First Come, First Serve Scheduling

SCHEDULE-FCFS(r, A)

J

R

**for** each process u in A

J.push(u)

let visited[1..A.size()] be a new array

initialize all elements in visited to zero

t

**while** J and R

p J.top()

**while** J and p.arrival\_time < t

r.waiting\_time += p.burst\_time

R.push(J.pop())

**if** R

r.cpu\_util += 1

p R.top()

p.burst\_time -= 1

**if** visited[p->id] == false

visited[p->id] true

r.response\_time += time – 1 – p.arrival\_time

**if** p.burst\_time == 0

r.turnaround\_time += time – p.arrival\_time

R.pop()

++time

r.cpu\_util /= time – 1

r.throughput = size / (time – 1)

r.turnaround\_time /= size

r.waiting\_time = r.turnaround\_time – r.waiting\_time / size

r.response\_time /= size

Round Robin Scheduling

SCHEDULE-RR(r, A)

J

R

**for** each process u in A

J.push(u)

let visited[1..A.size()] be a new array

initialize all elements in visited to zero

stop\_watch

t

**while** J and R

p J.top()

**while** J and p.arrival\_time < t

r.waiting\_time += p.burst\_time

R.push(J.pop())

**if** R

r.cpu\_util += 1

p R.top()

**if** stop\_watch == QUANTUM

R.push(R.pop())

stop\_watch

p R.top()

++stop\_watch

p.burst\_time -= 1

**if** visited[p->id] == false

visited[p->id] true

r.response\_time += time – 1 – p.arrival\_time

**if** p.burst\_time == 0

r.turnaround\_time += time – p.arrival\_time

R.pop()

stop\_watch

++time

r.cpu\_util /= time – 1

r.throughput = size / (time – 1)

r.turnaround\_time /= size

r.waiting\_time = r.turnaround\_time – r.waiting\_time / size

r.response\_time /= size

Shortest Job First Scheduling (Preemptive)

SCHEDULE-SJN(r, A)

J

R

**for** each process u in A

J.push(u)

let visited[1..A.size()] be a new array

initialize all elements in visited to zero

c NULL

t

**while** J or R or c != NULL

p J.top()

**while** J and p.arrival\_time < t

r.waiting\_time += p.burst\_time

R.push(J.pop())

**if** R or c != NULL

r.cpu\_util += 1

**if** c != NULL

c R.pop()

c.burst\_time -= 1

**if** visited[p->id] == false

visited[p->id] true

r.response\_time += time – 1 – c.arrival\_time

**if** c.burst\_time == 0

r.turnaround\_time += time – c.arrival\_time

c NULL

++time

r.cpu\_util /= time – 1

r.throughput = size / (time – 1)

r.turnaround\_time /= size

r.waiting\_time = r.turnaround\_time – r.waiting\_time / size

r.response\_time /= size

Shortest Job First Scheduling (Non-Preemptive)

SCHEDULE-SRTF(r, A)

J

R

**for** each process u in A

J.push(u)

let visited[1..A.size()] be a new array

initialize all elements in visited to zero

t

**while** J and R

p J.top()

**while** J and p.arrival\_time < t

r.waiting\_time += p.burst\_time

R.push(J.pop())

**if** R

r.cpu\_util += 1

p R.top()

p.burst\_time -= 1

**if** visited[p->id] == false

visited[p->id] true

r.response\_time += time – 1 – p.arrival\_time

**if** p.burst\_time == 0

r.turnaround\_time += time – p.arrival\_time

R.pop()

++time

r.cpu\_util /= time – 1

r.throughput = size / (time – 1)

r.turnaround\_time /= size

r.waiting\_time = r.turnaround\_time – r.waiting\_time / size

r.response\_time /= size