COS 417 Precept 2

Schedulers

Schedulers

- A vital part of an OS, responsible for handling how to run multiple processes on the CPU seemingly simultaneously

- Types of Schedulers
  - Round Robin (default xv6 scheduler)
  - Proportional Share scheduler:
    - Lottery Scheduler
    - Stride Scheduler (proj 2)

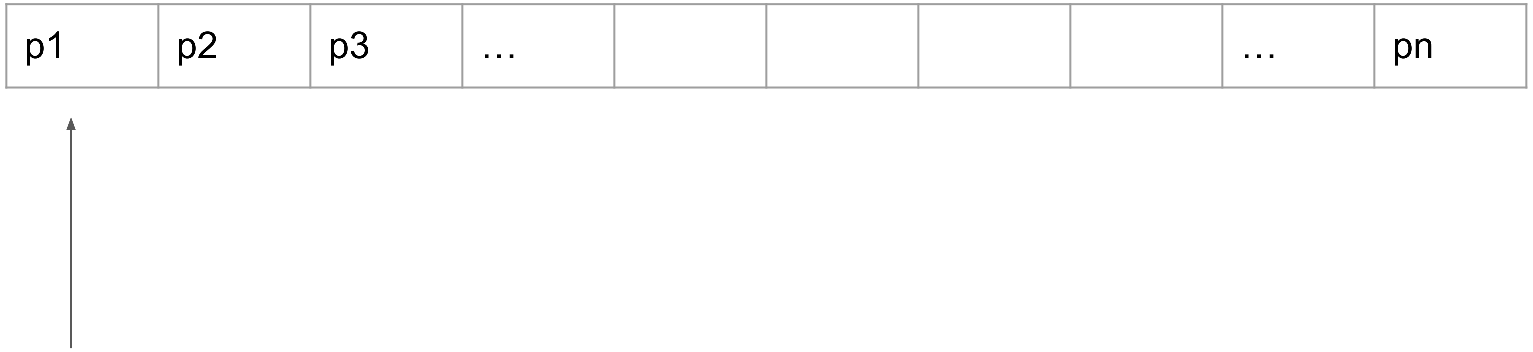
Link to chapter in book:
https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-lottery.pdf

Round Robin:

Each runnable process gets to run for some constant time slice/amount of ticks.

Tick 1

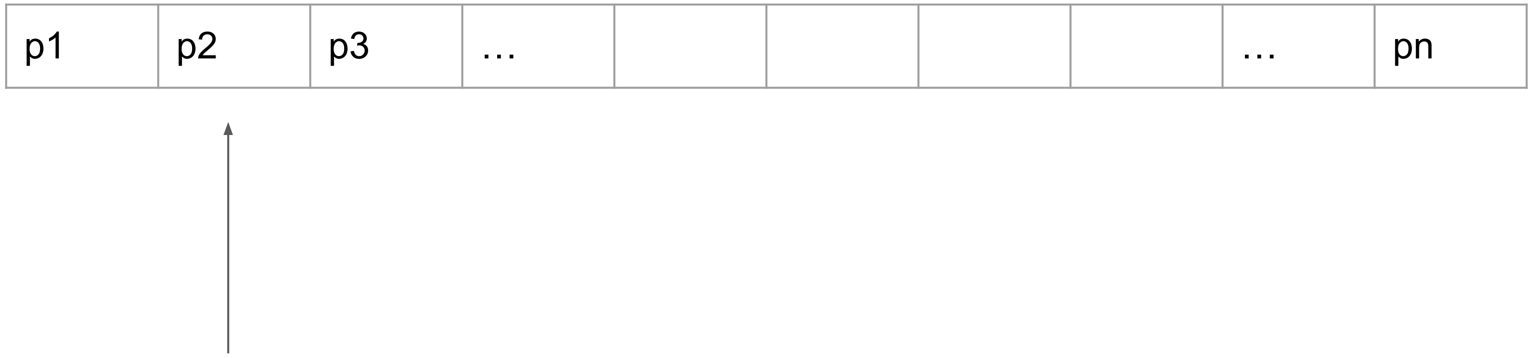
Process Queue:



Round Robin

Tick 2

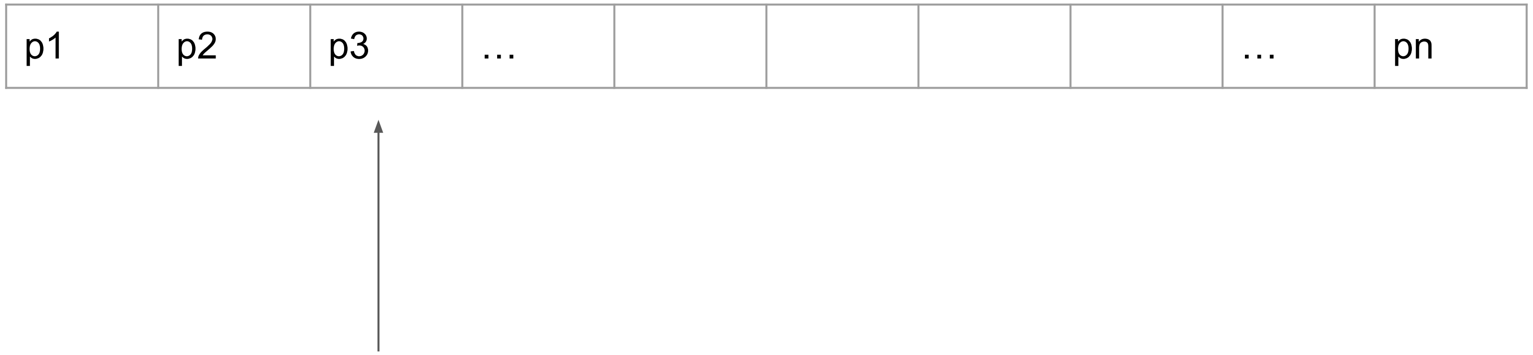
Process Queue:



Round Robin

Tick 3

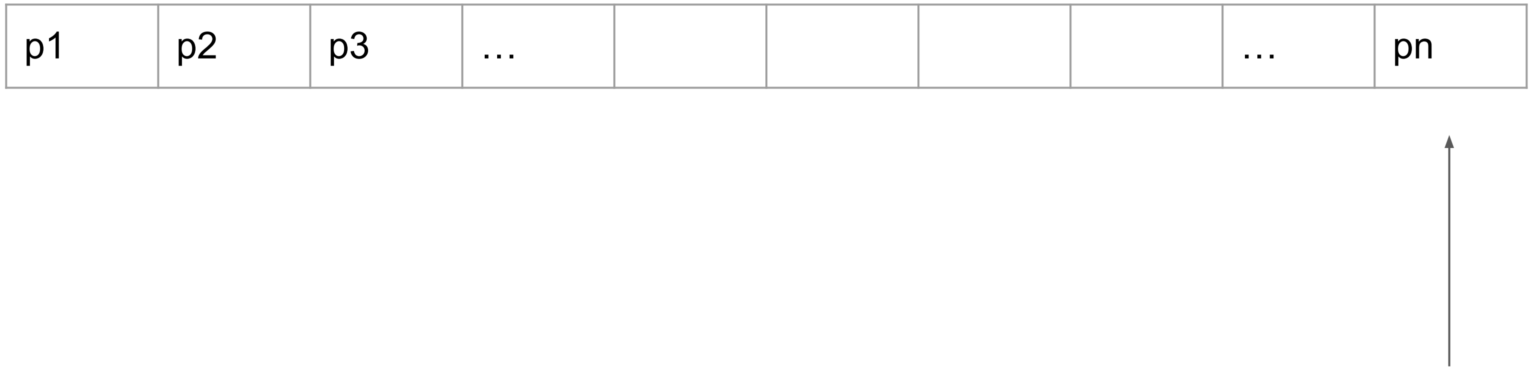
Process Queue:



Round Robin

Tick n

Process Queue:



Round Robin

Tick n+3?

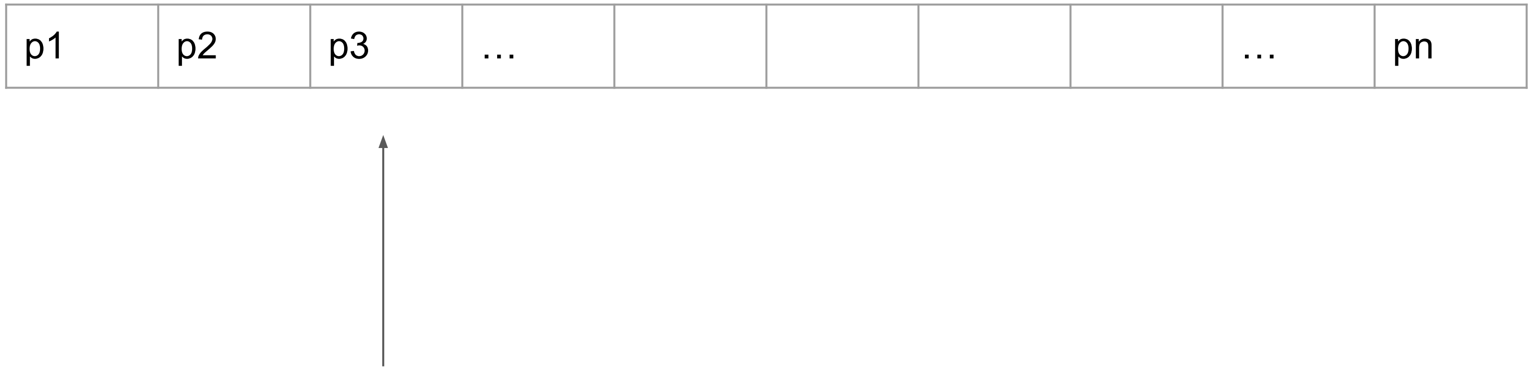
Process Queue:

p1	p2	p3	…					…	pn

Round Robin

Tick n+3

Process Queue:



Round Robin: potential issues?

- What if the time slice is too small/large?
- What if we want some processes to run more often?

Process Queue:

p1	p2	p3	…					…	pn

Proportional Share

- What if we want some processes to be prioritized, i.e. ran more often by the the scheduler than others?

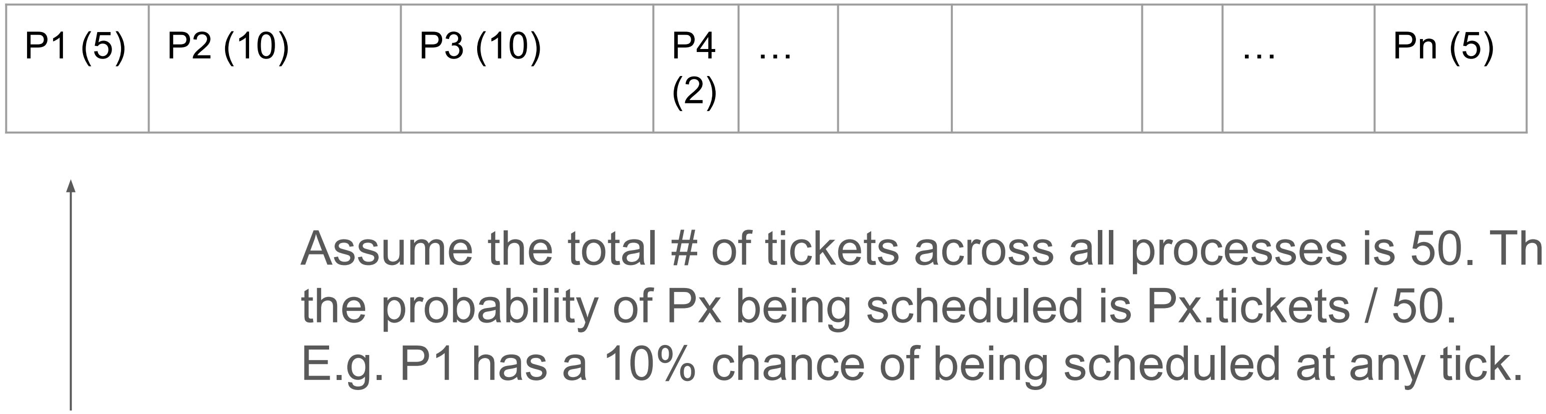
- Introducing **tickets**; where each process has a certain amount of tickets
- The more tickets, the more ‘prioritized’ the process is to run

- We’ll go over the following two types of proportional share schedulers:
    - Lottery
    - Stride

Lottery Scheduler:

The chance for each runnable process to run is proportional to the # of tickets it has. The # of tickets of each process is denoted inside the parentheses (not to scale).

Process Queue:



Assume the total # of tickets across all processes is 50. Then the probability of Px being scheduled is Px.tickets / 50.
E.g. P1 has a 10% chance of being scheduled at any tick.

Lottery Scheduler: potential issues?

- Inherent randomness. Can we get rid of that somehow?
- Not very predictable due to the above.

Process Queue:

P1 (5)	P2 (10)	P3 (10)	P4 (2)	…				…	Pn (5)

Stride Scheduler:

- Each runnable process has a # of tickets which determines its stride, specifically: stride = CONST / # of tickets
- Each runnable process also has a ‘pass’ value. A process that was just run will increment its ‘pass’ value by its stride value.
- At each tick, the scheduler chooses to run the process with the least ‘pass’ (with tie breakers).

- Objective of the scheduler is to make sure all runnable processes ‘keep up with each other’ and have the same pass value. If not, start running the processes who’s pass values are falling behind!

Assume CONST is 100

Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick 1

	Pass:
P1 →	0
P2 →	0
P3 →	0

Assume CONST is 100

Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick 2

	Pass:
P1	0
P2	20
P3	0

Assume CONST is 100

Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick 3

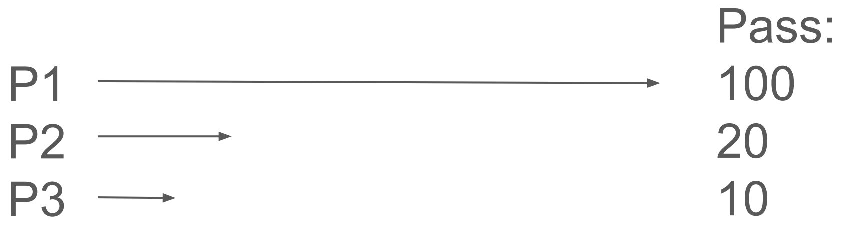
	Pass:
P1	0
P2	20
P3	10

Assume CONST is 100

Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick 4

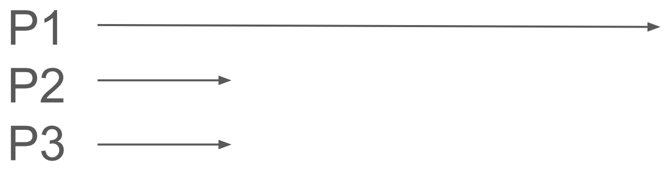


Assume CONST is 100

Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick 5



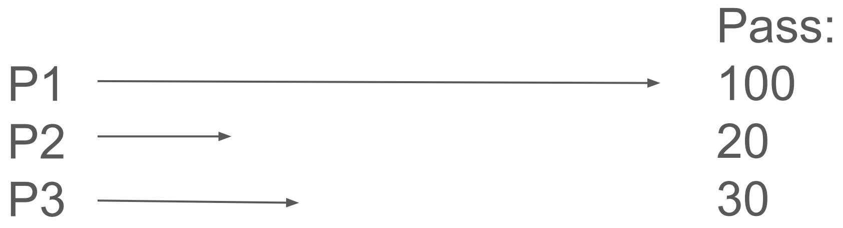
	Pass:
P1	100
P2	20
P3	20

Assume CONST is 100

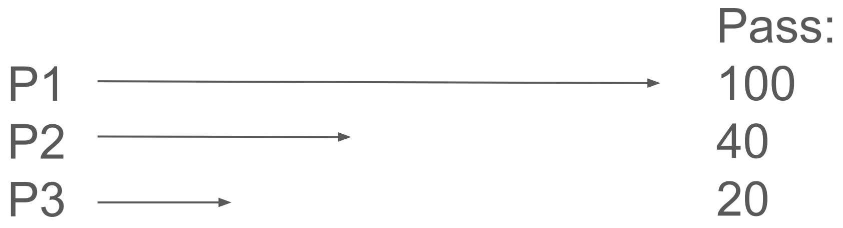
Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick 6



OR



Assume CONST is 100

Stride Scheduler

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

Tick n…eventually:

Pass:

P1 $\longrightarrow$ 100

P2 $\longrightarrow$ 100

P3 $\longrightarrow$ 110

…eventually, P2 and P3 will finally be scheduled enough to have a pass that's comparable to P1, finally giving P1 a chance to run soon.

Assume CONST is 100

Stride Scheduler: Determinism

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

We can easily tell how many times each process was scheduled, as well as in what order (given a starting point and tie-break rules)

E.g., given that at time N, all three processes have a pass value of 100. How many times did each process run since the start when all processes had 0 pass?

Assume CONST is 100

Stride Scheduler: Determinism

	Tickets	Stride
P1	1	100
P2	5	20
P3	10	10

We can easily tell how many times each process was scheduled, as well as in what order (given a starting point and tie-break rules)

E.g., given that at time N, all three processes have a pass value of 100. How many times did each process run since the start when all processes had 0 pass?

- P1: once
- P2: 5
- P3: 10
- (these just happen to be the same as the # of tickets because of how CONST is defined)