10/12 Threads

Concurrency

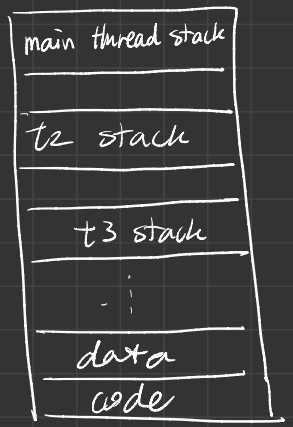
→ All about the structure

- divide a task into independent tasks
- open house example
  → greeter, checkins, booth, security, food
- helps manage the complexity & opens up opportunities to execute tasks simultaneously (parallelism)

Threads (mechanism for concurrency)

→ a unit of execution / task
- segnence of instructions

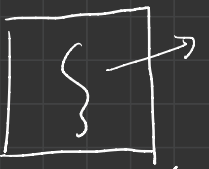
→ process = a container of resources
(VAS, fds, other OS ↘)
+ threads (executions)



VAS for multithreaded process

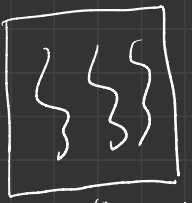
✡ per thread stack

✡ everything else is shared.



needs pc, sp, registers to have an execution

single threaded process



multithreaded process

→ Unit of scheduling.

→ runs in the process VAS.

→ 2 kinds of threads

① Kernel threads [managed and scheduled by the kernel]

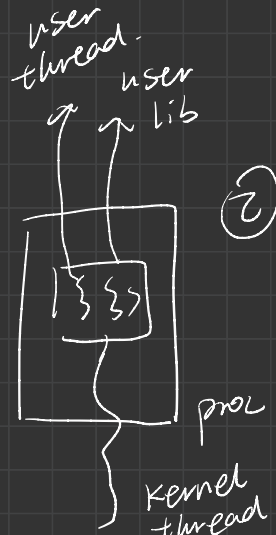
→ pthreads !

→ Thread control Block

{ tid, sp, registers, ptr to PCB }

→ Thread Life Cycle = Process Life Cycle

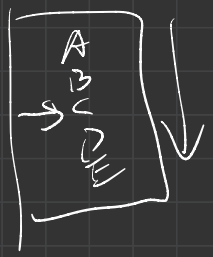
→ context switch = switch from one thread to another



② User threads (managed by user lib)

→ lightweight alternative, multiplexed on top of kernel threads

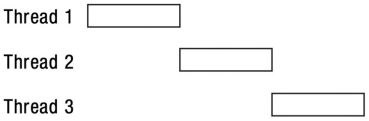
Programming / Execution Model of Threads.



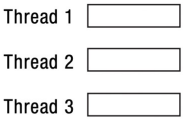
Programmer's View	Possible Execution #1	Possible Execution #2	Possible Execution #3
.	.	.	.
.	.	.	.
.	.	.	.
x = x + 1;	x = x + 1;	x = x + 1;	x = x + 1;
y = y + x;	y = y + x;	.....	y = y + x;
z = x + 5y;	z = x + 5y;	Thread is suspended.	.....
.	.	Other thread(s) run.	Thread is suspended.
.	.	Thread is resumed.	Other thread(s) run.
.	.	.....	Thread is resumed.
		y = y + x;	.....
		z = x + 5y;	z = x + 5y;

3 CPUs

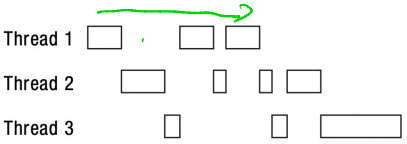
One Execution



Another Execution



Another Execution

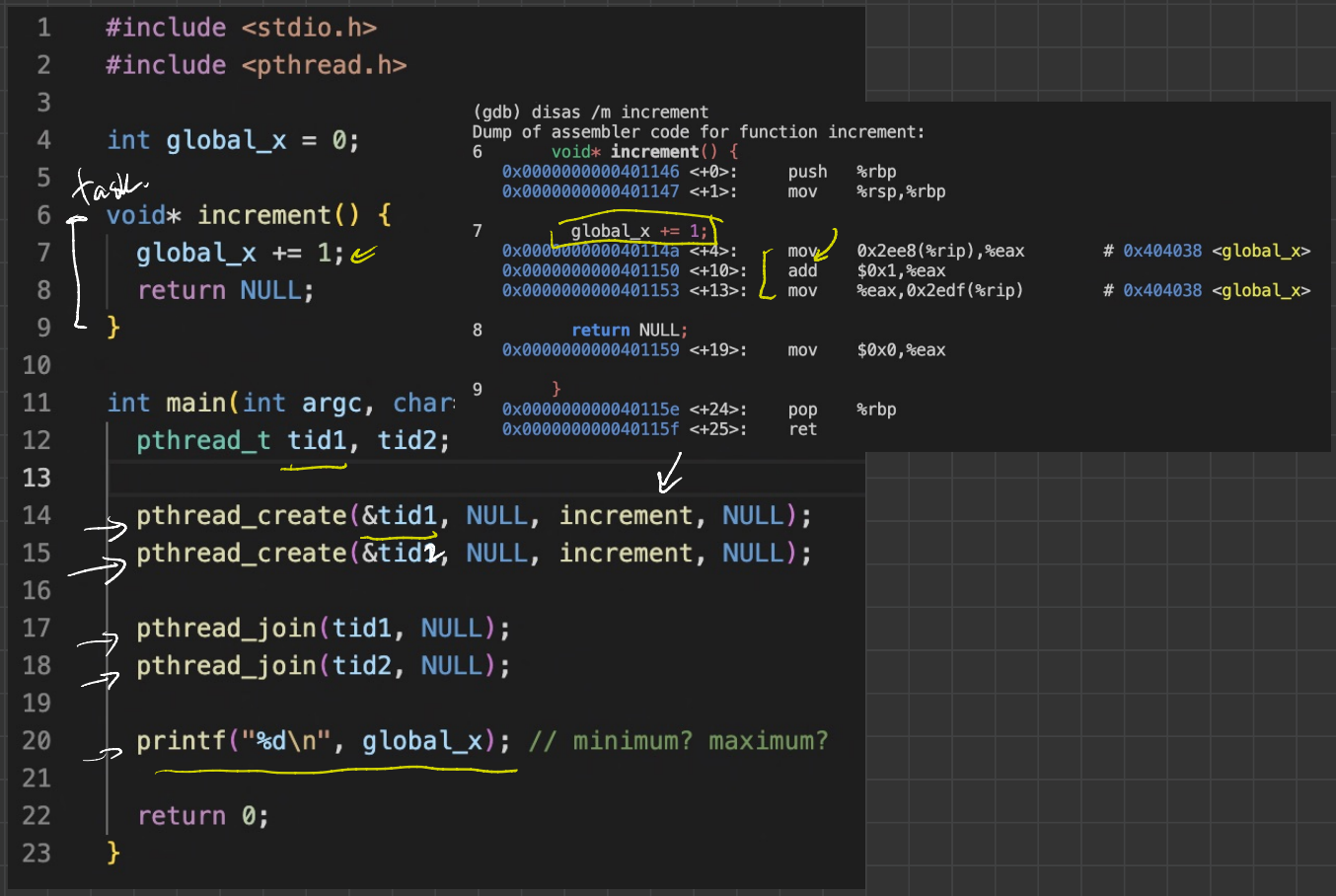


Threads API

pthread_create = Spawn

pthread_join = wait

pthread_exit = exit



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

int global_x = 0;

void* increment() {
  global_x += 1;
  return NULL;
}

int main(int argc, char
  pthread_t tid1, tid2;

  pthread_create(&tid1, NULL, increment, NULL);
  pthread_create(&tid2, NULL, increment, NULL);

  pthread_join(tid1, NULL);
  pthread_join(tid2, NULL);

  printf("%d\n", global_x); // minimum? maximum?

  return 0;
}
```

```
(gdb) disas /m increment
Dump of assembler code for function increment:
6       void* increment() {
   0x0000000000401146 <+0>:     push   %rbp
   0x0000000000401147 <+1>:     mov    %rsp,%rbp

7         global_x += 1;
   0x000000000040114a <+4>:     mov    0x2ee8(%rip),%eax        # 0x404038 <global_x>
   0x0000000000401150 <+10>:    add    $0x1,%eax
   0x0000000000401153 <+13>:    mov    %eax,0x2edf(%rip)        # 0x404038 <global_x>

8         return NULL;
   0x0000000000401159 <+19>:    mov    $0x0,%eax

9       }
   0x000000000040115e <+24>:    pop    %rbp
   0x000000000040115f <+25>:    ret
```