col7001$ ./a.out
Final counter = 2000000
(base) kolin@mosaic:~/col7001$ vi syncB.c
(base) kolin@mosaic:~/col7001$ less syncB.c
(base) kolin@mosaic:~/col7001$ gcc syncB.c
(base) kolin@mosaic:~/col7001$ ./a.out
```

19-Sept

Major will have -ve Marking
[+1 correct ; -2 for incorrect]

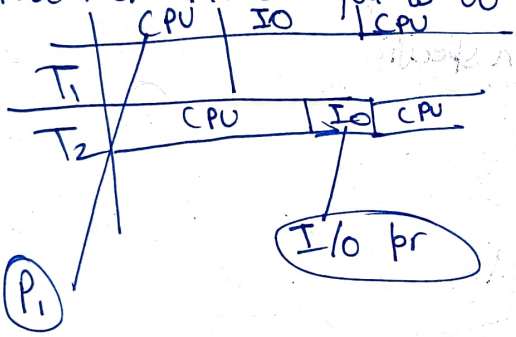
We now have lots of hardware

Cores, Processors, etc } interested in getting things done quickly

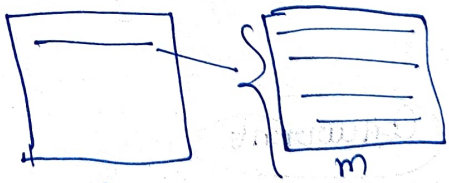
idea have Algo and do thing parallelly

we have { CPU bound → solving eq. — more CPU work
IO bound — more IO work
↳ watching movie is both
↳ typing is I/O

Processor Allows you to do multitasking / Multiprocessing



JOIN
[finding common row in two tables]
We find it based on one particular column or set of columns



Can be done in parallel
if 1 can be done
then all can be done
So it can be done in $O(n^2/p)$

Parallel
- Process (will require process/context switch which is more expensive) (write multithreaded program)
- thread (we use this because we are using common memory space) → Parallel JOIN

Is python program multi threaded?

Python vs Java VMs?

Java has Garbage collection its done while prog. is running on objects not been referenced

So there must be atleast one more thread doing that.

(It has to be in parallel context)

There must be one more thread, say your Java program Crashes, can you still stay in JVM

Suppose working in linux you are in user space, when prog crashes there you don't go to kernel you stay in user space

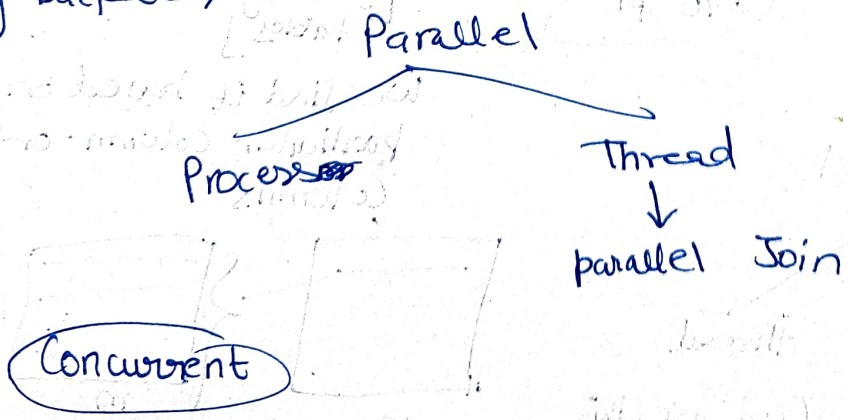
public static void main()

↑ method ↑ Data type

Suppose Java want to return Something like main()

Value of last prog stored in specific Register

Coming back to,



I can write a prog and make it Concurrent (Run at same time)

Concurrent ≠ parallel

↑

It does not make any assumption

Is python program multi threaded?

Python vs Java VMs?

Java has Garbage collection it's done while prog. is running on objects not been referenced

So there must be atleast one more thread doing that.

(It has to be in parallel context)

There must be one more thread, say your Java program Crashes, can you still stay in JVM

Suppose working in linux you are in user space, when prog crashes there you don't go to kernel you stay in userspace

public static void main()

↑ method ↑ Data type

Suppose Java want to return Something like main()

Value of last prog stored in specific Register

Coming back to,

Parallel

Process

Thread

↓

parallel Join

Concurrent

I can write a prog and make it Concurrent (Run at same time)

Concurrent ≠ parallel

↑

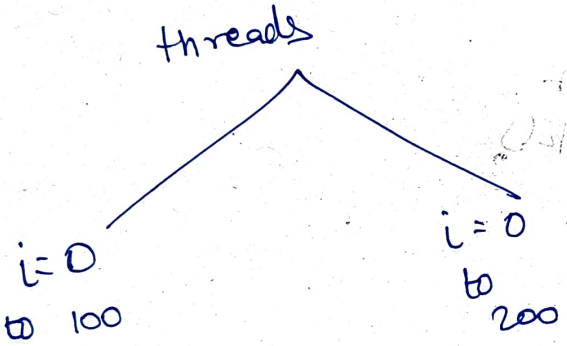
It does not make any assumption that hardware is concurrent

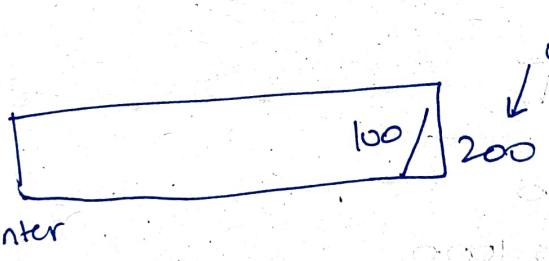
```
int count=0;
inc( )
{ for (i=1.... 1000)
  count++;
}
main()
{
  inc(); inc();
  printf(count)
}
```

```
int count=0;
inc()
{
for ( i = ____ 1000)
  count++;
}
main()
{
thread A
thread B
  printf(count)
}
```

Data race

Value will be between 1000 - 2000






```
load  R0 count
add   R0 R0 1
Store R0 count
bne
```

we can optimise this to :-

```
load  R0 count
add   R0 R0 1
bne
Store R0 count
```

If each thread was working independently we would have got 1000.

Moment we shared, ~~the~~ we get something more than ~~1~~ 1000 to 2000

to make it correct we could put lock before Count ++

```
int inc()
  {
   for( i=0 ____ 1000)
   }
     lock ()
      Counter++
      releaselock();
    }
  return Null
  }
```

T_1	T_2
0 - 600	→ 600
count = 600	0 to 1000
	↓
601 to 1000	1600
← 1600	
= 2000	

(Worked on code)

if lock is above for and we somehow exit before then it causes deadlock/ bug

Ex

```
~~inc()~~ inc()
{
  lock()
  for (i —  1000)
  {
    count++;
    if (i=500)
    { return ;
    }
  }
  release lock()
```

Thread 1 holds it and thread 2 never gets chance so its seems like deadlock but its a bug

How will we know prog went in deadlock?

OS gave us the lock, whatever OS gives should be ~~as~~ available in ——— (will discuss next week) (missed word)

C++ first line of code is main why?

↳ everything is object, every object has a constructor

memory allocation must happen.

Constructor function should be called

Compiler did memory allocation

Constructor is code running outside main

if happens in C++ has to be in Java