# *Raspoređivanje procesa*

Multiprogramiranjem se memorija deli na praticije u koje se smeštaju različiti procesi. Multiprogramiranjem se postiže bolje iskorišćenje procesora – ukoliko tekući proces čeka na resurs, procesor može izvršavati neki drugi proces koji se nalazi u stanju čekanja na procesor. Podela vremena (*time-sharing)* poseban je oblik multiprogramiranja koji omogućava da svaki korisnik radi interaktivno na računaru preko posebnog terminala kome pripada dodeljeno procesorsko vreme. Posle isteka vremenskog kvantuma, tj dodeljene količine procesorskog vremena, procesor se dodeljuje drugom terminalu.

Na jednoprocesorskim sistemima, u jednom trenutku može se izvršavati samo jedan proces. Na višeprocesorskim sistemima u jednom trenutku može se izvršavati onoliko procesa koliko u sistemu ima procesora. Ukoliko se na sistemu (bio jednoprocesorski ili višeprocesorski) izvršava višeprocesni operativni sistem (Windows 2003 Server, ili UNIX), veći broj procesa istovremeno čeka na procesor.

Operativni sistem treba da obezbedi odgovarajući mehanizam za dodelu procesora različitim procesima. Delovi tog mehanizma su redovi čekanja na procesor i planeri poslova.

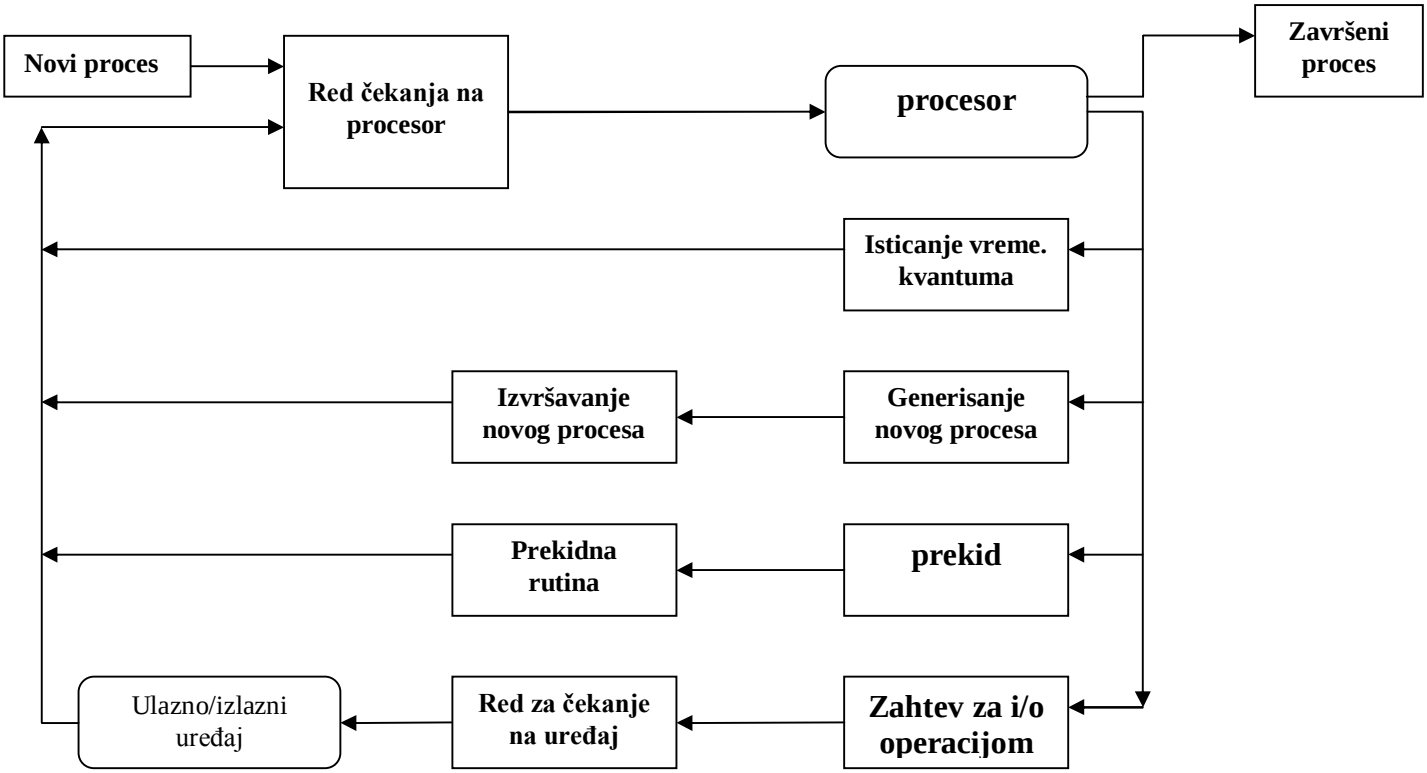
## *Redovi čekanja na procesor*

Proces u toku svog života prolazi kroz nekoliko redova čekanja. Nakon nastanka, proces se ubacuje u red čekanja za poslove (*job queue),* koji obuhvata sve postojeće procese na sistemu. Procesi se mogu nalaziti u raznim stanjima i u raznim lokacijama (u memoriji, disku, delimično u memoriji, delimično na disku). Svi procesi koju su spremni za rad i nalaze se u radnoj memoriji, čuvaju se u redu čekanja na procesor, tj u redu čekanja spremnih procesa. Redovi čekanja na procesor po pravilu se realizuju kao povezane liste, formirane od kontrolnih blokova procesa sa definisanim redosledom izvršavanja procesa. Redosled se zadaje preko zaglavlja liste (*queue header)*, koje sadrži informacije o početnom i poslednjem kontrolnom bloku u listi, i pokazivača na sledeći kontrolni blok.

Operativni sistem uvodi i poseban red čekanja za svaki ulazno-izlazni uređaj (*I/O queue)*. Svaki red čekanja na uređaj sadrži povezanu listu kontrolnih blokova procesa koji uređaj zahtevaju.

Postoje 2 vrste redova čekanja:

- Red čekanja na procesor
- Redovi čekanja na ulazno/izlazne uređaje




*Tehnike raspoređivanja poslova*

Novi proces se inicijalno postavlja u red čekanja za spremne procese u kome čeka dodelu procesora, nakon čega napušta red i počinje da se izvršava. Proces koji se nalazi u stanju izvršavanja može:

- Ostati bez procesora kada mu istekne vremenski kvantum
- Napraviti novi proces i čekati u blokiranom stanju da se novi proces završi
- Ostati bez procesora kada se desi prekid
- Postavi U/I zahtev, nakon čega se prebacuje u red čekanja na ulazno/izlazni uređaj, tj postaje blokiran

Proces se vraća u red čekanja na procesor sve dok se ne završi, posle čega oslobađa sve zauzete resurse.

***Planer poslova i dispečer sistema***

Proces u svom trajanju prolazi kroz razna stanja i redove čekanja. Programi za raspoređivanje odlučuju o tome kada će proces ući u neki red čekanja ili napustiti taj red. U literaturi se ove komponente operativnog sistema pominju pod imenima:

- **Planer poslova** (*job scheduler, long-term scheduler, high-level scheduler)*
- **Dispečer** (*dispatcher, short-term scheduler, low-level scheduler)*

Planer poslova, koji se u hijerarhijskom smislu nalazi iznad jezgra, obavlja sledeće funkcije:

- **Deli poslove u procese**
- **Na osnovu određenih algoritama dodeljuje prioritete procesima**
- **Dovodi procese u red čekanja na procesor**

Planer poslova se, praktično, poziva samo kada se pojave novi procesi ili kada jedan ili više procesa završe aktivnost. Planer poslova reguliše stepen multiprogramiranja tj broj simultanih procesa u memoriji. Od njega se očekuje da pravi dobru selekciju procesa koji će dobiti memoriju kako bi sistem što efikasnije funkcionisao. Planer poslova potiče sa velikih računarskih sistema (*mainframes)*, koje karakteriše grupna obrada poslova.

Na osnovu resursa koje dominantno koriste, tj broja i trajanja ciklusa, procesi se dele na:

- **Procese koji dominantno koriste procesor** (*CPU bound)*
- **Procese koji dominantno koriste ulazno-izlazne operacije** (*I/O bound*)

Zadatak dispečera sistema je da dodeljuje procesor procesima koji se nalaze u procesorskom redu. Dispečer dodeljuje procesor kad god tekući proces pređe iz stanja RUN u stanje WAIT ili READY. Dispečer utvrđuje kom je procesu najpovoljnije dodeliti procesor , tj koji je proces u stanju READY najvišeg prioriteta. Ukoliko se procesorski red formira sa inkrementalnim prioritetima (*dinamički izmenljiva lista)* najpovoljniji je prvi proces u redu. Najjednostavnije rečeno, **dispečer odlučuje koji će proces dobiti procesor**, **kada će ga dobiti i na koliko dugo**. Dispečer najpre utvrđuje da li procesor treba dodeliti tekućem procesu (čime bi se nastavilo njegovo izvršavanje) ili nekom drugom procesu iz reda. Ako se procesor dodeljuje tekućem procesu, kontrola izvršavanja procesa vraća se na adresu koju čuva prekidni mehanizam jezgra. Ukoliko se procesor dodeljuje nekom drugom procesu iz reda, dispečer najpre ažurira kontekst, tj okruženje tekućeg procesa u kontrolnom bloku, a zatim na osnovu konteksta novog procesa postavlja okruženje u kome se taj proces može izvršavati. Posle toga se kontrola izvršavanja prebacuje na ono mesto u novom procesu gde je izvršavanje bilo prekinuto.

U literaturi, planer poslova i dispečer navode se kao posebni programski moduli (*Silbershatz)* ili kao jedan modul (*Lister, Kvaternik*). U slučaju odvojenih modula, planer poslova niskog nivoa (*low-level scheduler, short-term scheduler)* bira proces iz reda čekanja na procesor po nekom kriterijumu. Nakon toga dispečer ažurira kontekst procesa, postavlja neophodno okruženje i predaje kontrolu nad procesorom izabranom procesu.

Na većini operativnih sistema, planer poslova je minimalan. Na interaktivnim sistemima (*time-sharing)* kao što je UNIX, na kojima planer poslova ne postoji, svaki proces se direktno ubacuje u red čekanja na procesor. Stabilnost ovakvih sistema zavisi od toga da li su procesorska snaga i količina sistemske memorije dovoljni da zadovolje tekući broj interaktivnih korisnika.

***Zamena konteksta procesa***

Prilikom dodele procesora drugom procesu, tj zamene procesa koji se trenutno izvršava, dispečer radi sledeće:

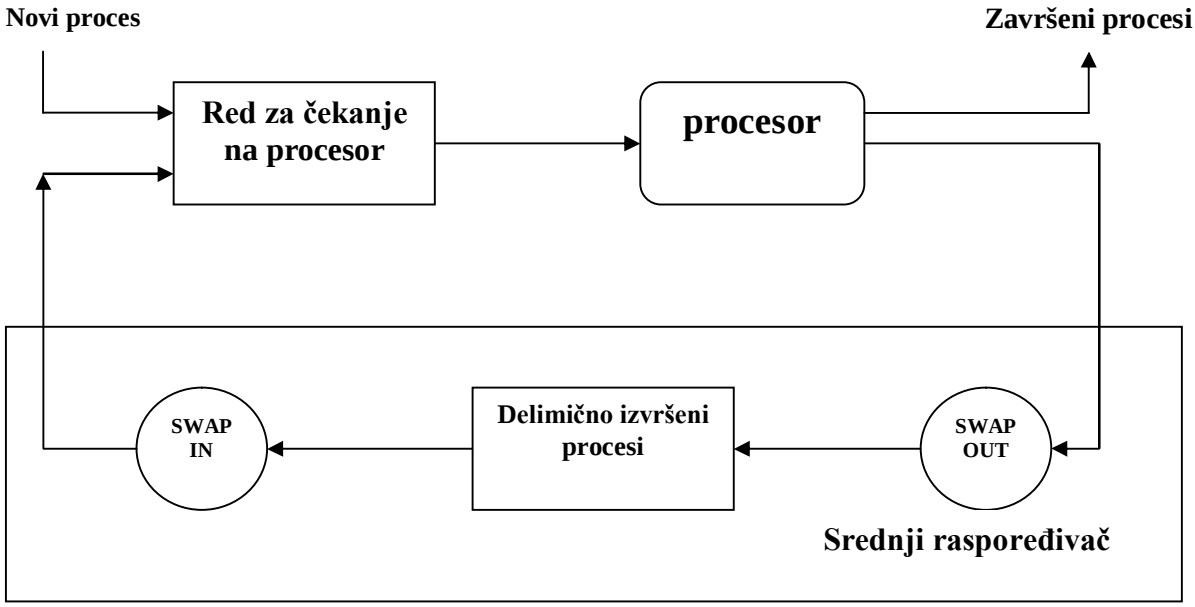
- Pamti stanje procesa koji se prekida, kako bi se proces mogao nastaviti kasnije
- Puni memoriju stanjem novog procesa kome se dodeljuje procesor

Navedene operacije su poznatije kao zamena konteksta procesa (*context switch*). ***Kontekst procesa čine podaci koji se čuvaju prilikom oduzimanja procesora, a koji omogućavaju nastavak izvršenja procesa, kao što su registri procesora, memorijske sekcije, lista otvorenih datoteka.***

Prilikom zamene konteksta, poslednje stanje tekućeg procesa se čuva u njegovom kontrolnom bloku. Poslednje stanje procesa koji će se dalje izvršavati rekonstruiše se iz odgovarajućeg kontrolnog bloka, nakon čega se procesu predaje kontrola za nastavak izvršenja. Prebacivanje konteksta je čist gubitak vremena i predstavlja premašenje sistema (*overhead)*, ali je neophodno radi omogućavanja multiprogramiranja. Premašenje zavisi od hardverskih performansi procesora i memorije, broja registara procesora koji se moraju sačuvati, optimalnih asemblerskih instrukcija i načina korišćenja raznih memorijskih tehnika.

***Srednji nivo raspoređivanja procesa***

Upotrebom *swap* tehnike, u interaktivne sisteme se uvodi nov nivo raspoređivanja procesa – srednji raspoređivač (*intermediate scheduler*).




*Srednji nivo raspoređivanja poslova*

Suština je u sledećem: svaki novostvoreni proces odlazi u red čekanja, nakon čega dispečer odlučuje kome će dodeliti procesor. Neki procesi mogu biti suspendovani, tj privremeno prekinuti i upisani na disk (*swap space*), čime se oslobađa memorija za druge procese. Kada se sa diska premeste u memoriju, suspendovani procesi se vraćaju u red čekanja na procesor. Funkcija suspenzije procesa (*swap-out),* funkciju povratka procesa u stanje spremnosti (*swap-in)* i izbor procesa za obe funkcije obavlja srednji raspoređivač.

## *Operacije nad procesima*

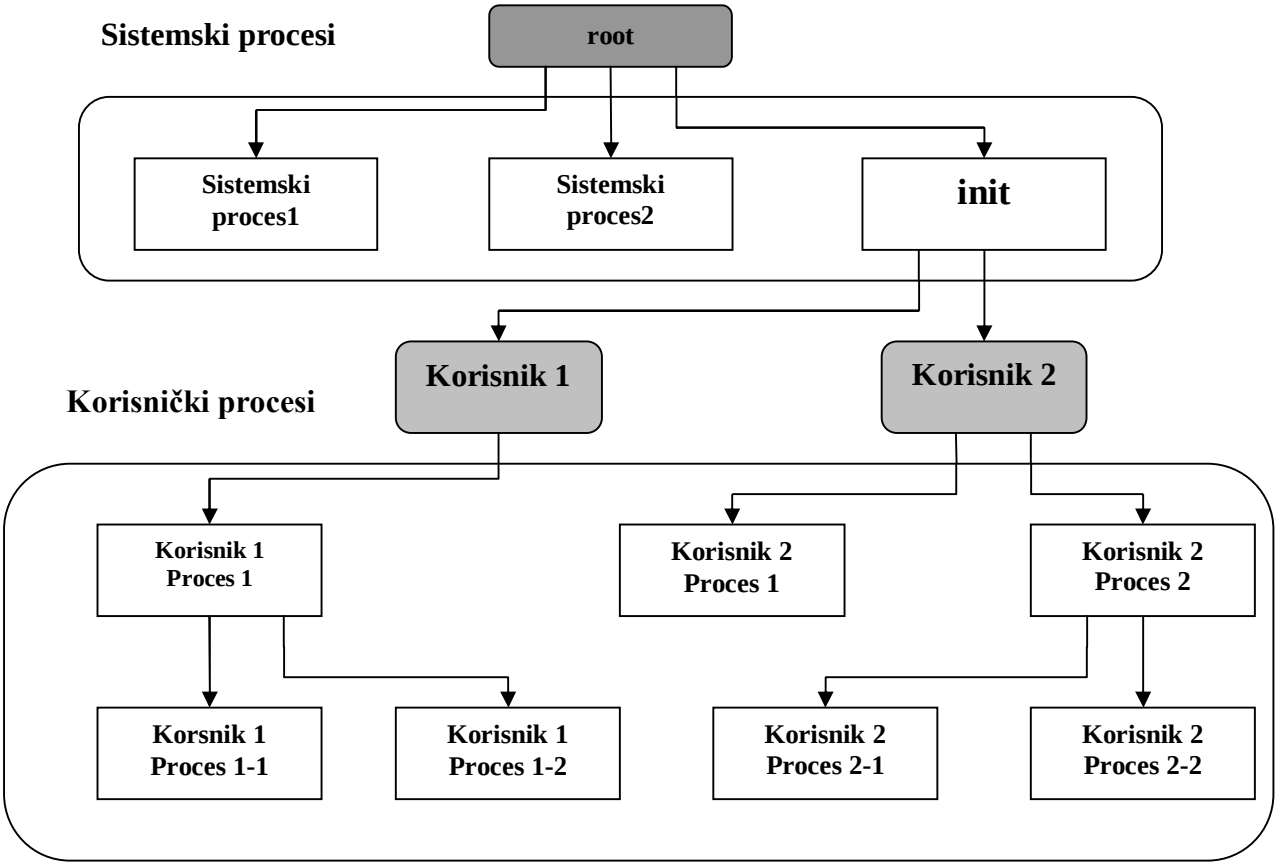
Ranije opisani koncept jezgra, uz korišćenje kontrolnih blokova procesa, dozvoljava sistemu da nad procesima izvrši sledeće opracije:

- Izrada novog procesa (pravljenjem kontrolnog bloka i ažuriranjem procesne liste)
- Izrada veza proces – proces roditelj

- Uništenje procesa (brisanjem kontrolnog bloka iz procesne liste)
- Promena stanja procesa (obavljanje tranzicija u dijagramu stanja, kao što je dodela procesora)
- Promena prioriteta procesa

***Izrada procesa***

Proces u toku svog izvršenja može, odgovarajućim sistemskim pozivom, praviti nove procese. Proces koji pravi nove procese jeste roditeljski proces (*parent process)*, dok se proces koji je roditelj napravio naziva dete proces (*child process)*. Svaki proces koji je dete drugog procesa može dalje stvarati nove procese – na taj način se formira hijerarhijska struktura procesa, tj stablo aktivnih procesa.




*Stablo aktivnih procesa*

Odnos proces – proces roditelj može se opisati na osnovu načina deljenja resursa i načina izvršavanja. Prema deljenju resursa , procesi roditelj i dete mogu se naći u sledećim relacijama:

- Procesi roditelj i dete dele sve resurse
- Procesi roditelj i dete dele podskup resursa roditeljskog procesa
- Procesi i roditelj ne dele resurse

To znači da, nakon formiranja, dete procesor može tražiti od operativnog sistema nove resurse, deo resursa roditeljskog procesa ili sve resurse koji pripadaju roditelju. Od resursa, svaki proces za svoj rad najčešće koristi registre procesora, memorijske sekcije, datoteke i ulazno-izlazne uređaje. Posebno je osetljiv memorijski adresni prostor, za čije se korišćenje primenjuju sledeće tehnike:

- Proces dete duplicira adresni prostor roditelja
- Adresni prostor deteta generiše se prema programu kojim se taj adresni prostor puni

Prema načinu izvršavanja, proces se sa svojim roditeljem može naći u sledećim relacijama:

- Proces roditelj nastavlja da se izvršava nezavisno i konkurentno s detetom
- Proces roditelj se blokira i čeka dok proces dete ne završi svoje aktivnosti

***Završetak procesa***

Proces završava svoje aktivnosti onda kada završi poslednju naredbu programa koji je učitan u adresni prostor tog procesa. Posle toga, koristeći sistemski poziv *exit,* proces traži od operativnog sistema da ga uništi, tj obriše. Pri tome, proces dete može vratiti, izlazne podatke roditeljskom procesu pomoću sistemskog poziva *wait*. Potom se oslobađaju svi resursi koji su pripadali procesu, kao što su fizička i virtuelna memorija, datoteke i ulazno-izlazni baferi. Ovo predstavlja tzv normalan završetak rada po obavljanju svih predviđenih instrukcija.

Proces može izazvati i nasilan završetak rada drugog procesa pomoću specijalnog sistemskog poziva *abort*. Po pravilu, proces roditelj može prekinuti izvršenje procesa koje on napravio, a to čini iz sledećih razloga:

- Proces dete je prilikom korišćenja nekog resursa (memorije, sistemskog-vremena) premašio kvotu
- Aktivnost koju dete proces obavlja više nije potrebna
- Proces roditelj je završio aktivnosti pre dece, što se smatra nenormalnom situacijom koju operativni sistemi ne dozvoljavaju – u tom se slučaju svi procesi deca moraju nasilno uništiti