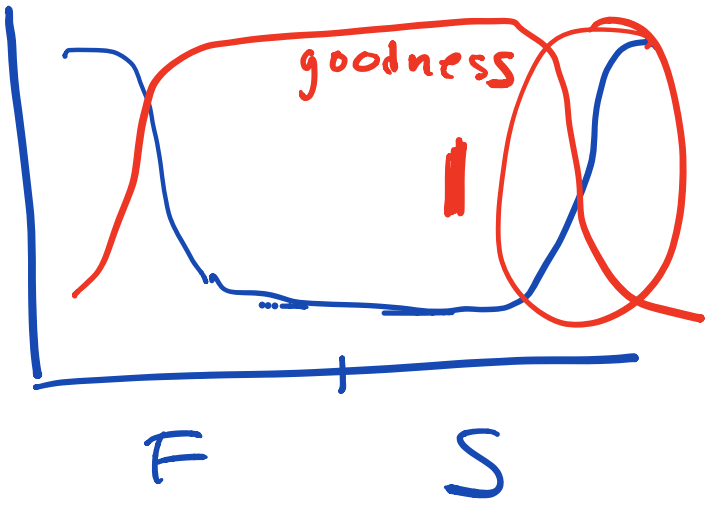
Today

weather

=> ignore good weather



OS: virtualization

=> interviews [concurrency } <- hardest

persistence

=> Today

-> intro: concurrency problems

-> code: examples

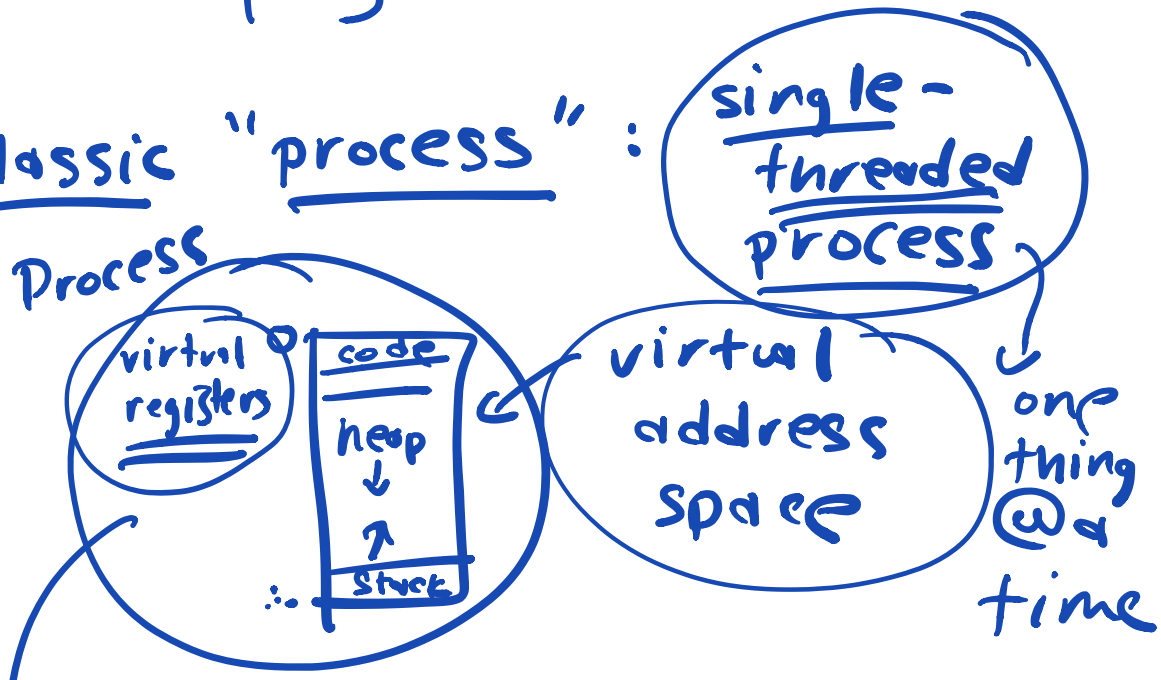
} lecture

{ C: common mistakes and other fun } Discussion

Concurrency: ~~X~~ 2 3

=) multi-threaded program

classic "process": single-threaded process



→ instruction ptr (PC), stack ptr, general purpose registers

OS: could switch

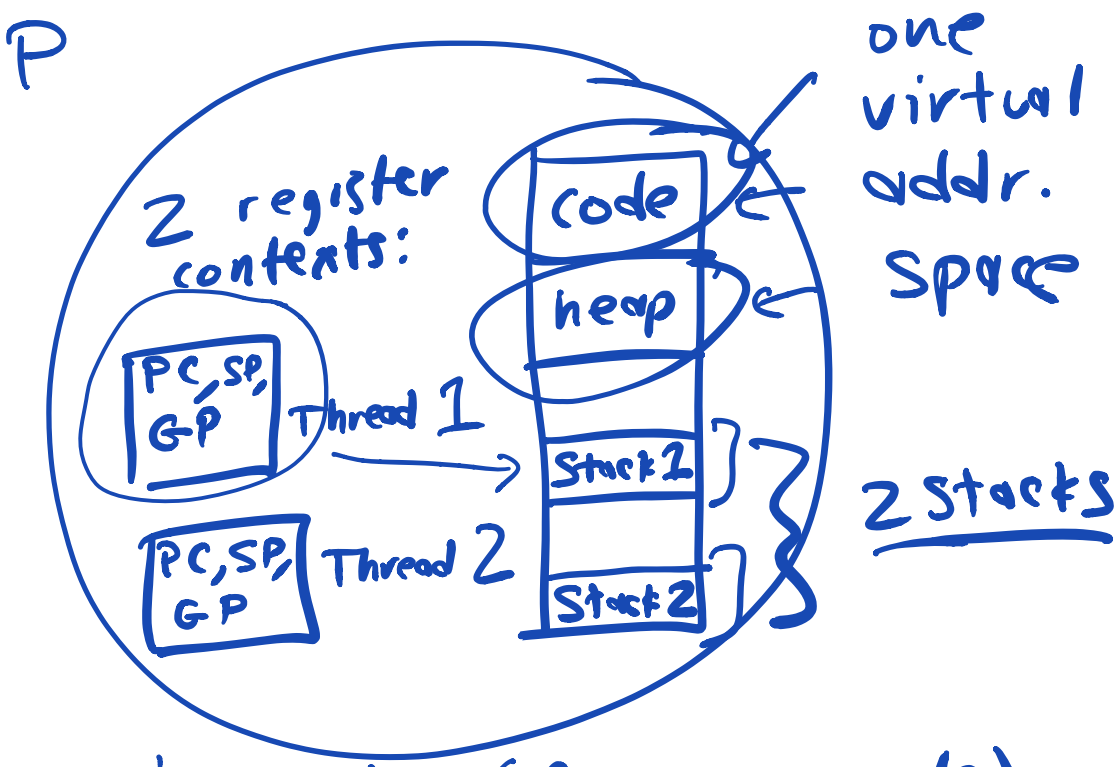
to another process

→ save registers (old),
restore registers (new)

→ switch page tables

want: many "activities" going on w/in process at same time:

⇒ multi-threaded process



2 threads (for example)

Thread 1 Thread 2 Thread 1

1 CPU: 

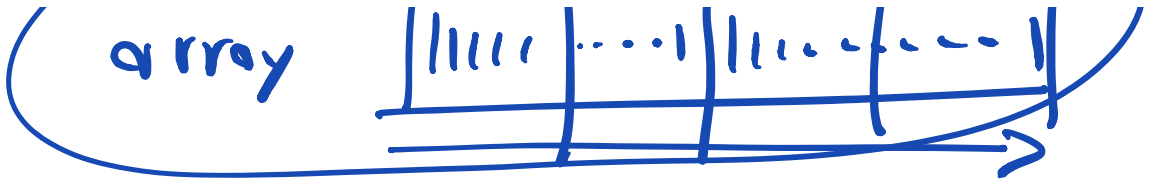
Context switch:
between Threads vs. Processes

→ both: save registers (old),
restore regs (new)

→ only switching between
processes: switch
address spaces
(i.e., change PTBR)

Why multi-threaded
process?

⇒ Parallelism
(to run prog faster)



4 CPUs:

=> create 4 threads

=> give each chunk of array + do work in parallel

=> Overlap (concurrent)

web server

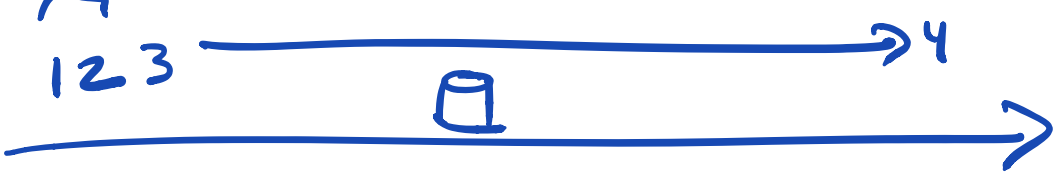
=> 1) get request

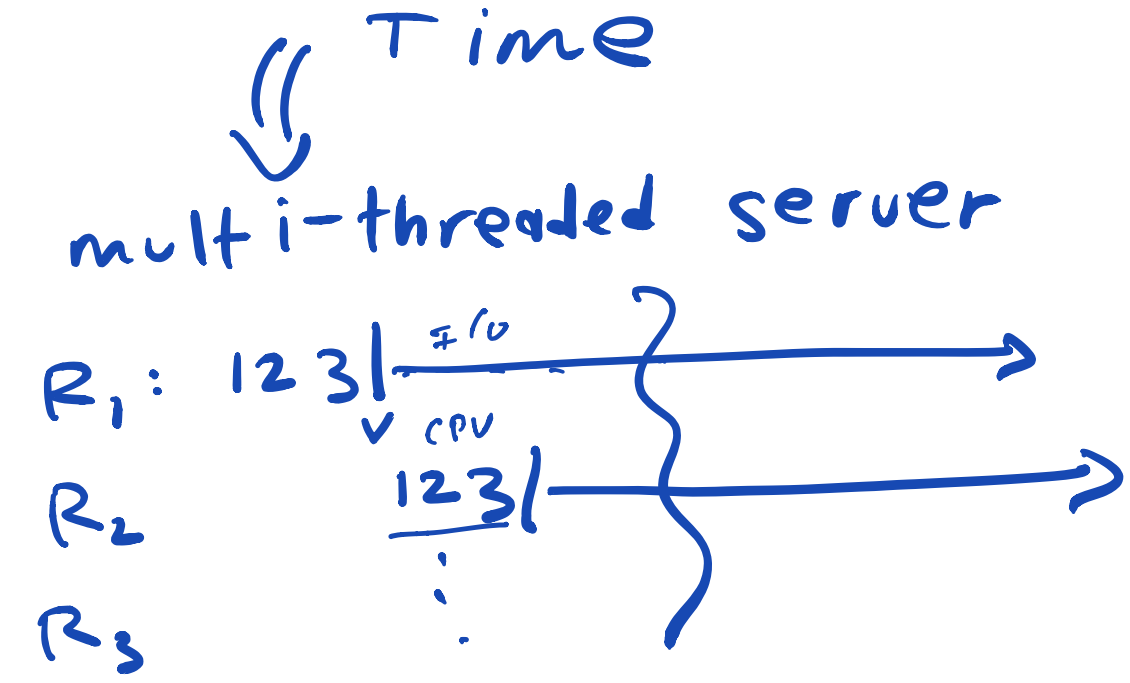
=> 2) parse

=> 3) read data from disk

=> 4) return data

(slow)
10 ms





Threads + Trust Model:

Threads w/in same process trust eachother