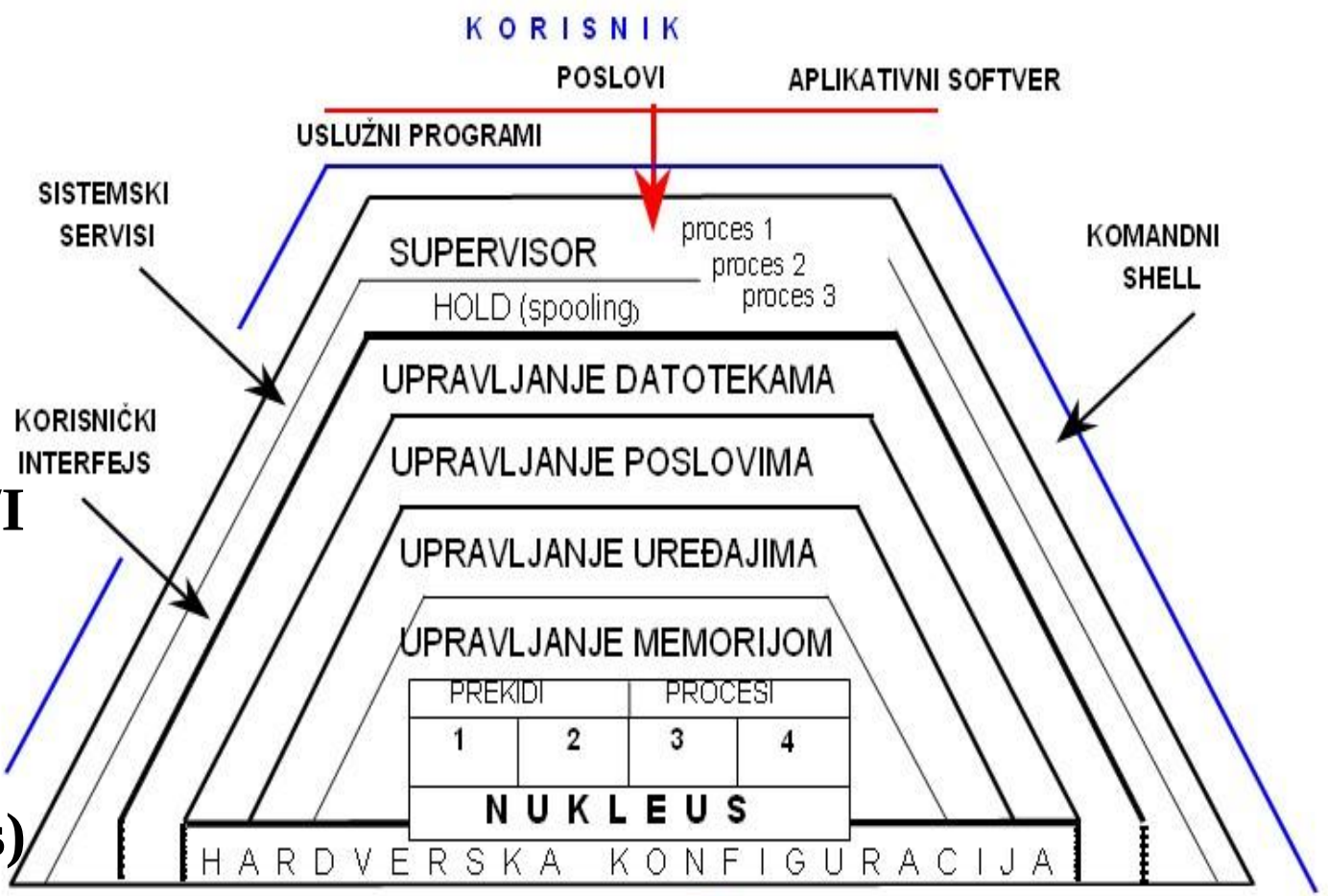
S A D R Ž A J

2.1 - Jezgro (*Nucleus* ili *Kernel*)

Računarski sistem posmatramo kao skup softverskih i hardverskih koponenti sa tačno definisanim ulogama i međusobnih veza

➢Struktura operativnog sistema može se podeliti na sledeće delove:

1. **Komandni interpretator**
2. **Upravljanje datotekama**
3. **Planiranje i evidencija**
4. **Upravljanje U/I uređajima**
5. **Upravljanje memorijom**
6. **Jezgro(nukleus)**



2.1 - Jezgro (*Nucleus* ili *Kernel*)

Predstavlja osnovni deo svakog operativnog sistema

- **najbliže je hardveru računara** (osim kod strukture sa *HAL - Hardware Abstraction Level* kod Windows OS*)*
- **konekcija** - obezbeđuje vezu između hardvera računara i ostalih slojeva operativnog sistema
- **apstrakcija hardvera** - omogućava programima da pristupe hardveru bez nekog njegovog detaljnog poznavanja
- **upravlja procesima**: kada i koji proces će se izvršavati kao i omogućava međusobnu komunikaciju između procesa
- **nije potrebno** samo ako programi ne žele da koriste njegove usluge i pristupaju direktno hardveru – nemogućnost prelaska iz jednog u drugi program (prilikom prelaska sa jednog na drugi program zahteva se resetovanje računara)

2.1 - Jezgro (*Nucleus* ili *Kernel*)

Osnovni zahtevi koji se zahtevaju od hardvera računara:

1. **Mehanizam prekida -** omogućuje izvršavanje prekidne rutine koja obezbeđuje prelazak sa korisničkog na upravljački program, tj. omogućuje izvršavanje više uporednih programa
2. **Zaštitni mehanizam adresiranja memorije** – sprečava pogrešno adresiranje, tj. mešanje podataka iz dva procesa
3. **Skup privilegovanih instrukcija** – instrukcije koje nisu dostupne svima, ali se zahteva da procesor ima dva načina rada: **korisnički** i **sistemski** režim
4. **Časovnik realnog vremena** – kontrola i evidentiranje potrošnje resursa računara za sve aktivne procese. Omogućuje raspoređivanje i izvršavanje raznih poslova u nekom trenutku.

2.1 - Jezgro (*Nucleus* ili *Kernel*)

Osnovni delovi jezgra:

1. **Prvi nivo obrade** (***FLIH**-First Level Interrupt Handler*) – određivanje izvora prekida i iniciranje servisa koji treba da se izvrše na osnovu tog prekida
2. **Dispečer sistema** (*low-level scheduler*) – dodeljuje procesor određenom procesu na osnovu nekog unapred definisanog algoritma za dodeljivanje
3. **Rutine za interprocesnu komunikaciju** – obezbeđuju da procesi koji se konkurentno (paraleno) izvršavaju mogu međusobno da komuniciraju putem: slanja poruka, semafora, imenovanih cevi (*named pipes*) ili korišćenjem zajedničke memorije

2.2 - Proces

Predstavlja deo programa koji je u stanju izvršavanja
Resurs koji se može dodeliti procesoru za izvršavanje
Resurs koji nešto radi na računaru
Aktivnost koja se sastoji od niza instrukcija koje procesor izvršava

- Svaki proces se sastoji i zauzima u memoriji tri različita memor.dela:

1. **Programski** - kod procesa (može biti zajednički za jedan program)
2. **Stek** – privremene podatke: lokalne promenljive
3. **Podatke** – globalne promenljive koje proces obrađuje

- Osim memorijskih delova proces obuhvata i vrednosti registara CPU
- Različiti nazivi za aktivnosti CPU: procesi (*processes*), poslovi (*jobs*), zadaci (*tasks*)

2.3-Predstavljanje procesa

- U svakom trenutku svi procesi su jedinstveno definisani **PCB** struktur.
- PCB (*Process Control Blok*) je deo radne memorije u kojoj se čuva:

- Jedinstveni identifikator procesa (PID) - uniformno definiše svaki proces
- Trenutno stanje procesa – spreman (*ready*), čeka (*wait*) i izvršava (*run*)
- Konteks (okruženje) – podaci o registrima CPU dok se proces izvršava (često se naziva i HPD - *Hardware Process Descriptor*)
- Prioritet procesa – važnost procesa u odnosu na ostale procese
- Memorijski ukazatelji – programski kod, podaci, mem.blokovi koji su zajednički sa ostalim procesima
- Listu otvorenih datoteka – datoteke koje koristi proces
- Status zauzetih U/I resursa – U/I zahtevi, U/I uređaji koji su dodeljeni
- Dodatne informacije-količinu procesor.vremena, vremenske limite, ...

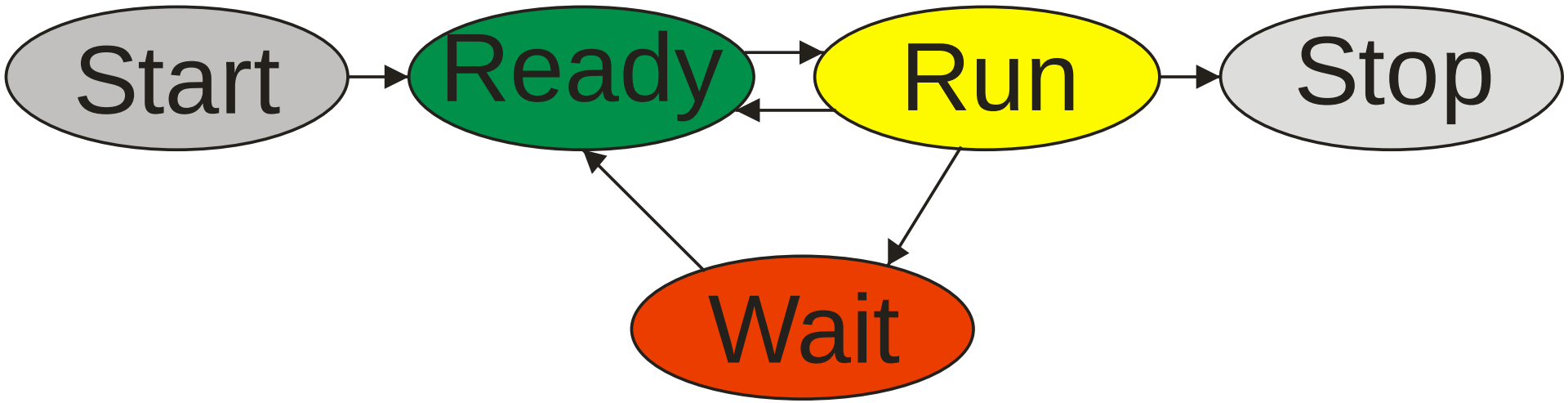
PCB se uvek nalazi u različitom memorijskom prostoru od koda procesa.

2.3 Stanja procesa

Proces u toku svog rada prolazi kroz nekoliko stanja:

1. **Start** - formiranje procesa koji ne mora da je u rad.memoriji
2. **Spreman**(*Ready*) - nalazi se u radnoj memoriji, ima sve resurse, i čeka da mu se dodeli procesor
3. **Izvršavanje** (*Run*) - procesor izvršava instrukcije procesa
4. **Čeka** (*Wait*) - proces nema sve resurse da bi se izvršio
5. **Stop** - kraj izvršenja procesa, brše se PCB iz liste aktivnih procesa pa proces može da se izbaci iz radne memorije
6. **Obustavljen/Spreman** (*Suspend/Ready*) - proces koji se nalazi u stanju spreman može biti obustavljen i izbačen iz memorije
7. **Obustavljen/Čeka** (*Suspend/Wait*) - proces koji se nalazi u stanju čeka može biti obustavljen i izbačen iz memorije

2.3 Dijagram stanja procesa



Start-Ready: prebacivanje procesa u radnu memoriju

Ready-Run: dodeljivanje procesora procesu

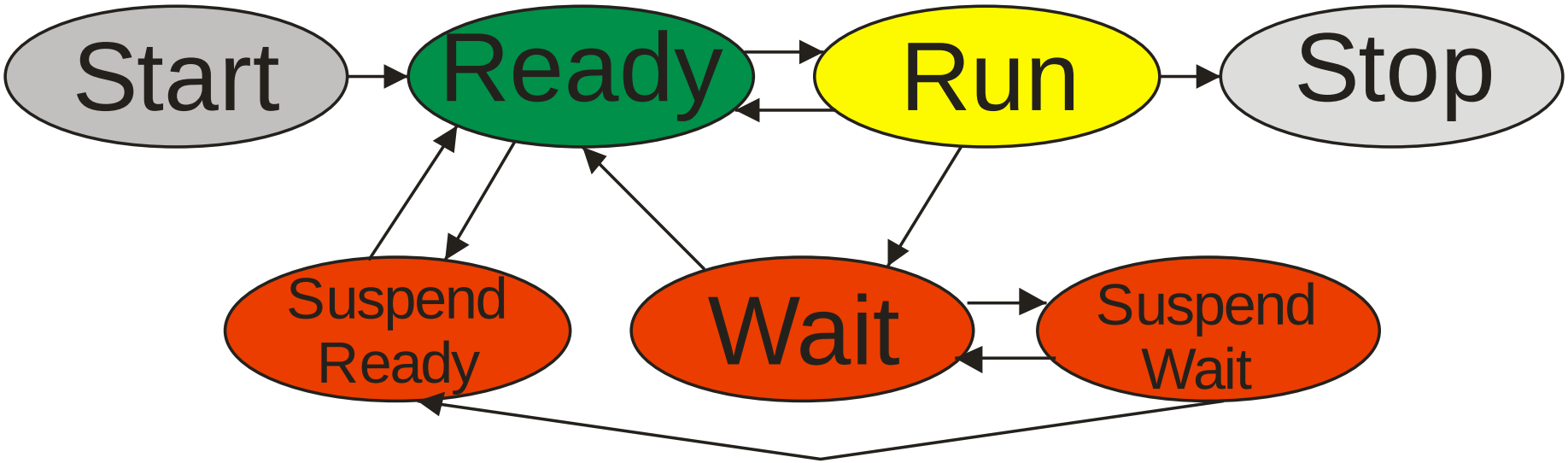
Run-Ready: oduzimanje procesora procesu (istek vremena, prekid)

Run-Wait: proces nema sve resurse da bi mogao da se kompletira/izvrši

Wait-Ready:proces je dobio resurs koji mu je neophodan da bi se izvršio

Run-Stop: proces završava rad (prirodno ili nasilno)

2.3 Dijagram stanja procesa



Ready-Suspend/Ready: veliki broj procesa u stanju Ready, izbegavanje zastoja, eksplicitno obustavljanje procesa

Suspend/Ready-Ready: eksciplitno na zahtev korisnika

Suspend/Wait-Wait: eksiplitno na zahtev korisnika

Wait-Suspend/Wait: veliki broj procesa u stanju Wait

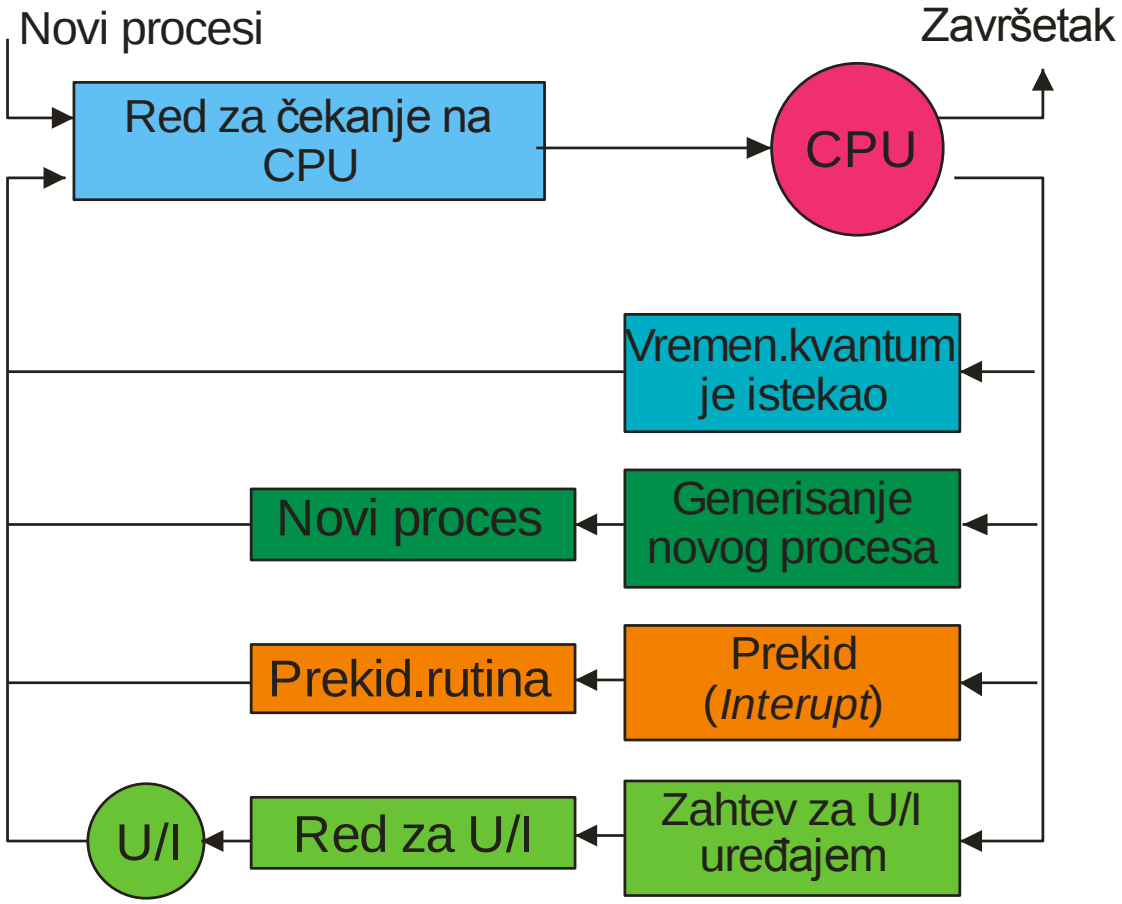
Suspend/Wait-Suspend/Ready: oslobođen je neophodni resurs za izvršenje procesa ali je proces i dalje obustavljen

2.4 Raspoređivanje procesa

- Procesi mogu da se nađu na raznim lokacijama u sistemu
- Svi spremni procesi nalaze se u radnoj (operativnoj) memoriji
- Svi procesi prolaze kroz nekoliko redova čekanja koji se realizuju kao povezane liste koje sadrže PCB procesa

Proces koji se izvršava može:

1. Ostati bez CPU
2. Napraviti novi proces
3. Biti prekinut – prekid
4. Zahtevati neki U/I resurs



2.4 Planer poslova i dispečer

- Dva osnovna programa za raspoređivanje procesa su: **planer poslova** i **dispečer sistema**
- Funkcije **planera poslova** (*job scheduler*):
  1. Podela poslova na procese
  2. Na osnovu određenih algoritama pravi redosled izvršavanja procesa
  3. Smešta procese u red čekanja na procesor
- Poziva se samo kada dođe neki novi proces ili kada se proces završi
- Uloga **dispečera sistema**(*dispatcher*) je da dodeli procesor procesu:
  1. Izborom najprioritetnijeg procesa
  2. Promenom konteksta procesa – potrebno je da upamti stanje tekućeg procesa (kako bi mogao da ga vrati kasnije kada se završi prekidni proces) i da napuni memoriju parametrima novog (prekidnog) procesa koji treba da se izvršava

2.5 Operacije nad procesima

1. Kreiranje novog procesa
2. Pravljenje veze proces-proces roditelj
3. Zaustavljanje (terminiranje) rada procesa
4. Izbacivanje procesa (brisanje PCB iz reda za čekanje na CPU)
5. Promena stanja procesa
6. Promena prioriteta procesa
7. Izbor procesa
8. Blokiranje procesa
9. Deblokiranje procesa
10. Obustavljanje (*suspend*) procesa
11. Reaktiviranje procesa
12. Vraćanje procesa iz obustavljenih stanja

2.6 Rutine za interproc.komunika.

Svi procesi u odnosu na međusobnu zavisnost toka izvršavanja dele se:

1. Nezavisne-ne utiču na izvršenje drugih procesa, niti drugi procesi utiču
2. Kooperativne - predstavljaju procese koji imaju međusobni uticaj jedni na druge, dele zajedničke podatke ili bilo kakve resurse.Karakteriše ih:
   a) Deljenje informacija - obezbediti ravnopravni pristup podacima
   b) Ubrzanje rada - ima smisla kod višeprocesnih sistema kao i višestrukim U/I (RAID struktura)
   c) Modularnost - podela sistemskih funkcija na procese i niti
   d) Pogodnost - omogućavanje da se više poslova odvija istovremeno

Potrebno je da zavisni procesi mogu da međusobno komuniciraju

Komunikacija između procesa vrši se pomoću:

1. Korišćenjem zajedničke memorije - bafera
2. Slanjem poruka
3. Semaforske tehnike
4. Korišćenjem imenovanih cevi (*named pipe*)

2.6 Korišćenje zajedničkog bafera

Bafer je realizovan kao cirkularna struktura od N elemenata (0, N-1)

Postoje 2 pokazivača: **in** (prvo slobodno mesto) i **out** (prvo puno mesto)

Proizvođač

```
item sledeći_proizvedeni;
while (1) {
  /*proizvođač proizvodi*/
  while (((in+1)%N)==out);
   /*bafer je pun, čeka se da potrošač nešto uzme iz bafera*/
   buffer[in]=sledeći_proizvedeni;
   in=(in+1)%N;
}
```

Protrošač

```
item sledeći_potrošen;
while (1) {
  while (in==out);
   /*bafer je prazan, čeka se da proizvođač nešto stavi u bafer*/
  sledeći_potrošen=buffer[out];
  out=(out+1)%N;
/*potrošač koristi(uzima)*/
}
```

2.6 Slanje poruka

- Primenjuje se u distribuiranim sistemima kada se procesi odvijaju na dva nezavisna računara koji su fizički odvojeni
- Moraju da budu podržane sledeće dve operacije:
  1. **Slanje poruke** (*send massage*)
  2. **Prijem poruke** (*receive message*)
- Poruke mogu da budu:
  1. Fiksne dužine (manje fleksibilne ali se lakše implementiraju)
  2. Promenljive dužine (fleksibilnije su ali se teže implementiraju)
- Dve šeme za realizaciju ove metode:
  1. Sinhrono ili blokirajuće slanje i primanje poruke
  2. Asihrono ili neblokirajuće slanje i primanje poruke
- Komunikacija može da bude:

1. Direktna
2. Indirektna putem poštanskih sandučića (*mailbox*) ili priljučka (*ports*)

2.6 Semafori

- Semafor predstavlja celobrojnu nenegativnu promenljivu koja označava neki resurs i omogućava komunikaciju između procesa koji ga koriste
- Dve nedeljive operacije karakterišu svaki semafor:
  1. **Signal**(s) – povećava vrednost semafora za 1, tj. s=s+1
  2. **Wait**(s) – smanjuje vrednost semafor za 1, tj. s=s-1
- Ove operacije ne mogu se podeliti na više ciklusa, tako da dva procesa ne mogu istovremeno izvršavati ove operacije nad istim semaforom
- Naredbe moraju da se implementiraju u jezgru OS jer imaju direktne veze sa dispečerom sistema jer uvode/izvode resurse iz stanja čekanja
- Vrednost semafora može se odrediti u svakom trenutku sa formulom

$$\mathbf{val(s)=c(s) + ns(s) - nw(s)}$$

c(s) – početna vrednost semafora, ns(s) – broj operacija **signal**
nw(w) – broj operacija **wait**

2.6 Named pipes

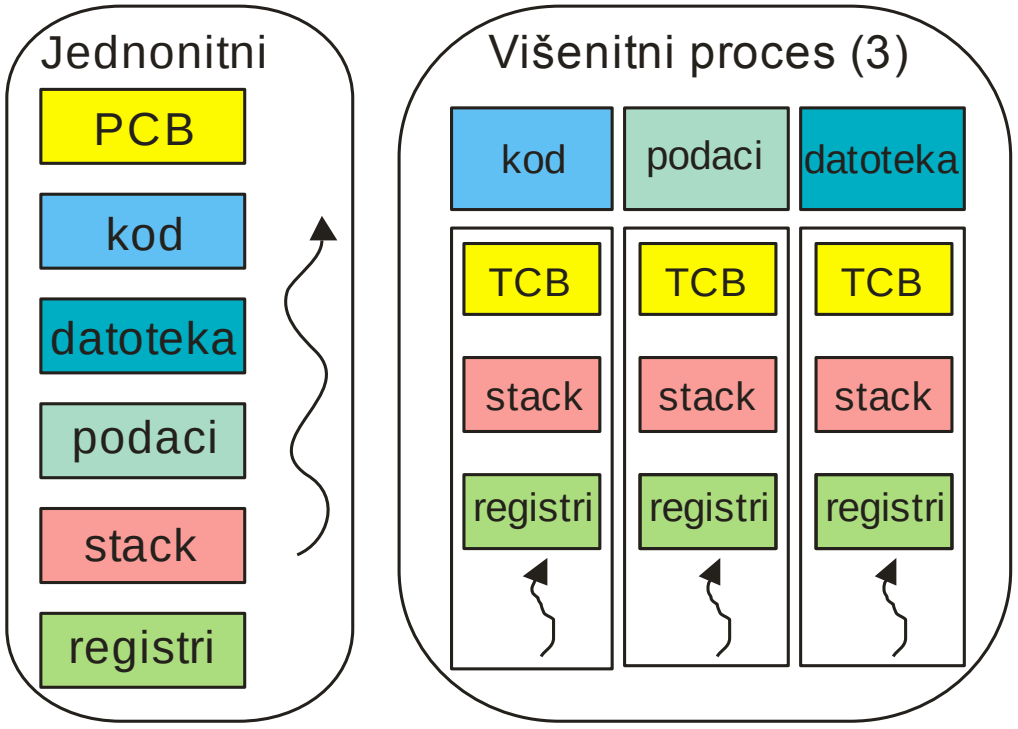
- Omogućavaju sekvencijalnu komunikaciju između 2 povezana procesa
- Predstavljaju standardno sredstvo za međuprocesnu komunikaciju (IPC) na Windows, kao i na Unix platformama.
- Svaki kreirani ***pipe*** ima unapred utvrđenu veličinu u bajtovima i njemu se pristupa kao običnom fajlu
- Predstavlja tehniku koja je serijskog tipa – podaci se čitaju istim redosledom kako su i upisani - "***First-In, First-Out*** "
- Jedan proces otvara FIFO za upis, drugi proces otvara FIFO za čitanje
- Za upis i čitanje koriste se funkcije niskog nivoa (write, read)
- Kada proces želi da upiše nešto u ***pipe*** to se odmah izvršava ukoliko ima prostora u njemu, suprotno proces se blokira.
- Slično je kod čitanja ***pipe*** ukoliko se čita manje bajtova nego što se nalazi u ***pipe***, suprotno, ako je broj bajtova veći proces se blokira
- Operativni sistem vodi računa da se samo jedna operacija može izvršiti u isto vreme (*mutual exclusion*)

2.7 Laki i teški procesi

- Laki procesi, niti (*threads*) predstavljaju najmanji programski blok koji planer (*scheduler*) može raspoređivati a koji može da obavi neki posao
- Prebacivanje konteksta je brže a komunikacija između niti je znatno jednostavnija jer one dele isti memorijski prostor
- Teški procesi mogu imati više niti koje se paralelno izvršavaju

Prednosti višenitnog koncepta

- Smanjenje vremena odziva
- Ekonomičnost – u prostoru, resursima i ušteda u vremenu
- Bolje iskorišćenje višeprocesorske arhitekture



Hvala na pažnji !!!

Pitanja

? ? ?