Agenda for Today

1. Context Demo
2. Q&A
3. Real Schedulers: MLFQ, Lottery, Stride, and CFS
4. Wrap Up
Context Demo
Lab Questions

If we swap away from a task because it needs to sleep, how/when do we determine that we should swap back to the task?

We only swap to another task when the current task blocks/exits. Each time the task we're currently running blocks/exits, we need to look for a task to swap to. As we loop over other tasks, we have to check to see if they are now runnable (e.g. they've slept long enough).

Why is there a sleep_ms function if we aren't supposed to use it?

You could call sleep_ms if you can't find a task to swap to, just to pause briefly. Or you can just loop.
Lab Questions

When you are looking for another task to pick when the current one sleeps/exits, how do we know we'll actually get to all the tasks?

We don't have to worry too much about this in our implementation, but it would be a good idea to start checking other tasks at the next index past the current task.

Is it okay to modify the task_create function?

Yes. You probably need to set the state of each new task inside task_create.
Real Schedulers
How would you describe MLFQ to a 161 student?

We have three queues of different priority. The top queue is the highest priority. All new tasks go in the top queue. Tasks are chosen in round-robin fashion, choosing only from the highest non-empty priority level. When a task is run, it is given a length of time to run (a quantum). If the task blocks before its quantum is exhausted, it can stay at its current priority. Otherwise the task is downgraded. That way, short-running tasks run in the high priority queue, and longer-running tasks run with lower priority. Periodically, tasks are bumped back up to the top priority queue.
Lottery Scheduling (AM)

How would you describe lottery scheduling to a 161 student?
Each process has some lottery tickets. The more tickets it has, the higher priority the process is.
The scheduler picks a random number, and runs the process holding that numbered ticket for some quantum.
Analogy: Hunger Games.
How would you describe stride scheduling to a 161 student?

Processes still have a number of tickets. Instead of picking tickets at random, each process is assigned a *stride*, which is inversely proportional to its number of tickets.

Each process has a counter that starts at zero. When a process runs, we add its stride to its counter.

We choose the next process to run by finding the process with the lowest counter.

**Why?** Like lottery scheduling, this adds the idea of priority. Stride scheduling is like lottery scheduling for people who don't like randomness. Guarantees fairness over short time scales, unlike Lottery Scheduling.
How would you describe the completely-fair scheduler to a 161 student?

Weighted round-robin with time slices that change to fit all the jobs' proportions into a fixed time (called scheduler latency).

There's a minimum granularity parameter. We never run a process for less than that time.

Priorities for processes are determined by weights, which come from a process' nice value. Jobs that are less nice are given a higher priority, so they accumulate vruntime more slowly and can run more often.
Aesthetics (AM)

Which scheduler (MLFQ, Lottery, Stride, or CFS) do you like? Why?
MLFQ: 5
Lottery: 4
Stride: 8
CFS: 8
Aesthetics (PM)

Which scheduler (MLFQ, Lottery, Stride, or CFS) do you like? Why?
MLFQ: 8
Lottery: 4
Stride: 6
CFS: 2
Wrap Up
Reminders

Lab
The worm lab is due on Wednesday

Reading
Read about Concurrency and the Threads API for Monday

Mentor Office Hours
Sunday 1–4pm
Tuesday 8–9pm
Wednesday 8–9pm