Midterm happens in class this Thursday: 75min, 1 cheatsheet (2, sided, letter-sized, NO SCREENSHOT, please submit cheatsheet)

bring pen/pencil, no calculator/phones

please write answers i could understand , if you find anything unclear, write your assumptions

read the entire exam before start writing

do not write in the back of the exam

coding questions:

all the functions you need are provided

Topics we covered

- Process: process view of the machine
    - one of key abstractions
    - each process thinks they have exclusive access to both cpu and memory
    - and this is a illusion
    - memory layout, what exactly is the "view of the memory"
        - text/code, data, heap, stack, environment
        - what are the operations need to be done before/after function call
        - %rbp, %rsp, %rip ...
    - PCB: process control block
        - process state, counter, registers, memory management, etc.
        - OS maintain all these PCB
    - how a process is created? system calls
        - fork, exec, wait, ...
        - user mode ->(system call) kernel mode
        - the difference between system call and functoon calls
        - open, read, write, close,...
- Concurrency
    - to run multiple processes at the same time: fork + exec
    - to run multiple thread within a single process: threading library
    - Race conditions: data races

  - how do we (try our best) to avoid race conditions:
    - mutex: only one thread access a critical section at a time
    - init, lock, unlock
    - disable interrupts, spinlocks, peterson algorithms, ...
    - condition variables: wait for a condition to become true
    - wait, signal, broadcast
    - wait : while loop!
    - monitor (programming paradigm): Mike Dahlin's coding standard for concurrency programming
      - all methods calls are protected by a mutex
  - Deadlock
    - mutual exclusion, hold and wait, no preemption, circular wait
    - how to avoid deadlock
      - enforce a partial order on the lock you acquire
- Therac-25 Case Study
- Scheduling
  - preemptive and non-preemptive scheduling
  - Metrics: turnaround time, waiting/response time, system throughput, fairness
  - algorithms
    - FCFS
    - SJF, STCF
    - RR
    - MLFQ
    - fair share schedulers
- Virtual memory
  - purpose: memory protection, illusion of large memory, make memory usage more efficient
  - address translation: VM address -> PM address
  - key structure: page tables
    - multi-level page table
    - translation process from L1 -> L4
    - TLB: what policies do we use
  - Page fault
    - invalid access
    - memory is not presented in ?