scheduling

scheduling = after context switch, which thread to run next, how long...

scheduler = determines which thread to run first from ready queue, OS component

↳ run whens = when a thread yields, exists, preempted or blocked on I/O, timer...

- preemptive scheduling = timer interrupt force context switch
- non-preemptive scheduling = process must yield / block voluntarily

* batch sche. = non-preemptive, no other jobs run if they block

* interactive sche = preemptive, other jobs do run if they block

dispatcher = gives control of the CPU to the process selected by the scheduler

↳ switching context → switching to user mode → jumping in the user program to restart it

* dispatch latency = time it takes stop one start another

scheduling goals = determining what to prioritize while scheduling

↳ CPU utilization: percentage of time that CPU is runnin user-jobs

↳ throughput ⇒ # processes completed per unit of time : 120 jobs → 1 minute ⇒ 2 jobs/sec

↳ turnaround time ⇒ the duration btw submitted and completed

↳ waiting time in the ready queue

↳ load average = # jobs in the ready queue

* while running high priority jobs prevent starvation of low priority jobs with deadlines (sometimes it is permitted)

first-come-first-served (FCFS)

↳ only used in batch scheduling, non-preemptive → no starvation

↳ in case of I/O of some job = can be switched to next job, put current to the end (not all versions do that, just waits)

A=13 B=4 C=4

A=13 B=13+4=17 C=13+4+4=21 } 51/3=17

turnaround time — avg

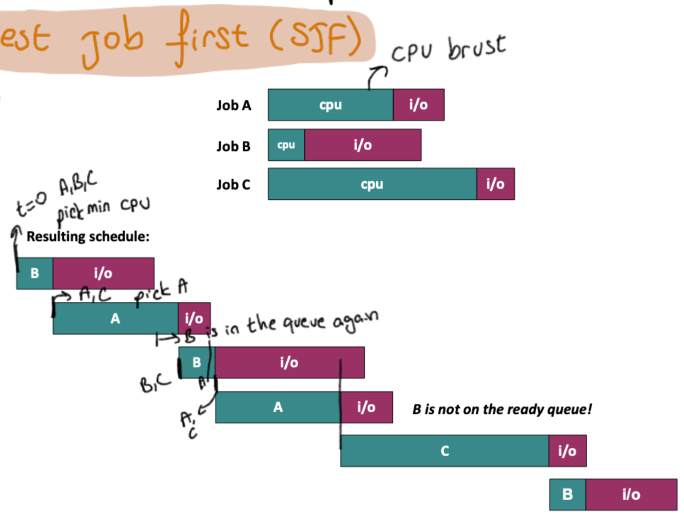
A=21 B=11 C=12 } 44/3=14..

robin-round (RR)

↳ FCFS with preemptive, CPU quantum (one block) (switch if i/o)

shortest job first (SJF)

↳



(mostly longer) CPU bound = CPU brust time (expected) > i/o time (A-C) → compiler, games (examples)

I/O bound = i/o time > CPU time (B) → web browser

↳ next process is run when the current terminates or I/O

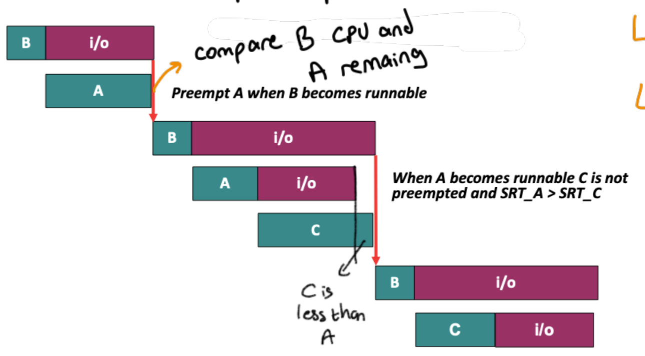
↳ non-preemptive, runs until it blocks for I/O

↳ procrastination (long ones done later)

↳ good for interactive programs (shorter waits for i/o)

shortest remaing time first (SRTF)

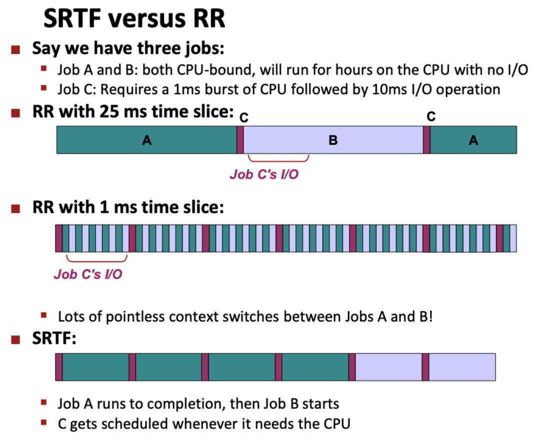
↳ SJF with preemptive



↳ if shorter CPU brust job becomes runnable, run it

↳ there is check every time there is new one in the ready queue, or exists, or i/o

	FCFS	RR	SJF	SRTF
Preemptive?	N	Y	N	Y



in a version

When is the scheduler called?	FCFS	RR	SJF	SRTF
Current process exits	Y	Y	Y	Y
Current process goes for I/O	N	Y	Y	Y
A new process is added	N	N	N	Y
Timer interrupt goes off	N	Y	N	N
A process returns from I/O	-	N	N	Y

priorities = in linux [0,99] each thread, with nice() can be adjusted (new is [-20,20])

multi-level feedback queues (MLFQ) = give higher priority to i/o bound jobs

↳ increase priority if short CPU, decrease if long CPU

priority inversion = C lock R, A uses R, priorities A > B > C

↳ if C starts first B executes before A like it has high priority until C terminates

solution = priority inheritance, execute C before B since R is used by high priority A

↳ C inherits A's priority

lottery scheduling = randomized priority schedule, higher ones have more tickets → higher priority

↳ A[0,29] (30) B[30,40] (10) C[40-99] (60) priorities = C > A > B round 1 45 → execute C..