Scheduling policies for interactive systems

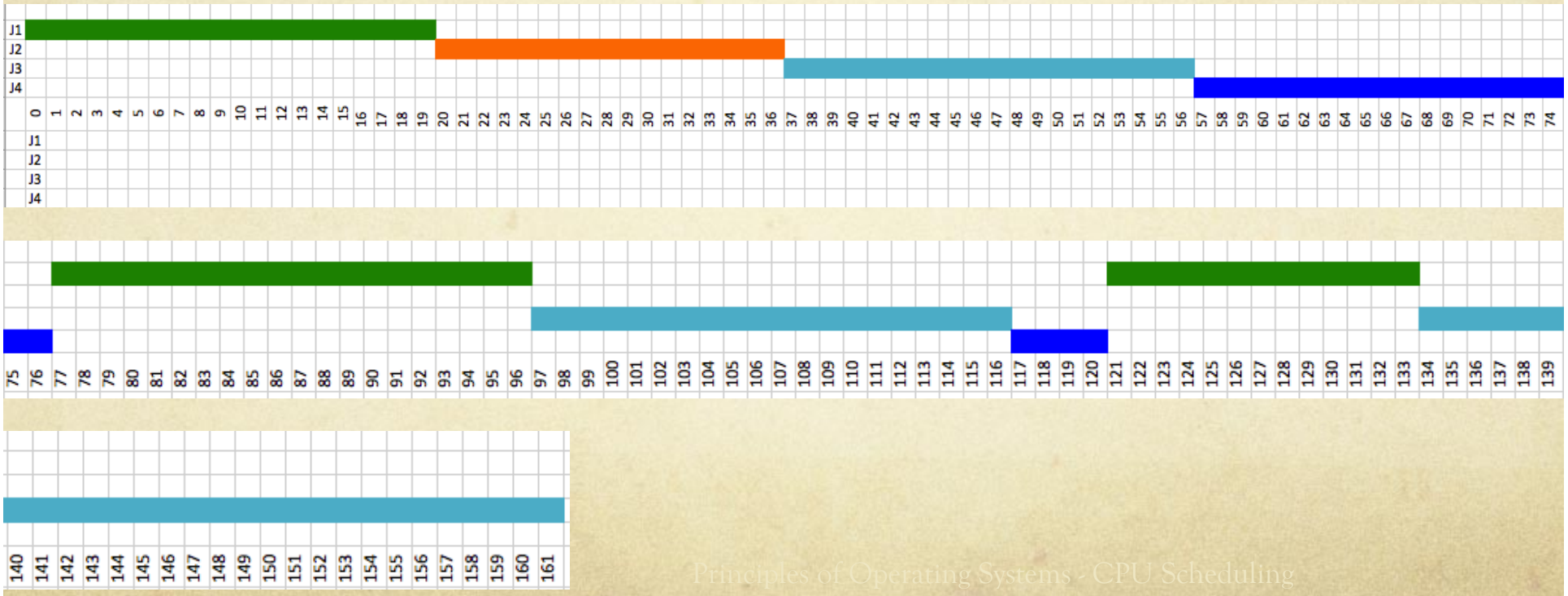
Round Robin (*RR*)

- Active jobs kept in *ready* queue
- Each job gets small unit (*quantum*) of CPU time
  - Quantum usually 10-100 milliseconds
- After quantum, job is *preempted* & next job in queue is scheduled
- *Pros*: simple; ensures fairness
- *Cons*: assumes all jobs are equally important
- Need to choose quantum *carefully*!
  - Too *short* → too much switching overhead
  - Too *long* → system not responsive

Round Robin Example

Quantum = 20

Job	CPU burst
J1	53
J2	17
J3	68
J4	24



Priority based scheduling

- *Priority* value (integer) associated with each job
- Job with *highest* priority is scheduled
- Schedule could be *preemptive* or *non-preemptive*
- Priority assignment based on *external* factor
  - Importance of job to user [typically *static* assignment]
  - *Cons*: low-priority jobs can *starve*
- Priority assignment based on *internal* factor
  - Aging
  - %CPU time used in last X hours

  ] [typically *dynamic* assignment]

Hybrid approaches may be used for mixed job types

Multilevel scheduling

- Ready queue *partitioned* into separate queues
  - E.g., system processes, foreground (*interactive*), background (*batch*)
- *Each* queue has its *own* scheduling algorithm
  - Example: foreground (*RR*), background(*FCFS*)
- Scheduling must be done between queues
  - *Fixed* priority – e.g, serve all foreground, then background
    - Possibility of starvation!
  - Time slice – e.g. 80% foreground (*RR*), 20% background (*FCFS*)