Worm
Quiz

Scheduling (MLFQ, proportional share)
Threads
Midterm Exam
Cooperative Scheduling

• Long-running processes give up CPU.
  ▪ A friendly task blocks repeatedly, e.g., by making system calls.
  ▪ yield system call
  ▪ Illegal operation e.g., dividing by zero or accessing an invalid page
Context Switch

• An OS piece of code that saves the context of a currently running process and restores the context of another process.

• Context consists of information such as
  – Register values e.g, program counter, stack pointer
  – Memory maps
  – Process ID
Context Switch

Process $P_1$ executes on the processor

After an interrupt, the kernel decides to dispatch a new process and initiates a context switch

Kernel stores process $P_1$'s execution context to its PCB in memory

Kernel loads process $P_2$'s execution context from its PCB in memory

Process $P_2$ executes on the processor
Lab

• Implement a cooperative scheduler

• After a task starts, it runs to completion (exits) or performs a blocking operation:
  – Wait for another task
  – Sleep for a fixed amount of time
  – Wait for user input
Lab

• Scheduler starts running another task
  – Scheduler checks if any of previously blocked task can unblock (e.g., user input is available)
  – Perform checking in the round robin way

• Each task has
  – a given function. The function runs when the scheduler starts the task.
  – a scheduling state e.g., ready, running
    ➢ The task is in the ready state when it’s created.
Lab

• Test the scheduler with the given test files
  – A test file consists of a main function that creates tasks and calls blocking functions.

• Play worm game after passing test files