Upravljanje procesima

Vladimir Filipović

Procesi

- Jedan od ključnih koncepata, kada su operativni sistemi, ali i računarski sistemi, u pitanju, predstavljaju procesi.
- Proces se definiše kao program u izvršavanju.
  - Kada programer napiše program (izvorni kôd) na nekom programskom jeziku, on je u stvari napisao tekst kojim se neki algoritam implementira na tom jeziku.
  - Kompilacijom, odnosno prevođenjem napisanog izvornog programa na mašinski jezik, nastaje izvršni program, fajl na nekoj od sekundarnih memorija.
  - Pokretanje izvršnog programa podrazumeva njegovo učitavanje u primarnu (radnu) memoriju računara i izvršavanje na procesoru – program postaje proces.

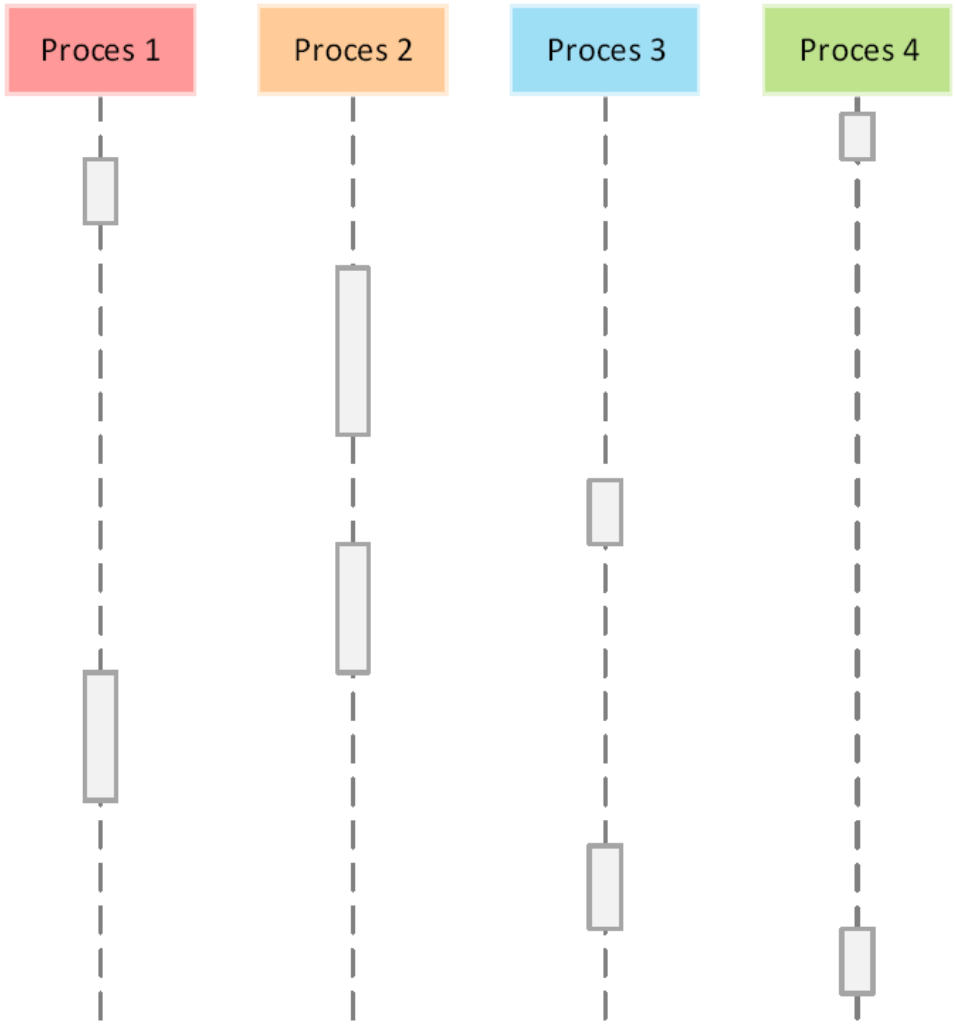
Procesi (2)

- Operativni sistem je dužan da vodi računa o procesu, tj. da mu obezbedi resurse koji su potrebni za izvršavanje.
- Procesu mogu biti potrebni resursi poput procesora, memorije, ulazno-izlaznih uređaja, fajlova, itd. da bi obavio zadatak zbog kojeg je pokrenut.
- Dakle, zadatak operativnih sistema je da obezbede uslove da se procesi mogu efikasno izvršavati.

Procesi (3)

- Procesi mogu biti korisnički ili sistemski.
- Efikasno izvršavanje procesa najčešće podrazumeva da se izvršavaju konkurentno i/ili paralelno.
- Na sistemima čiji procesori imaju samo jedno jezgro, dodeljivanjem procesorskog resursa procesima koji se izvršavaju konkurentno operativni sistem čini računarski sistem produktivnijim. Stvara se privid da se procesi izvršavaju u isto vreme - pseduoparalelno. Upravljanje procesima na ovakav način predstavlja osnovni zadatak za operativne sisteme koji podržavaju multiprogramiranje.

Procesi (4)



Pseudoparalelno izvršavanje procesa

Procesi (5)

- Razvojem procesora i povećavanjem broja jezgara omogućeno je da se i više procesa na različitim jezgrima izvršava paralelno. Principi funkcionisanja su slični kao i u slučaju kada postoji samo jedno jezgro ali su mogućnosti veće.
- Osim izvršnog koda i podataka, bitne informacije o toku izvršavanja procesa sadrže i registri, a posebno programski brojač koji vodi računa o tome dokle se stiglo sa izvršavanjem procesa. Takođe, bitne informacije za svaki proces su i podaci o otvorenim fajlovima, informacije o dozvolama i vlasniku procesa itd.

Procesi (6)

- Zadatak operativnih sistema je da obezbede efikasne mehanizme za:
  - Kreiranje i brisanje procesa;
  - Upravljanje procesima;
  - Komunikaciju između procesa;
  - Sinhronizaciju procesa;

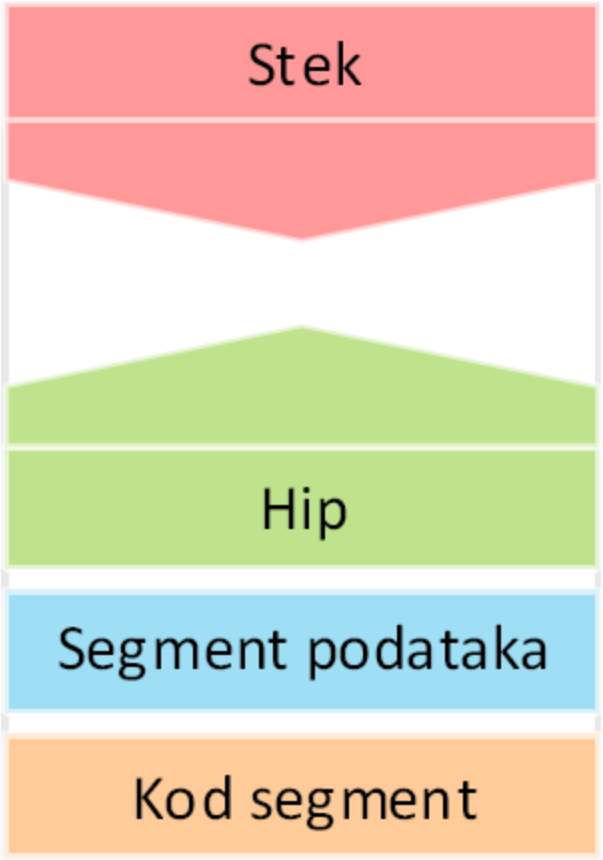
Procesi u memoriji

- Za razliku od programa koji je pasivan i opisuje skup instrukcija koje treba izvršiti, proces je aktivan i izvršava se, korak po korak, po algoritmu koji je implementiran u programu.
- Programski brojač za svaki proces vodi računa o tome dokle se stiglo sa izvršavanjem, tj. koja je sledeća instrukcija.
- Instrukcije se nalaze u memoriji zajedno sa ostalim podacima koji su potrebni procesu za izvršavanje.

Procesi u memoriji (2)

- Po pokretanju i učitavanju u memoriju programi, odnosno tada već procesi, obično raspolažu sa četiri dela memorije koji se nazivaju segmenti:
  - Stek (Stack) - sadrži lokalne promenljive i parametre funkcija;
  - Hip (Heap) memorija u koju se smeštaju podaci koji se generišu u toku izvršavanja procesa;
  - Segment podataka (Data segment) - sadrži globalne promenljive;
  - Kôd segment (Text segment) - sadrži instrukcije koje proces treba da izvrši.

Procesi u memoriji (3)



Proces u radnoj memoriji

Stanja procesa

- Operativni sistemi definišu skup stanja u kojima se, u zavisnosti od trenutne aktivnosti i situacije u sistemu, procesi mogu naći.
- Većina sistema podržava sledeća osnovna stanja:

  Novi, Spreman, Izvršavanje, Čekanje, Završen

Stanja procesa (2)

- Novi – proces je upravo kreiran, operativi sistem je napravio dokumentaciju koja se odnosi na njega i on prelazi u spremno stanje.
- Spreman – procesor čeka da operativni sistem donese odluku da mu bude dodeljen procesor.
- Izvršavanje – proces se izvršava na procesoru.
- Čekanje – proces se izvršavao ali je za njegov dalji rad potreban neki resurs koji nije slobodan tako da on čeka da se stvore uslovi da bi mogao da nastavi sa radom.
- Završen – proces je završio sa izvršavanjem i trebalo bi ga izbaciti iz sistema.

Stanja procesa (3)

Iz trenutnog stanja proces prelazi u neko drugo u sledećim situacijama:

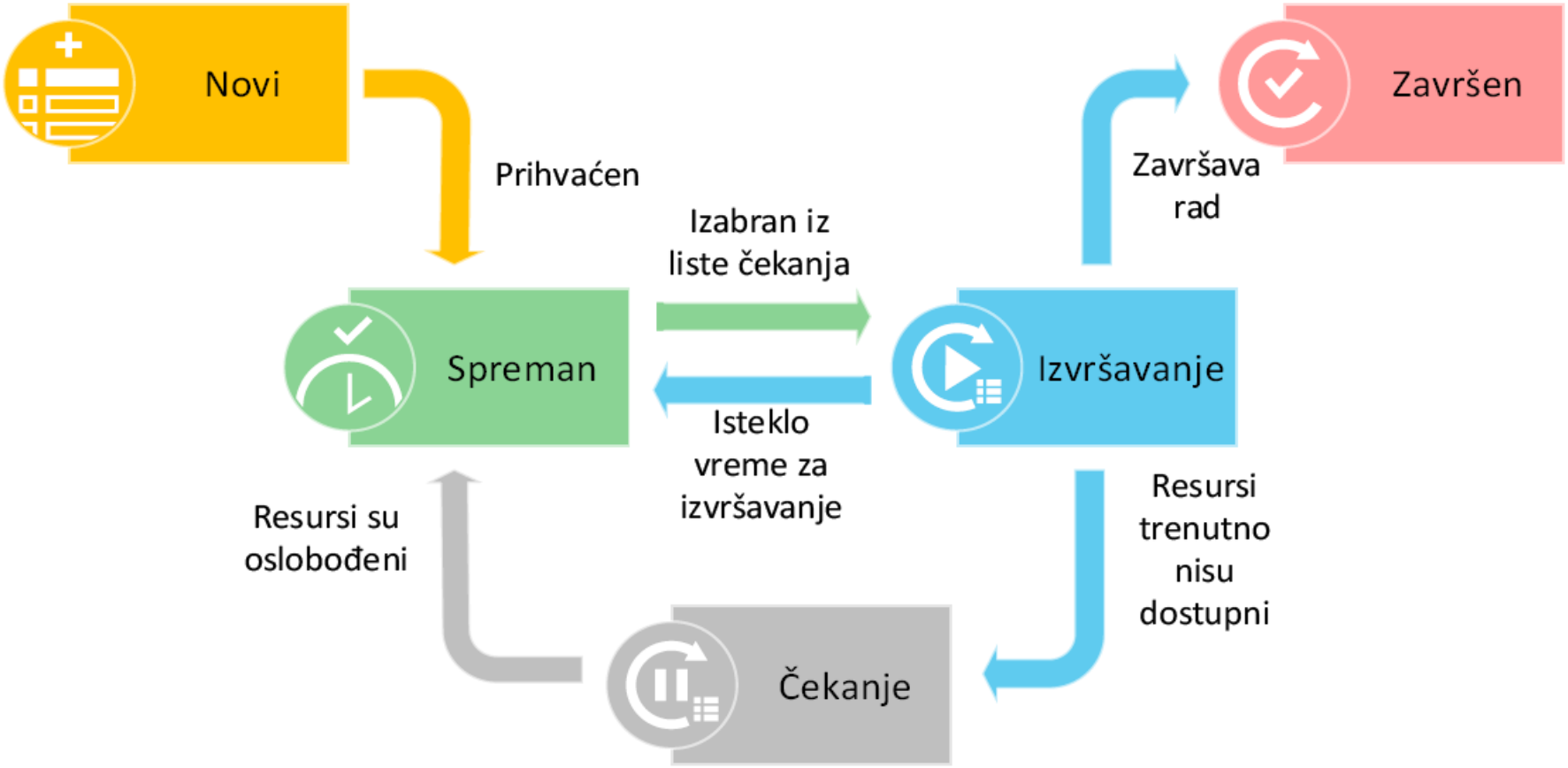
1. Proces prelazi iz stanja Spreman u stanje Izvršavanje kada se procesor oslobodi a operativni sistem, po nekom kriterijumu, izabere njega iz liste spremnih procesa i dodeli mu procesor.
2. Proces prelazi iz stanja Izvršavanje u stanje Čekanje u trenutku kada su mu za dalje izvršavanje potrebni resursi koji trenutno nisu dostupni.
   Na primer, ako treba da odštampa rezultate a štampač je u tom trenutku zauzet.

Stanja procesa (4)

Iz trenutnog stanja proces prelazi u neko drugo u sledećim situacijama:

3. Proces prelazi iz stanja Izvršavanje u stanje Spreman kada istekne vreme unapred određeno prilikom dodeljivanja procesora ili ako operativni sistem donese odluku da se proces prekine da bi neki drugi proces došao do procesora. Tada proces ponovo prelazi u listu procesa koji konkurišu da dobiju procesor.
4. Proces prelazi iz stanja Čekanje u stanje Spreman, kada dođe do potrebnih resursa i spreman je za dalji rad. Pošto se u tom trenutku ne zna da li je procesor slobodan, proces se stavlja u listu spremnih procesa a operativnom sistemu se prepušta odluka kada će mu dozvoliti da ponovo dobije procesor.

Stanja procesa (5)

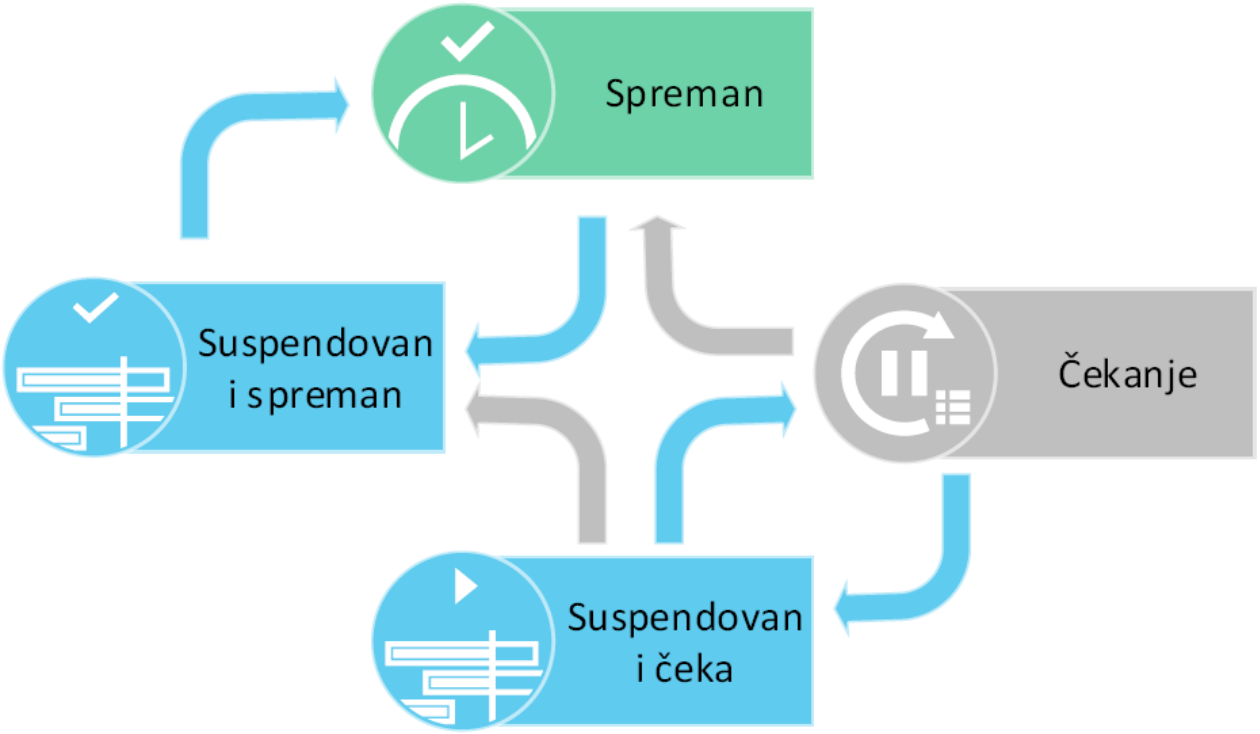


Stanja procesa

Stanja procesa (6)

- Osim osnovnih stanja, operativni sistemi često podržavaju i suspendovana stanja u koje procesi mogu da pređu na zahtev korisnika ili operativnog sistema.
- Suspendovati se mogu procesi koji su u stanju Spreman (prelazi u stanje Suspendovan i spreman) ili Čekanje i (prelazi u stanje Suspendovan i čeka).
- Suspendovani proces oslobađa resurse koje je zauzimao pre suspenzije i prestaje da konkuriše za druge resurse koji su mu potrebni za izvršavanje ali i dalje ostaje proces.

Stanja procesa (7)



Suspendovana stanja

Stanja procesa (8)

- Često se suspendovani procesi prebacuju na disk do prestanka suspenzije, čime se oslobađa i deo radne memorije koji su zauzimali i na taj način ostavlja više prostora za druge procese.
- Do suspendovanja spremnog procesa, odnosno do prelaska procesa iz stanja Spreman u stanje Suspendovan i spreman, najčešće dolazi zbog preopterećenosti sistema usled velikog broja spremnih procesa, pa je privremeno suspendovanje nekog od procesa poželjno kako bi se operativnom sistemu olakšao rad.

Stanja procesa (9)

- Suspendovanje procesa koji je u stanju Čekanje i njegovo prebacivanje u stanje Suspendovan i čeka se obično vrši da bi se oslobodili resursi kojima on raspolaže i time sprečilo zaglavljivanje ili ubrzao rad sistema.
- Kao što je već istaknuto, suspendovanjem procesa se oslobađaju i svi resursi koje je on posedovao čime se drugim procesima omogućava da dođu do njih.

Stanja procesa (10)

- Proces koji je u stanju Suspendovan i čeka prelazi u stanje Suspendovan i spreman, ako se stvore uslovi, tj. oslobode resursi na koje čeka (zbog kojih se našao u stanju Čekanje), tako da po povratku iz suspenzije može da bude u stanju Spreman.
- Prekidanje suspenzije procesa koje inicira korisnik ili operativni sistem realizuje se njegovim prelaskom iz stanja Suspendovan i čeka u stanje Čekanje i prelaskom iz stanja Suspendovan i spreman u stanje Spreman.

Kontrolni blok procesa

- Da bi operativni sistemi bili u mogućnosti da vode preciznu evidenciju o procesima kao i da bi se olakšala implementacija mulitiprogramiranja, obično se koriste strukture podataka koje čuvaju informacije o svakom pokrenutom procesu.
- Ova dinamička struktura se često naziva Kontrolni blok procesa i sadrži najznačajnije podatke koji omogućavaju identifikaciju i upravljanje procesima.

Kontrolni blok procesa (2)

U većini implementacija operativnih sistema kontrolini blok procesa sadrži sledeće informacije o procesima:

- Jedinstveni identifikator procesa (PID) – broj koji proces dobija u trenutku pokretanja, tj. svog nastanka. Svi procesi na jednom sistemu imaju različite PID-ove, čime je sistemu omogućeno da ih lako razlikuje.
- Stanje procesa – informacija o trenutnom stanju u kojem se proces nalazi. Na primer, proces se može naći u nekom od stanja koja su definisana u prethodnom odeljku: Spreman, Izvršavanje, Čekanje, Suspendovan i spreman, itd.
- Programski brojač – čuva informaciju o sledećoj instrukciji koju proces treba da izvrši.

Kontrolni blok procesa (3)

U većini implementacija operativnih sistema kontrolini blok procesa sadrži sledeće informacije o procesima:

- Sadržaj registara procesora – vrednosti koje se nalaze u registrima, kako bi proces posle gubitka prava da koristi procesor, mogao da nastavi sa radom tamo gde je zaustavljen.
- Prioritet procesa – informacija o važnosti procesa u odnosu na ostale procese u sistemu.
- Adresa memorije gde se nalazi proces – pokazivači na adrese u memoriji gde se nalaze segmenti procesa.
- Adrese zauzetih resursa – kako bi operativni sistem imao informaciju o tome na kojim lokacijama treba da traži potrebne podatke.

Kontrolni blok procesa (4)

Procesi	Memorija	Fajlovi
PID	Pokazivač na tekst segment	Koreni direktorijum
Programski brojač	Pokazivač na segment podataka	Radni direktorijum
Registri	Pokazivač na stek segment	Fajl deskriptori
Status procesa	Pokazivač na hip	ID korisnika
Pokazivač na stek		ID grupe
Prioritet procesa		
Grupa kojoj pripada proces		
Vreme pokretanja procesa		
Signali		

Kontrolni blok procesa (5)

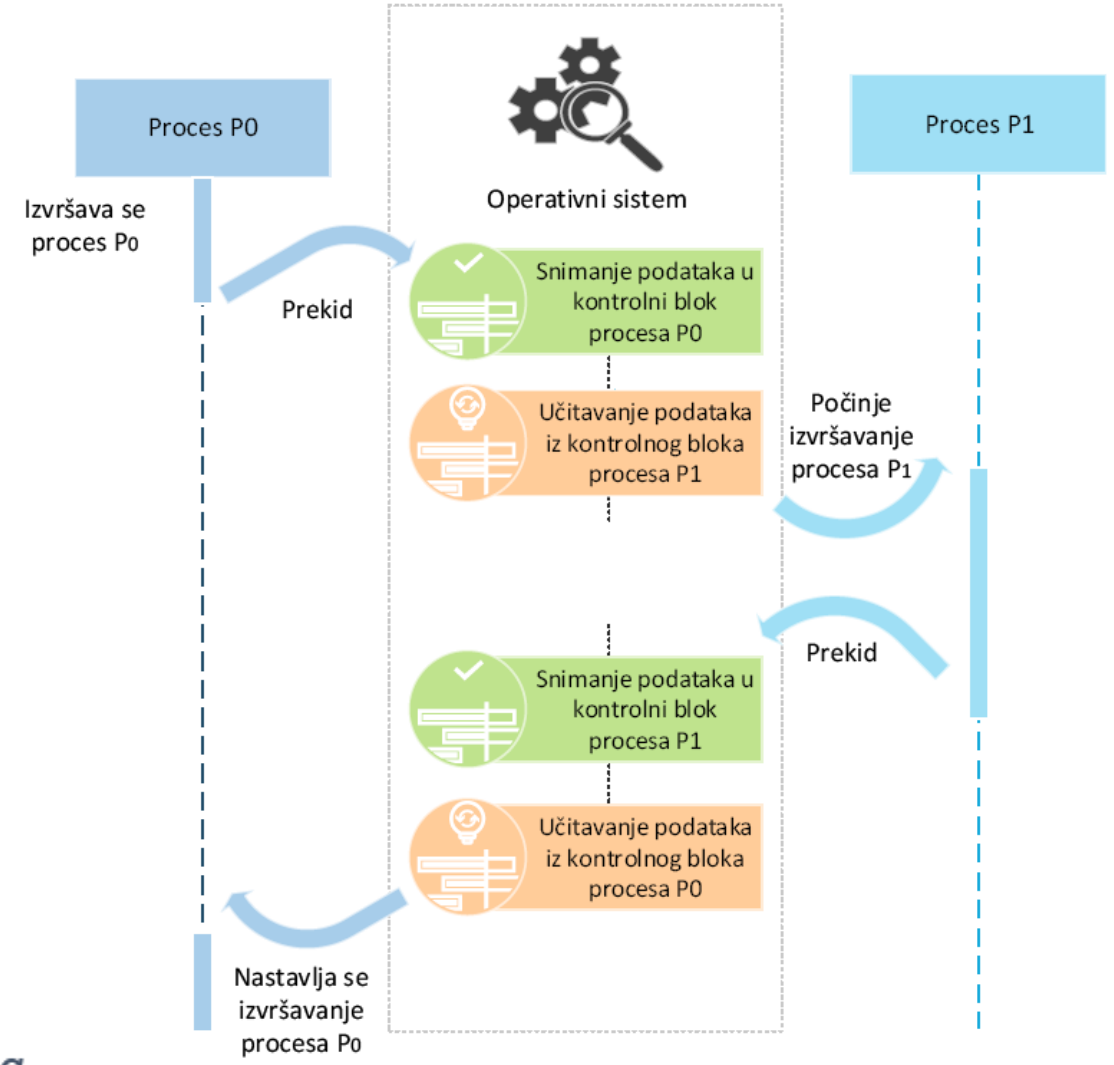
Operativni sistemi obezbeđuju efikasne mehanizme za manipulaciju kontrolnim blokovima procesa kako bi se realizovali sledeći zadaci:

- ✓ Kreiranje kontrolnog bloka za novi proces;
- ✓ Uništavanje kontrolnog bloka procesa koji se izvršio;
- ✓ Menjanje stanja procesa;
- ✓ Menjanje prioriteta procesa;
- ✓ Izbor procesa za izvršavanje.

Kontrolni blok procesa (6)

- Prebacivanje konteksta predstavlja postupak kojim se proces koji se trenutno izvršava na procesoru prekida, pamte njegovi parametri, a zatim se umesto njega pokreće neki drugi proces i pri tome se učitavaju njegovi parametri.
- Modul koji je odgovoran da izvršava prebacivanje konteksta često se naziva Dispečer (dispatcher). On je zadužen za punjenje registara procesa, prebacivanje u korisnički režim rada i skok na odgovarajuću lokaciju u korisničkom programu kako bi se on nastavio.

Kontrolni blok procesa (7)



Prebacivanje konteksta

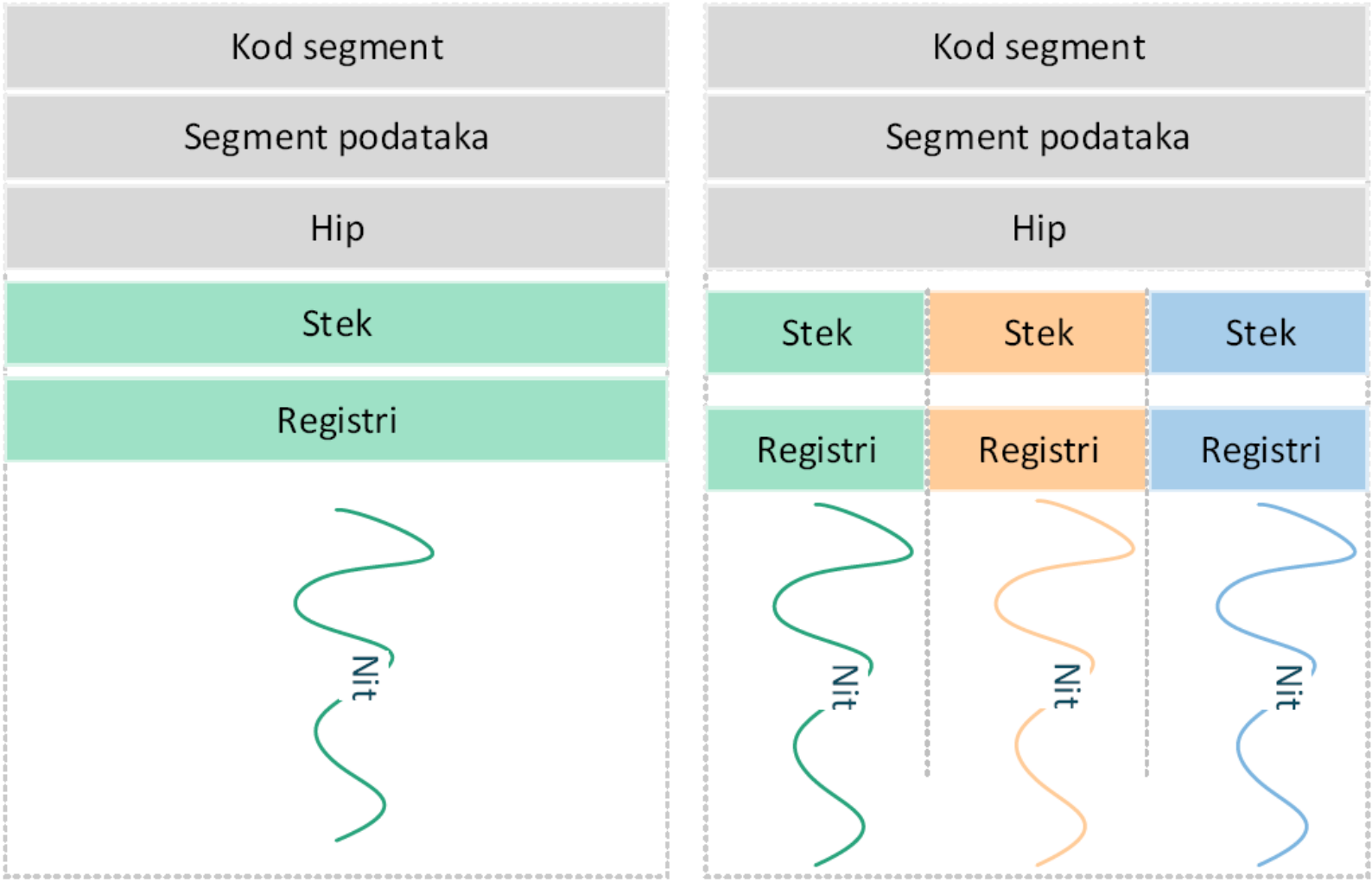
Niti

- Tradicionalni pristup podrazumeva da procesi imaju svoj memorijski prostor, resurse i da imaju samo jednu jedinicu izvršavanja, tj. da se svi poslovi procesa izvršavaju sekvencijalno.
- Savremeni operativni sistemi obično podržavaju koncept niti (threads), koje predstavljaju osnovne jedinice za izvršavanje u okviru procesa.
- Niti su delovi jednog procesa i izvršavaju se korišćenjem resursa koji su mu pridruženi. Osim resursa procesa kom pripadaju, niti imaju i sopstvene resurse.

Niti (2)

- Svaka nit poseduje svoje registre, programski brojač i stek, a razlikuje ih i jedinstveni identifikator (TID - thread ID).
- Kôd segment, segment podataka, podaci o otvorenim fajlovima, itd. zajednički su za sve niti jednog procesa.
- Ovakvim pristupom smanjuje se zauzeti prostor i omogućava se da više niti obavlja različite zadatke u okviru jednog procesa.
- Takođe, korišćenjem niti pruža se mogućnost da se bolje iskoriste prednosti koje donosi višeprocesorska arhitektura.

Niti (3)



Jedna nit procesa i više niti istog procesa

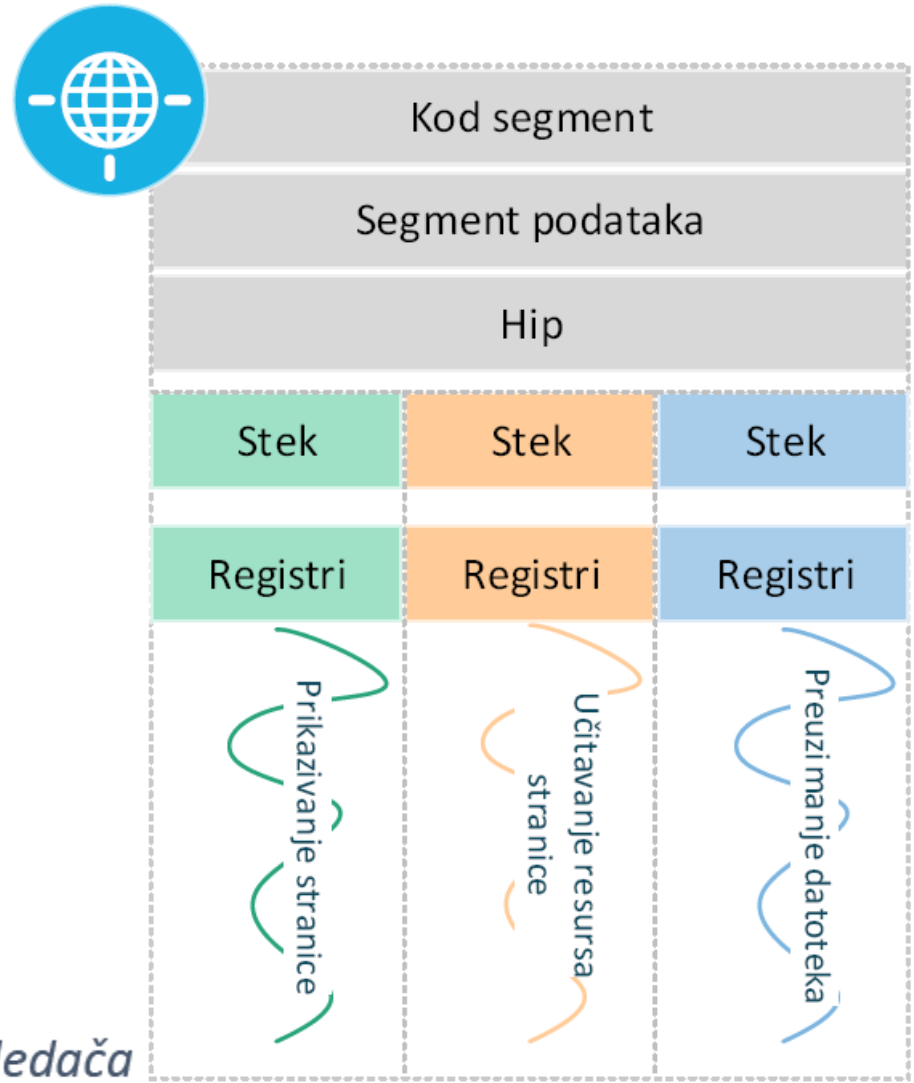
Niti (4)

- Rad sa više niti (multithreading) podrazumeva mogućnost operativnog sistema da podrži konkurentno ili paralelno izvršavanje više niti.
- Na početku izvršavanja svaki proces dobija svoj memorijski prostor i kontrolnu (inicijalnu) nit. Ova nit ima zadatak da obavi potrebne inicijalizacije i kreira ostale niti koje su potrebne za izvršavanje procesa.
- S obzirom da niti imaju sve karakteristike procesa, pri čemu neke resurse dele sa drugim nitima, često se nazivaju i lakim (lightweight) procesima.

Niti (5)

- Upotreba niti se može se ilustrovati na primeru veb pregledača. Naime, oni se mogu implementirani tako da koriste bar tri niti:
  - Jedna nit služi za prikazivanje hiperteksta u okviru prozora,
  - druga učitava podatke sa nekog veb servera i
  - treća služi za preuzimanje datoteke preko mreže.
- Ove niti dele podatke kroz zajedničke resurse i istovremeno se mogu izvršavati na različitim jezgrima istog procesora.

Niti (6)



Niti veb pregledača

Niti (7)

- Korišćenje niti donosi mnoge prednosti.
- Niti omogućavaju značajne uštede memorijskog prostora i vremena izvršavanja.
  - One dele memoriju i neke resurse koji pripadaju istom procesu tako da zauzimaju manje prostora nego kada su u pitanju nezavisni procesi.
  - Takođe, iz istog razloga, niti se kreiraju mnogo brže od procesa.
  - Prebacivanje konteksta između niti istog procesa je brže od prebacivanja konteksta između procesa jer se prebacuju samo resursi koji su jedinstveni za nit – stek, registri i programski brojač.

Niti (8)

- Niti pružaju mogućnost aplikacijama da nastave rad u situacijama kada se izvršavaju dugotrajne operacije koje bi, bez podele poslova procesa na niti, privremeno zaustavile izvršavanje ostalih delova procesa.
- Slično, korišćenjem niti omogućava se rad procesa čiji su delovi potpuno blokirani.
- Razvojem i ekspanzijom višeprocesorskih sistema do izražaja naročito dolaze prednosti koje donosi korišćenje ovog koncepta jer se više niti može izvršavati istovremeno.

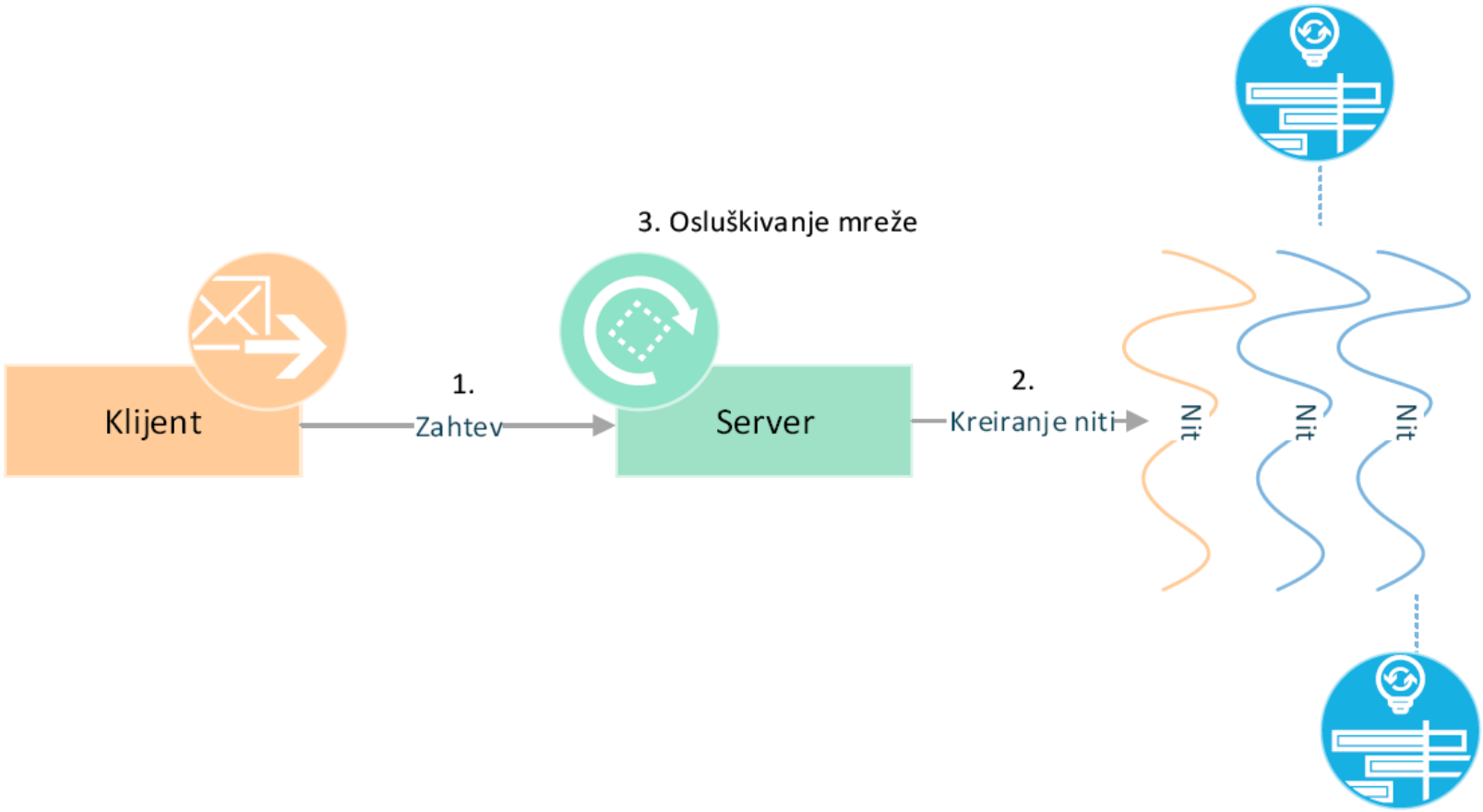
Niti (9)

- Na primeru serverske aplikacije koja ima zadatak da opsluži veliki broj klijenata (klijentskih aplikacija) u kratkom vremenskom intervalu može se dobro ilustrovati pogodnosti koje donosi korišćenje niti.
- Tradicionalni pristup, bez korišćenja niti, je podrazumevao da serverski proces u beskonačnoj petlji čeka da se pojavi klijent, a onda se za svakog novog klijenta kreira novi proces kojem će prepustiti opsluživanje klijenta.

Niti (10)

- Korišćenjem niti, problem se rešava na sličan način ali se umesto različitih procesa koristi samo jedan u okviru kojeg se može kreirati veliki broj niti.
- Jedna nit procesa je zadužena da „osluškuje“ mrežu, tj. da prihvata klijente pri čemu se za svakog novog klijenta kreira nova nit.
- Kreiranje niti je brže i zahteva manje prostora nego kada su u pitanju procesi, tako da ovakav pristup omogućava da se mnogo brže odgovori na zahteve ali i da ih se opsluži mnogo veći broj klijenta.

Niti (11)



Niti – serverska aplikacija

Korisničke niti i niti jezgra

- U zavisnosti od toga da li se nitima upravlja sa korisničkog ili sistemskog nivoa, niti se nazivaju korisničke niti ili niti jezgra.
- Pri tome, pristup procesoru i priliku da se izvršavaju imaju samo niti jezgra, tako da je potrebno napraviti odgovarajuću korespodenciju između korisničkih i niti jezgra.
- Ovo se postiže preslikavanjem (mapiranjem) korisničkih u niti jezgra.

Korisničke niti i niti jezgra (2)

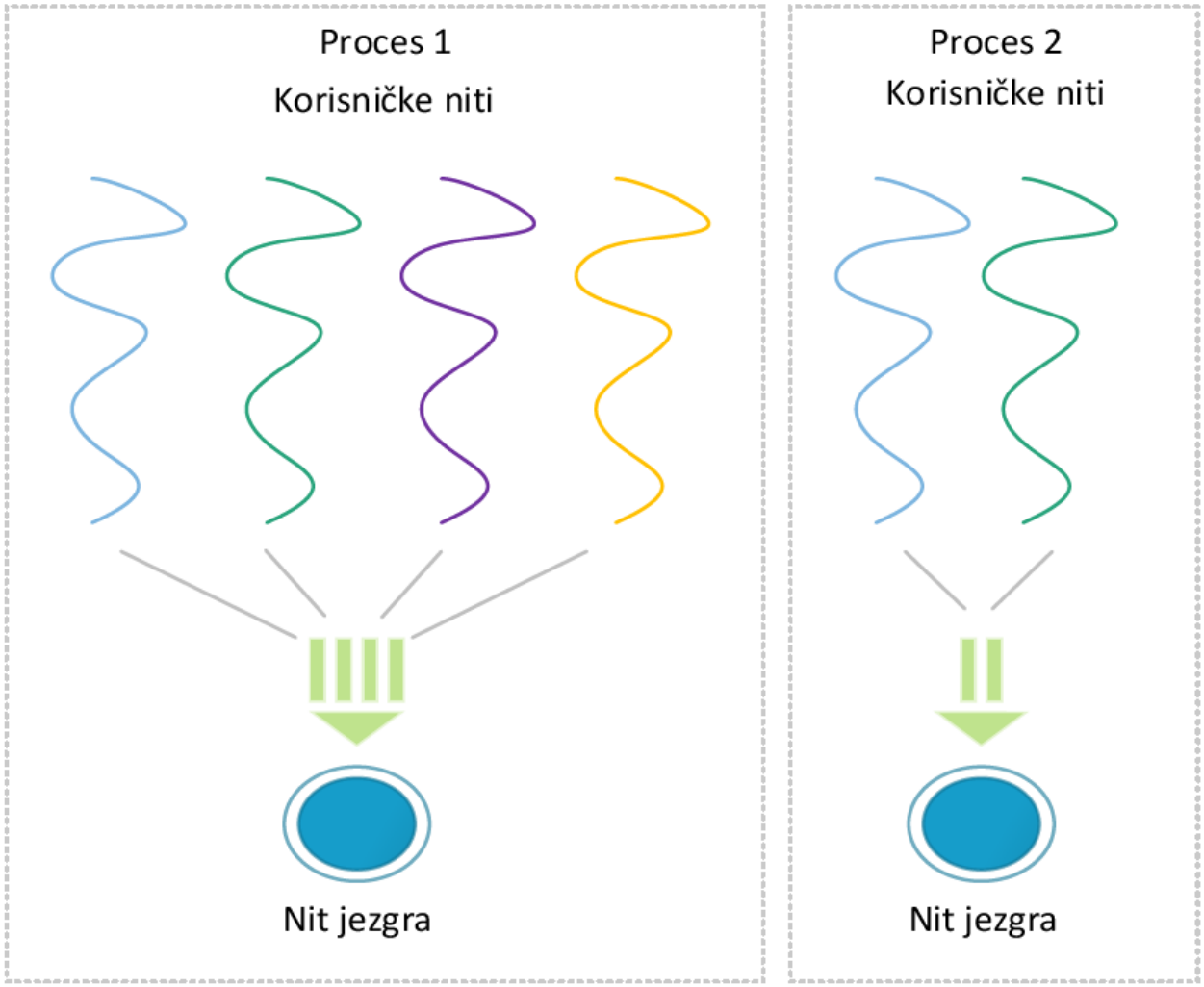
Najčešće podržana preslikavanja su:

- Preslikavanje više u jednu;
- Preslikavanje jedna u jednu;
- Preslikavanje više u više;

Korisničke niti i niti jezgra (3)

- Preslikavanje više u jednu (many-to-one) podrazumeva da se više korisničkih niti, sve koje pripadaju jednom procesu, preslikaju u jednu nit jezgra.
- Broj niti jezgra je u ovom slučaju jednak broju procesa koji postoje u sistemu.
- Dakle, jezgro manipuliše isključivo sa procesima, tako da sve niti jednog procesa izvršava kroz jednu nit jezgra.
- Nitima se upravlja iz korisničkog režima bez uticaja jezgra - odluke o tome koja će se korisnička nit izvršavati donose se na korisničkom nivou.

Korisničke niti i niti jezgra (4)



Preslikavanje više korisničkih niti u jednu nit jezgra

Korisničke niti i niti jezgra (5)

- Međutim, jezgro operativnog sistema i dalje upravlja procesima i time na indirektan način donosi odluke koja će se od niti jezgra izvršavati na procesoru.
- Obično se kreiranje, raspoređivanje i upravljanje korisničkim nitima omogućava implementacijom posebnih biblioteka čija je osnovna funkcija da na korisničkom nivou pruže podršku za rad sa nitima.

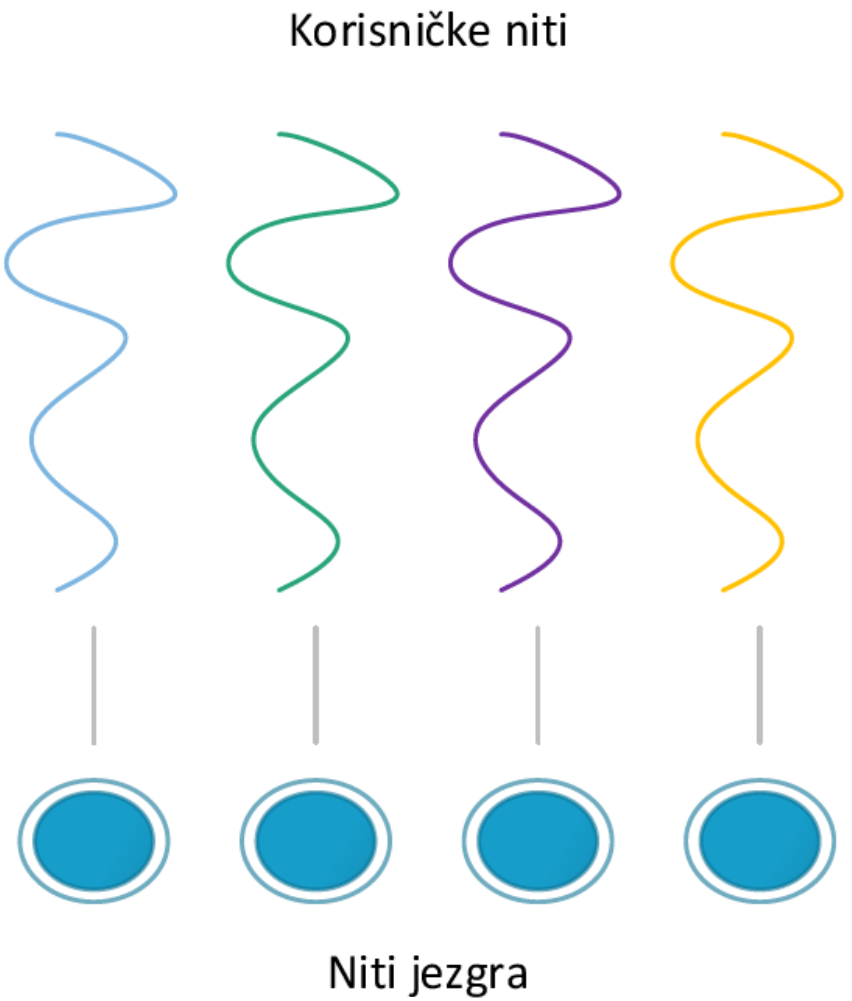
Korisničke niti i niti jezgra (6)

- Nedostatak ovakvog pristupa je što u situacijama kada se neka od korisničkih niti blokira, blokira se i odgovarajuća nit jezgra, tako da dolazi do blokiranja celog procesa tj. svih niti koje ga čine.
- Nadalje, pošto samo jedna nit može pristupiti jezgru u jednom trenutku, ovakvim pristupom (tj. preslikavanjem više u jednu) ne mogu da se iskoriste prednosti računarskih sistema sa višeprocesorskom arhitekturom.

Korisničke niti i niti jezgra (7)

- Preslikavanje jedna u jednu (one-to-one) podrazumeva da se svaka korisnička nit preslika u jednu nit jezgra.
- Pri tome se upravljanje nitima potpuno prepušta jezgru operativnog sistema.
- Niti jezgra se kreiraju sporije i ima ih manje ali, za razliku od korisničkih niti, jezgro ima bolju podršku za rad sa ovim nitima, a nad njima ima i veći stepen kontrole.

Korisničke niti i niti jezgra (8)



Preslikavanje jedne korisničke niti u jednu nit jezgra

Korisničke niti i niti jezgra (9)

- Ovim pristupom se rešavaju glavni problemi prethodnog modela:
  - Obezbeđuje se konkurentnije izvršavanje niti, jer postoji mogućnost da druge niti istog procesa, nastave aktivnosti u slučaju kada se jedna od njih blokira.
  - Takođe, omogućava se da se više niti jezgra izvršavaju paralelno na sistemima sa više procesora.

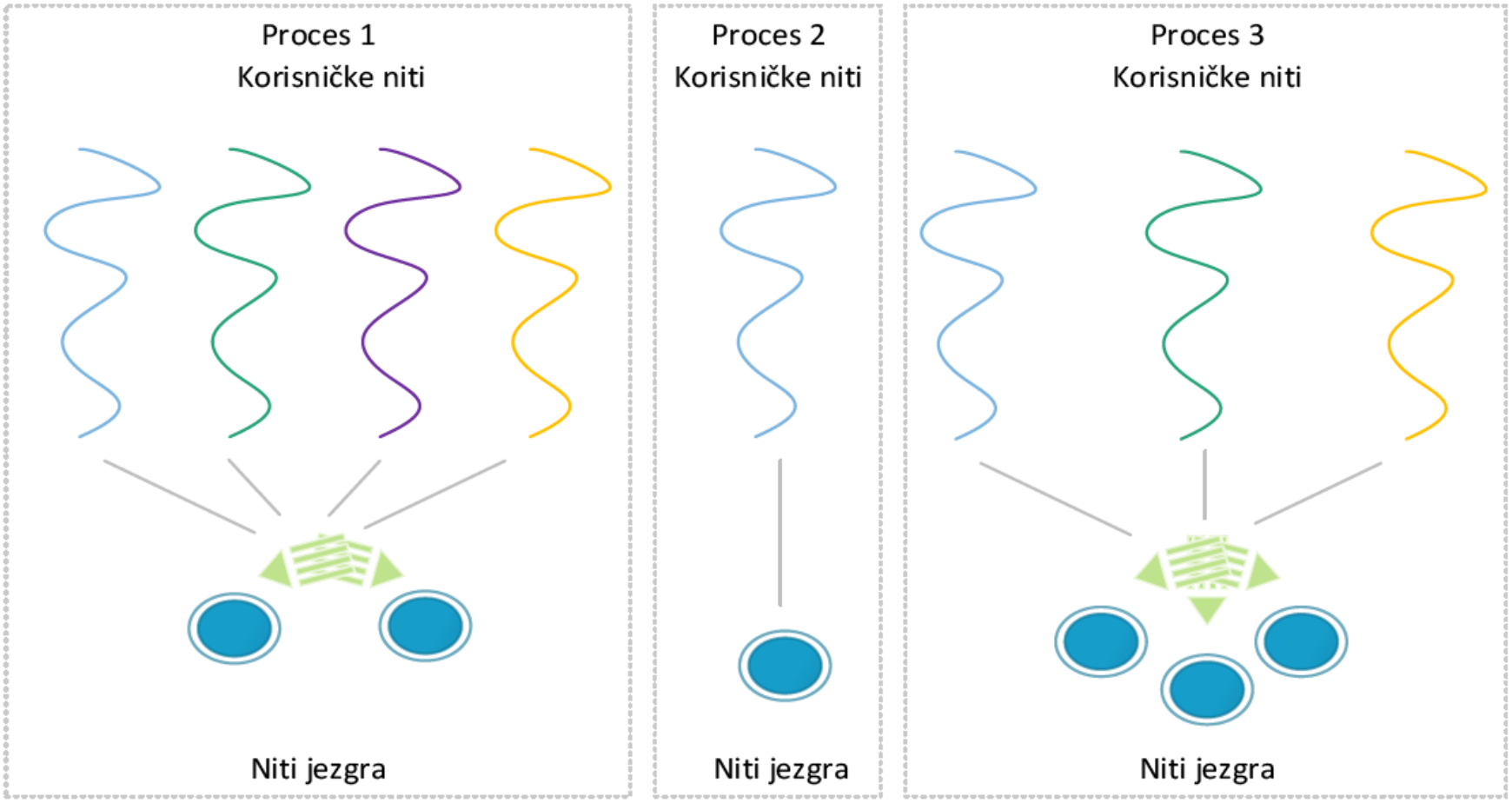
Korisničke niti i niti jezgra (10)

- Najveća mana ovakvog pristupa je ograničavanje broja niti jezgra a samim time i broja niti u korisničkom delu.
- Naime, zbog toga što se nezanemarljivo malo vremena i prostora troši pri stvaranju i održavanju niti jezgra, mnogi operativni sistemi imaju ograničenje dozvoljenog broja ovakvih niti.

Korisničke niti i niti jezgra (11)

- Preslikavanje više u više (many-to-many) je hibrid prethodna dva pristupa jer podrazumeva da se, u zavisnosti od potreba procesa i mogućnosti sistema, korisničke niti jednog procesa preslikavaju u manji (ili isti u ekstremnom slučaju) broj niti jezgra.
- Broj niti kao i način raspoređivanja korisničkih na niti jezgra se određuje na korisničkom nivou, dok se upravljanje prepušta jezgru operativnog sistema.
- Broj niti jezgra koje će biti kreirane zavisi od konkretnog procesa ali često i od broja procesora u sistemu.

Korisničke niti i niti jezgra (12)



Preslikavanje više korisničkih niti u više niti jezgra

Korisničke niti i niti jezgra (13)

- U sistemima sa više procesora prirodno je da se korisničke niti procesa preslikaju u više niti jezgra.
- Ovo je najkompleksniji model za implementiranje ali je i najkvalitetniji jer se može prilagoditi potrebama procesa kao i karakteristikama sistema.

Redovi procesa

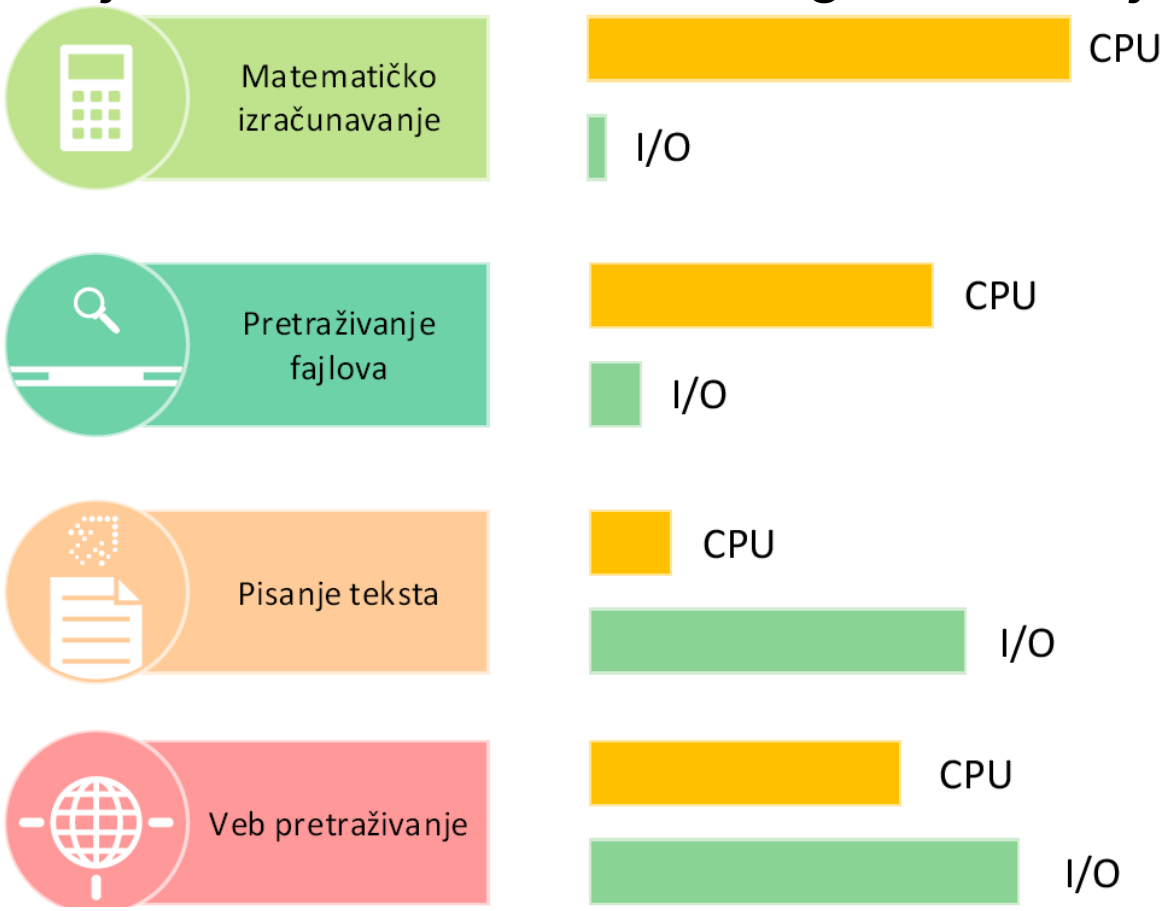
- Jedan od najvažnijih zadataka operativnih sistema je da maksimalno povećaju efikasnost kada je izvršavanje procesa u pitanju.
- Ovo podrazumeva i da se procesor iskoristi na što bolji način, odnosno da se na njemu stalno izvršava neki od procesa ali i da se pri tome procesi smenjuju.
- Izvršavanje procesa se obično sastoji od naizmeničnog korišćenja procesora i čekanja na ulazno-izlazne (U/I) operacije.

Redovi procesa (2)

- Da procesor ne bi bio besposlen U/I operacija procesa koji se izvršava na njemu, nekom drugom procesu se dozvoljava da koristi procesor.
- Kada procesi obave ulazno-izlazne operacije onda im je ponovo potreban procesor do sledeće ulazno-izlazne operacije i tako do kraja izvršavanja.
- Na ovaj način se procesi izvršavaju pseudo-paralelno, a u situacijama kada procesor ima više jezgara ili kada u sistemu postoji više procesora se radi o paralelnom izvršavanju procesa.

Redovi procesa (3)

Važno je istaći da neki procesi više vremena troše na U/I operacije, dok drugi najviše vremena u toku svog izvršavanja koriste procesor.



Potrošnja vremena različitih procesa

Redovi procesa (4)

- Imajući u vidu činjenicu da postoje različite potrebe procesa kada su procesor i U/I uređaji u pitanju, operativni sistemi obično, od svih procesa koji bi trebalo da se izvrše, biraju podskup koji će se učitati u memoriji i izvršavati.
- Posle početne selekcije, operativni sistem ima zadatak da određuje redosled kojim procesi dobijaju procesor i ostale uređaje i koliko vremena mogu da koriste dobijene resurse.
- Poslovi ovakvog tipa se obično poveravaju posebnim modulima operativnog sistema koji se nazivaju planeri.

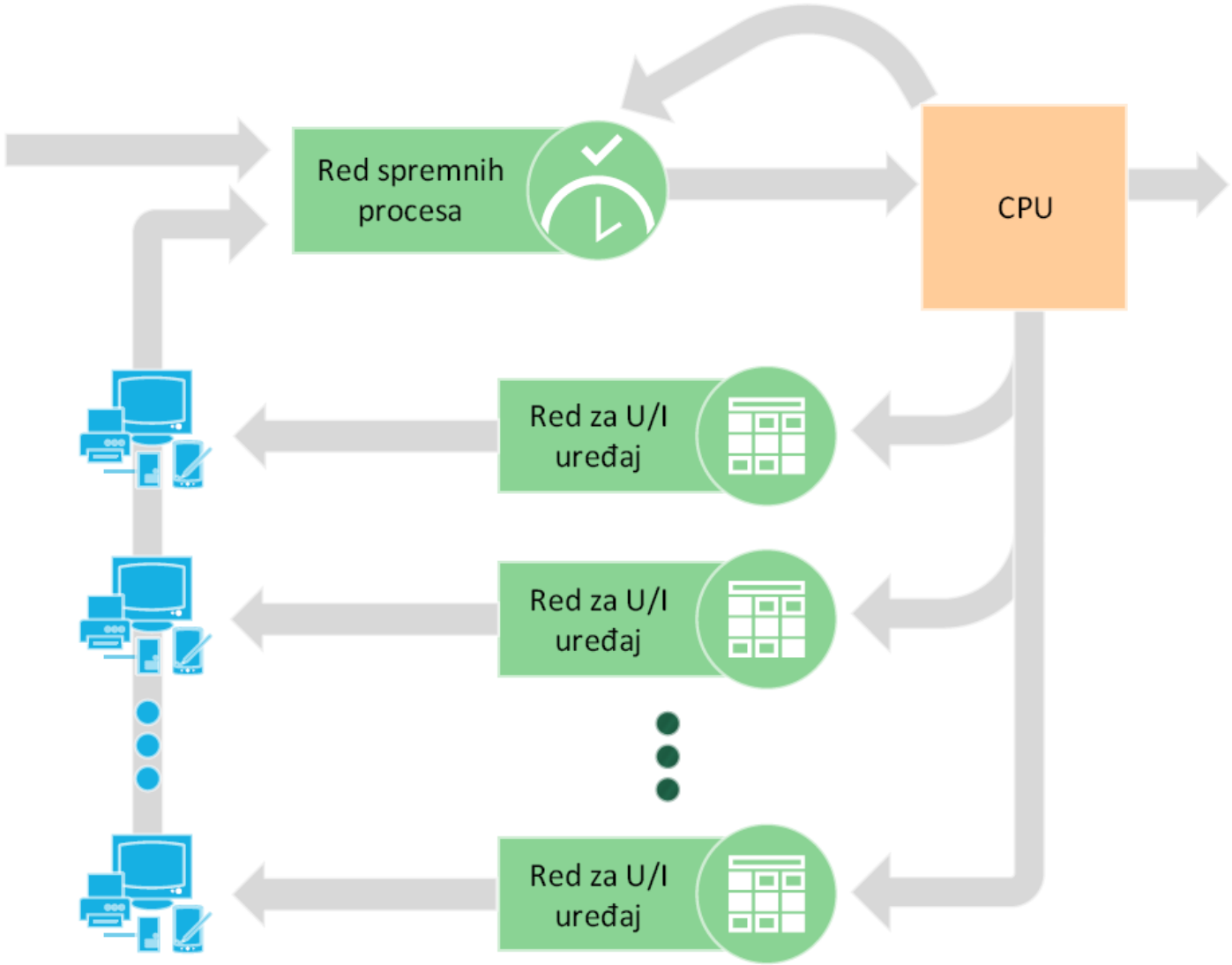
Redovi procesa (5)

- Da bi se olakšalo upravljanje procesima uobičajeno je da se formira nekoliko redova procesa:
  - Početni red, u koji se smeštaju procesi po pokretanju naziva se red poslova. Ovaj red sadrži skup svih procesa u sistemu.
  - Red spremnih procesa sadrži procese koji su izabrani po nekom kriterijumu koji je određen na nivou operativnog sistema. Procesi u ovom redu su spremni za izvršavanje (nalaze se u stanju Spreman) i smešteni su u glavnu memoriju. Izbor procesa koji će ući u red spremnih trebalo bi da zavisi i od potreba tih procesa.

Redovi procesa (6)

- Da bi se olakšalo upravljanje procesima uobičajeno je da se formira nekoliko redova procesa:
  - Redovi procesa koji čekaju na neke od uređaja, a koji se formiraju za svaki od uređaja ponaosob. Procesi koji se nalaze u ovim redovima su u stanju Čekanje.
- Redovi se obično implementiraju kao povezane liste kontrolnih blokova procesa.
- Redovi ne moraju uvek biti organizovane po FIFO (First In, First Out) principu.

Redovi procesa (7)



Red procesa

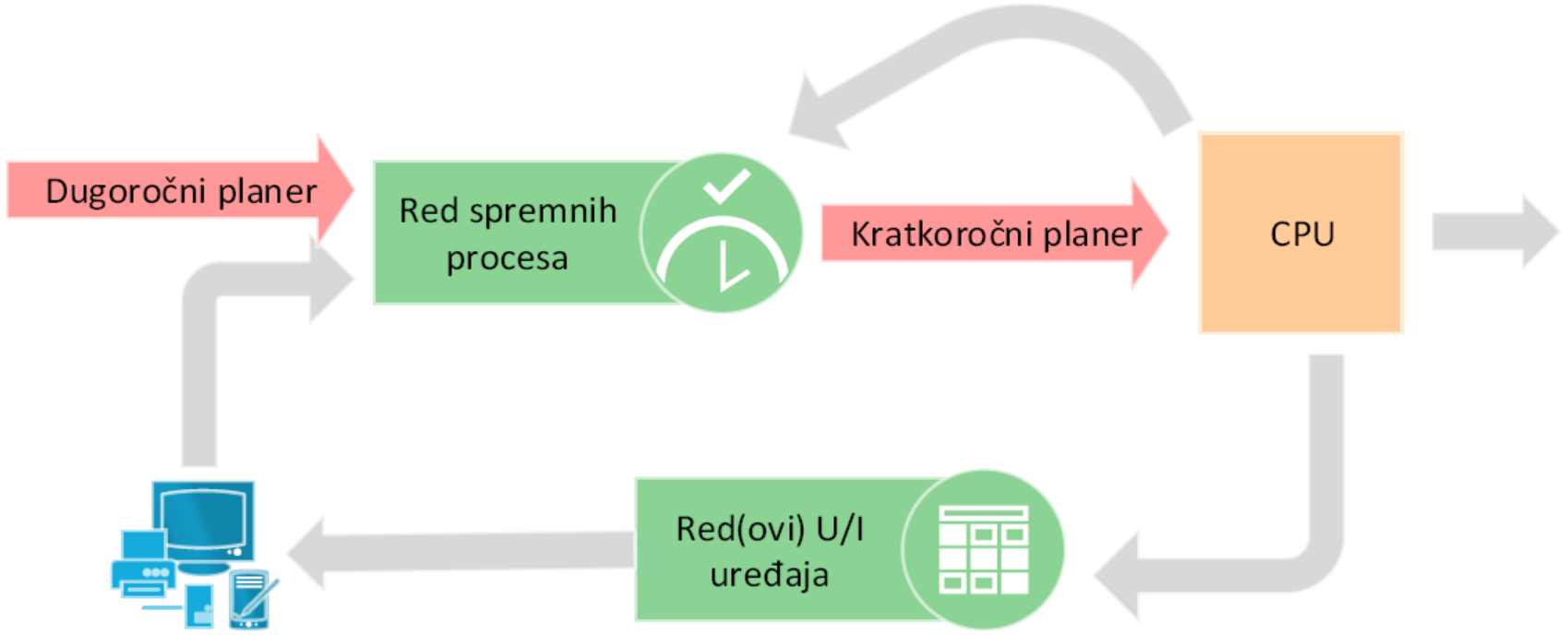
Planeri

- Kao što je već istaknuto, zadatak operativnog sistema je da omogući što veću produktivnost sistema, odnosno da organizuje kretanje procesa kroz različite redove i time omogući njihovo što brže izvršavanje.
- Odabir procesa i njihovo raspoređivanje po redovima ali i organizaciju čekanja vrše planeri.
- Obično postoje bar dve vrste planera: Dugoročni i Kratkoročni.
- Uz njih se često implementira i Srednjoročni planer.

Planeri (2)

- Funkcija Dugoročnih planera je da od skupa svih procesa koji bi trebalo da se izvrše (iz Reda poslova) izaberu one koji će da se aktivno uključe u sistem i počnu sa izvršavanjem.
- Drugim rečima, ovi planeri bi trebalo da naprave dobar odabir procesa za Red spremnih procesa.
- S druge strane, Kratkoročni planeri imaju zadatak da donose odluke o tome koji će se od spremnih procesa izvršavati i koliko dugo će svaki od njih imati procesor na raspolaganju.
- Kratkoročni planer se često naziva planer procesora.

Planeri (3)



Dugoročni i kratkoročni planer

Planeri (4)

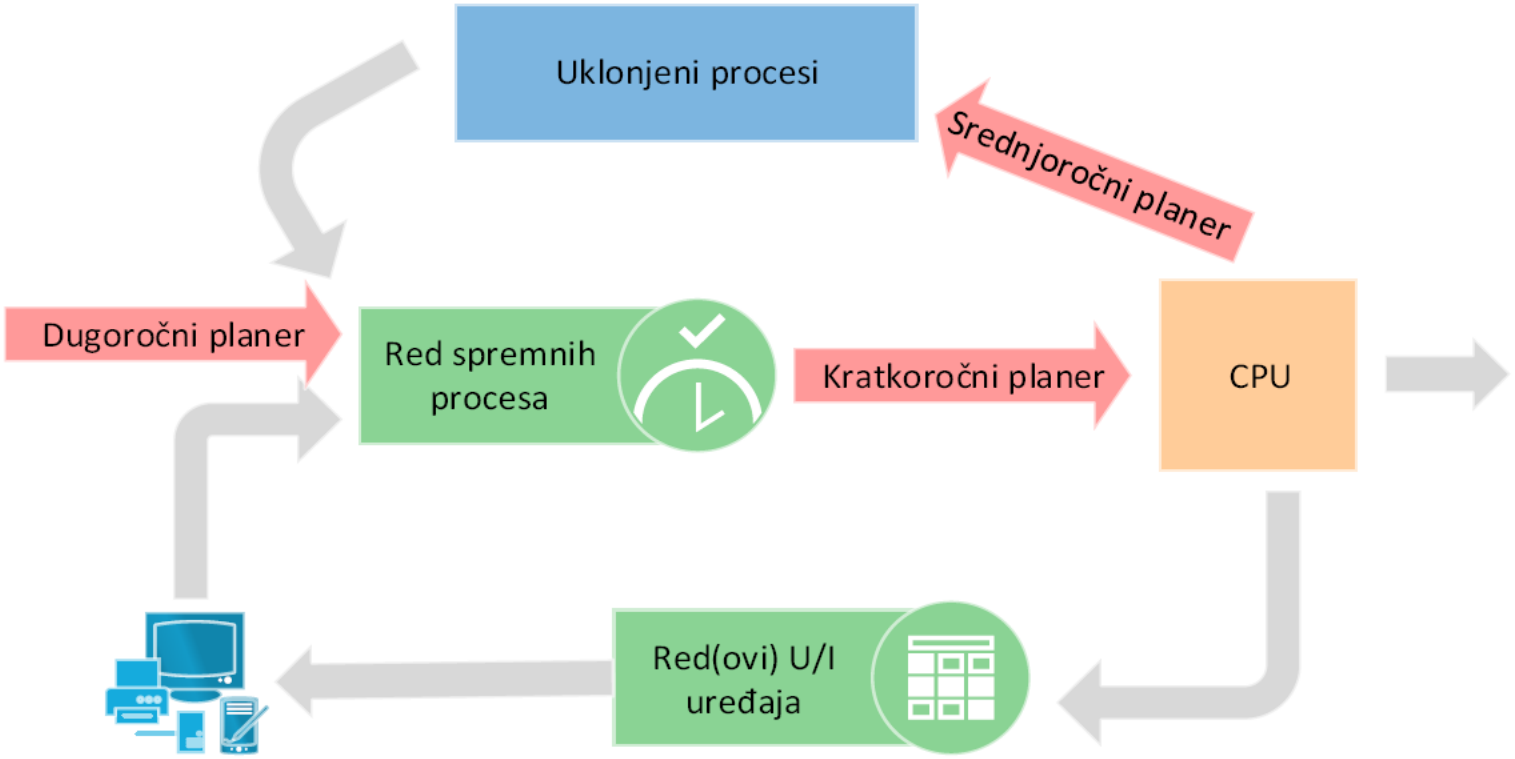
- Kratkoročni planeri se često pozivaju, pa je potrebno da budu implementirani tako da što brže donose odluke.
- Sa druge strane, dugoročni planeri se ne pozivaju učestalo kao kratkoročni, tako da se može dozvoliti njihov nešto sporiji rad.
- U praksi, za dugoročne planere je važnije da naprave dobru kombinaciju poslova nego da uštede na vremenu, dok je kod kratkoročnih planera brzina rada važnija od optimalnosti donete odluke.

Planeri (5)

- Ponekad je skup izabranih procesa u memoriji takav da jedni druge ometaju i zaglavljuju, čime umanjuju efikasnost celokupnog sistema.
- Tada je, često, korisnije suspendovati neke od procesa iz memorije nego forsirati viši stepen multiprogramiranja na uštrb efikasnosti.
- Posle izvesnog vremena, kada se steknu uslovi za to, uklonjeni procesi se mogu ponovo vratiti u memoriju.
- Ovo se naziva prebacivanje (swapping), a izvodi se da bi se poboljšala kombinacija izabranih poslova ili da bi se oslobodila memorija kod zagušenja.

Planeri (6)

- Za zadatke prebacivanja, obično se na nivou operativnog sistema implementira poseban planer koji se naziva srednjoročni planer.



Srednjoročni planer

Višeprocesorski sistemi

- Višeprocesorski sistemi omogućavaju veću efikasnost sistema ali zahtevaju kompleksnije algoritme za raspoređivanje.
- Obično su procesori identičnih ili sličnih karakteristika tako da procesi mogu da konkurišu za bilo koje od njih.
- U takvim situacijama se, pri raspoređivanju procesa, mogu implementirati rešenja koja podrazumevaju jedan zajednički red čekanja za sve procese ili različite redove čekanja procesa za svaki od procesora.

Višeprocesorski sistemi (2)

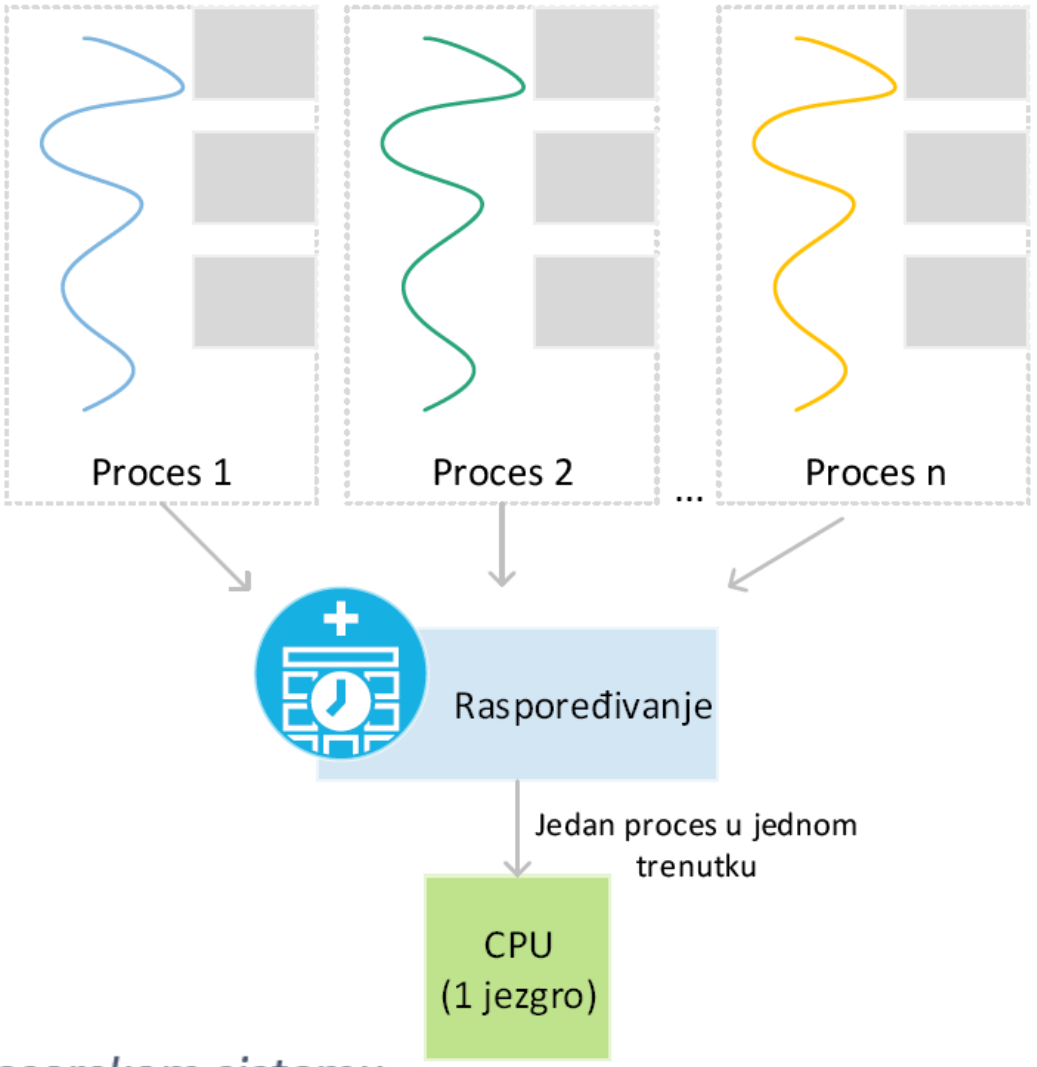
- Pristup sa zajedničkim redom čekanja omogućava ravnomerno opterećenje procesora i češće se koristi.
- Pri implementaciji rešenja sa zajedničkim redom čekanja, postoje dva pristupa:
  - Simetrično multiprocesiranje;
  - Asimetrično multiprocesiranje.

Višeprocesorski sistemi (3)

- Simetrično multiprocesiranje podrazumeva da su procesori ravnopravni i da se procesi raspoređuju na nekog od njih u zavisnosti od primenjenog algoritma za raspoređivanje. Praktično, može se reći da procesori biraju proces koji će izvršavati.
- Asimetrično multiprocesiranje, se zasniva na ideji da se jedan procesor proglasi glavnim i da njegov zadatak bude izvršavanje koda jezgra operativnog sistema a samim tim i da vodi računa o U/I operacijama. Ostali procesori izvršavaju isključivo korisničke procese dodeljene od strane glavnog procesora.

Višeprocesorski sistemi (4)

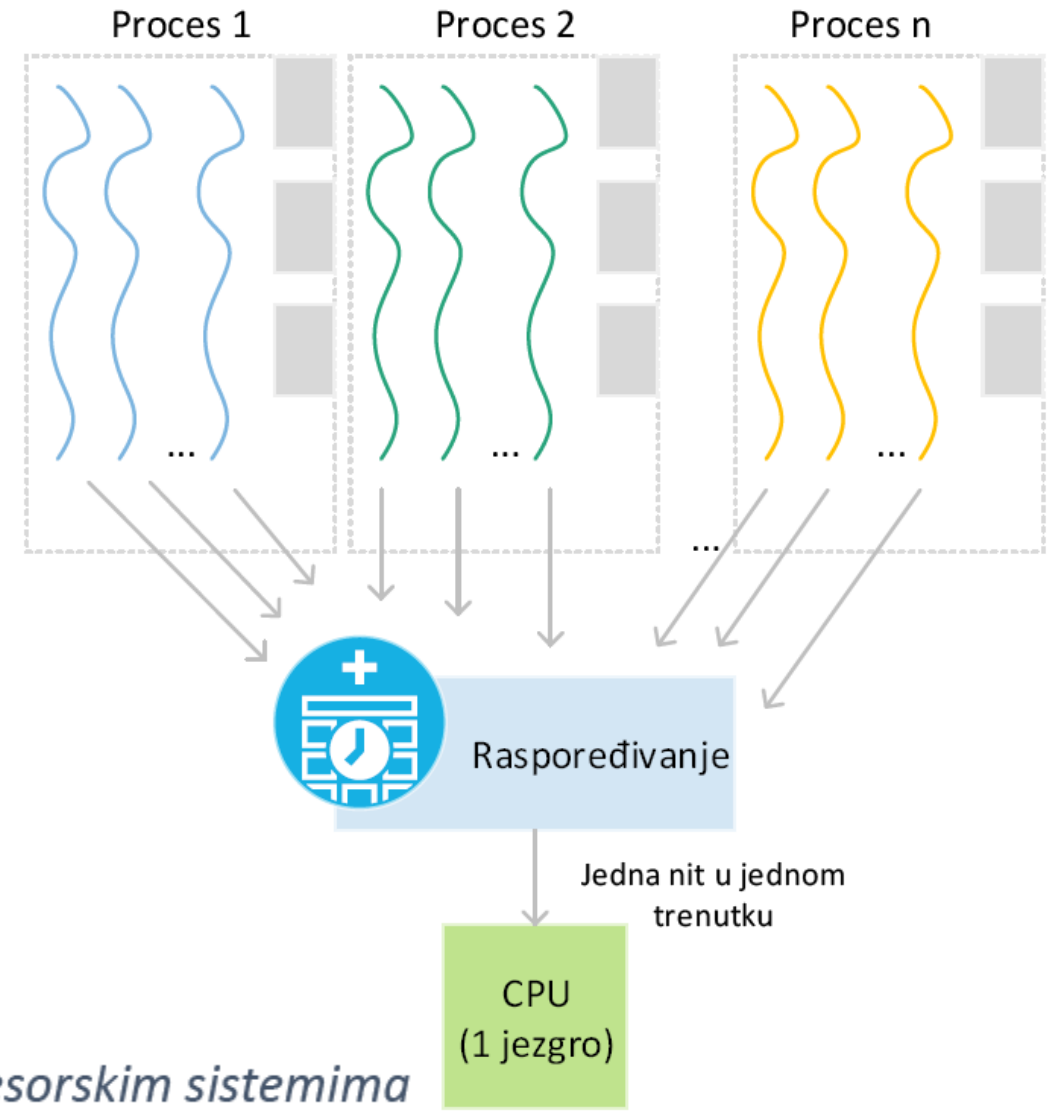
Operativni sistemi koji podržavaju samo procese i izvršavaju se na jednoprocesorskim sistemima vrše odabir procesa koji će se izvršavati na procesoru i pri tome određuju trajanje njihovih aktivnosti na procesoru.



Raspoređivanje procesa na jednoprocesorskom sistemu

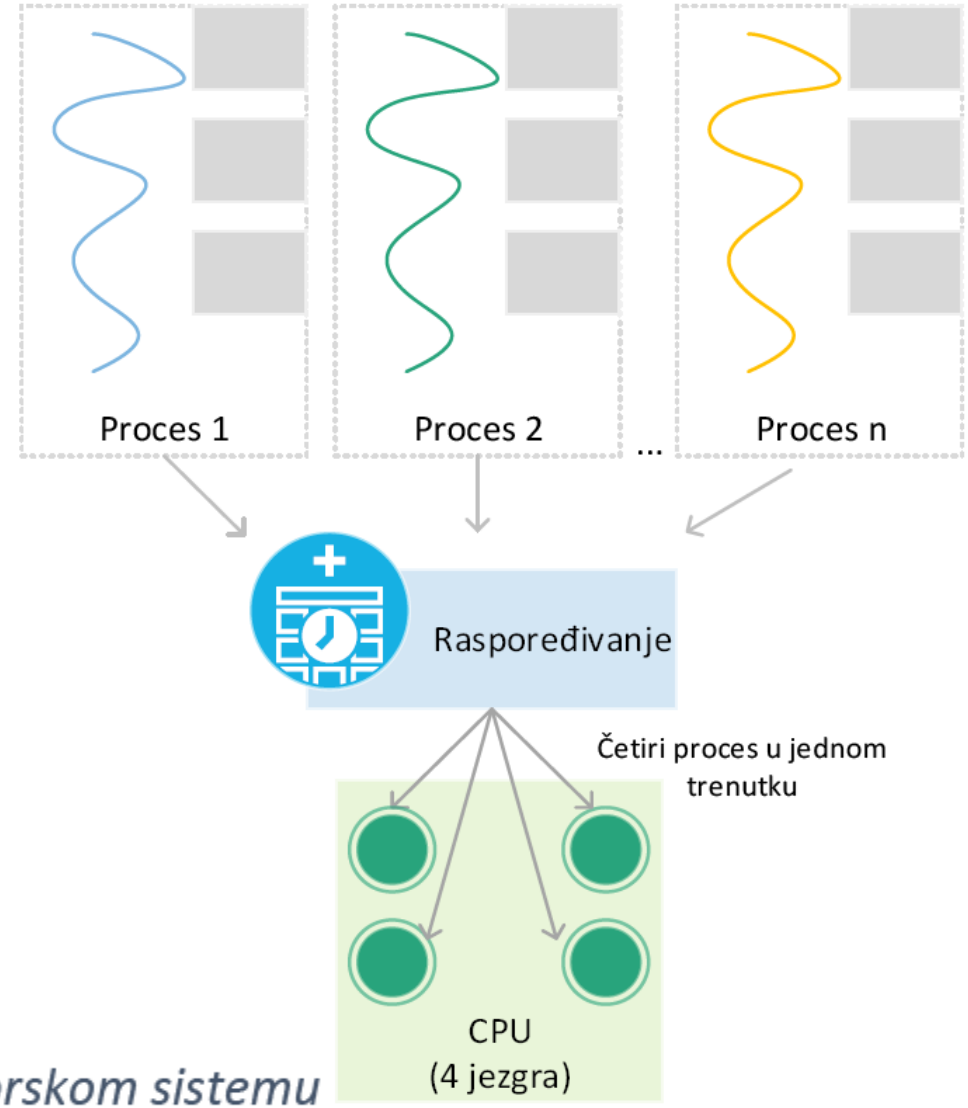
Višeprocesorski sistemi (5)

Operativni sistemi koji podržavaju koncept niti i izvršavaju se na računaru sa jednim procesorskim jezgrom, slično kao i u prethodnom primeru, obezbeđuju da se niti mogu smenjivati stvarajući privid da se izvršavaju paralelno.



Višeprocesorski sistemi (6)

U sistemima sa više procesora, procesi se mogu izvršavati na različitim procesorima (jezgrima) paralelno što sistem čini efikasnijim u odnosu na prethodne.

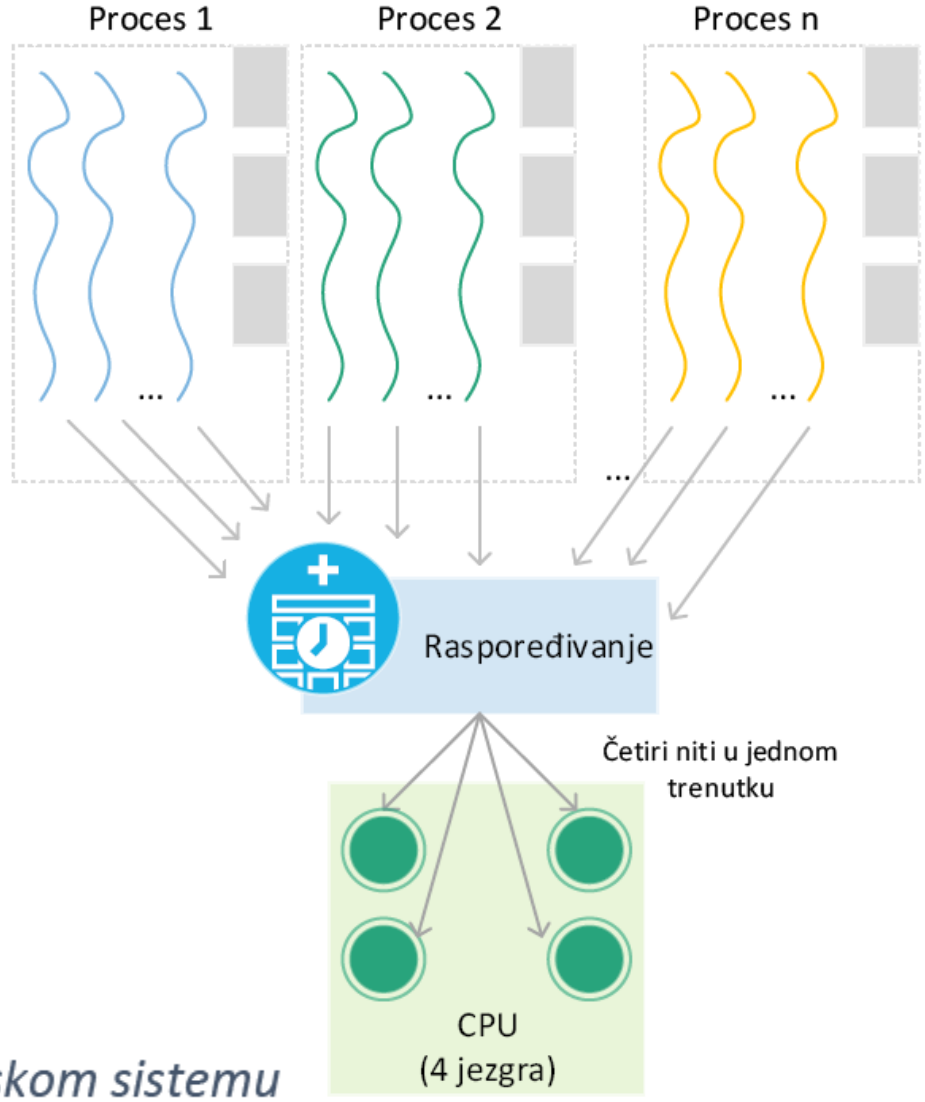


Raspoređivanje procesa na višeprocesorskom sistemu

Višeprocesorski sistemi (7)

Operativni sistemi koji podržavaju rad sa nitima, a izvršavaju se na višeprocesorskim sistemima su najefikasniji ali i najosetljiviji jer zahtevaju pažljivo planiranje.

Naime, potrebno je obezbediti sinhronizaciju niti istog procesa koje se mogu izvršavati na različitim procesorima.



Raspoređivanje niti na višeprocesorskom sistemu

Raspoređivanje procesa

- Pod raspoređivanjem procesa se podrazumeva donošenje odluka o toku izvršavanja (promeni stanja) procesa koji se nalaze u memoriji.
- Raspoređivanje bi trebalo da bude što bliže optimalnom, jer od njega zavisi efikasnost sistema.
- Dobro organizovano raspoređivanje procesa može u značajnoj meri poboljšati brzinu i produktivnost sistema.

Raspoređivanje procesa (2)

- Za algoritme koji će se opisivati se podrazumeva da se operativni sistem izvršava na jednoprocesorskom sistemu, a pojam proces se odnositi na procese, poslove procesa i niti.
- Postoje različiti algoritmi raspoređivanja koji mogu favorizovati određene klase procesa u odnosu na druge, tako da je dobro poznavanje ovih algoritama bitno za odabir prave strategije pri dizajniranju planera operativnih sistema.

Raspoređivanje procesa (3)

Kvalitet algoritama se obično ocenjuje na osnovu:

- Iskorišćenost procesora – odnosi se na procenat koliko je procesor bio zaposlen u određenom vremenskom periodu. Teoretski ovaj procenat može da varira od 0% do 100%.
- Propusna moć je mera koja predstavlja broj procesa koji se mogu završiti u jedinici vremena.
- Vreme obilaska podrazumeva vreme koje protekne od momenta pokretanja procesa do njegovog završetka. Ovo vreme predstavlja sumu vremena koje proces provede u čekanju da se smesti u memoriju, čekanju u redu spremnih procesa, vremena tokom kojeg se izvršavao na procesoru i vremena tokom kojeg je obavljao ulazno-izlazne operacije.

Raspoređivanje procesa (4)

Kvalitet algoritama se obično ocenjuje na osnovu kriterijuma kao što su:

- Vreme čekanja – ukupno vreme koje proces provede čekajući u Redu spremnih procesa. Ono ne podrazumeva vreme izvršavanja procesa na procesoru kao i vreme koje je proces potrošio na U/I uređajima.
- Vreme odziva se odnosi na vreme koje protekne od prijavljivanja procesa do trenutka kada se proizvede prvi izlaz programa. Pri tome se ne računa i vreme potrebno da sa taj izlaz prikaže na nekom od izlaznih uređaja.
- Količnik dužine vremena obilaska i trajanja izvršavanja procesa takođe može biti veoma koristan parametar jer dobro opisuje situaciju u sistemu.

Raspoređivanje procesa (5)

- Optimizacija raspoređivanja procesa u sistemu podrazumeva pronalaženje rešenja koje odlikuje:
  - Maksimalna iskorišćenost procesora;
  - Maksimalna propusna moć;
  - Minimalno vreme obilaska;
  - Minimalno vreme čekanja;
  - Minimalno vreme odziva;
  - Konvergencija količnika dužine vremena obilaska i vremena izvršavanja procesa ka 1.
- Najčešće je nemoguće istovremeno uticati na sve navedene parametre, tako da se, akcenat može staviti na neke od njih.

Raspoređivanje procesa (6)

- U narednom delu biće prikazani neki od tradicionalnih algoritama planiranja.
- Važno je napomenuti da neki od algoritama imaju i verzije koje dozvoljavaju prekidanje procesa. Prekidanje procesa podrazumeva prebacivanje konteksta procesa koji se izvršava i učitavanje drugog procesa, na osnovu odluke planera, pre nego što je prvi proces iskoristio procesor onoliko koliko mu je bilo potrebno da obavi posao, da dođe do ulazno-izlaznih operacija ili da se izvrši do kraja.

Raspoređivanje procesa (7)

- Algoritam koji se zbog svoje jednostavnosti i pravednosti obično prvi nameće je FCFS (First Come, First Served).
- On je zasnovan na ideji da se procesor dodeljuje procesima po redu kako su ga zatražili, tj. prijavili se za njegovo korišćenje.
- FCFS algoritam se jednostavno implementira korišćenjem FIFO (First In, First Out) strukture.
- FCFS algoritam nema verziju koja bi dozvoljavala prekidanje jer bi to bilo u suprotnosti sa glavnim principom na kojem je zasnovan.

Raspoređivanje procesa (8)

- Prednosti ovog algoritma su laka implementacija i pravednost, ali je vreme čekanja često veliko.
- Takođe, ovakav pristup može dovesti do takozvanog „efekta konvoja“, situacije kod koje dosta procesa čeka da se izvrši jedan vremenski zahtevan proces.

Raspoređivanje procesa (9)

- SPF (Shortest-Process-First) algoritam se zasniva na ideji da se favorizuju procesi čiji je sledeći zahtev za procesorom (procesorskim vremenom) manji.
- Svakom procesu se pridružuje očekivano vreme za njegovo sledeće izvršavanje (aktivnost) na procesoru, a procesor se dodeljuje redom procesima koji zahtevaju manje vremena.
- Ako dva procesa imaju isto očekivano vreme, obično se primenjuje FCFS algoritam za odluku koji proces će dobiti prednost.

Raspoređivanje procesa (10)

- SPF algoritam se može implementirati bez prekidanja, ali i u verziji sa prekidanjem. Razlika između ova dva pristupa je u tretmanu novih procesa koji imaju kraći sledeći zahtev za procesorskim vremenom u odnosu na proces koji se izvršava na procesoru.
- Najveći problem, kada je ovaj algoritam u pitanju, predstavlja određivanje dužine sledećeg zahteva za procesorom, odnosno procena dužine trajanja sledeće aktivnosti procesa na procesoru.
- U takvoj situaciji efikasnost sistema zavisi od mogućnosti da se realno proceni trajanje posla.

Raspoređivanje procesa (11)

- Algoritam sa prioritetima podrazumeva da se svakom procesu pridruži njegov prioritet i da se kao sledeći za izvršavanje bira onaj koji ima najviši prioritet.
- Ukoliko se dogodi da dva procesa imaju isti prioritet bira se onaj koji je pre došao u sistem.
- SPF algoritam je specijalan slučaj algoritma sa prioritetima gde se prioriteti procesa definišu tako da budu obrnuto proporcijalni očekivanom vremenu izvršavanja procesa.
- FCFS algoritam je specijalan slučaj algoritma sa prioritetima gde je prioritet vreme dolaska u sistem.

Raspoređivanje procesa (12)

- Prioriteti se mogu definisati interno ili eksterno.
  - Interno definisani prioriteti koriste neku merljivu veličinu za izračunavanje prioriteta procesa.
    Često se pri određivanju prioriteta uz obzir uzimaju vremenske granice, memorijski zahtevi, broj otvorenih fajlova, prosečno trajanje aktivnosti na procesoru, prosečno trajanje ulazno-izlaznih operacija, količnik poslednje dve veličine, itd.
  - Eksterni prioriteti se nameću na osnovu kriterijuma koji su spoljašnji u odnosu na operativni sistem odnosno, uglavnom su „političke“ prirode.
    U tom slučaju na prioritet procesa mogu uticati vremenski rokovi, važnost klijenta, vrednost projekta, itd.

Raspoređivanje procesa (13)

- Za algoritam sa prioritetima postoji verzija sa i bez prekidanja.
- Najveća mana ovog algoritma je potencijalno „izgladnjavanje“ procesa - može se dogoditi da procesi nižeg prioriteta dugo (teoretski i beskonačno) čekaju na procesor jer u sistem stalno pristižu procesi višeg prioriteta.
- Ovaj problem se može rešiti povećavanjem prioriteta procesa sa porastom njegovog vremena provedenom u redu za čekanje.

Raspoređivanje procesa (14)

- Kružni algoritam (Round Robin) je zasnovan na ideji za opsluživanje računara kao kada je multikorisnički interaktivni rad (timesharing) u pitanju.
- Svaki proces dobija procesor na unapred zadati vremenski trenutak koji se naziva kvantum vremena (time quantum).
  Kada to vreme istekne, proces se prekida i stavlja na kraj reda spremnih procesa a procesor na kvantum vremena dobija sledeći proces iz reda spremnih procesa.

Raspoređivanje procesa (15)

- Ako ima n spremnih procesa, a vremenski kvantum je dužine *q*, onda svaki proces dobija 1/*n* procesorskog vremena najviše po *q* u jednom prolazu.
- Performanse kružnog algoritma veoma zavise od kvantuma vremena.
  - Ako taj kvantum vremena veći od svih mogućih zahteva za procesorom (teži beskonačnosti), ovaj algoritam se ponaša kao FCFS algoritam.
  - Ukoliko je vremenski kvantum veoma mali, kružni algoritam se naziva deljenje procesora i teoretski izgleda da svaki od n procesa u redu ima sopstveni procesor koji je n puta sporiji od procesora koji se nalazi u sistemu.

Raspoređivanje procesa (16)

- Na kraju svakog kvantuma vremena vrši se prebacivanje konteksta procesa.
- Trajanje prebacivanja konteksta zavisi od brzine memorije, broja registara i postojanja specijalnih instrukcija.
- Poželjno je da se kvantum vremena izabere tako da bude znatno duži od vremena potrebnog za prebacivanje konteksta.
- Izbor kvantuma vremena je veoma delikatan posao jer vidimo da je bolje imati duži kvantum vremena, ali ako je kvantum preveliki onda algoritam konvergira ka slabom FCFS algoritmu.

Raspoređivanje procesa (17)

- Redovi u više nivoa su algoritam koji je zasnovan na ideji da se red spremnih procesa podeli u više redova sa različitim prioritetima.
- Poslovi koji pristižu svrstavaju se u odgovarajući red na osnovu nekog kriterijuma.
- Svaki red može imati sopstveni algoritam planiranja, a između samih redova postoji jasno definisana razlika u prioritetu.
- Posao koji je prvi u svom redu može dobiti procesor na korišćenje isključivo ako su redovi višeg prioriteta prazni.

Raspoređivanje procesa (18)

- Redovi u više nivoa sa povratnom vezom su modifikacija prethodnog algoritma.
- Kod redova u više nivoa sa povratnom vezom dozvoljeno je kretanje procesa između redova.
- Ideja je da se u toku izvršavanja izdvoje procesi sa različitim trajanjem aktivnosti na procesoru tako da se procesi koji koriste previše procesorskog vremena premeste u redove sa nižim prioritetom.
- Ovakvim pristupom, procesi orijentisani više ka ulazno-izlaznim operacijama i interaktivni procesi se drže u redovima višeg prioriteta.

Raspoređivanje procesa (19)

- Svaki proces u višim redovima dobija određeni kvantum vremena i kada ga iskoristi premešta se u red na nižem nivou..

Zahvalnica

Najveći deo materijala iz ove prezentacije je preuzet iz knjige „Operativni sistemi“ autora prof. dr Miroslava Marića i iz slajdova sa predavanja koje je držao prof. dr Marić.

Hvala prof. dr Mariću na datoj saglasnosti za korišćenje tih materijala.