

# OPERATIVNI SISTEMI

Slajdovi su kreirani na osnovu knjige “Operativni sistemi, principi unutrašnje organizacije i dizajna, 7. izdanje“, William Stallings, CET, Beograd, 2013.

Jednoprocesorsko raspoređivanje

# Procesorsko raspoređivanje

- Cilj procesorskog raspoređivanja je da se procesoru/processorima dodeljuju procesi za izvršavanje
  - ▣ tako da se ispune ciljevi sistemi, kao što su vreme odziva, propusna moć i efikasnost procesora

# Ciljevi raspoređivanja

- Raspoređivanje treba da
  - ▣ Obezbedi fer deljenje vremena među procesima
  - ▣ Spreči gladovanje procesa
  - ▣ Obezbedi efikasno korišćenje procesora
  - ▣ Uspostavi privilegije među procesima
    - Npr veći prioritet procesima sa bliskim rokom završetka
  - ▣ Pomenute poslove obavlja uz uvođenje što manje dodatnog opterećenja

# Tipovi procesorskog raspoređivanja

- Dugoročno raspoređivanje
  - ▣ Stvaranje novog procesa
  - ▣ Oduka da se proces doda u skup procesa koji konkurišu za izvršavanje
    - Prema redosledu dolaska ili
    - Prema nekom kriterijumu (prioritet, očekivano vreme izvršenja, zahtevi za U/I, ...)
  - ▣ Upravlja stepenom multiprogramiranja
  - ▣ Što više procesa, manji procenat vremena u kojem proces može da se izvršava

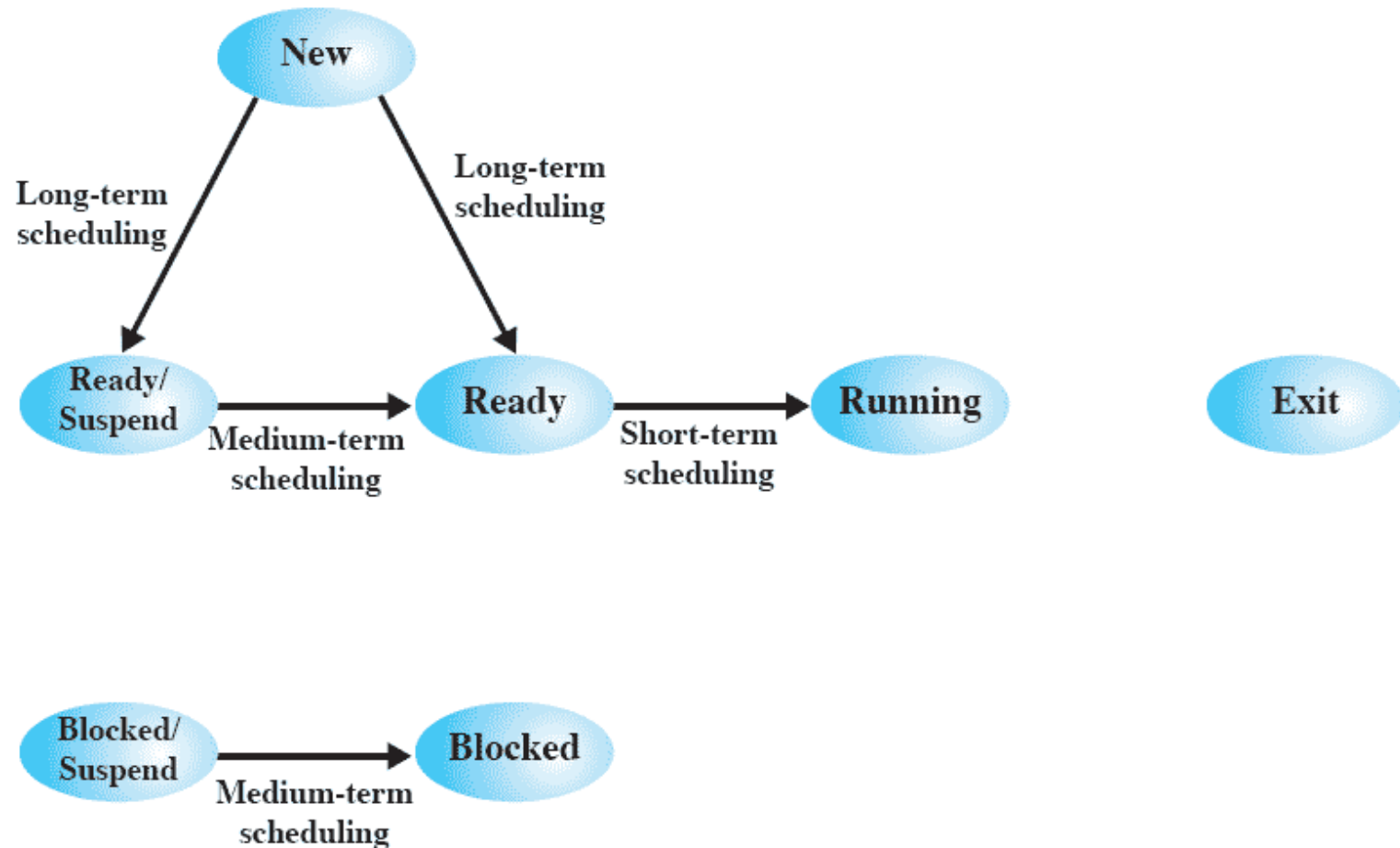
# Tipovi procesorskog raspoređivanja

- Srednjoročno raspoređivanje
  - ▣ Deo funkcije razmene
  - ▣ Odluka da se proces doda broju procesa koji su delimično ili potpuno u glavnoj memoriji
  - ▣ Određuje kada proces treba da bude zamenjen (suspendovan) na disk
  - ▣ Raspoređivanje se vrši tako da se zadovolji ciljni stepen multiprogramiranja

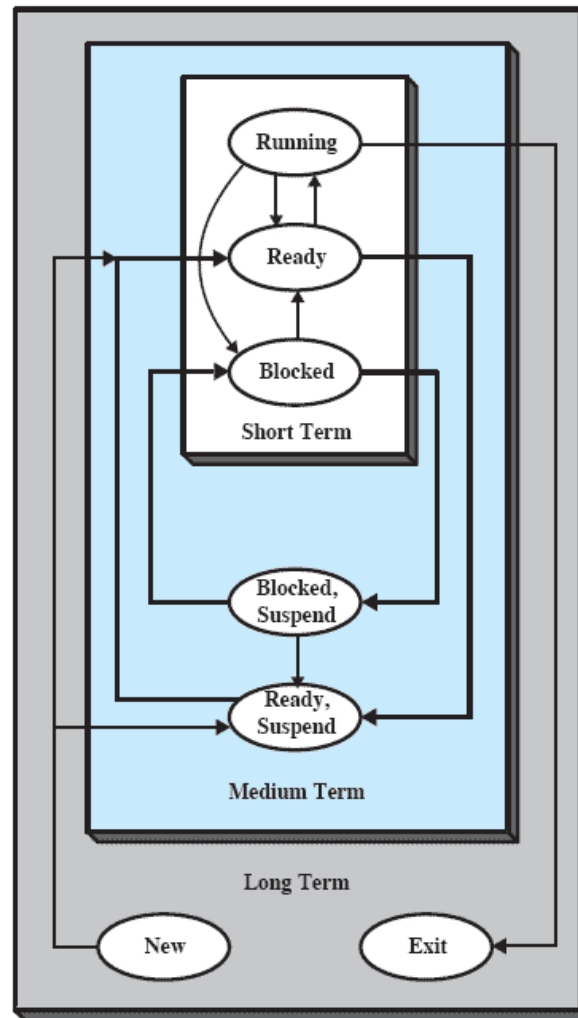
# Tipovi procesorskog raspoređivanja

- **Kratkoročno raspoređivanje**
  - ▣ Naziva se i dispečer
  - ▣ Najčešće se izvršava
  - ▣ Instrukcije kojeg procesa će procesor da izvršava
  - ▣ Aktivira se (ako je potrebno) kada se desi događaj
    - Prekidi generatora takta
    - U/I prekidi
    - Pozivi operativnog sistema
    - Signaliziranja (npr. od strane semafora)

# Raspoređivanje u životnom ciklusu procesa



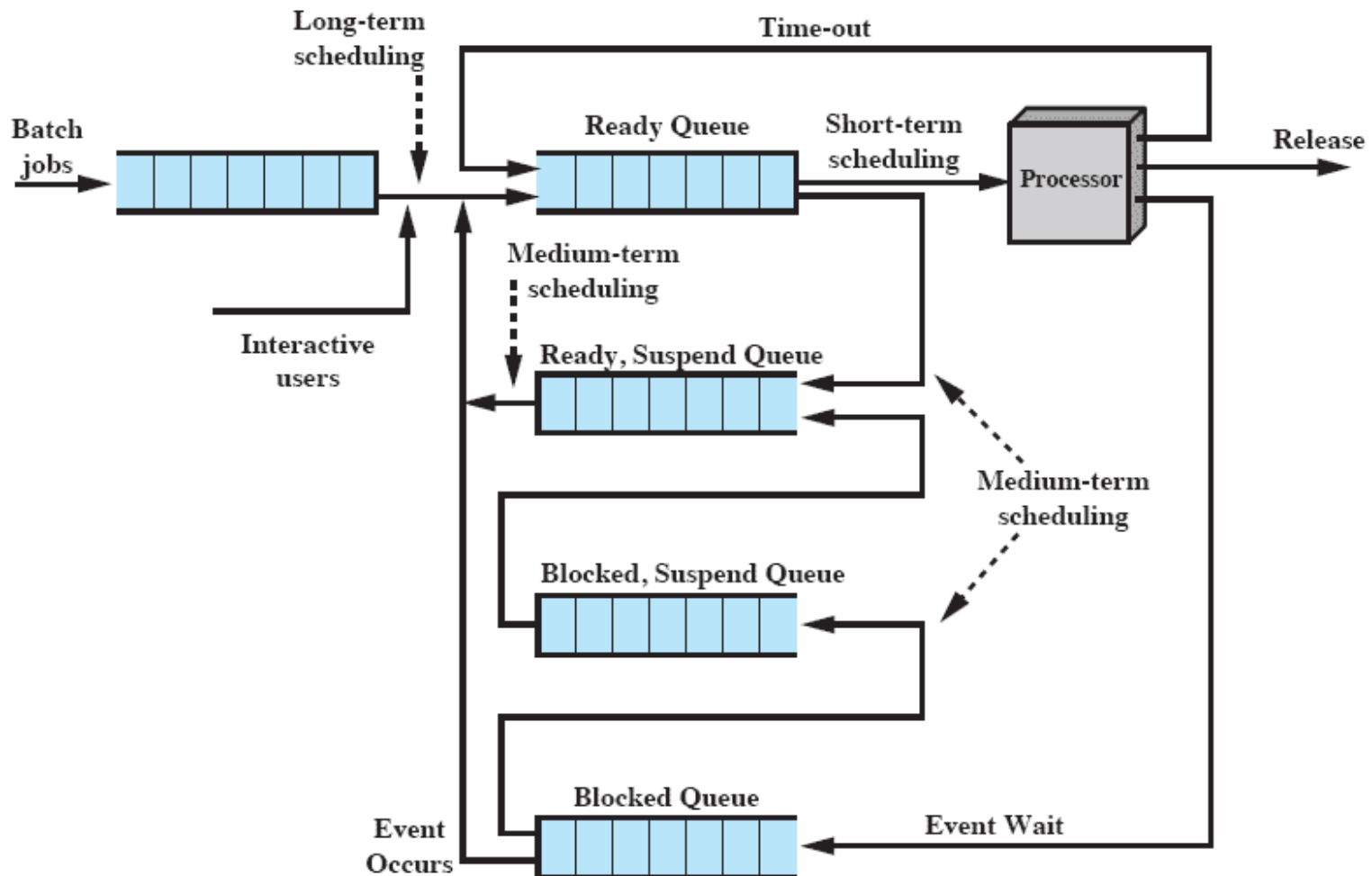
# Raspoređivanje u životnom ciklusu procesa





# Raspoređivanje u životnom ciklusu procesa

- Raspoređivanje je upravljanje redovima čekanja



# Kriterijumi kratkoročnog raspoređivanja

- Glavni cilj je da se procesorsko vreme dodeljuje na način da optimizuje različite aspekte ponašanja sistema
- Ustanovljava se skup kriterijuma na osnovu kojih se ocenjuju različite politike raspoređivanja

# Dimenzije kriterijuma

## Stanovište korisnika vs stanovište sistema

- Korisnički orijentisani kriterijumi
  - ▣ Ponašanje programa kako ga vidi korisnik
  - ▣ Npr. vreme odziva
    - Vreme proteklo od podnošenja zahteva do dobijanja odziva kao izlaza
    - Cilj je politika raspoređivanja koja obezbeđuje dobro vreme odziva



# Dimenzije kriterijuma

## Stanovište korisnika vs stanovište sistema

- **Sistemske orijentisane kriterijume**
  - ▣ Usredsređeni na delotvorno i efikasno iskorišćenje procesora
  - ▣ Npr. propusna moć
    - Broj obrađenih procesa u jedinici vremena
    - Cilj je politika raspoređivanja koja će uvećati propusnu moć



# Dimenzije kriterijuma

Povezano vs nepovezano sa performansama

- Razlikujemo kriterijume
  - ▣ Koji se odnose na performansu i
  - ▣ Koji nisu povezani sa performansom
- Kriterijumi vezani za performansu
  - ▣ Obično su kvantitativni i merljivi
  - ▣ Npr. vreme odziva i propusna moć
- Kriterijumi koji se ne odnose na performansu
  - ▣ Kvalitativni i teže merljivi
  - ▣ Npr. predvidljivost
    - Da usluga pokazuje iste karakteristike tokom vremena

# Kriterijumi kratkoročnog raspoređivanja

- Korisnički orijentisani, u vezi sa performansom
  - ▣ Vreme prolaska zadatka
    - Vreme od podnošenja zahteva do završetka procesa
    - Uključuje i vreme koje proces provede čekajući na resurse
  - ▣ Vreme čekanja
    - Vreme koje je proces proveo u redovima čekanja
  - ▣ Vreme odziva
    - Vreme od podnošenja zahteva do početka primanja odziva
  - ▣ Rokovi
    - Politika raspoređivanja treba da obezbedi da se proces završi u zadatom roku (ako takav rok postoji)
    - Drugi ciljevi se podređuju ispunjenju roka

# Kriterijumi kratkoročnog raspoređivanja

- Korisnički orijentisani, ostali kriterijumi
  - ▣ Predvidljivost
    - Posao treba da se završi sa približno istim vremenom i troškovima bez obzira na opterećenje sistema

# Kriterijumi kratkoročnog raspoređivanja

- Sistemski orijentisani, u vezi sa performansom
  - ▣ Propusna moć
    - Broj završenih procesa u jedinici vremena
    - Cilj je da se uveća propusna moć
  - ▣ Iskorišćenje procesora
    - Procenat vremena u kojem je procesor zauzet
    - Cilj je da procesor bude što više zauzet



# Kriterijumi kratkoročnog raspoređivanja

## □ Sistemski orijentisani, ostali

### □ Pravičnost

- Procesi treba da ravnopravno dele procesorsko vreme (osim ako je upravljanjem drugačije određeno)
- Procesi ne smeju da gladuju

### □ Primena prioriteta

- Ako su procesima dodeljeni različiti prioriteti, politika raspoređivanja treba da daje prednost procesima sa višim prioritetom

### □ Uravnoteženje resursa

- Raspoređivanje treba da optimizuje upotrebu resursa
- Npr. prednost se može dati procesima koji manje koriste preopterećene resurse

# Režim izbora procesa za izvršavanje

- Određuje trenutke u vremenu kada se vrši izbor narednog procesa kojem će biti dodeljen procesor
- Bez prekidanja
  - ▣ Proces se izvršava dok god se sam ne završi ili se sam blokira da sačeka U/I ili uslugu OS
- Sa prekidanjem
  - ▣ OS može da prekine proces koji se izvršava i postavi ga u stanje spreman
  - ▣ Na osnovu prekida generatora takta OS može da prekine proces
    - Ako mu je isteklo maksimalno vreme neprekinutog izvršavanja
    - Ako raspoređivač želi da da prednost drugom procesu
    - ...

# Upotreba prioriteta

- Procesima se dodele prioriteti
  - ▣ Proces višeg prioriteta ima prednost

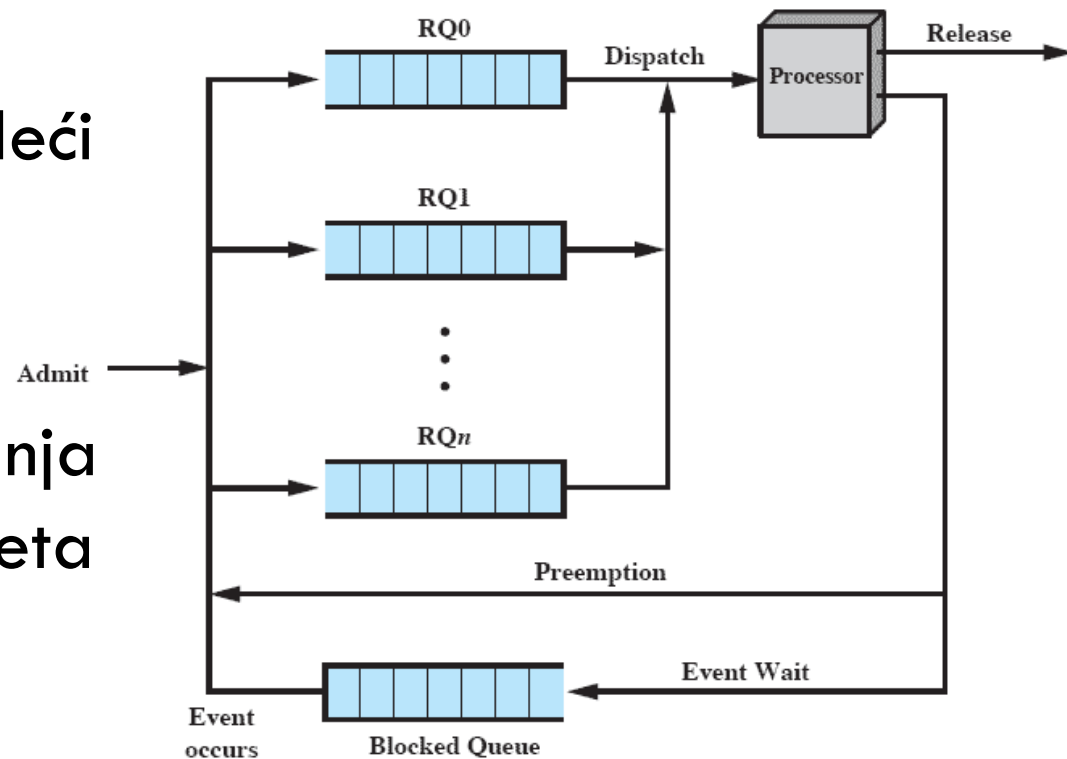


# Prioritet

- Obično brojna vrednost
  - ▣ Nema jedinstvenog pravila da li niži broj predstavlja viši ili niži prioritet
- Može biti definisan
  - ▣ Interno
    - Prioritet proističe i izračunava se iz podataka procesa
    - Npr. Na osnovu veličine memorijskih zahteva, količine utrošenog procesorskog vremena itd.
  - ▣ Eksterno
    - Prioritet je definisan u procesu izvan OS
    - Npr. na osnovu važnosti procesa, tipa procesa itd.

# Upotreba prioriteta

- Raspoređivač bira uvek proces sa višim prioritetom kao sledeći za izvršavanje
- Različiti redovi čekanja za svaki nivo prioriteta



# Upotreba prioriteta

## □ Gladovanje

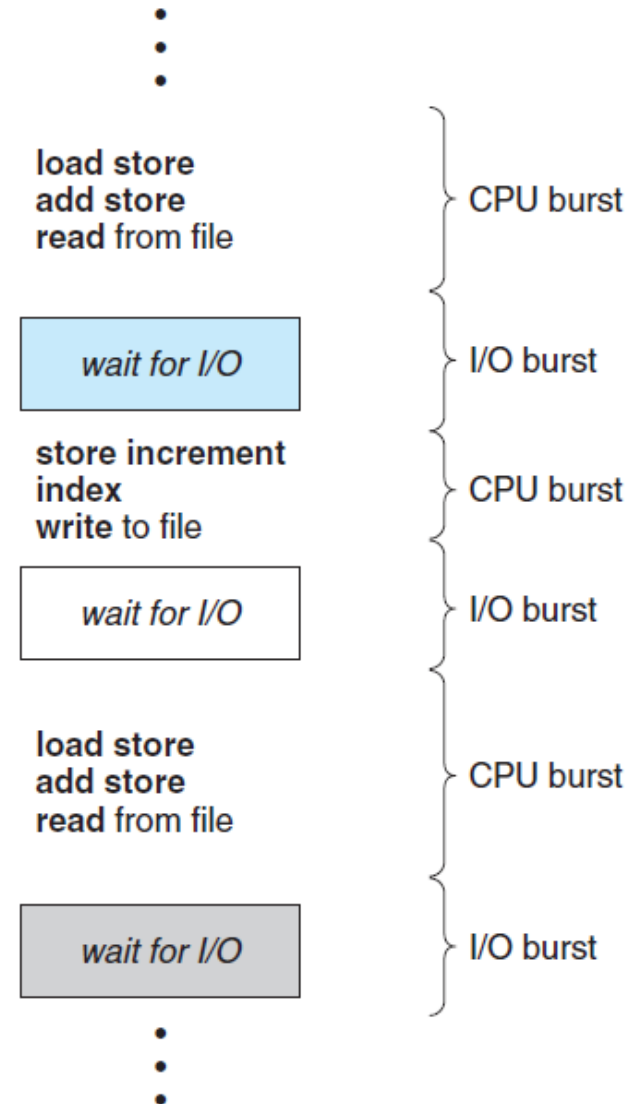
- ▣ Procesi sa nižim prioritetom ne dobijaju procesor ako uvek ima onih sa višim
- ▣ Jedno rešenje je promena prioriteta zavisno od starosti i istorije izvršavanja

Postoji priča da je kada je 1973. jedan IBM server na MIT-u ugašen, evidentiran proces niskog prioriteta startovan 1967. koji još nije stigao da se izvrši



# Ciklusi izvršavanja procesa

- CPU-burst za neki proces
  - ▣ Vreme proteklo od trenutka dobijanja procesora do gubljenja procesora
- CPU-burst se završava blokiranjem procesa ili vraćanjem u red spremnih zbog isteka određenog vremena
- Proces u toku životnog ciklusa obično više puta dobija i gubi procesor
  - ▣ CPU-burst se menja u toku izvršavanja
- Procesi sa intenzivnom potrebom procesora
  - ▣ Dugačak CPU-burst
- Procesi sa intenzivnim U/I
  - ▣ Kratak CPU-burst



# Politike raspoređivanja procesa

1. Po redosledu dolaska
2. Kružno dodeljivanje
3. Najkraći proces sledeći
4. Najkraće preostalo vreme
5. Sledeći sa najvećim odnosom odziva
6. Povratna sprega



# Parametri za testiranje politika

- Posmatramo jedan CPU-burst za svaki proces
- Trenutak dolaska
  - ▣ Trenutak u kojem proces postaje spreman
- Trenutak završetka
  - ▣ Trenutak kada je izvršavanje instrukcija kompletirano
- Vreme usluge ( $T_s$ )
  - ▣ Vreme potrebno procesoru da izvrši instrukcije procesa
- Vreme prolaska ( $T_r$ )
  - ▣ Vreme koje proces provodi u sistemu
  - ▣ Na vreme usluživanja dodato vreme čekanja
- Normalizovano vreme prolaska ( $T_r / T_s$ )
  - ▣ Relativno kašnjenje procesa
  - ▣ U najboljem slučaju je 1.0, inače je veće od toga

*Primer*

Proces	Trenutak dolaska	Vreme usluživanja
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

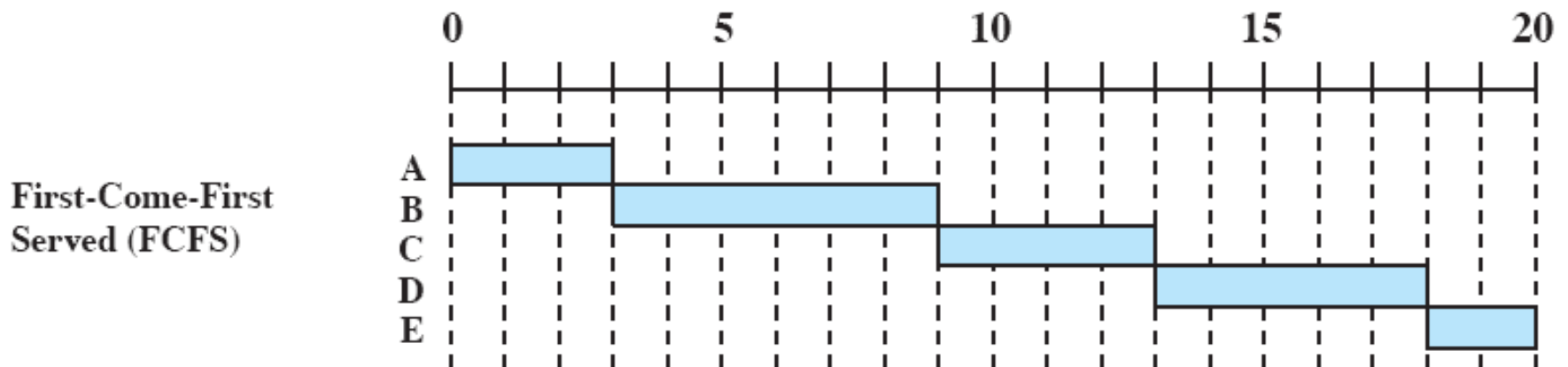
# Prema redosledu dolaska

- FCFS – *First-come-first-served*



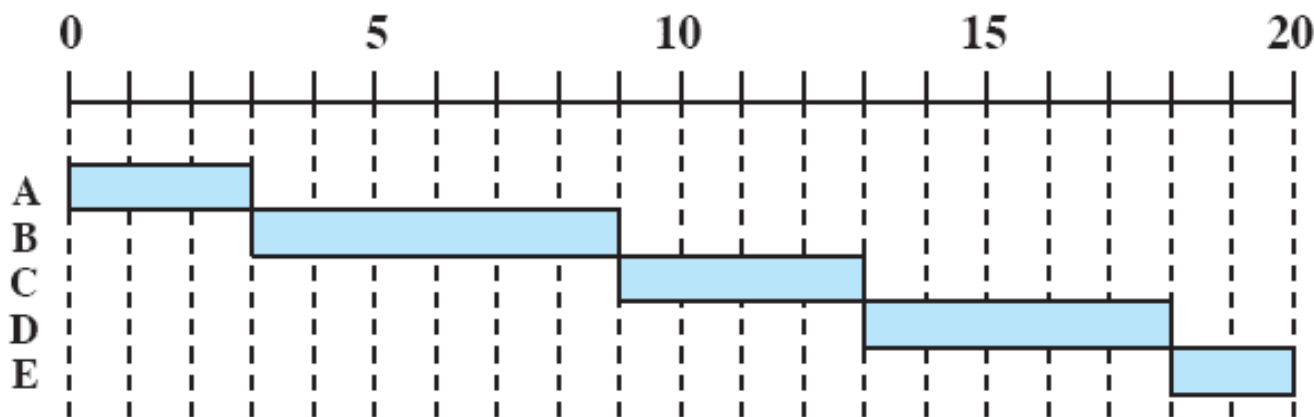
# Prema redosledu dolaska

- Proces se stavlja u red spremnih procesa
- Politika bez prekidanja
- Kada tekući proces prestane da se izvršava bira se proces koji je najduže bio u redu čekanja



# Prema redosledu dolaska

First-Come-First  
Served (FCFS)



Proces	A	B	C	D	E	Prosek
Vreme završetka	3	9	13	18	20	
Vreme prolaska zadatka ( $T_r$ )	3	7	9	12	12	8.60
$T_r / T_s$	1.00	1.17	2.25	2.40	6.00	2.56

# Prema redosledu dolaska

- Kratki procesi moraju dugo da čekaju da se izvrše
  - ▣ Kada dočekaju izvršiće se za kratko vreme
  - ▣ Loš odnos vremena prolaska i vremena usluge
- Bolje prolaze procesi koji intenzivno koriste procesor
  - ▣ Procesi sa dosta U/I kratko koriste procesor, a nakon U/I ponovo moraju dugo da čekaju procesor
- FCFS nije dobra politika sama po sebi
  - ▣ U kombinaciji sa prioritetima je efektivna

# Kružno dodeljivanje

- RR – *round robin*



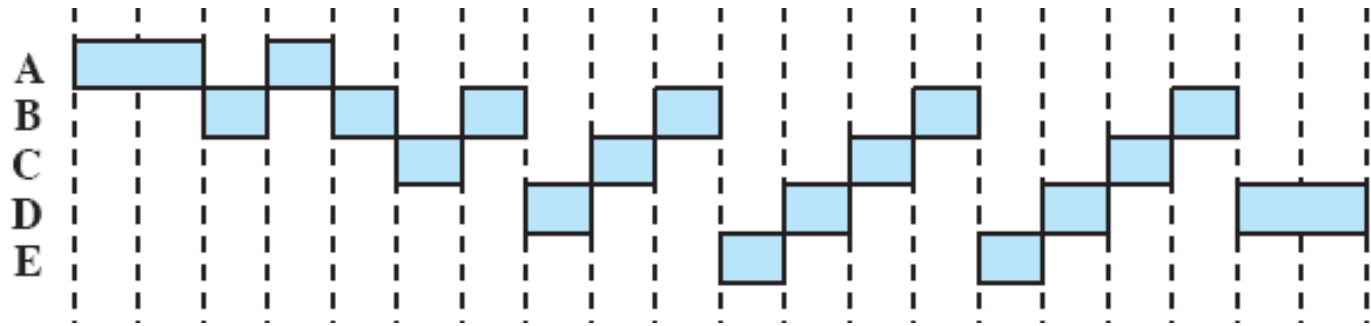
# Kružno dodeljivanje

- Prekidna politika
- Deljenje na vremenske isečke (*time slicing*)
  - ▣ Svaki proces dobija isečak vremena pre nego što bude prekinut
  - ▣ Generator takta periodično izaziva prekid
  - ▣ Ako je istekao vremenski isečak trenutno izvršavani proces se stavlja u red čekanja
  - ▣ Sledeći spremni proces se bira na osnovu FCFS

# Kružno dodeljivanje

□  $q$  – dužina vremenskog isečka (quantum)

Round-Robin  
(RR),  $q = 1$



Proces	A	B	C	D	E	Prosek
Vreme završetka	4	18	17	20	15	
Vreme prolaska zadatka ( $T_r$ )	4	16	13	14	7	10.80
$T_r / T_s$	1.33	2.67	3.25	2.80	3.50	2.71



# Kružno dodeljivanje

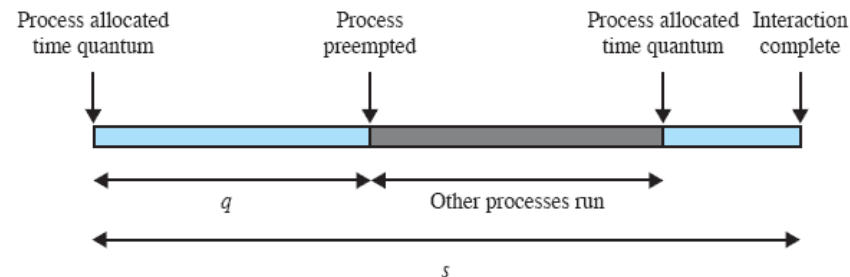
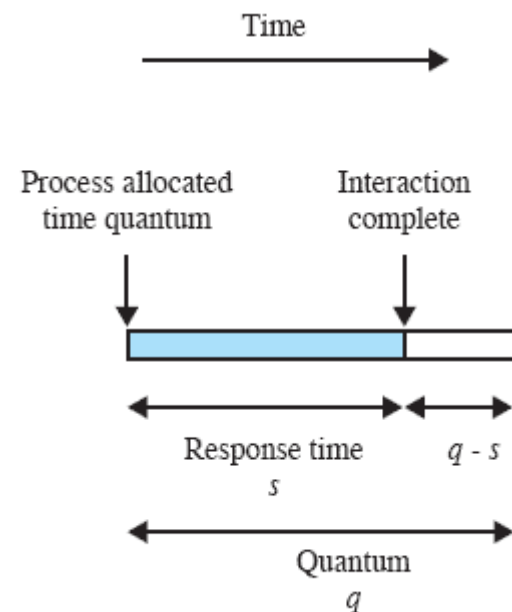
- Bolje prolaze procesi sa intenzivnom upotrebom procesora
  - ▣ Iskoriste ceo vremenski isečak
  - ▣ Odmah se postavljaju u red spremnih procesa
- Procesi sa dosta U/I
  - ▣ Kratko koristi isečak
  - ▣ Blokira se zbog U/I
  - ▣ Kada se završi U/I dodaje se u red spremnih

# Kružno dodeljivanje – veličina isečka

- Kako izabrati dužinu vremenskog isečka?
- Ako je isečak suviše mali
  - ▣ Česta prekidanja i komutacije
  - ▣ Dodatna režija i usporenje zbog toga
- Ako je isečak suviše veliki
  - ▣ RR se degeneriše u FCFS

# Kružno dodeljivanje – veličina isečka

- Ako je isečak veći od vremena za interakciju sa procesorom
- Ako je isečak manji od vremena za interakciju sa procesorom



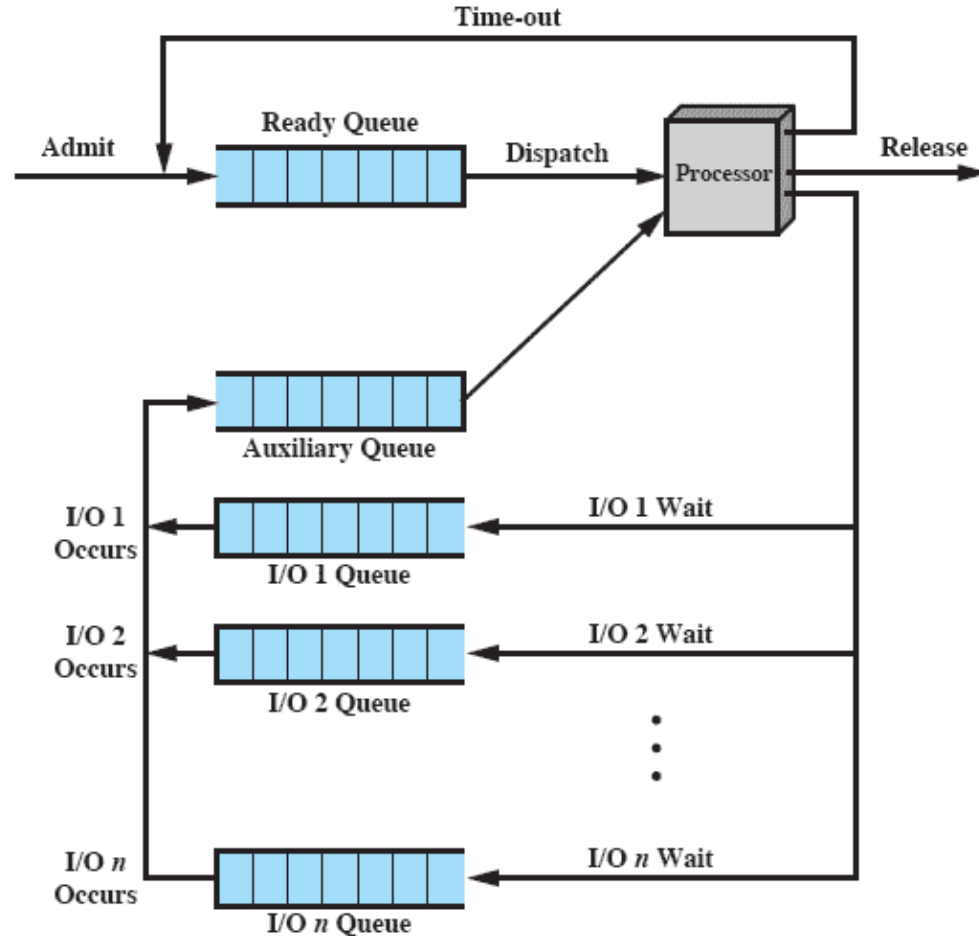
# Kružno dodeljivanje – veličina isečka

- Isečak bi trebao da bude neznatno veći od prosečnog vremena za interakciju sa procesorom
  - ▣ Na ovaj način proces uglavnom obavi posao ili se blokira zbog U/I pre isteka vremenskog isečka
- U većini modernih sistema vremenski isečak je
  - ▣ Između 10 i 100 ms
- Procenjeno je da isečak treba da bude toliki da 80% CPU-bursts bude kraće od isečka

# Virtuelno kružno dodeljivanje

- VRR – *Virtual round robin*
- Rešava nepravedan tretman procesa sa dosta U/I kod klasičnog RR algoritma
- Nakon završetka U/I operacije
  - ▣ Proces se ubacuje u poseban FCFS red čekanja
  - ▣ Dispečer daje prednost procesima iz ovog reda u odnosu na glavni red spremnih procesa
  - ▣ Procesi iz pomoćnog reda dobijaju isečak vremena umanjen za vreme koliko su se prethodni put izvršavali kad su uzeti iz reda spremnih procesa

# Virtuelno kružno dodeljivanje



# Najkraći proces sledeći

- SPN – *Shortest process next*



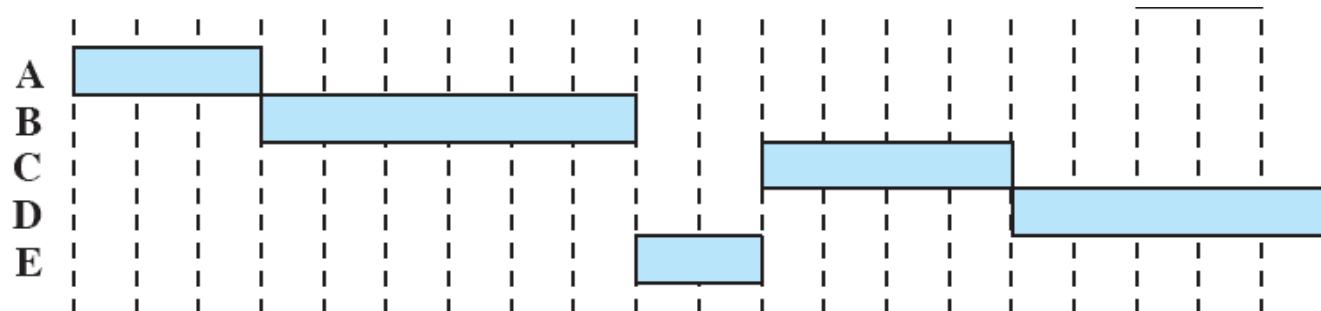
# Najkraći proces sledeći

- Politika bez prekidanja
- Bira se proces sa najkraćim očekivanim CPU-burst
  - ▣ Prikladnije ime za ovaj algoritam bi bilo  
*Shortest CPU-burst next*
- Kratak proces skače na čelo reda ispred dugih procesa



# Najkraći proces sledeći

Shortest Process  
Next (SPN)



Proces	A	B	C	D	E	Prosek
Vreme završetka	3	9	15	20	11	
Vreme prolaska zadatka ( $T_r$ )	3	7	11	14	3	7.60
$T_r / T_s$	1	1.17	2.75	2.80	1.50	1.84

# Najkraći proces sledeći

- Smanjuje se predvidljivost vremena prolaska
  - ▣ Posebno za duže procese
- Dužim procesima preči gladovanje
  - ▣ Kraći procesi stalno uskaču ispred njih u redu čekaanja
- Nije poželjna za sisteme sa deljenjem vremena jer ne podržava prekide

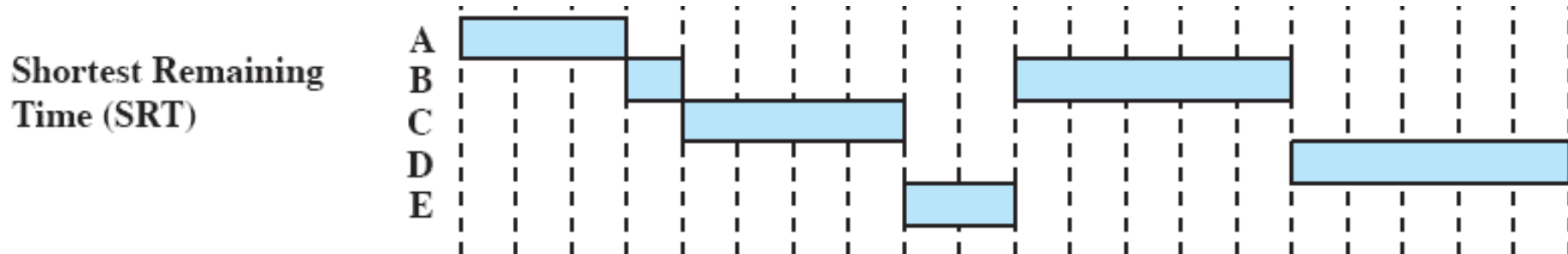
# Procena očekivanog CPU-burst

- Potrebno je proceniti koliki će biti sledeći CPU burst za proces
  - ▣ Može na osnovu trajanja prethodnih CPU-bursts procesa
- Najjednostavnija varijanta
  - ▣ Prosečno trajanje svih prethodnih CPU-bursts procesa
- Druga varijanta
  - ▣ Dati veću težinu skorijim instancama
  - ▣ Eksponencijalno usrednjavanje
  - ▣ Što je posmatranje starije to se manje uzima u obzir u proračunu srednje vrednosti

# Najkraće preostalo vreme

- SRT – *Shortest remaining time*
- Politika sa prekidanjem
- Prekidna varijanta SPN
- Kada se pojavi novi proces u redu spremnih procesa
  - ▣ Dobiće procesor umesto tekućeg procesa ako ima kraći očekivani CPU-burst

# Najkraće preostalo vreme



Proces	A	B	C	D	E	Prosek
Vreme završetka	3	15	8	20	10	
Vreme prolaska zadatka ( $T_r$ )	3	13	4	14	2	7.20
$T_r / T_s$	1	2.17	1	2.80	1	1.59

# Najkraće preostalo vreme

- Nema dodatnih prekida
  - ▣ Nema potrebe za prekidima u svakom taktu, jer je već najkraći proces dobio procesor
  - ▣ Prekid nastaje samo pojavom novog procesa
  - ▣ Samo novi proces u redu može da ima kraći CPU-burst
- Dodatna režija za evidenciju i predviđanje CPU-burst
- Radi veoma dobro u većini slučajeva
  - ▣ Ako se dobro predvidi CPU-burst

# Sledeći sa najvećim odnosom odziva

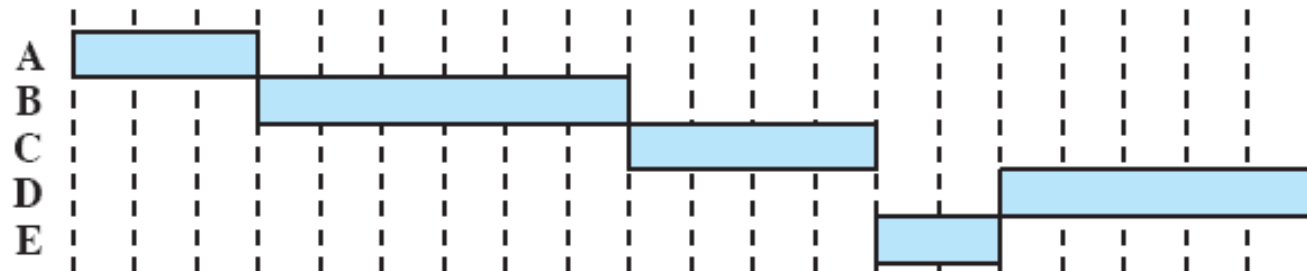
- HRRN – *Highest response ratio next*
- Politika bez prekidanja
- Za svaki proces se računa proporcija R

$$R = \frac{w+s}{s}$$

- ▣ w – vreme koje je proces proveo u čekanju na procesor do sada
  - ▣ s – očekivano vreme usluživanja
- Kada se tekući proces završi ili blokira
  - ▣ Bira se proces sa najvećom vrednošću R

# Sledeći sa najvećim odnosom odziva

Highest Response  
Ratio Next (HRRN)



Proces	A	B	C	D	E	Prosek
Vreme završetka	3	9	13	20	15	
Vreme prolaska zadatka ( $T_r$ )	3	7	9	14	7	8.00
$T_r / T_s$	1	1.17	2.25	2.80	3.5	2.14

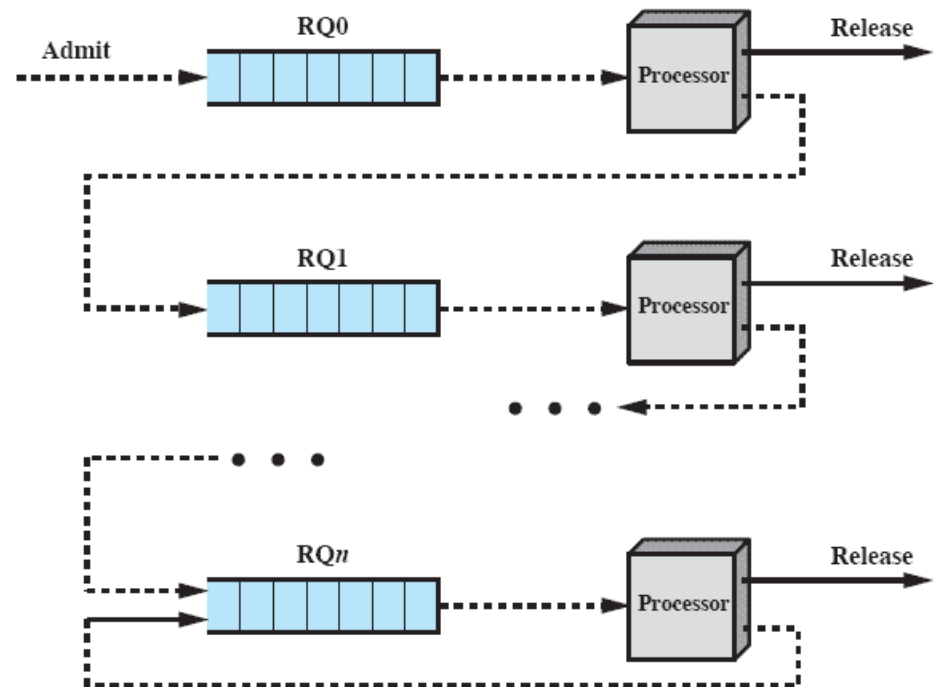


# Sledeći sa najvećim odnosom odziva

- Dobar tretman za kraće procese
  - $s$  je malo
  - $R$  je veliko zbog malog broja u imeniocu
- Dobar tretman za procese koji dugo čekaju
  - $w$  je veliko
  - $R$  je veliko zbog uvećanog brojioca
- Tako će se približno ravnopravno tretirati duži i kraći poslovi

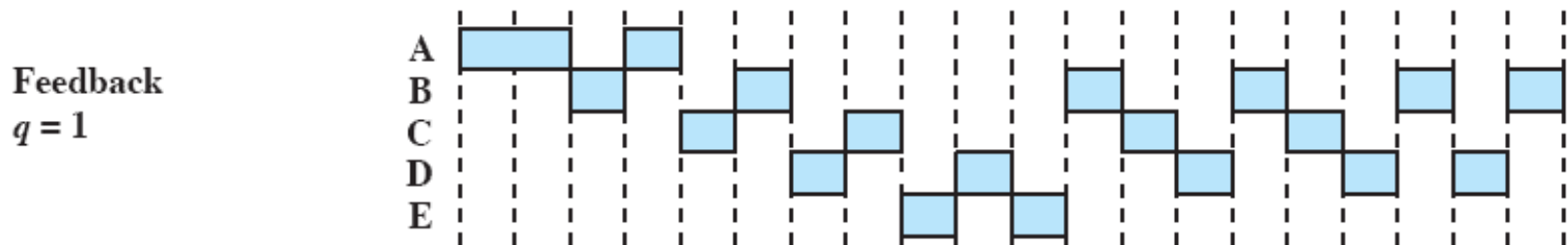
# Povratna sprega

- Prekidna politika
- Svaki put kada se proces prekine stavlja se u FCFS red čekanja nižeg prioriteta
- Brže dobijaju procesor noviji procesi
- Brže prolaze kraći procesi



# Povratna sprega

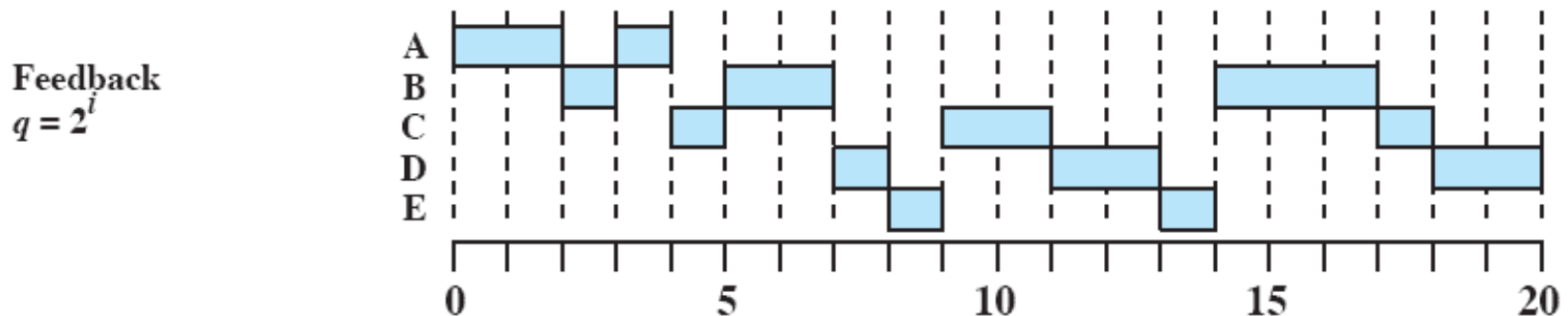
- Kvantum da svima bude isti
  - ▣ Dugi procesi mogu da gladuju



Proces	A	B	C	D	E	Prosek
Vreme završetka	4	20	16	19	11	
Vreme prolaska zadatka ( $T_r$ )	4	18	12	13	3	10.00
$T_r / T_s$	1.33	3.00	3.00	2.60	1.5	2.29

# Povratna sprega

- Kvantum da se menja dinamički
  - ▣ Kvantum obrnuto proporcionalan prioritetu
  - ▣ Procesi iz manje prioritetnih redova kada dobiju procesor imaju šansu da ga duže koriste



Proces	A	B	C	D	E	Prosek
Vreme završetka	4	17	18	20	14	
Vreme prolaska zadatka ( $T_r$ )	4	15	14	14	6	10.60
$T_r / T_s$	1.33	2.50	3.50	2.80	3.00	2.63

# Raspoređivanje sa fer deobom

- FSS – *Fair-share scheduler*
- Procesi se klasifikuju u grupe
  - ▣ Npr procesi koje je kreirala jedna aplikacija su ista grupa
- Raspoređivanje se vrši uzimajući u obzir i grupu kojoj proces pripada
- Prioritet procesa se izračunava na osnovu
  - ▣ Osnovnog prioriteta samog procesa
  - ▣ Dotadašnjeg vremena korišćenja procesora od strane procesa
  - ▣ Dotadašnjeg vremena korišćenja procesora od strane grupe kojoj proces pripada

# Linux raspoređivač

- *Completely Fair Scheduler*
  - ▣ Postoji od Linux kernela verzije 2.6.23
- Zasniva se na ideji da se procesima dodeljuje procenat procesorskog vremena zavisno od
  - ▣ Trenutnog broja procesa u sistemu
  - ▣ Prioriteta procesa
- Vremenski isečak se dodeljuje dinamički da bi se postigla planirana proporcija u korišćenju ukupnog procesorskog vremena

# Linux raspoređivač

- U sistemu sa  $N$  procesa istog prioriteta
  - ▣ Svaki proces bi dobio  $1/N$  procesorskog vremena
- Prioriteti služe kao težinski faktor pri rasporedeli procesorskog vremena
  - ▣ *nice* vrednost definiše prioritet
    - Procesi sa većom vrednošću imaju manji prioritet jer su više obzirni prema drugim procesima
    - Opseg *nice* vrednosti za standardne procese je od -20 do +19 (podrazumevana vrednost je 0)

# Linux raspoređivač

- Maksimalno kašnjenje
  - ▣ Konfigurabilna vrednost koja određuje dužinu vremenskog intervala u kojem svaki proces mora dobiti procesor bar jednom
  - ▣ Primer
    - Maksimalno dozvoljeno kašnjenje je 10 ms
    - Dva procesa u sistemu sa odnosom prioriteta 4:1
    - Prvi proces dobija isečak 8 ms, a drugi 2 ms



# Linux raspoređivač

- Sa prevelikim brojem procesa u sistemu
  - isečak bi postao veoma mali
  - troškovi komutacije bi bili preveliki
- Minimalna granularnost
  - Vrednost koja određuje minimalnu dužinu isečka bez obzira na broj procesa
  - Ovim se za veliki broj procesa gubi ravnopravnost po cenu upotrebljivosti sistema

# Linux raspoređivač

- Način odabira sledećeg procesa se ne zasniva na organizovanju procesa u redove
- Procesima se dodeljuje brojna vrednost
  - Virtuelno vreme izvršavanja
    - Računa se na osnovu prioriteta svih procesa u sistemu i dotadašnjeg vremena korišćenja procesora

# Linux raspoređivač

- Virtuelna vremena izvršavanja se organizuju u binarno stablo pretrage
- Za sledeći proces se bira onaj koji ima najmanje virtuelno vreme izvršavanja (proces krajnje levo u stablu)

