ORARI I CPU-së

KAPITULLI 6

Prof. Ass. Dr. Isak Shabani

Orari i CPU

- Konceptet Bazë
- Kriteri i Orarit
- Algoritmet e Orarit
- Thread Orari
- Orari për Shumë Procesorë (ang. Multiple-Processor)
- Shembuj të Sistemeve Operative
- Vlerësimi i Algoritmeve

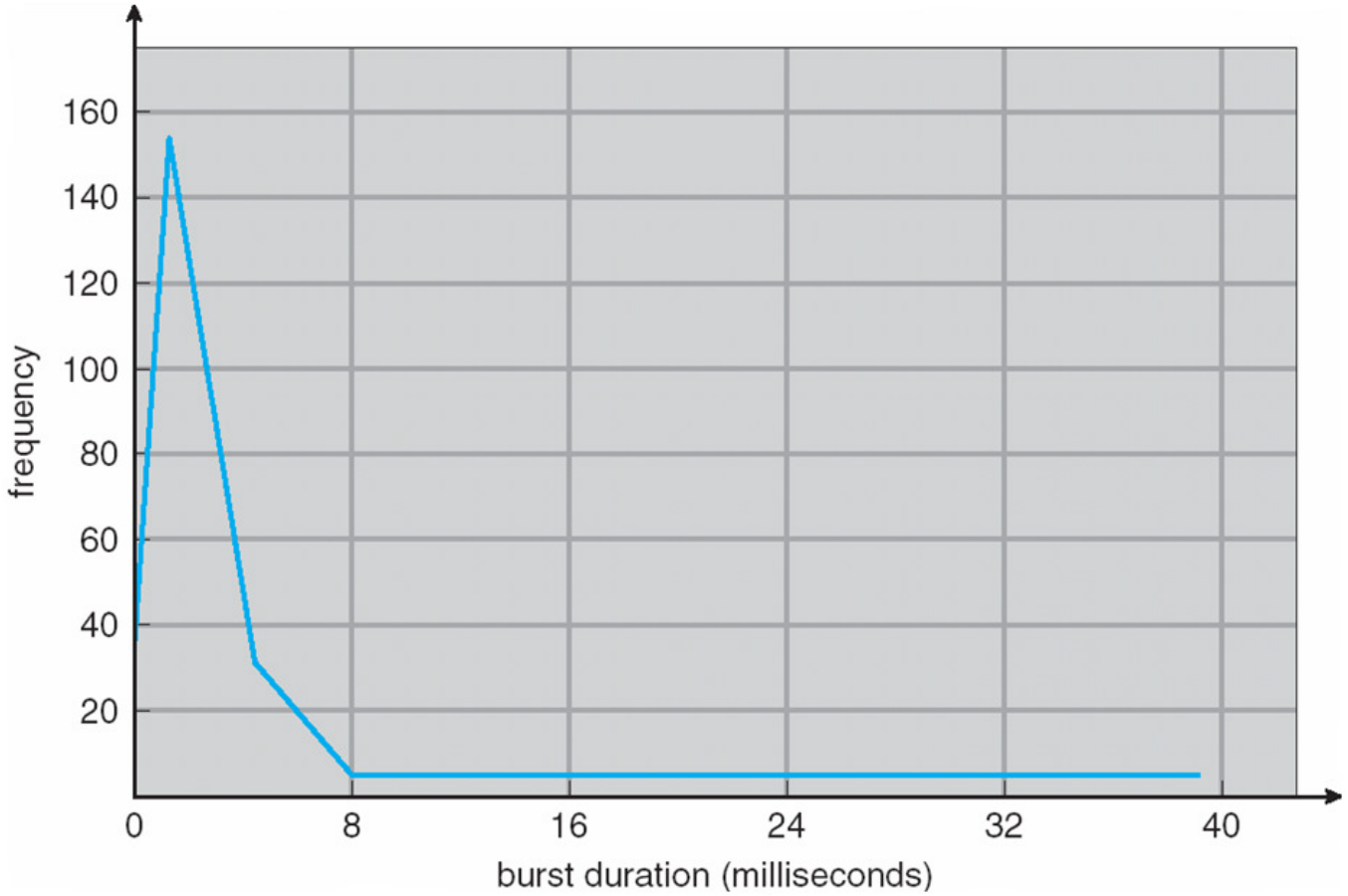
Objektivat

- Hyrje në orarin e CPU, i cili është bazë e sistemeve operative të multiprogramuara
- Të përshkruhen algoritme të ndryshme të orarit të CPU
- Të diskutohen kriteret vlerësuese për zgjedhjen e algoritmit të orarit të CPU për sistem të caktuar

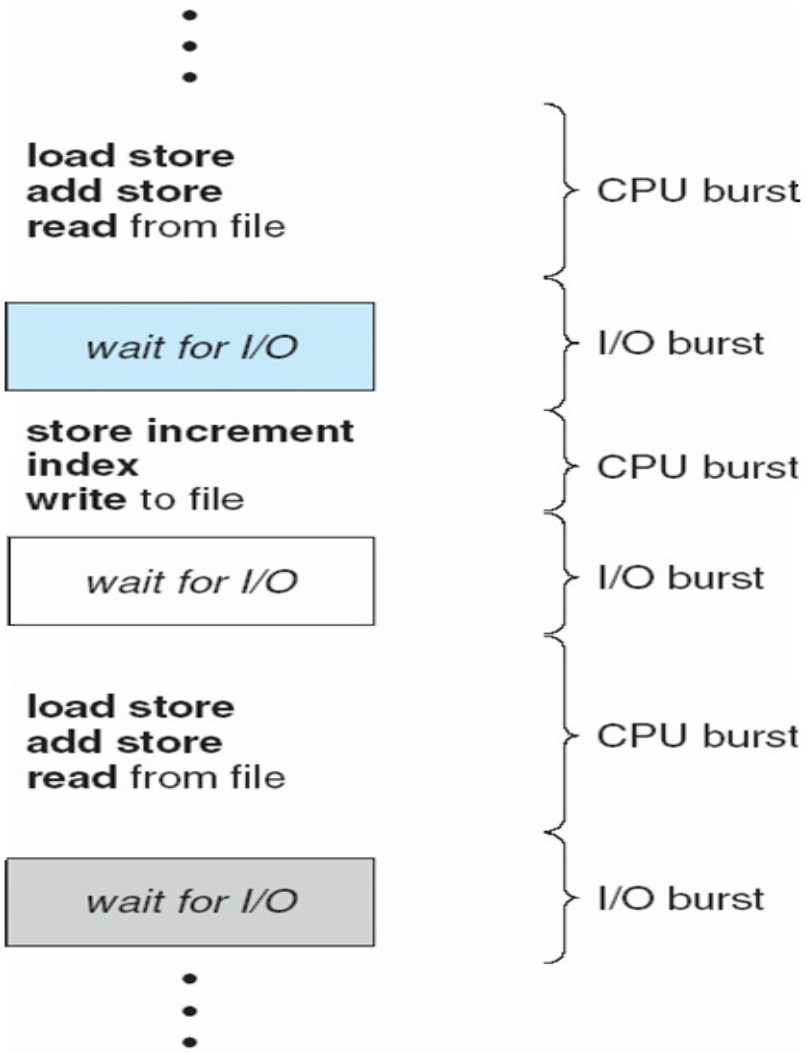
Konceptet Bazike

- Utilizim maksimal i CPU arrihet përmes multiprogramimit
- CPU–I/O Burst Cycle – Ekzekutimi i procesit përmban cikël të ekzekutimeve të CPU dhe pritjeve I/O
- Shpërndarja e **CPU burst**

Histogrami i Kohës së CPU-burst



Sekuenca Alternative e CPU dhe I/O Bursts



Orari i CPU

- Zgjedhë në mes të proceseve në memorie të cilat janë të gatshme për ekzekutim, dhe e alokon CPU te një prej tyre
- Vendimet e orarit CPU mund të zënë vend kur procesi:
  1. Kalon nga gjendja e lëshimit (running) në gjendjen pritëse (waiting state)
  2. Kur kalon nga gjendja e lëshimit në gjendje të gatshme (ready state)
  3. Kalon nga pritja (waiting) në të gatshme (ready)
  4. Përfundon (ang. Temrminates)
- Orari nga 1 dhe 4 është **nonpreemptive**
- Të gjitha oraret tjera janë **preemptive**

Dispatcher

- Moduli Dispatcher i jep kontrollin e CPU proceseve të zgjedhura nga orari afat-shkurtërë kjo përfshinë:
  - Ndryshimi i kontekstit
  - Kalimi në user mode
  - Kërcimi në lokacionin e duhur në programin e shfrytëzuesit për të rinisur atë program
- **Dispatch latency** – koha për të cilen dispatcheri ndalon një proces dhe e fillon një tjetër në lëshim

Kriteri i Orarit

- **Utilizimi i CPU** – mbajë CPU të ngarkuar sa më shumë të jetë e mundur
- **Throughput** – # numri i procesve që kompletojnë ekzekutimin e tyre për njësi kohore
- **Turnaround time** – sasia e kohës për të ekzekutuar një proces të caktuar
- **Waiting time** – koha për të cilën procesi ka qenë në pritje në radhë të proceseve të gatshme
- **Response time** – koha e nevojshme nga koha kur është bërë kërkesa gjerë në përgjigjen e parë, jo output (për ambientet me kohë-të-ndarë)

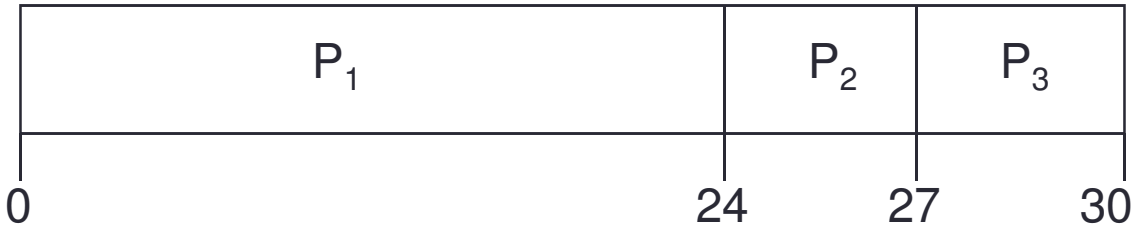
Kriteri i Optimizimit të Algoritmit të Orarit

- Utilizim Max i CPU
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Orari First-Come, First-Served (FCFS)

Proceset	Burst Time
P_1	24
P_2	3
P_3	3

- Të supozojmë se proceset arrijnë në rënditjen: P_1 , P_2 , P_3 Gantt Charti për orarin do të jetë:



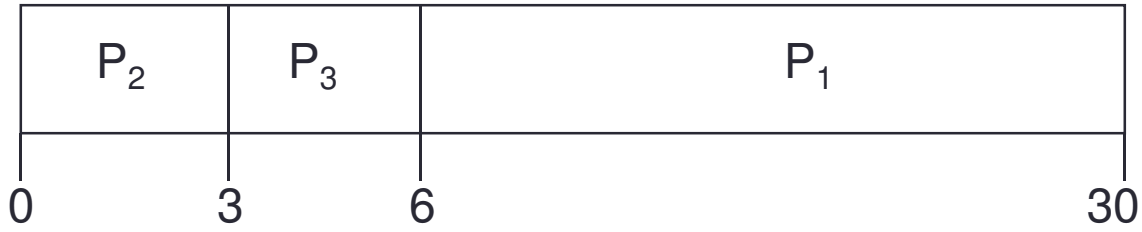
- Koha e pritjes për P_1 = 0 P_2 = 24 P_3 = 27
- Koha mesatare e pritjes: (0 + 24 + 27)/3 = 17

Orari FCFS (Vazh.)

Të supozojmë se proceset arrijnë në këtë rënditje:

$$P_2 , P_3 , P_1$$

- Gantt chart-i për orarin është:



- Koha epritjes për $P_1 = 6$ *ë* $P_2 = 0$ *ë* $P_3 = 3$
- Koha mesatare e pritjes: $(6 + 0 + 3)/3 = 3$
- Shumë më mirë se rasti paraprak
- *Efekti i konvoit* është procesi i shkurtër prapa procesit të gjatë

Orari Shortest-Job-First (SJF)

- Shoqëro me secilin proces gjatësinë e CPU burst-it të tij vijuës.
- Përdor këto gjatësi për të bërë orarin e procesit me kohë më të shkurtër
- SJF është optimal – jep kohën mesatare minimale të pirtjes për bashkësinë e proceseve të dhëna
  - Vështirësia është në njohjen e gjatësisë së CPU kërkesës vijuëse

Shembull i SJF

Proceset	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- Chart për orarin SJF

- Koha mesatare e pritjes = (3 + 16 + 9 + 0) / 4 = 7

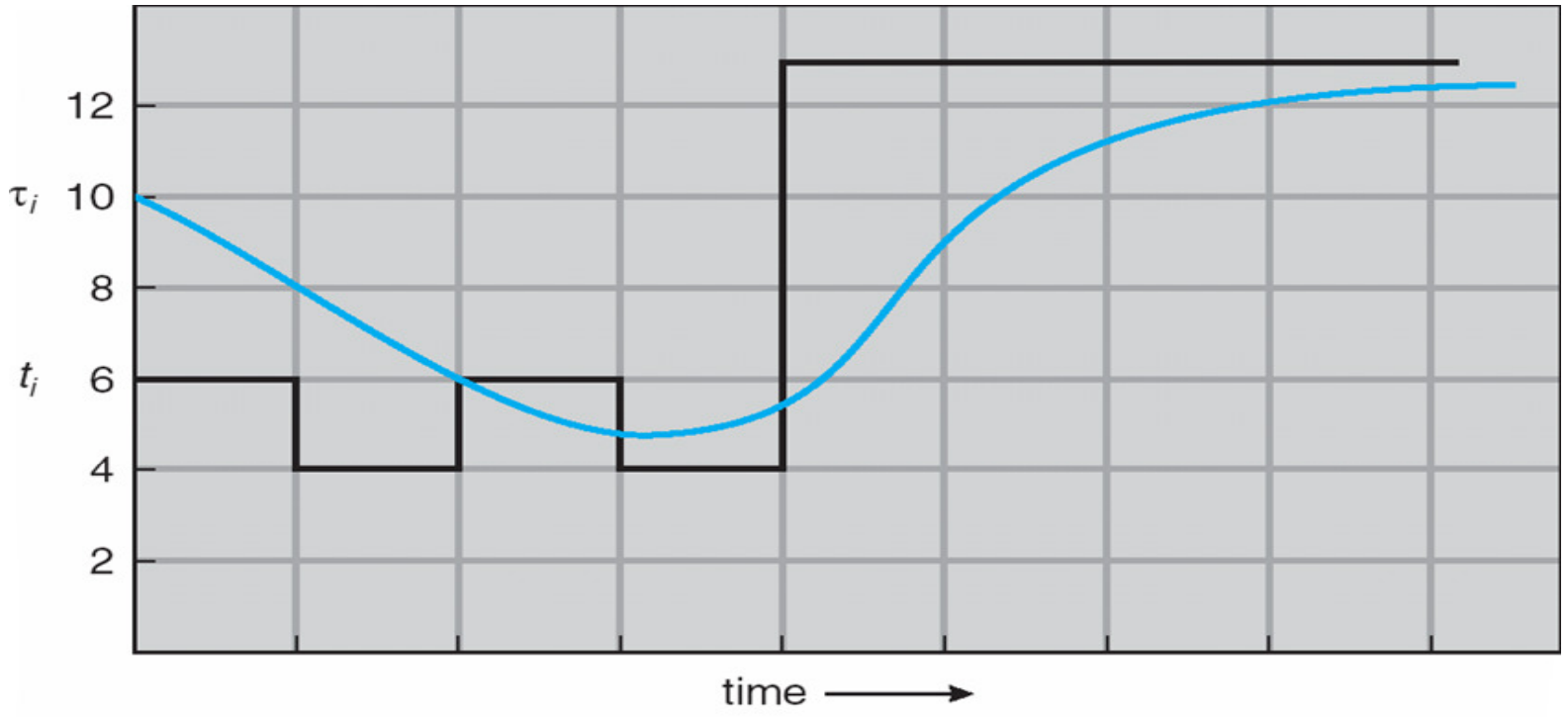
Përcaktimi i Gjatësisë së CPU Burst-it vijuës

- Mund të përcaktohet vetëm gjatësia
- Mund të arrihet duke shfrytëzuar gjatësinë e CPU burst-ave paraprak, duke shfrytëzuar mesataren eksponencial

1. $t_n = \text{gjatësia aktuale e } n^{th} \text{ CPU burst}$
2. $\tau_{n+1} = \text{vlera e predikuar për CPU burst të ardhshëm}$
3. $\alpha, 0 \le \alpha \le 1$
4. Definim :

$$\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n.$$

Parashikimi i Gjatësisë së CPU Burst të Ardhshëm



CPU burst (t_i)	6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Shembuj të Mesatarës Eksponenciale

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Historia e fundit nuk numërohet
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Vetëm CPU burst aktual i fundit numërohet
- Nëse e zgjerojmë formulën, marrim:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_{n-1} + \ldots$$
$$+ (1 - \alpha)^j \alpha\, t_{n-j} + \ldots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

- Gjetëras të dyjat α dhe $(1 - \alpha)$ janë më të vogla ose të barabarta me 1, secili term suksesiv ka më pak peshë se paraardhësi i tij

Orari Prioritar

- Numër prioriteti (integer) i shoqërohet secilit proces
- CPU alokohet tek procesi me prioritet më të lartë (numri i plotë më i vogël ≡ prioriteti më i lartë)
  - Preemptive
  - Nonpreemptive
- SJF është orar me prioritet ku prioriteti është koha e predikatit të CPU burst-it të ardhshëm
- Problemi ≡ **Starvation** – proceset me prioritet të ulët mund të mos ekzekutohen fare
- Zgjidhje ≡ **Aging** – me progresin e kohës rrit prioritetin e procesit

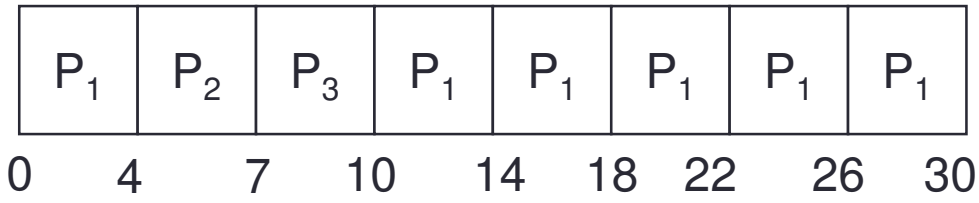
Round Robin (RR)

- Secili proces merr një njësi të vogël të kohës së CPU (*time quantum*), zakonisht 10-100 millisekonda. Pas kalimit të kësaj kohe, procesi është premtpuar (ndërprerja e procesit) dhe vendoset në fund të radhës së gatshme.
- Nëse janë *n* procese në radhë të gatshme dhe time quantum është *q*, atëherë secili proces merr 1/*n* të kohës së CPU në copëza të së paku *q* njësi kohe përnjëherë. Asnjë proces nuk pret më shumë se (*n*-1)*q* njësi kohore.
- Performanca
  - *q* e madhe $\Rightarrow$ FIFO
  - *q* e vogel $\Rightarrow$ *q* duhet të jetë e madhe në lidhje me ndryshimin e kontekstit, përndryshe ngarkesa është shumë e madhe

Shembull i RR me Time Quantum = 4

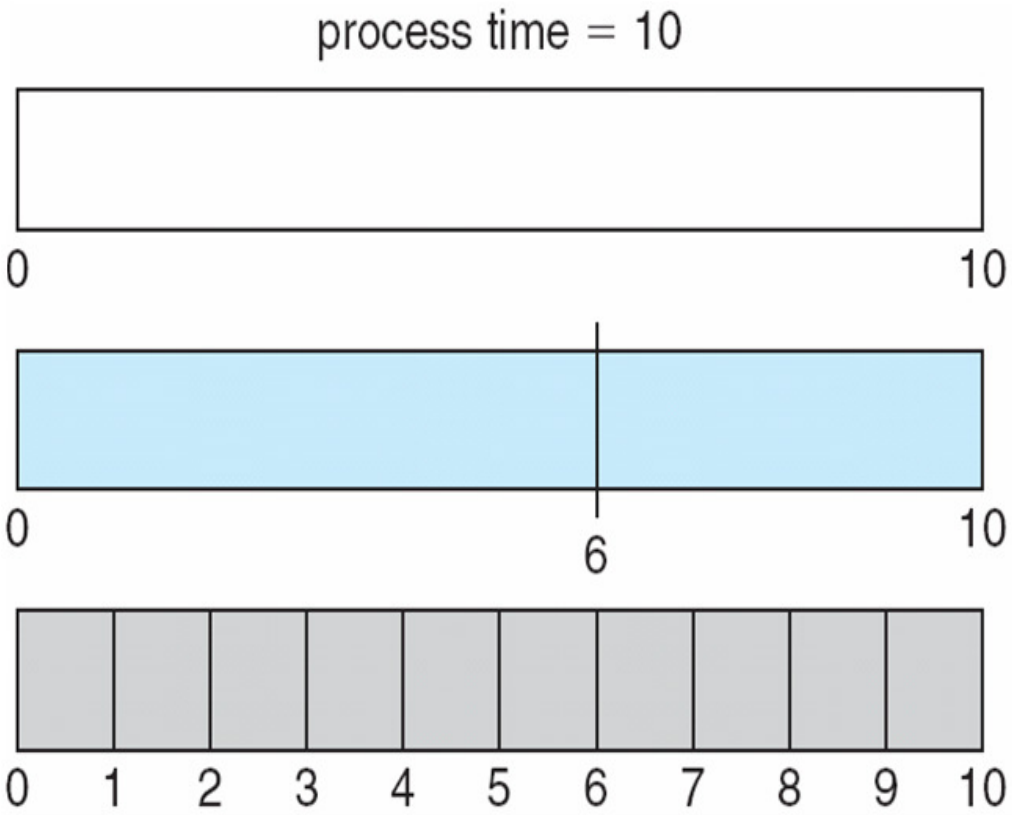
Proceset	Burst Time
P_1	24
P_2	3
P_3	3

- Gantt charti është:



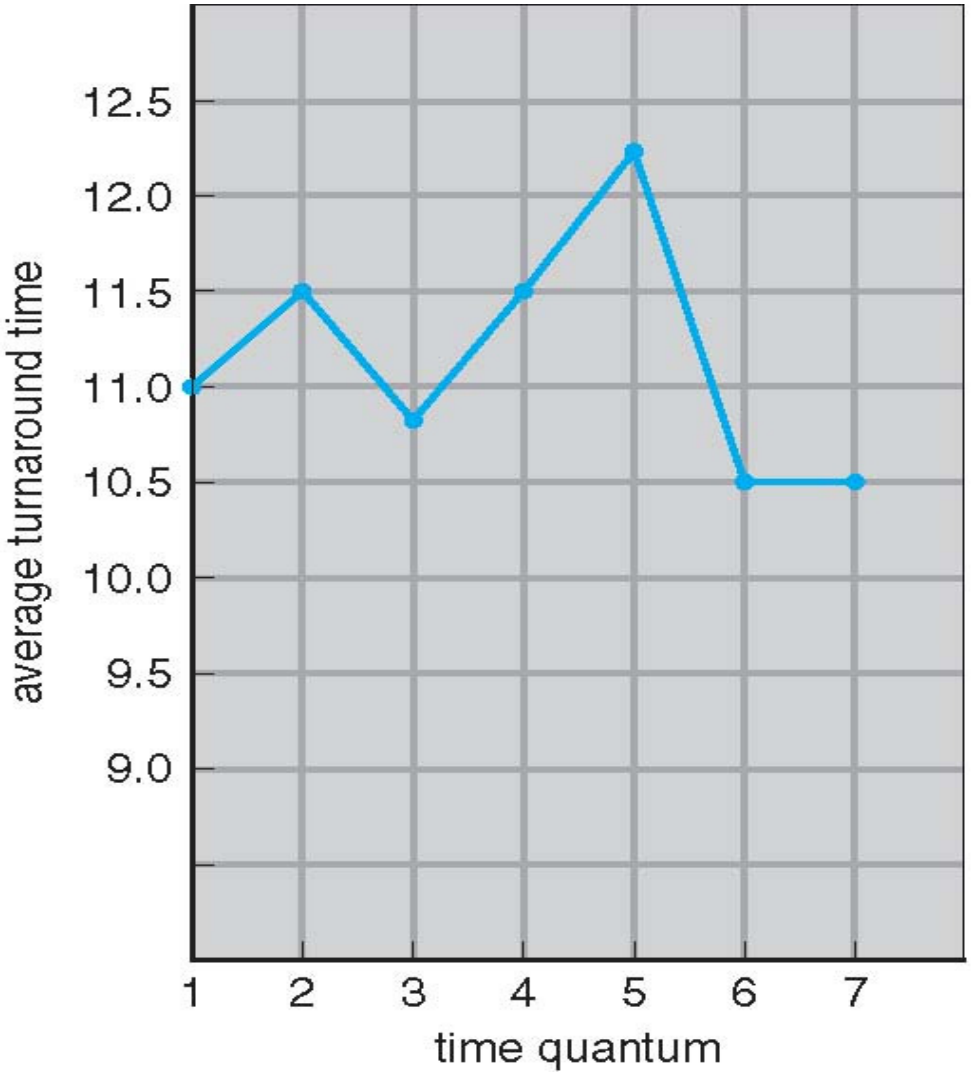
- Zakonisht, mesatarja më e lartë e turnaround se SJF, por *përgjegjësi* më të mirë

Time Quantum dhe Koha e Ndryshimit të Kontekstit



quantum	context switches
12	0
6	1
1	9

Koha Turnaround varet nga Koha e Kuantumit

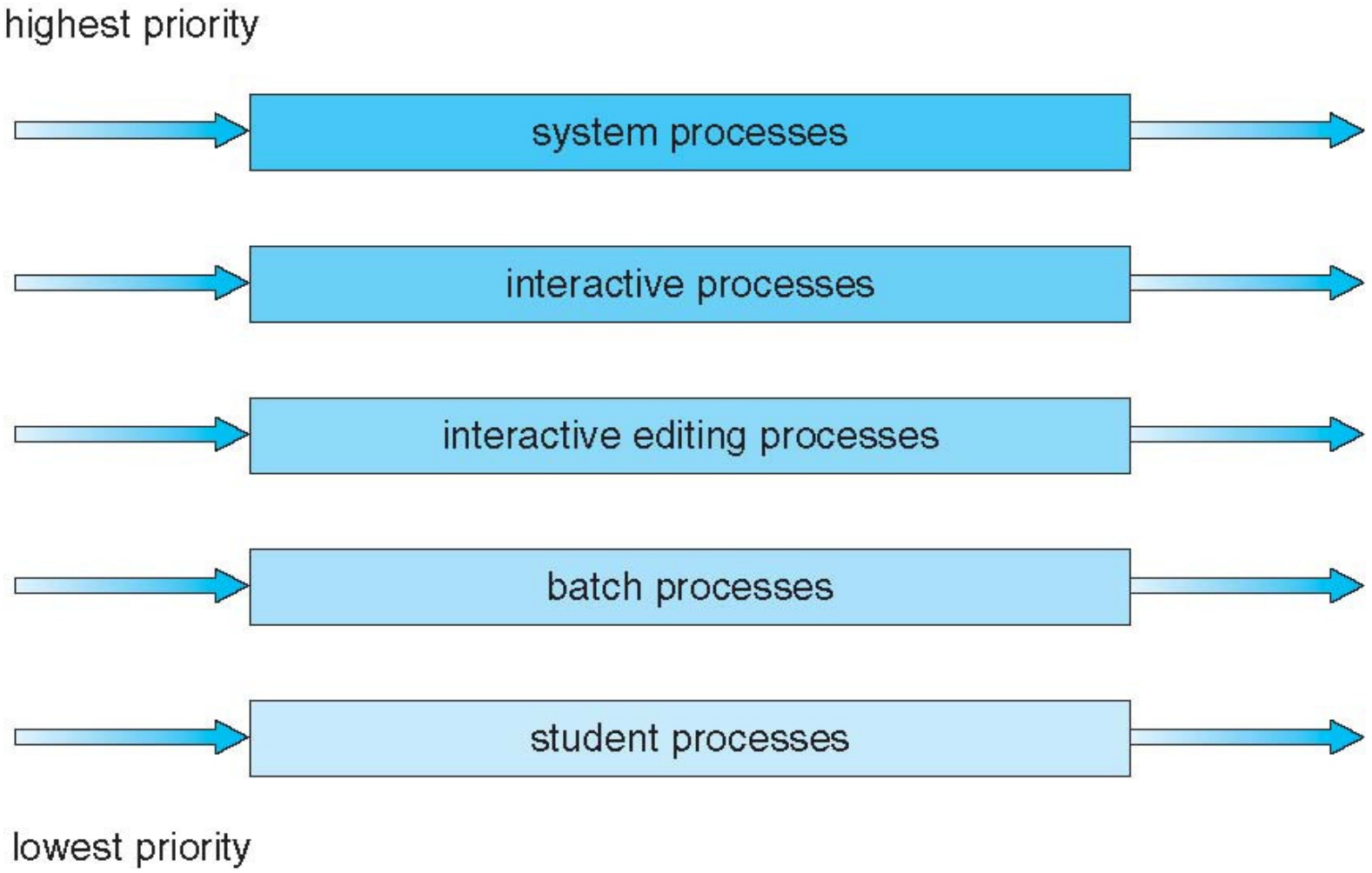


process	time
P_1	6
P_2	3
P_3	1
P_4	7

Radha Shumë-nivelëshe

- Radha Ready është e ndarë në radhë parciale:
  foreground (interaktive)
  background (batch/grumbuj)
- Secila radhë ka algoritmin e vet të orarit
  - sfondi– RR
  - prapavija– FCFS
- Orari duhet të bëhet në mes të radhëve
  - Orarë fiks të prioriteteve (p.sh., shërbe të gjitha nga sfondi atëherë nga prapavija). Mundësi e starvation.
  - Ndarja Kohore – secila radhë merr sasi të caktuar të kohës së CPU të cilën mund ta vendos në orar në mes të proceseveë p.sh., 80% në sfond në RR
  - 20% në prapavijë në FCFS

Orarët me Radhë Shumë-Nivelëshe



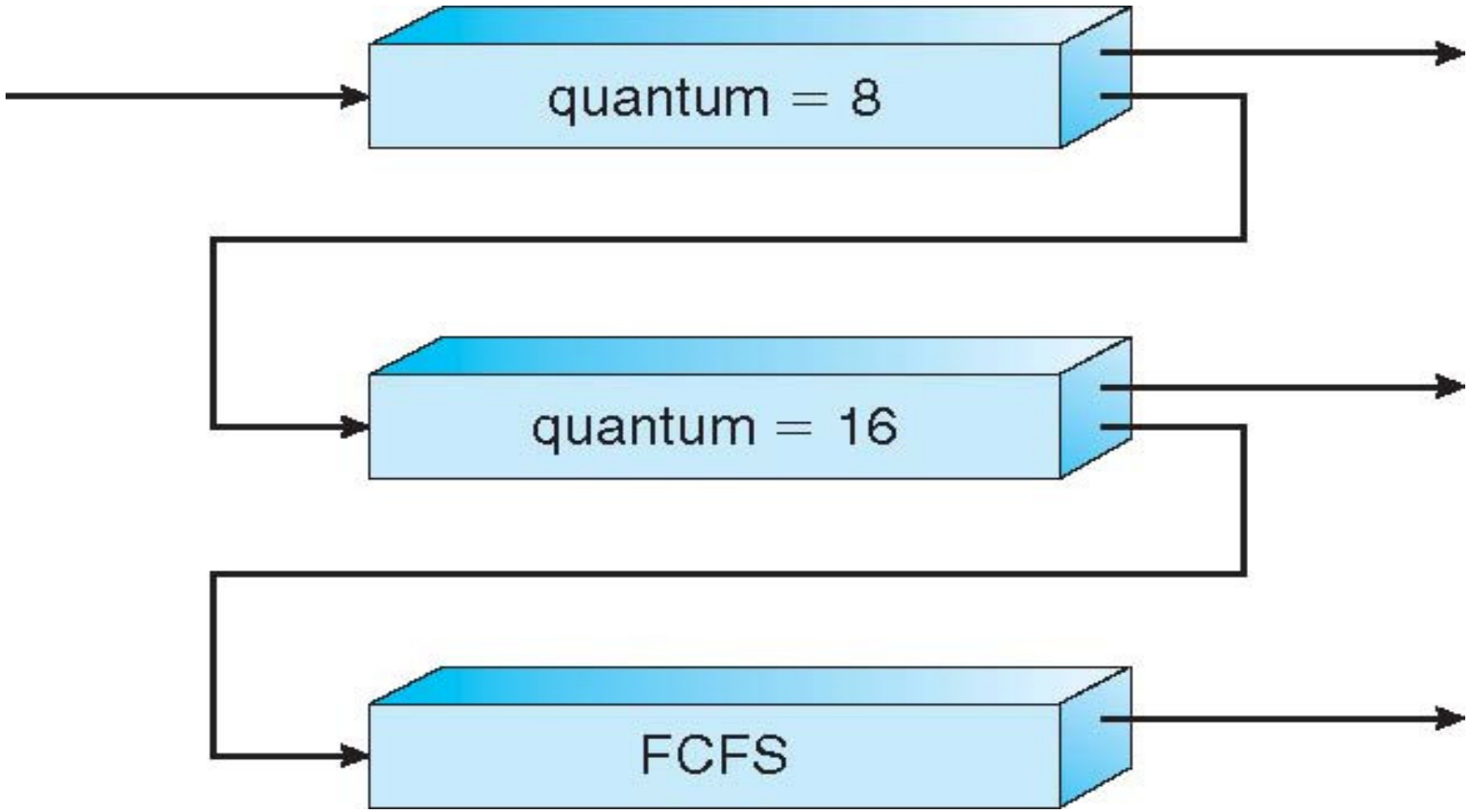
Radhët Kthyëse Shumë-Nivelëshe

- Procesi mund të lëvizë në mes të radhëve të ndryshmeë mplakja (aging) mund të implementohet në këtë mënyrë
- Orari me Radhë-Kthyëse shumë-nivelëshe i definuar me parametrat vijuës:
  - Numër të radhëve (ang.,queues)
  - Algoritma të orarit për secilën radhë
  - Metoda e shfrytëzuar për të përcaktuar kur të bëhet ngritja e procesit
  - Metoda e shfrytëzuar për të përcaktuar kur të bëhet zbritja në pozitë e procesit
  - Metoda e shfrytëzuar për të përcaktuar se cila radhë e procesit do të futet kur ai proces ka nevojë për shërbim

Shembull i Radhës Kthyëse Shumë-nivelëshe

- Tri radhë:
  - Q_0 – RR me time quantum 8 ms
  - Q_1 – RR time quantum 16 ms
  - Q_2 – FCFS
- Orari
  - Një punë e re futet në radhën Q_0 *e cila shërbhet* FCFS. Kur e fiton CPU, puna merr 8 milisekonda. Nëse nuk përfundon në 8 milisekonda, puna lëvizë në radhën Q_1.
  - Në Q_1 puna prap shërbehet në FCFS dhe merr 16 milisekonda shtesë. Nëse ende nuk përfundon, preemptohet dhe lëvizet në radhën Q_2.

Radhës Kthyëse Shumë-Nivelëshe



Orari i Fijeve (Threads)

- Dallimi në mes të fijeve të nivelit-shfrytëzues dhe kernel-nivelit
- Shumë-në-një dhe shumë-me-shumë modelet, livraria e fijeve bën orarin e fijeve të nivelit-shfrytëzues që të lëshohen në LWP
  - E njohur si **process-contention scope (PCS)** gjerësa gara e orarit është përbrenda procesit
- Fija Kernel me orar në CPU të gatshme është **system-contention scope (SCS)** – gara në mes të fijeve në sistem.

Orari Pthread

- API lejon specifikimin e PCS ose SCS gjatë krijimit të fijes
  - PTHREAD SCOPE PROCESS bën orarin për fijet duke shfrytëzuar PCS orarin
  - PTHREAD SCOPE SYSTEM bën orarin e fijeve duke shfrytëzuar SCS orarin.

API i Orarit Pthread

```
#include <pthread.h>
#include <stdio.h>
#define NUM THREADS 5
int main(int argc, char *argv[])
{
     int i
    pthread t tid[NUM THREADS]
    pthread attr t attr
    /* get the default attributes */
    pthread attr init(&attr)
    /* set the scheduling algorithm to PROCESS or SYSTEM */
    pthread attr setscope(&attr, PTHREAD SCOPE SYSTEM)
    /* set the scheduling policy - FIFO, RT, or OTHER */
    pthread attr setschedpolicy(&attr, SCHED OTHER)
    /* create the threads */
    for (i = 0ë i < NUM THREADSë i++)
                    pthread create(&tid[i],&attr,runner,NULL)
```

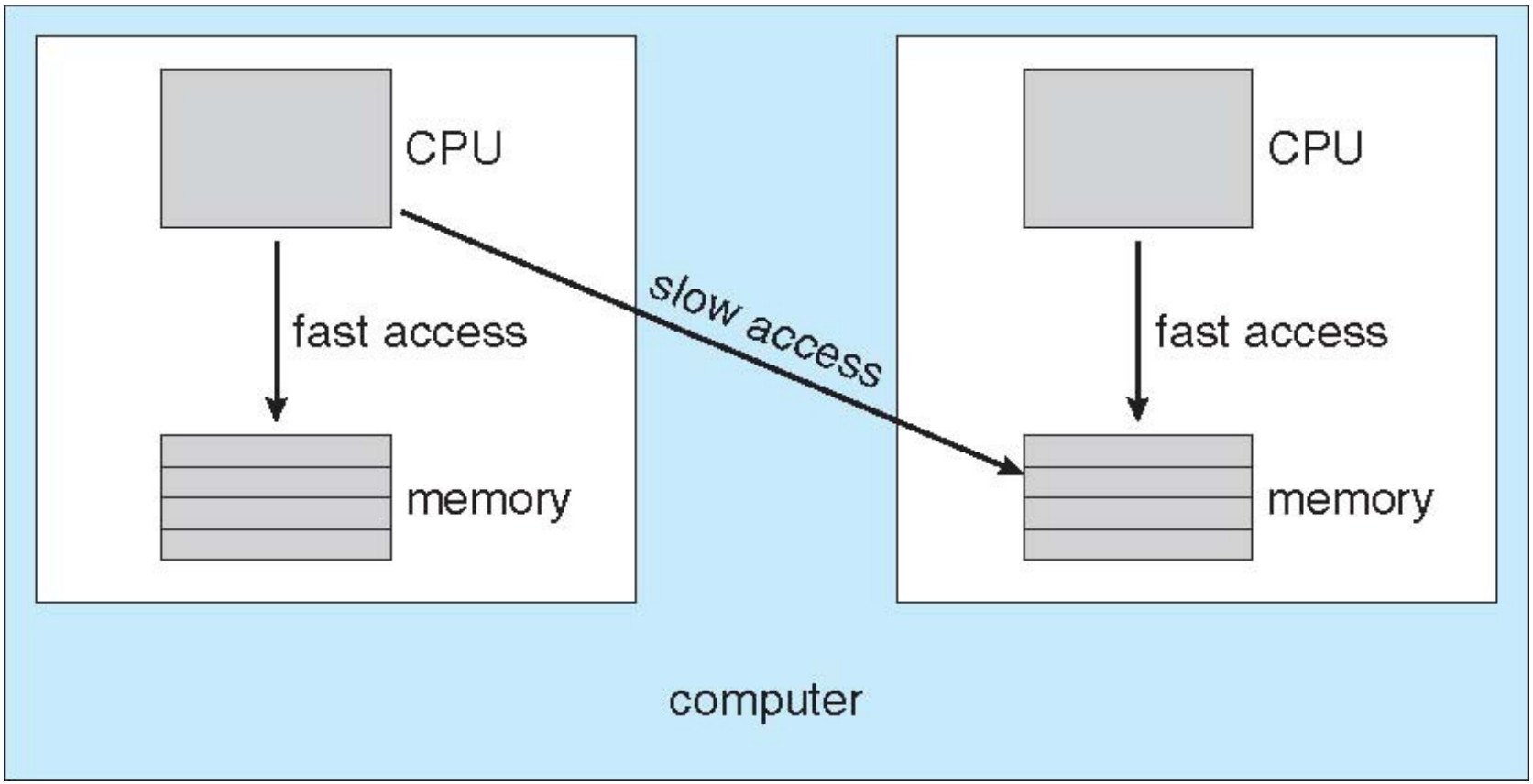
API i Orarit Pthread

```
    /* now join on each thread */
    for (i = 0ë i < NUM THREADSë i++)
                pthread join(tid[i], NULL)ë
}
 /* Each thread will begin control in this function */
void *runner(void *param)
{
    printf("I am a thread\n")ë
    pthread exit(0)ë
}
```

Orari për Shumë-Procesorë

- Orari i CPU është më kompleks kur ka shumë CPU
- **Procesorë homogjenë** përbrenda multiprocesorit
- **Multiprocesimi asimetrik** – vetëm një proces u qaset strukturës së bazës së të dhënave, duke e lehtësuar nevojën për ndarjen e të dhënave
- **Multiprocesimi simetrik (SMP)** – secili procesor e bën orarin për vetvetën, të gjitha proceset në radhën e gatshme të përbashkët, ose seicla e ka radhën e saj private të procesve të gatshme
- **Afiniteti i Procesorit**– procesi ka afnititet për procesor në të cilin aktualisht është duke u ekzekutuar
  - **Afinitet të butë**
  - **Afinitet të fortë**

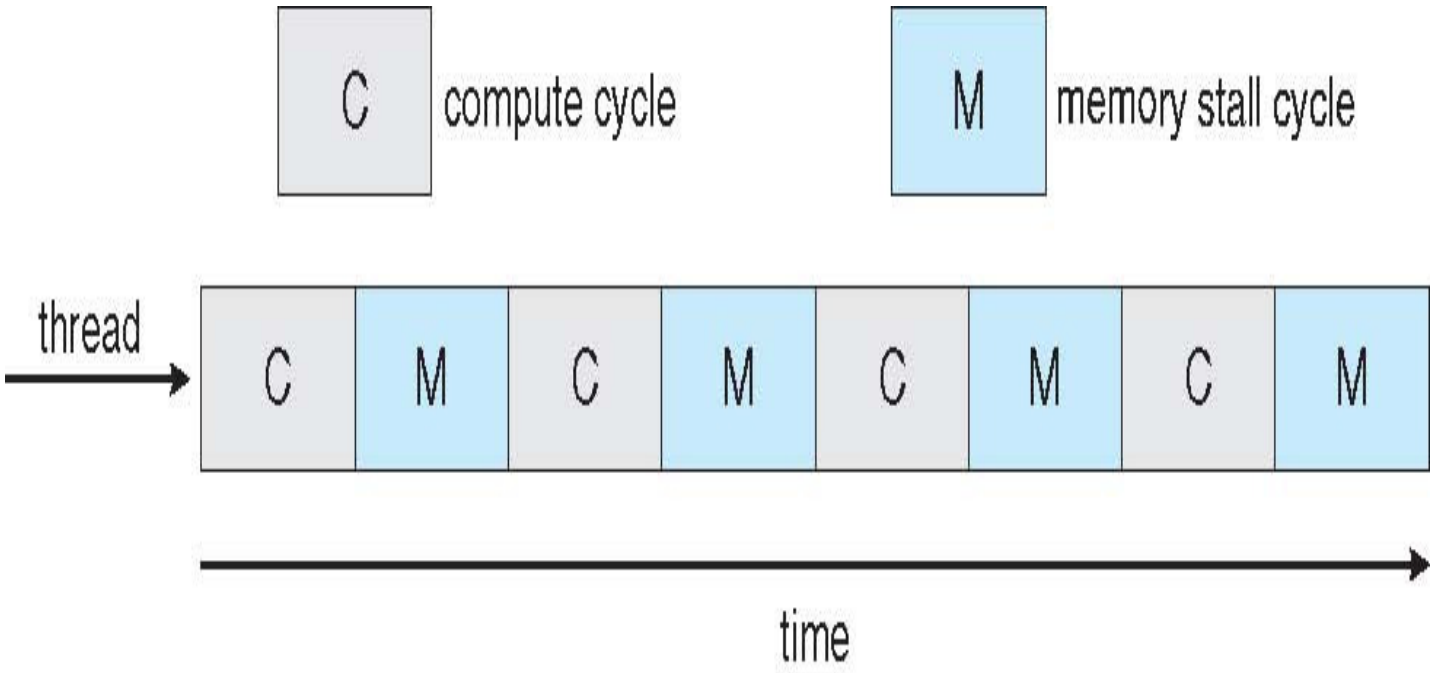
NUMA dhe Orari i CPU



Procesorët Shumë-Shtresash

- Trend i fundit vendosja e shumë shtresave të procesorëve në një chip të njëjtë fizik
- Më i shpejtë dhe konsumon më pak energji
- Shumë fije për shtresë dhe duke u rritur
  - Merr përparësi të memories për të bërë progres në fijën tjetër gjerësa marrja memorike ndodhë.

Sistemet Shumë-Shtresash Shumë-Fijesh



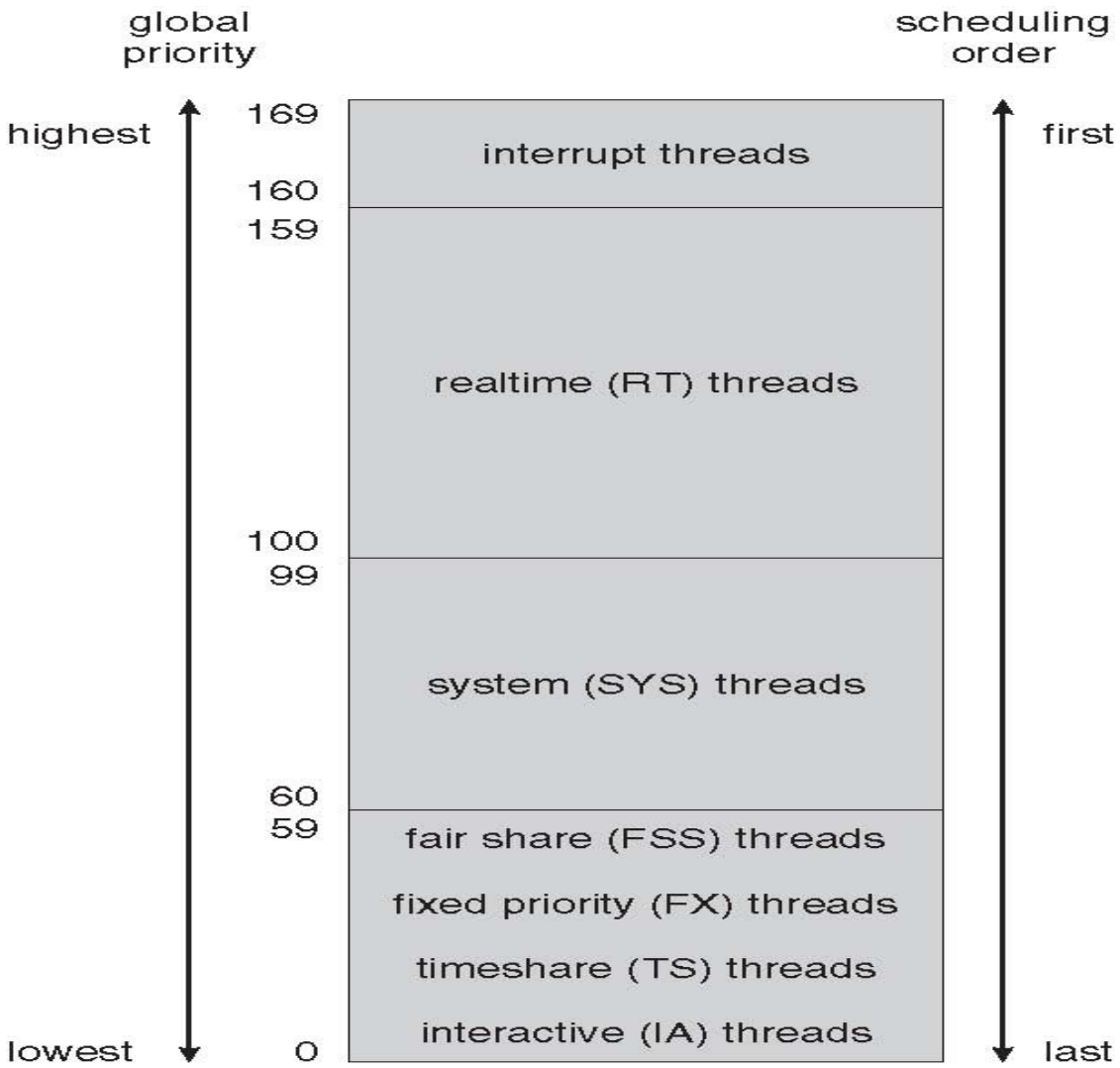
Shembuj të Sistemve Operative

- Orari i Solaris
- Orari Windows XP
- Orari Linux

Tabela e Dërgesës (Dispatch) në Solaris

priority	time quantum	time quantum expired	return from sleep
0	200	0	50
5	200	0	50
10	160	0	51
15	160	5	51
20	120	10	52
25	120	15	52
30	80	20	53
35	80	25	54
40	40	30	55
45	40	35	56
50	40	40	58
55	40	45	58
59	20	49	59

Orari Solaris



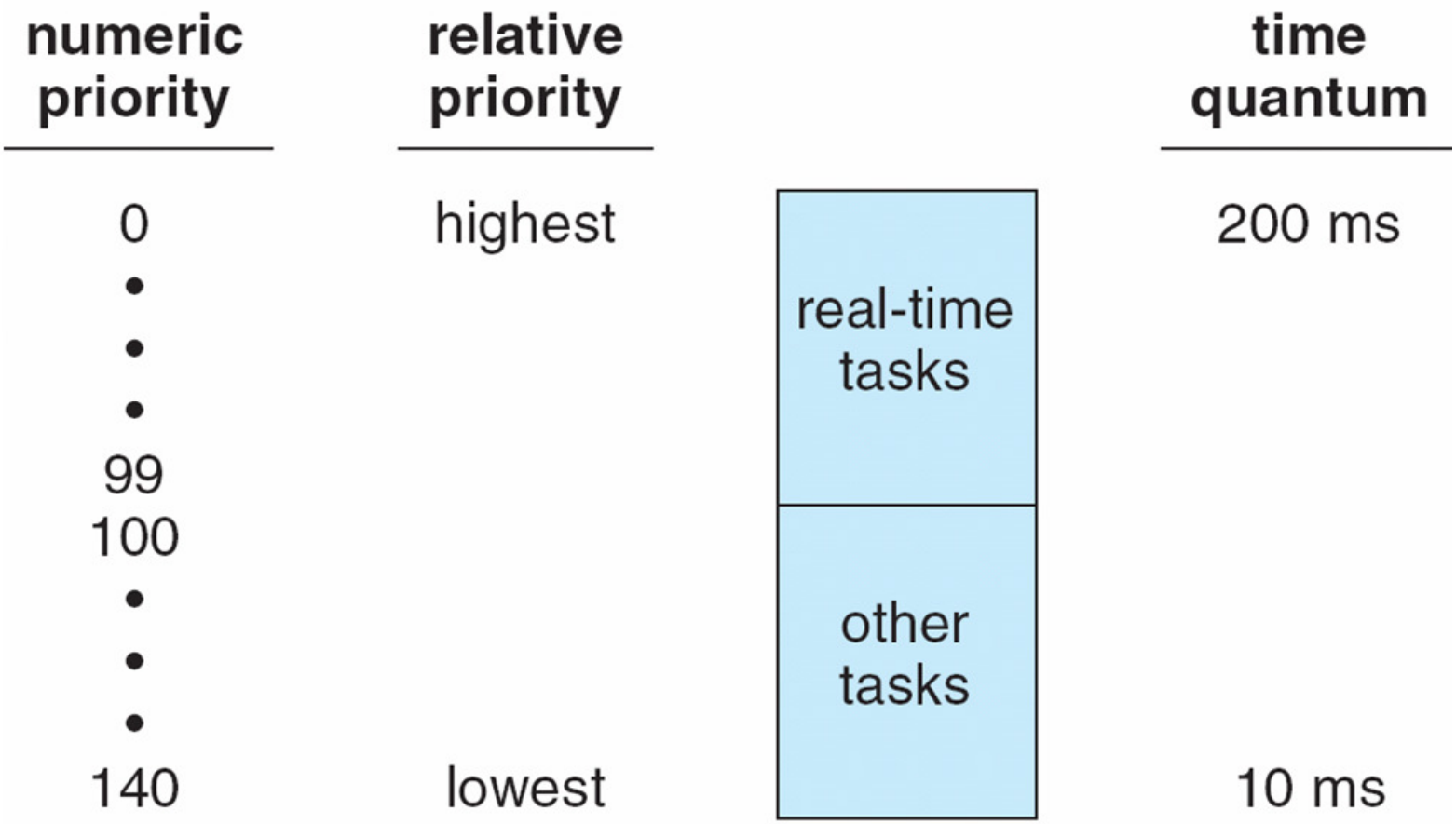
Prioritetet Windows XP

	real-time	high	above normal	normal	below normal	idle priority
time-critical	31	15	15	15	15	15
highest	26	15	12	10	8	6
above normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
below normal	23	12	9	7	5	3
lowest	22	11	8	6	4	2
idle	16	1	1	1	1	1

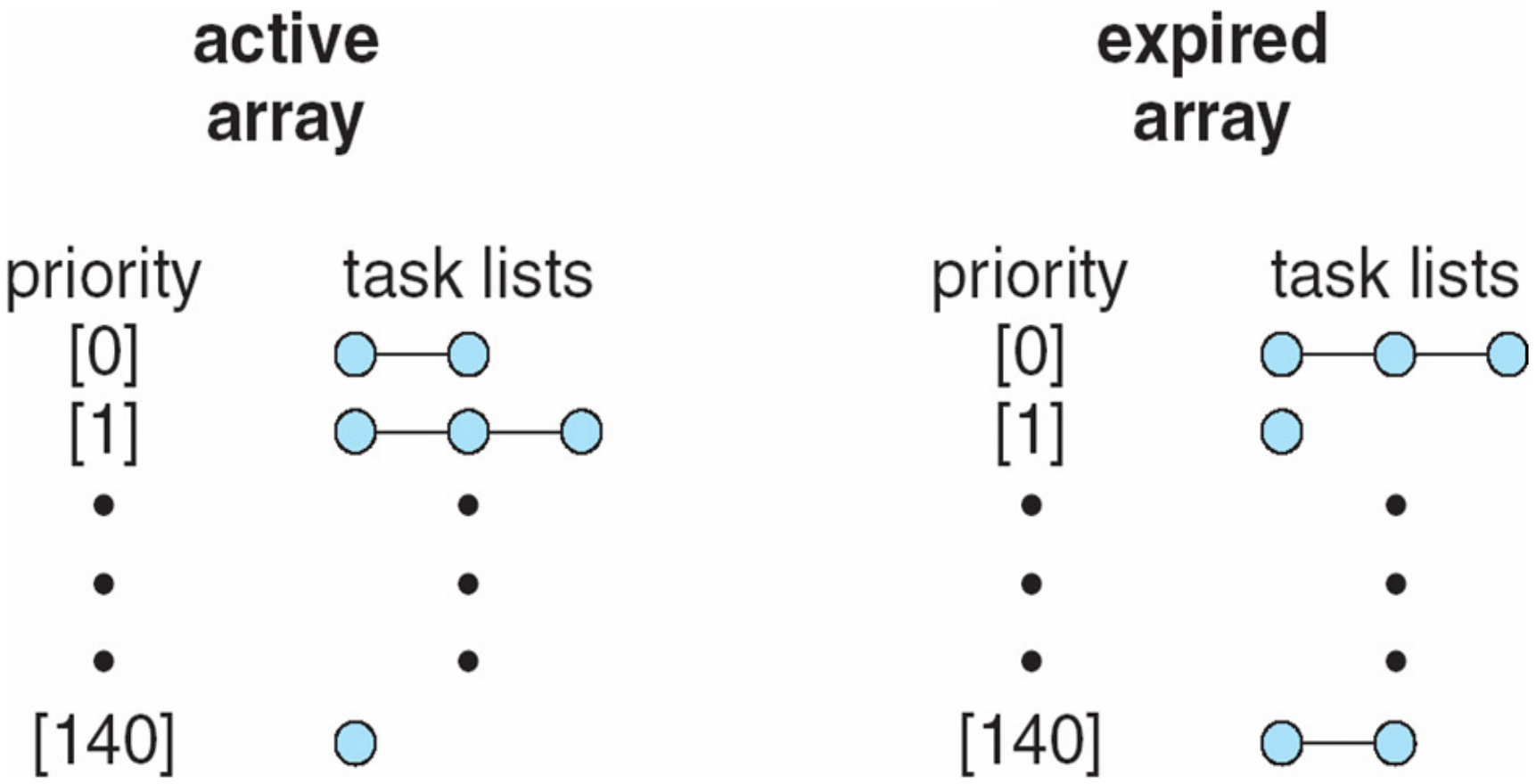
Orari Linux

- Renditje konstante *O*(1) e kohës së orarit
- Dy rangje prioriteti: ndarje-kohore dhe kohë-reale
- Rangu **Kohë-reale** nga 0 në 99 dhe dhe vlerë e **mirë** nga 100 në 140
- (figura 5.15)

Prioritetet dhe gjatësia Time-slice



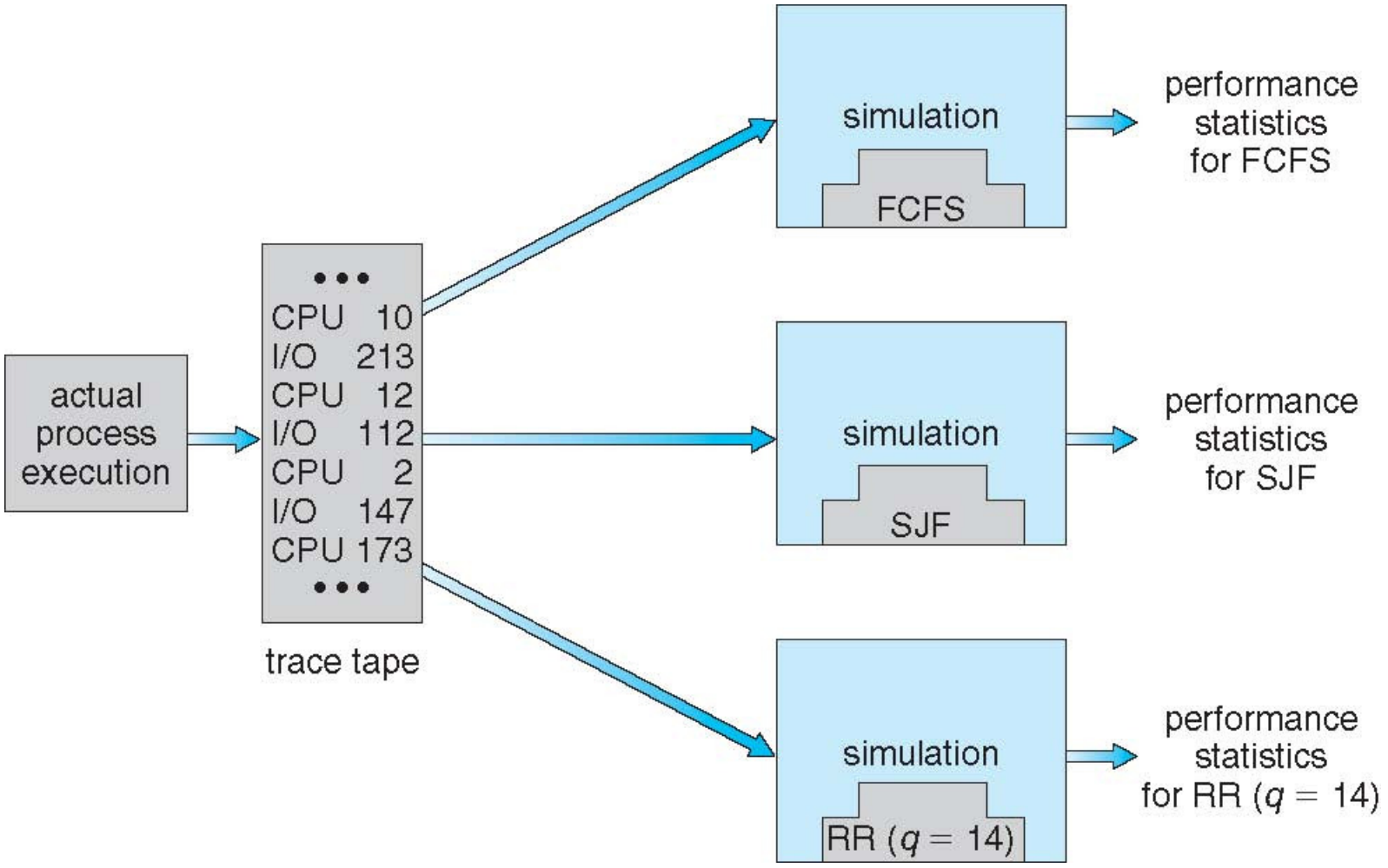
Lista e Detyrave të indeksuara Sipas Prioriteteve



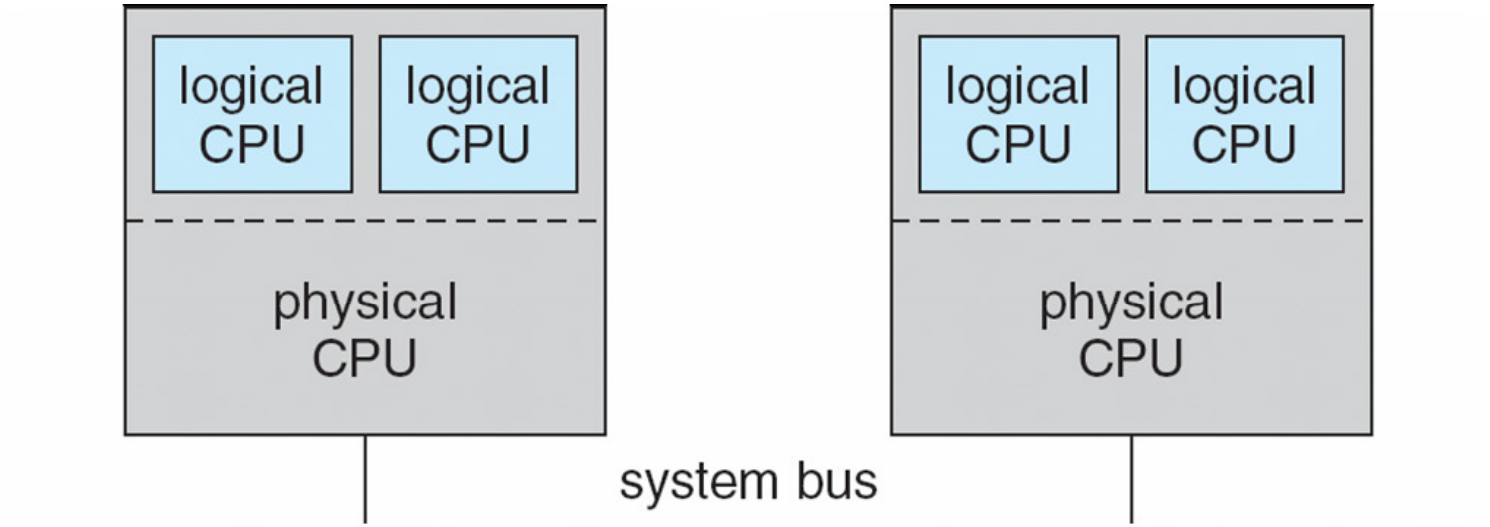
Vlerësimi Algoritmit

- Model deterministik – merr ngarkesë të parapërcaktuar dhe definon performancën në secilin algoritëm për atë ngarkesë
- Modelet e radhëve
- Implementimi

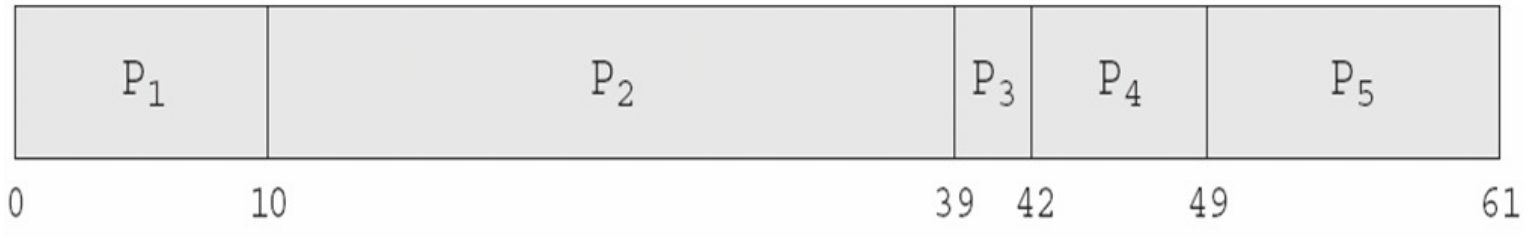
Vlerësimi i Orarëve të CPU me simulim



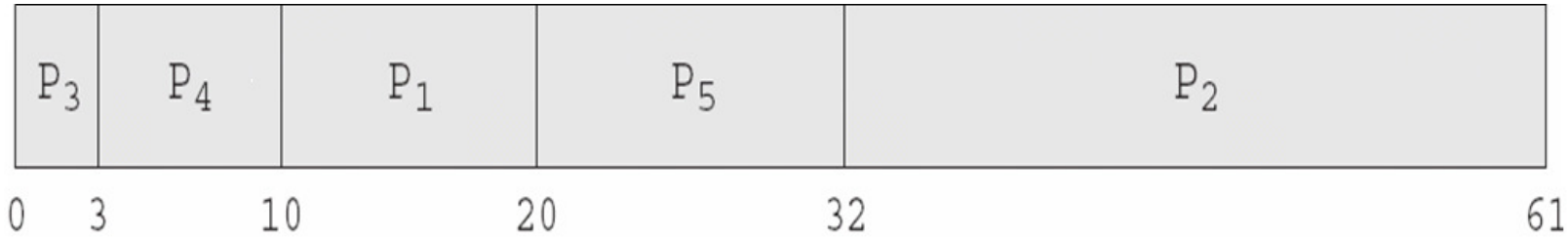
5.08


logical CPU
logical CPU
physical CPU
logical CPU
logical CPU
physical CPU
system bus

In-5.7



In-5.8



In-5.9

Orari ne Solaris 2