S A D R Ž A J

3.1-Raspoređivanje sa predpražnjenjem

Prebacivanjem CPU-a sa procesa na proces povećava se produktivnost sistema

- **Planiranje je osnovna funkcija** operativnog sistema, jer se skoro svi računarski resursi planiraju pre upotrebe
- CPU mora da odgovori na trapove, programske zahteve i U/I prekide
- Osnovni zadatak upravljanja procesorom je prebacivanje procesora sa **aktivnog procesa na** neki od **spremnih procesa.**
- O izboru spremnog procesa, na koga se prebacuje procesor, brine se poseban proces koji se zove **raspoređivanje** (*scheduling).*
- Cilj raspoređivanja je da **ravnomerno rasporedi procesorsko vreme** između istovremeno postojećih procesa koji su u ***Ready*** stanju.
- Omogućuje da se zadovolje principi ***multiuser*** i ***multitasking*** rada
- **Asihroni** i **sinhroni** procesi

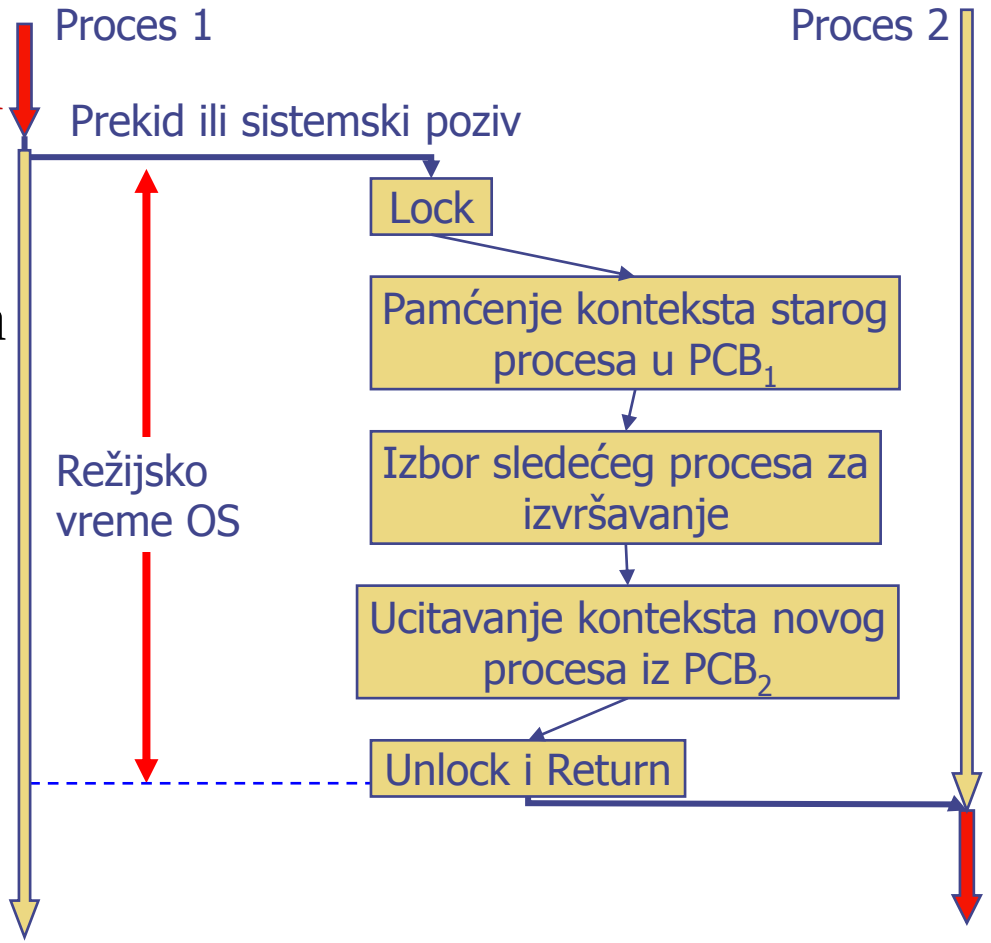
3.1-Raspoređivanje procesa

Procesor se dodeljuje drugom procesu kada:

1. Proces pređe u stanje čekanja na resurs
2. Proces *roditelj* čeka da se završi proces *dete*
3. Prilikom prelaska iz stanja RUN u STOP
4. Prilikom prelaska iz stanja RUN u READY
5. Prilikom prelaska iz stanja WAIT u READY

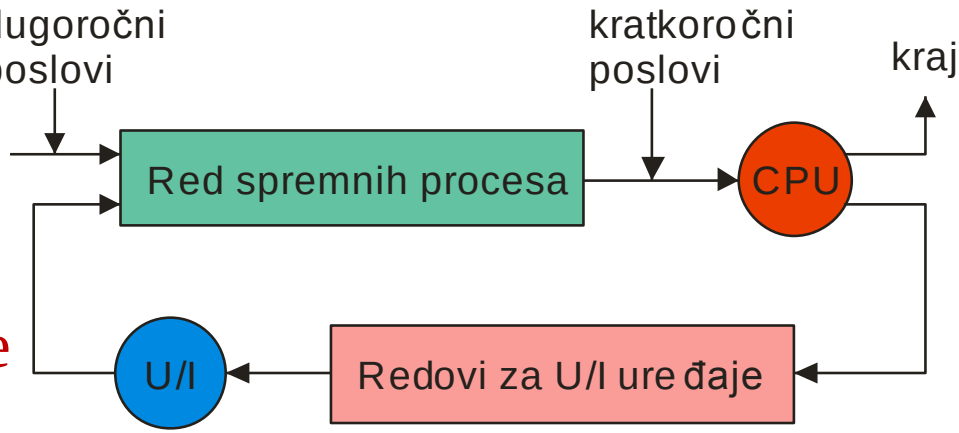
3.1-Raspoređivanje procesa

- Pod raspoređivanjem procesa podrazumevamo promenu aktivnog kontrolnog bloka procesa (**PCB**) na koji pokazuje CPU.
- Ova promena naziva se **kontekstno prebacivanje** (*context switch*).
- Kada CPU prelazi na drugi proces, sistem mora da upamti status starog procesa i napuni podatke za izvršavanje novog procesa - status novog procesa
- Vreme kontekstnog prebacivanja je režija OS – trošak sistema, tj. sistem ne radi koristan posao za korisnika.
- Period trajanja kontekstnog prebacivanja zavisi od hardverske podrške.

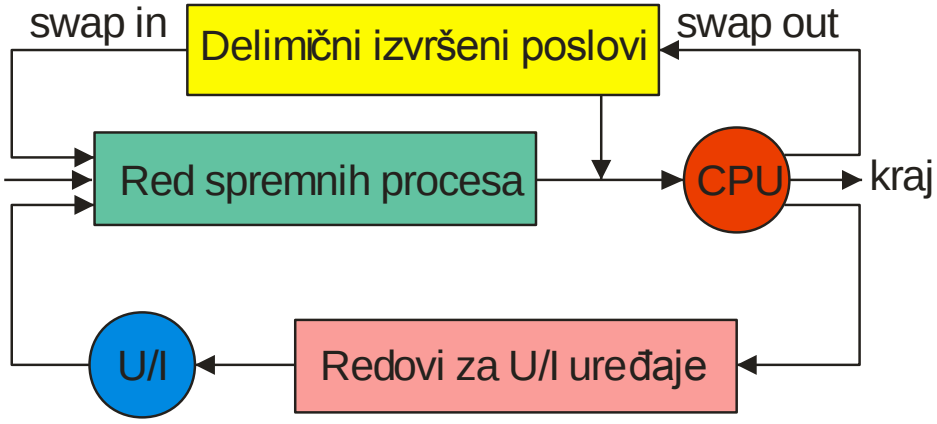


3.1 Vrste i uloga planera

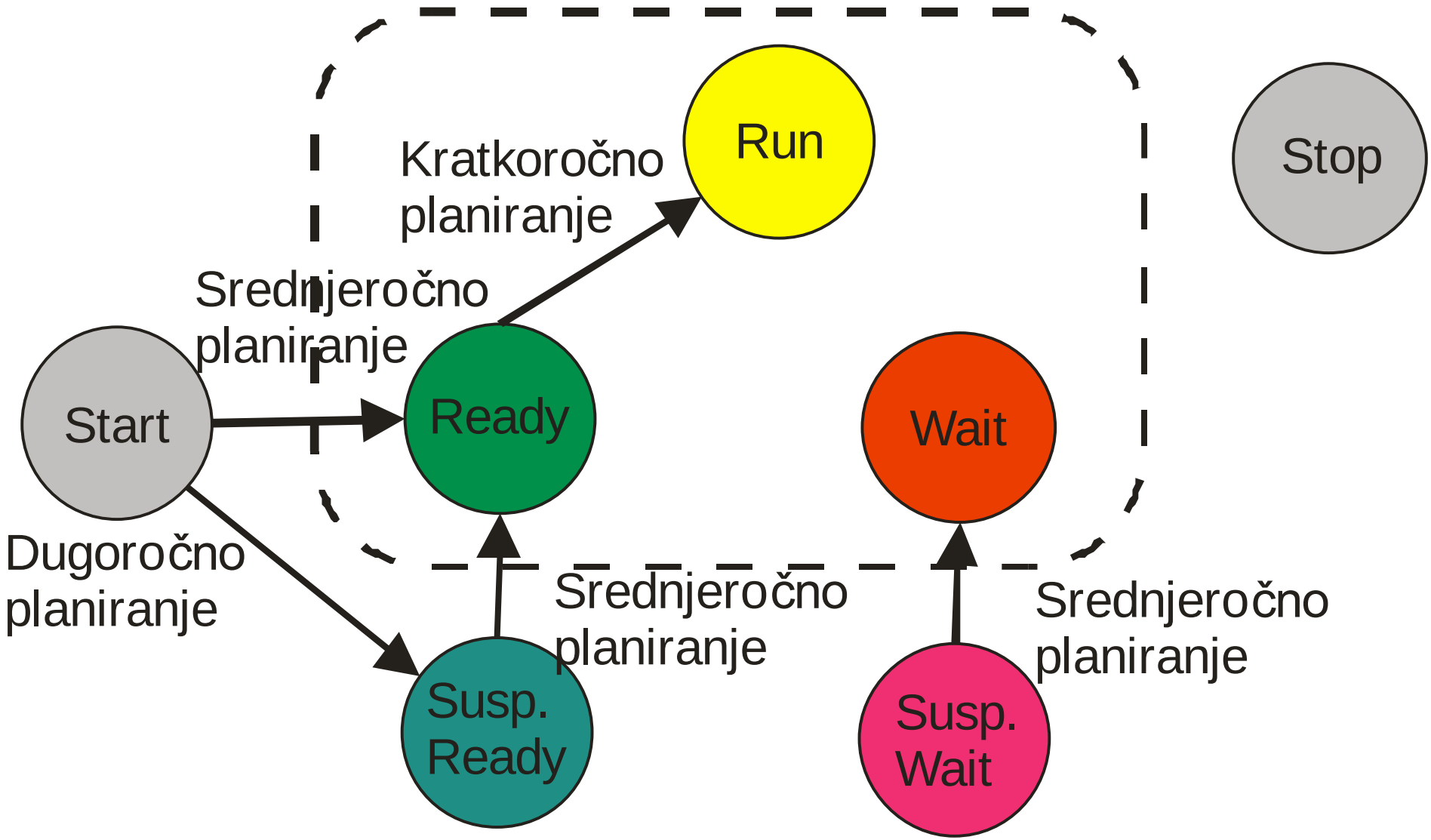
- Operativni sistem ima više planera: ***dugoročni, srednjeročni*** **i** ***kratkoročni***.
- **Dugoročni planer** (*long term* ili planer poslova) određuje koji će se poslovi propustiti u sistem.



- U paketnim sistemima ima više poslova koji su u sistemu nego onih koji mogu da budu izvršeni. Poslovi koji čekaju kopiraju se na disk.
- Dugoročni planer bira poslove iz pula poslova i puni ih u memoriju.
- **Srednjeročno planiranje**(*medium)*se odnosi na suspendovane procese
- **Kratkoročni planer** (*short term planer CPU-a*) bira jedan od tih spremnih procesa koji se nalaze u memoriji i dodeljuje mu procesor.
- Osnovna razlika ova dva planera je u frekvenciji njihovog izvršavanja.



3.1 Vrste i uloga planera



3.1-Kriterijumi dodele procesora

Korisnički kriterijumi:

- **Predvidljivost** – potrebno je unapred poznavati karakteristike procesa (vreme za njegovo izvršavanje)
- **Rokovi** – vreme za koje neki posao treba da se završi
- **Vreme obilaska** – vreme koje protekne od trenutka nailska posla pa do njegovog završavanja
- **Vreme čekanja** – vreme koje protekne od trenutka nailaska posla pa sve dok mu se ne dodeli procesor tj. ne počne da se izvršava
- **Vreme odziva** – to je vremenski interval od trenutka kada je korisnik završio unošenje svog zahteva do trenutka kada sistem počinje da daje neki odgovor ta taj zahtev.
- **Normalizovano ukupno vreme** - to je odnos ukupnog vremena prema vremenu usluge. Ta vrednost ukazuje na relativno kašnjenje koje doživljava proces. Minimalna vrednost za taj odnos je **1** (proces je odmah izvršen), a veća vrednost odgovara smanjenju nivoa usluge.

3.1-Kriterijumi dodele procesora

Sistemski kriterijumi:

- Iskorišćenost procesora – varira od 0% do 100 %
- Propusna moć – broj poslova koji se mogu završiti u jedinici vremena
- Pravičnost – ravnopravno izvršavanje svih procesa koji se nalaze u redu spremnih (*ready*) procesa
- Uravnoteženja resursa– ravnomerno korišćenje resursa
- Na osnovu prioriteta – omogućiti da se procesi većeg prioriteta ranije završe (sistemski poslovi, *real time* zadaci)

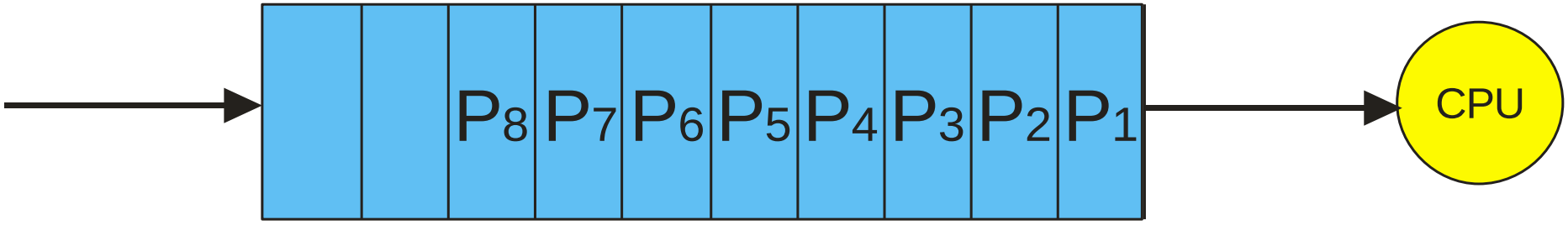
3.2 - Algoritmi za dodelu

1. FCFS – *First Come First Servered*
2. SJF – *Shortest Job First*
   - *Sa predpražnjenjem*
   - *Bez predpražnjenja*
3. Na osnovu prioriteta
4. RR – *Round Robin*
5. Višenivovski redovi
6. Višenivovski redovi sa povratnom spregom
7. *Real-Time*
8. Multiprocesorski

3.2-First Come, First Serverd

- Najednostavniji algoritam planiranja poslova
- Procesor se dodeljuje onom procesu koji ga je prvi zahtevao.
- Implementacija ovog algoritma se lako vrši korišćenjem FIFO redova.
- Kada se proces ubacuje u red spremnih procesa njegov PCB se stavlja na kraj reda a kada se procesor oslobodi on se dodeljuje prvom procesu koji se nalazi na početku reda
- Performanse ovog algoritma su veoma slabe.

FIFO (*First Input First Output*)



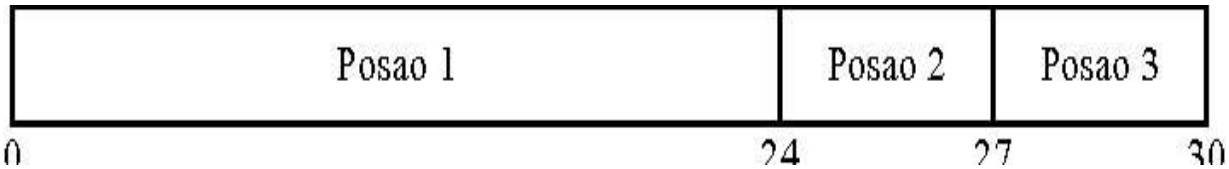
3.2-First Come, First Serverd

PRIMER:

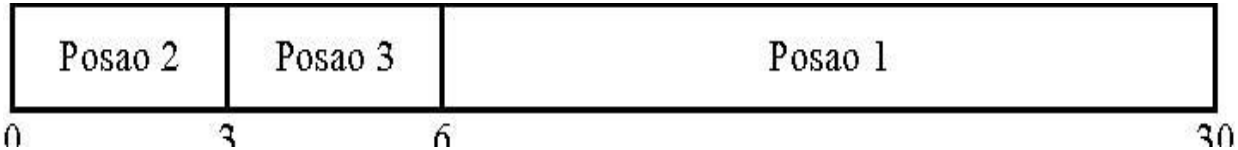
Posmatrajmo sledeća tri posla prikazana u tabeli:

Posao	Vreme izvršavanja
1	24
2	3
3	3

- Ako su ovi poslovi pristigli u redosledu navođenja i opslužuju se na osnovu **FCFS** algoritma, rezultat ovoga vidimo u sledećoj ***Gantovoj*** karti:



- Vreme obilaska za prvi posao je 24, za drugi 27 a za treći 30.
- Srednje vreme obilaska je 27.
- Ako su poslovi pristigli u redosledu 2, 3, 1, dobićemo sledeću Gantovu kartu:

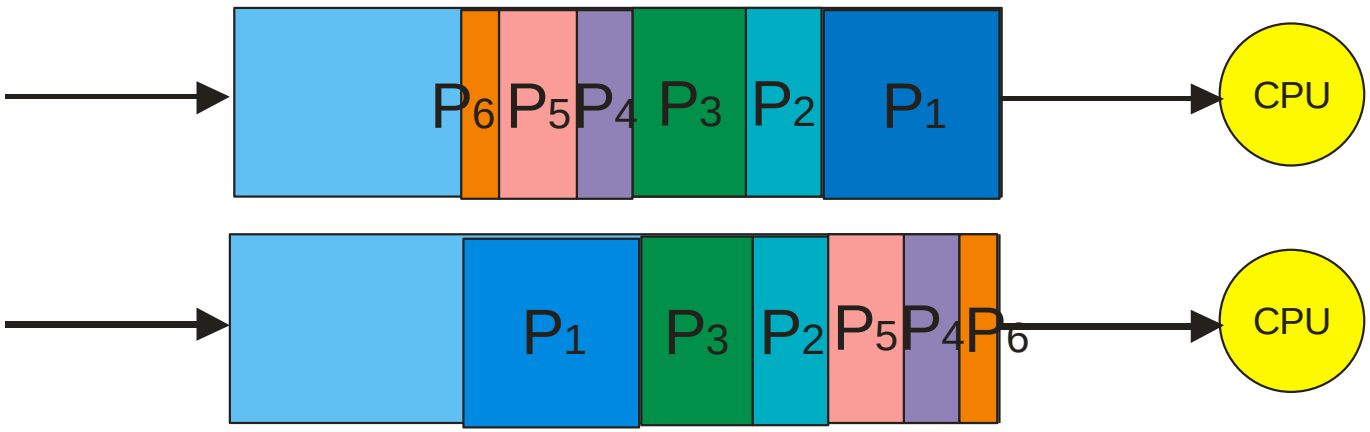


- Sada je srednje vreme obilaska jednako 13 što je značajno smanjenje.
- Srednje vreme obilaska za ovaj algoritam u opštem slučaju nije minimalno moguće, a može značajno da varira u zavisnosti od konkretnog slučaja.

3.2-Shortes Job First (SJF)

- Različit pristup planiranju poslova prisutan je kod SJF (*Najkraći posao prvo*)
- Svakom poslu se pridružuje i podatak o dužini njegovog sledećeg CPU ciklusa.
- Kada je procesor slobodan on se dodeljuje onom poslu kod koga je ta dužina najmanja.
- Ukoliko postoje dva ili više poslova iste dužine, kao sekundarni kriterijum koristi se FCFS.

SJF (*Shoties Job First*)



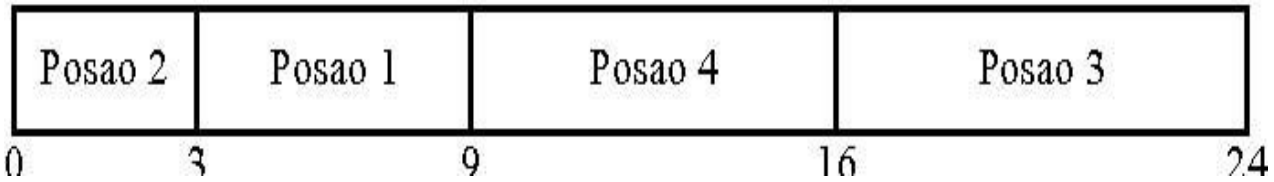
3.2-Shortes Job First (SJF)

PRIMER:

Posmatrajmo sledeća tri posla prikazana u tabeli:

Posao	Vreme izvršavanja
1	6
2	3
3	8
4	7

Upotreba SJF algoritma daje sledeću Gantovu kartu:



Za srednje vreme obilaska ovde se dobija da je (3+9+16+24)/4=**13**

- Ovaj algoritam je dokazano optimalan, jer daje minimalno srednje vreme čekanja za neki skup poslova.
- U dokazu se pokazuje da premeštanje kraćeg posla pre dužeg smanjuje vreme čekanja kraćeg posla u većoj meri nego što povećava vreme čekanja dužeg posla.
- Prema tome, smanjuje se srednje vreme čekanja.
- Problem je što mi ne znamo uvek koliko će pojedini posao da traje tako da je primena ovog algoritma dosta otežana

3.2-Shortes Remaining Time First

- Do sada opisani algoritmi su algoritmi bez prekidanja.
- SJF algoritam može biti realizovan i bez prekidanja i sa prekidanjem.
- Dilema nastaje kada novi posao može da ima kraći CPU ciklus nego što je potrebno za dovršetak posla kome je već dodeljen procesor.
- SJF sa prekidanjem će prekinuti tekući posao-proces koji se trenutno izvršava, dok SJF bez prekidanja dozvoljava da se dovrši tekući posao pa tek onda prelazi na izvršavanje sledećeg procesa.
- Varijanta SJF algoritma sa prekidanjem naziva se **SRTF** (*Shortest-Remainig-Time-First*).
- Algoritam na osnovu prioriteta takođe može biti sa i bez prekidanja.
- Kada posao pristiže u red spremnih procesa njegov prioritet se poredi sa prioritetom posla koji se izvršava. Ako je prioritet novog posla viši, tekući posao će se prekinuti i procesor dodeliti novom poslu.
- Algoritmi bez prekidanja (naročito FCFS) nisu pogodni za sisteme sa deljenjem procesorskog vremena, gde je neophodno da svaki korisnik dobije deo procesorskog vremena u pravilnim intervalima.

Raspoređivanje na osnovu prioriteta

- Prioritet se pridružuje svakom poslu, a CPU se dodeljuje poslu sa najvišim prioritetom.
- Poslovi sa istim prioritetom se opslužuju po FCFS algoritmu.
- SJF je u stvari algoritam sa prioritetima gde je prioritet p jednak recipročnoj vrednosti predviđenog trajanja τ CPU aktivosti, tj. $p = 1/\tau$.
- Prioriteti se mogu definisati interno ili eksterno.
- Interni prioriteti koriste neku merljivu veličinu za izračunavanje prioriteta procesa (vremenske granice, memorijski zahtevi, broj otvorenih fajlova, odnos srednjeg trajanja U/I aktivnosti i srednjeg trajanja CPU aktivnosti).

Raspoređivanje na osnovu prioriteta

- Eksterni prioriteti postavljaju se na osnovu kriterijuma koji su spoljašnji u odnosu na operativni sistem, kao što su vrsta i iznos plaćene naknade za korišćenje računara, važnost naručioca posla.
- Glavni problem sa ovim algoritmom je pojava umiranja od gladi.
- 1973 godine na MIT-u, kada je računar IBM 7094 trebao da bude zamenjen, pronađen posao niskog prioriteta koji je prijavljen na sistem 1967 godine i još uvek se nije izvršio.
- Rešenje ovog problema je starenje.
- Tehnika starenja predviđa da se postepeno povećava prioritet onih poslova koji dugo čekaju u sistemu.

3.2 Round Robin algoritam

- Kružni algoritam (round-robin) je algoritam projektovan specijalno za sisteme sa deljenjem procesorskog vremena.
- Definiše se mala jednica vremena (kvantum vremena) koja se kreće između 10 i 100 milisekundi.
- Red spremnih procesa realizovan je kao kružni red.
- Planer procesa svakom procesu iz reda dodeljuje CPU na korišćenje tokom intervala koji je manji ili jednak kvantumu vremena. Kada se ovaj algoritma implementira, red spremnih procesa se realizuje kao FIFO red procesa.
- Novi proces se dodaje na kraj reda.
- Planer procesa bira prvi proces iz reda, postavlja tajmer na vreme od jednog kvantuma i dodeljuje CPU procesu.
- Ako proces ima CPU ciklus kraći od vremenskog kvantuma, u tom slučaju on sam oslobađa CPU izdajući U/I zahtev ili terminiranjem.
- Ako proces ima CPU ciklus veći od vremenskog kvantuma, tajmer će, po isteku kvantuma vremena, da izazove prekid.

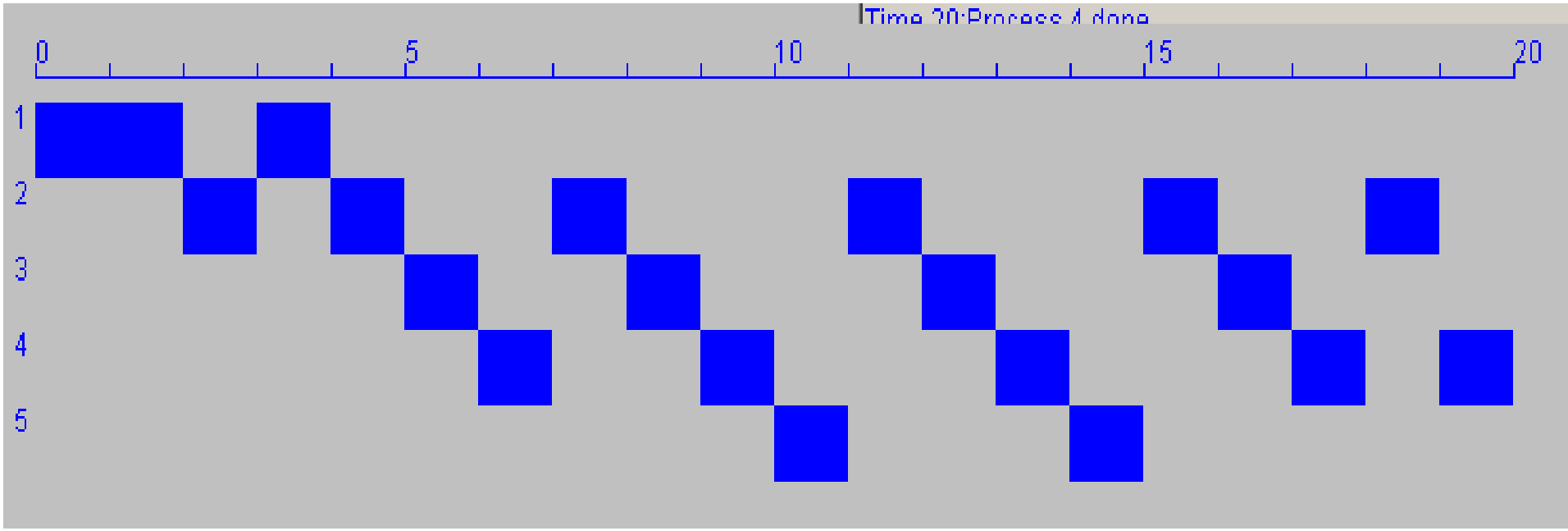
3.2 Round Robin algoritam

- Performanse ovog algoritma veoma zavise od kvantuma vremena.
- Ako kvantum vremena teži beskonačnosti, ponašanje ovog algoritma teži ponašanju FCFS algoritma.
- Ako je vremenski kvantum veoma mali (1 μs), Round-Robin se naziva „deljenje procesora" i teoretski izgleda da svaki od ***n*** procesa u redu ima sopstveni procesor koji je ***n*** puta sporiji od stvarnog procesora.
- Prebacivanje konteksta je sa stanovišta izvršavanja procesa potpuno nekoristan posao tj. nepotrebno gubljenje vremena.
- Trajanje prebacivanja konteksta zavisi od brzine memorije, broja registara i postojanja specijalnih instrukcija.
- Kvantum treba da je znatno duži od vremena za promenu konteksta.
- Vreme obilaska takođe zavisi od kvantuma vremena.
- Vreme obilaska je kraće ako većina poslova svoju CPU aktivnost završava za jedan kvantum vremena.
- Zato je bolje imati duži kvantum vremena, ali to onda konvergira ka slabom FCFS algoritmu. U praksi se najčešće teži da kvantum vremena u 80% slučajeva bude duži od trajanja CPU aktivnosti.

3.2 Round Robin algoritam

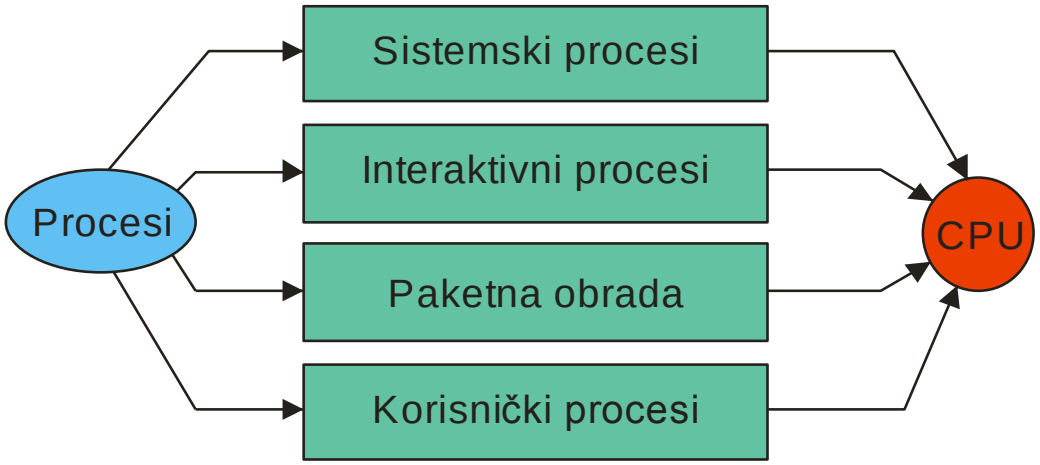
Primer: RR sa kvantumom = 1

Proces	Vreme dolaska	CPU vreme
P1	0	3
P2	2	6
P3	4	4
P4	6	5
P5	8	2



Raspoređivanje u više redova čekanja

- Procesi se mogu klasifikovati u više grupa koji imaju različite prioritete izvršavanja.



- Algoritmi ovog tipa (*Multylevel queue scheduling*) dele procese u više razdvojenih redova koji imaju različit prioritet izvršavanja
- Svaki red čekanja može da ima svoj potpuno nezavistan algoritam za raspoređivanje procesa procesoru koji odgovara njegovim potrebama
- Za izbor procesa koji se stvarno dodeljuje procesoru može se koristiti *Round Robin* ili prema prioritetu.

3.3 Višenivoski redovi

- Ova klasa algoritama planiranja kreirana je za situacije kada se poslovi mogu lako klasifikovati u različite grupe.
- Uobičajena je podela na poslove u prvom planu (interaktivni) i poslove u pozadini (paketni).
- Ova dva tipa poslova imaju sasvim različite zahteve u pogledu vremena odziva i mogu da imaju različite algoritme planiranja, a uz to interaktivni poslovi mogu imati eksterno dodeljen viši prioritet od poslova u pozadini.
- Red spremnih procesa podeljen u više redova kod ovog algoritma
- Poslovi koji pristižu svrstavaju se u odgovarajući red.
- Svaki red ima sopstveni algoritam planiranja, a između samih redova postoji fiksiran prioritet.
- Na taj način posao koji je prvi u svom redu može dobiti procesor na korišćenje isključivo ako je red višeg prioriteta prazan

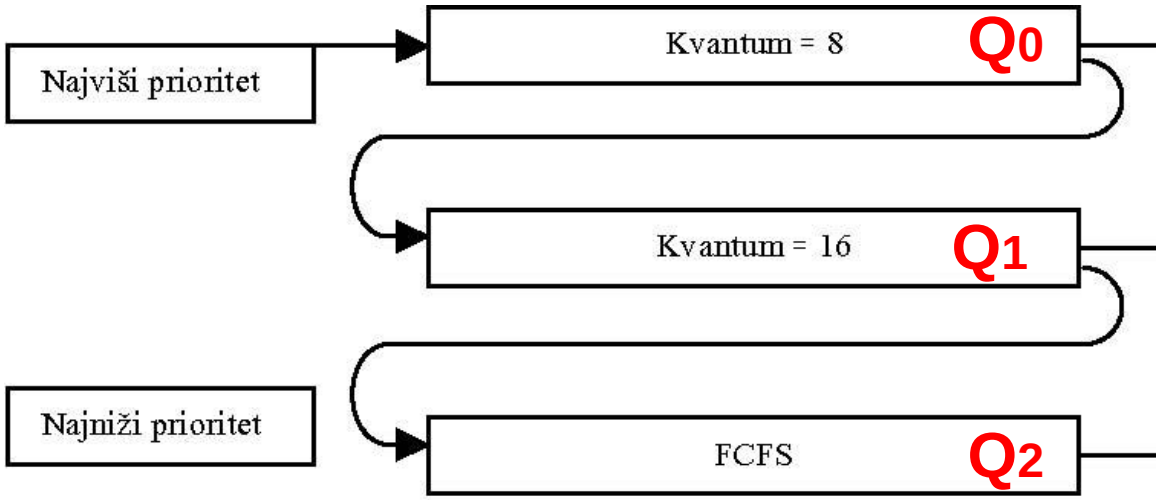
3.3 Višenivoski redovi sa povratnom spregom (Multilevel Feedback Queue)

- Kod prethodnog algoritma poslovi se dodeljuju redovima po njihovom ulasku u sistem i oni ne mogu prelaziti iz reda u red.
- Ponekad se zahteva dodatna fleksibilnost koja omogućava prelazak procesa iz jednog reda u drugi.
- Izdvajaju se poslovi sa različitim karakteristikama CPU, I/O aktivnosti
- Ako posao koristi previše CPU vremena premešta se u red sa nižim prioritetom dok se I/O procesi prebacuju u red sa visokim prioritetom.
- Uvodi se vremenska komponeneta koja sprečava gladovanje – ako proces čeka suviše dugo premešta se u red višeg prioriteta.
- ***Multilevel-feedback-queue*** raspoređivač definišu sledeći parameteri:
  - ✓ Broj redova
  - ✓ Algoritmi raspoređivanja za svaki red
  - ✓ Metod koji definiše kada se proces premešta u prioritetniji red
  - ✓ Metod koji definiše kada se proces spušta u red sa nižim prioritetom
  - ✓ Metod koji definiše u koji red se proces upisuje kada ulazi u sistem

3.3 Višenivoski redovi sa povratnom spregom - primer

Tri reda:

- Q_0 – time quantum 8 ms
- Q_1 – time quantum 16 ms
- Q_2 – FCFS



Sl. 3.8. Višenivovski red sa povratnom spregom.

Raspoređivanje

- Novi posao upisuje se u red Q_0 koji se opslužuje po FCFS.
- Kada dobije CPU, posao radi 8 ms.
- Ako za to vreme ne završi sa radom, premešta se u red Q_1.
- U redu Q_1 posao se još jednom opslužuje po FCFS algoritmu, ali sada dobija 16 ms.
- Ako ni tada ne završi sa radom, istiskuje se i upisuje u red Q_2.

3.4 Real Time scheduling

- *Scheduling* algoritmi za rad u realnom vremenu u principu dosta se razlikuju u odnosu na *scheduling* politike kod OS opšte namene.
- Oni ne dozvoljavaju da neki proces ne bude opslužen za mnogo dugo vreme, ali dozvoljavaju da se neki procesi izvršavaju češće od drugih.
- Kod RT *scheduling*-a izrazit je problem **krajnjih-rokova** (*deadlines*).
- Cena koja treba da se plati ako se propuste krajnji rokovi je različita od sistema do sistema, ali RT *scheduling* algoritmi su tako projektovani da zadovolje zahteve u pogledu krajnjih rokova, a time i ispune zahteve u pogledu propusnosti (*throughput*) u radu sistema.
- Globalno posmatrano RT *scheduling* algoritme delimo na ***soft*** i ***hard***.
- *Hard*-RT - moraju da izvrše neki proces u propisanom terminu da bi izbegli grešku. Raketni sistemi su tipični reprezenti hard-RT sistema.
- *Soft*-RT sistemi se koristi da ukaže na bilo koji sistem koji preferira da ispuni krajnje rokove ali ako to ne ostvari ne dolazi do pogrešnog rada sistema. Primer, korisnički interfejs.

3.5 Raspoređivanje u višeprocesorskim sistemima

- Raspoređivanje procesa kod višeprocesorskih sistema je komplikovanije jer se zahteva striktna sinhronizacija procesa
- Formira se zajednički red za čekanje
- Tehnike koje se tada koriste odnose se na:
  1. Simetrično multiprocesiranje - svi procesori su ravnopravni i pretražuju red i uzimaju procese koje će izvršavati
  2. Asimetrično procesiranje – postoji master CPU koji određuje koji će proces biti dodeljen određenim CPU (*master/slave*). Ovde master CPU izvršava sistemske procese a ostali CPU korisničke

Hvala na pažnji !!!

Pitanja

? ? ?