

Uvod u procese

- **Pojam** procesa
- **Raspoređivanje** procesa
- **Operacije** nad procesima
- **Saradnja** među procesima
- **Interprocesna komunikacija (IPC)**
- **Komunikacija u klijent-server sistemima**

Pojam procesa

- **Operativni sistem izvršava mnoštvo programa:**
 - ☞ **Grupni sistemi – jobs - poslovi**
 - ☞ **Time-shared sistemi – korisnički programi ili zadaci (tasks)**
 - ☞ **U literaturi se termini job, task i process koriste identično.**
- **1. def: Proces je program u stanju izvršavanja (running)**
 - ☞ (kontekst procesa: proces se izvršava u svom kontekstu).
- **Proces sadrži:**
 - ☞ **CPU registre** {programski brojač i ostali CPU registri}
 - ☞ **Memorijsku sekciju**
 - 📄 tekst
 - 📄 stek (stack)
 - 📄 sekcija podataka
 - ☞ **I/O resursi:** {datoteke, I/O uređaji}
 - ☞ **OS struktura**
- **2. def: Proces je unija 4 konteksta:**
 - ☞ **Register context**
 - ☞ **Memory context**
 - ☞ **IO context**
 - ☞ **OS kontekst**

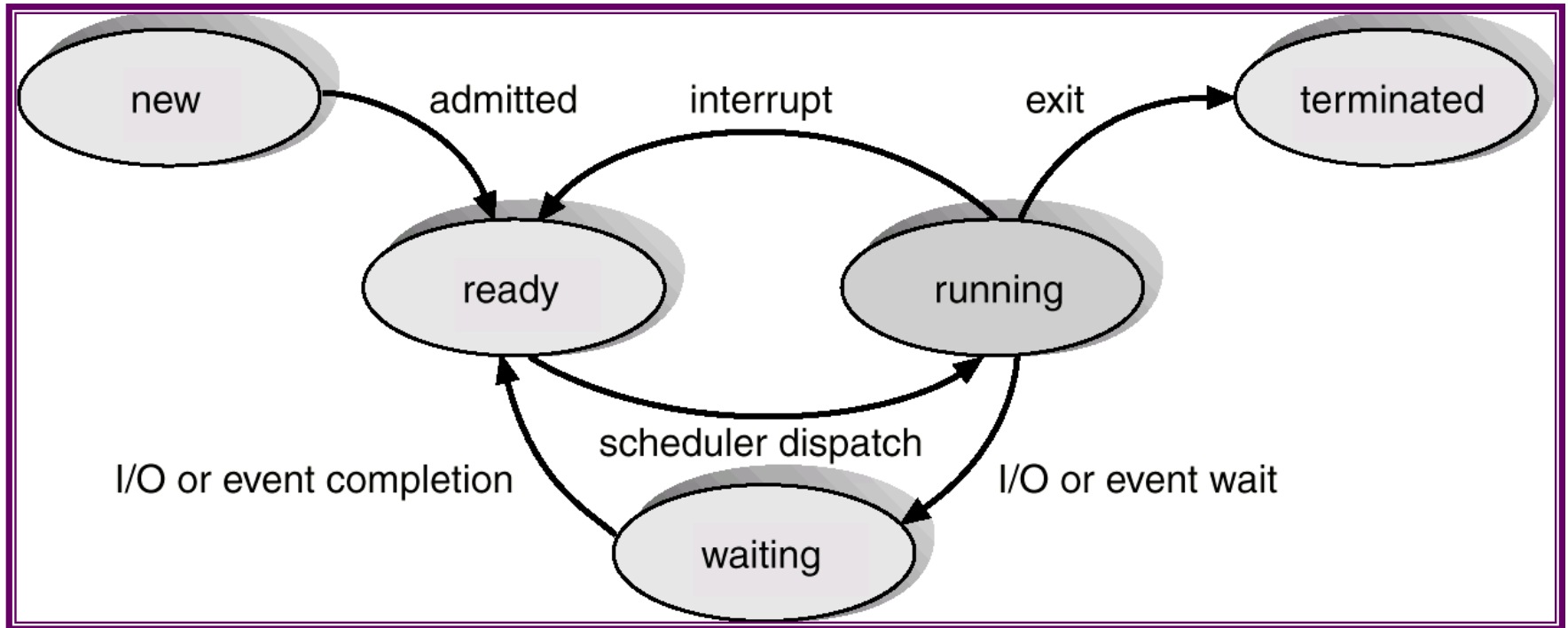
Stanja procesa

■ U toku izvršavanja proces menja stanja:

- ☞ **New (novi):** Proces je upravo kreiran.
- ☞ **Ready (spreman):** Proces je spreman za rad ali čeka da mu se dodeli CPU.
- ☞ **Running (izvršava se):** Procesove Instrukcije se upravo izvršavaju (odnosi se na CPU)
- ☞ **Waiting (čekanje):** Proces čeka da se neki događaj izvrši.
- ☞ **Terminated (završio):** Proces je završio izvršavanje.

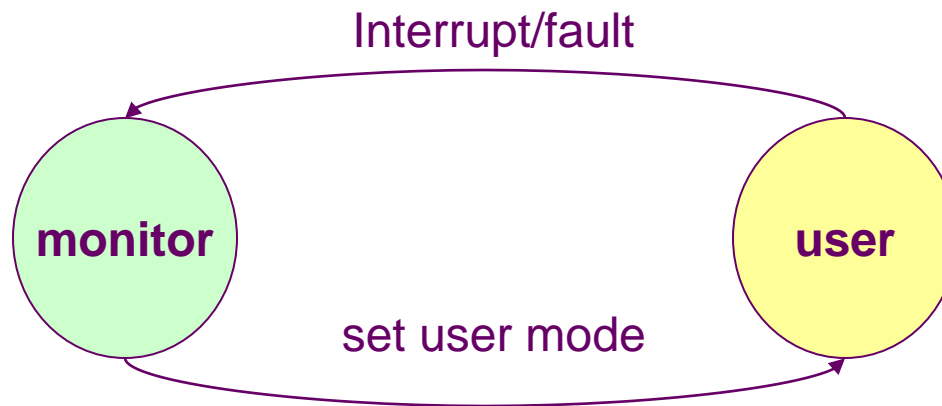
Dijagram stanja procesa

- **3. def: Proces je objekat koji funkcioniše po ovom dijagramu**



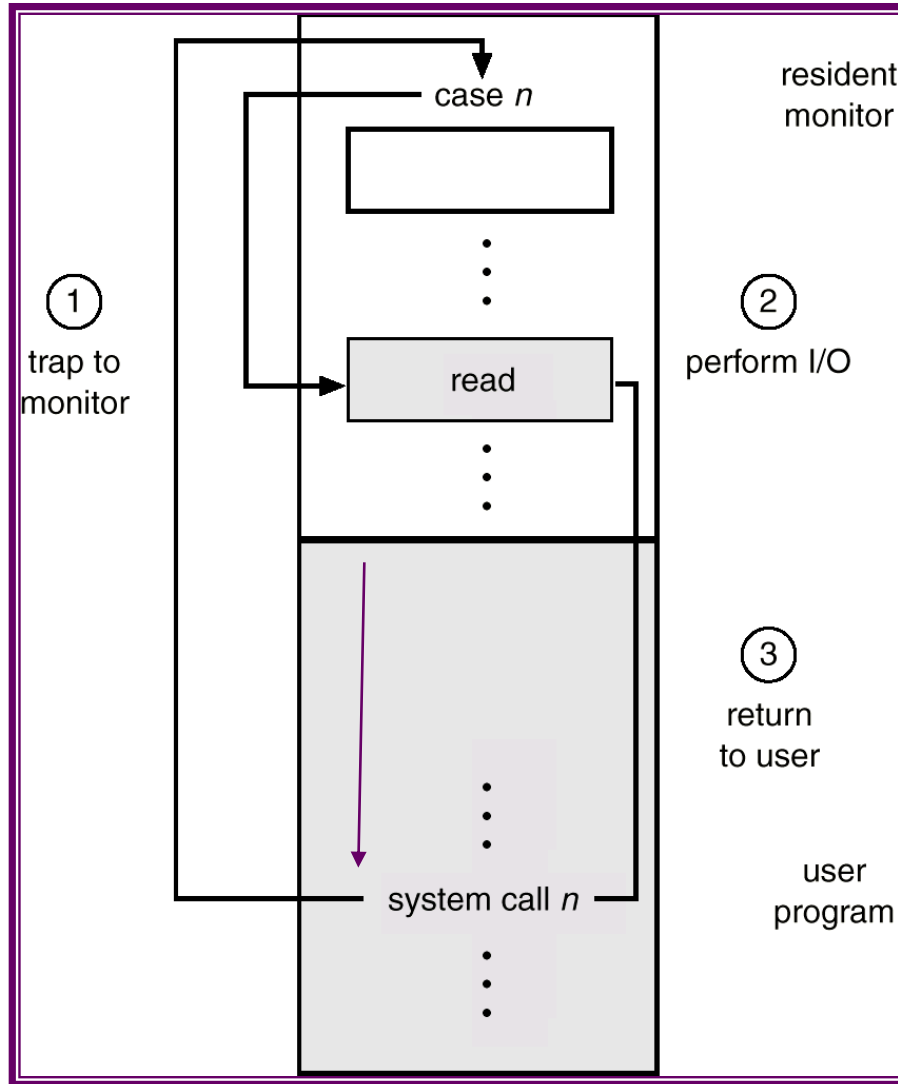
Running stanje i Dual-Mode Operation

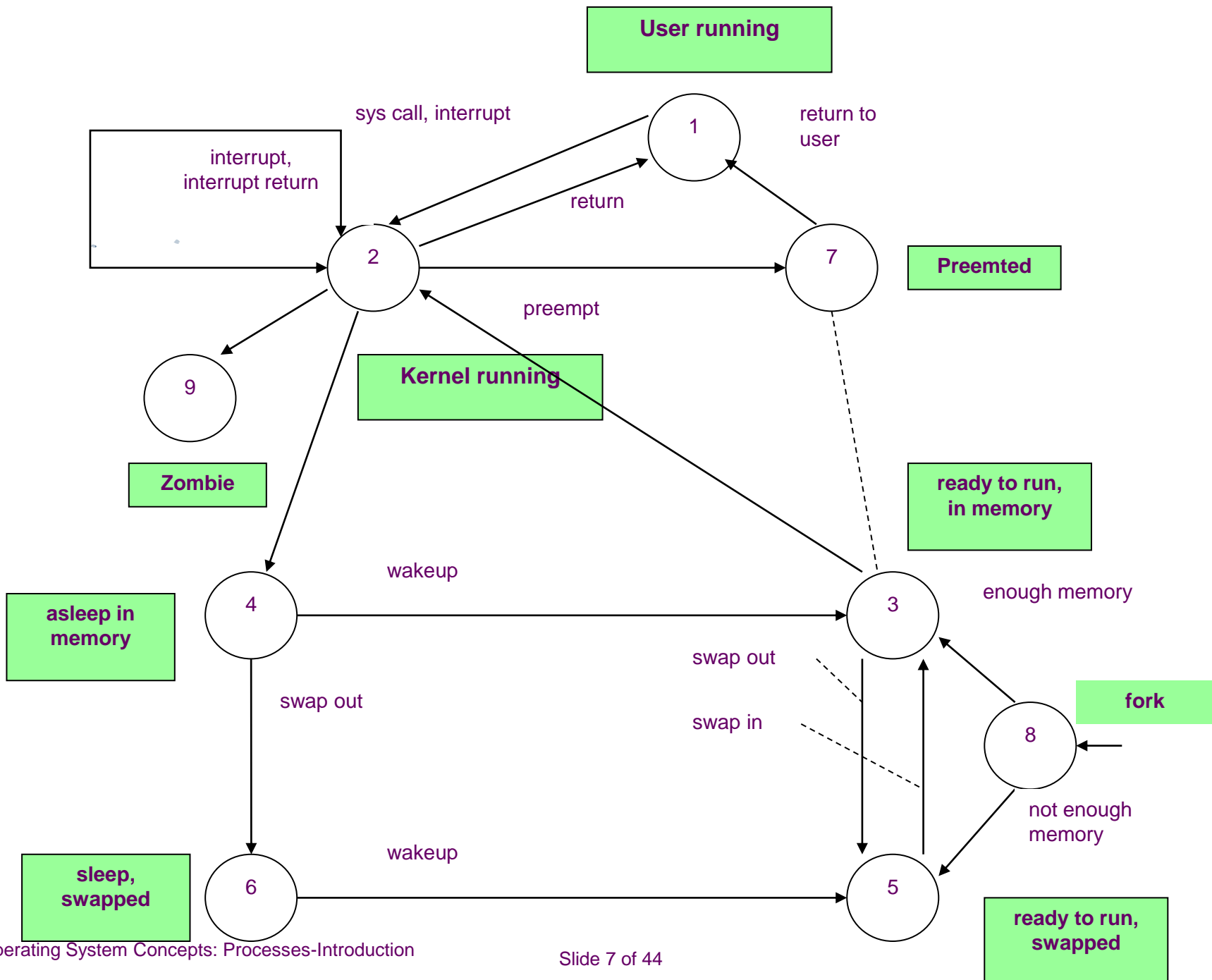
- **Mode Bit** se ubacuje u računarski hardver
- da ukazuje na trenutni mode:
 - ☞ **monitor mode (0)**
 - ☞ ili
 - ☞ **user mod (1)**
- Kada se **javi greška ili prekid hardver** prebacije u monitor mod.



Privilegovane instrukcije mogu dati rezultat samo u monitor modu.

Sistemiški pozivi



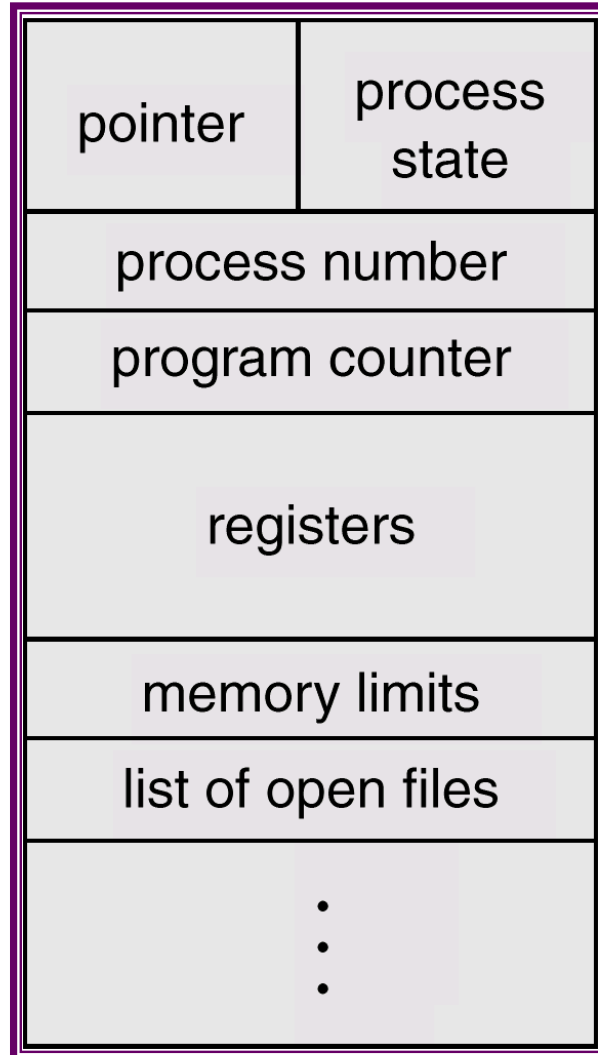


PCB (Process Control Block)

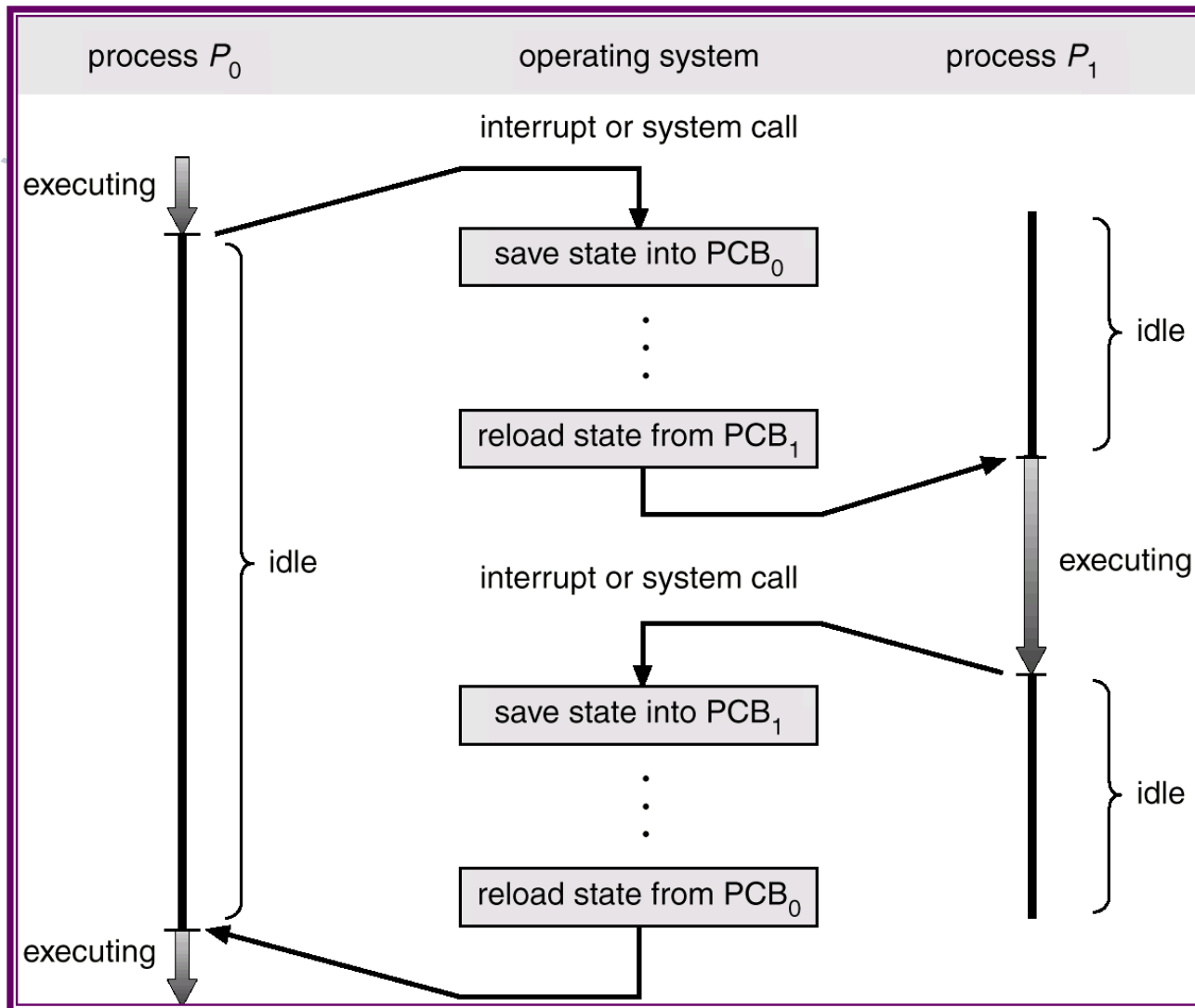
Informacije vezane za svaki proces:

- Stanje procesa
- Programski brojač
- Sadržaj CPU registara
- Informacije o memoriji procesa
- Lista otvorenih datoteka
- Statističke informacije
- Status I/O resursa

PCB (Process Control Block)



CPU prebacivanje od procesa do procesa

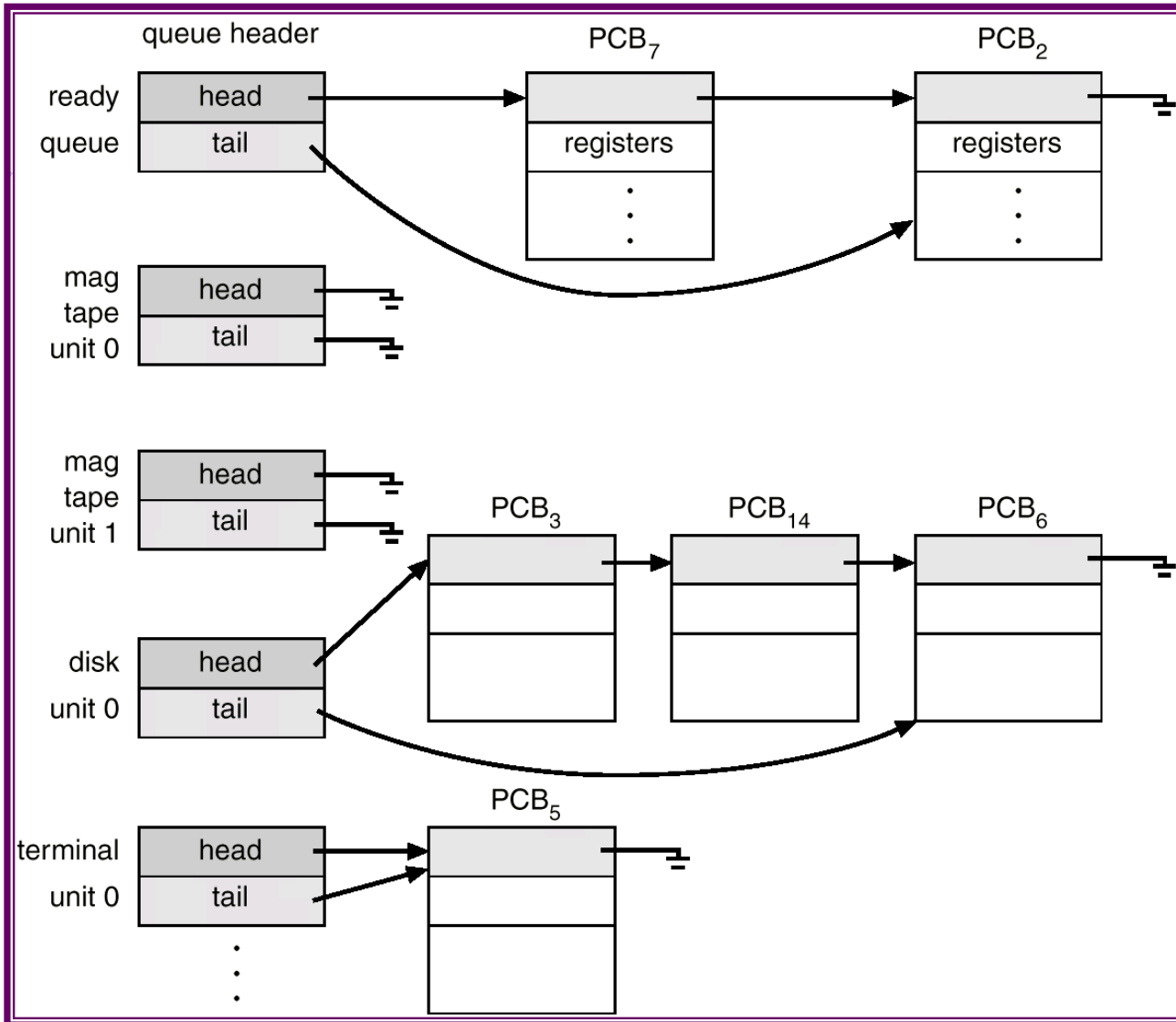


**Context
switch**

Queue: red čekanja

- Red čekanja za poslove (*Job queue*):
 - ☞ obuhvata sve procese u sistemu
- Red čekanja za spremne procese (*Ready queue*):
 - ☞ obuhvata sve procese
 - 📄 smeštene u glavnoj memoriji
 - 📄 spremne i čekaju na izvršenje
- Redovi čekanja za I/O uređaje (*Device queues*):
 - ☞ obuhvata procese koji
 - 📄 čekaju na dodelu I/O uređaja
- Proces **se kreće** između različitih **redova čekanja**

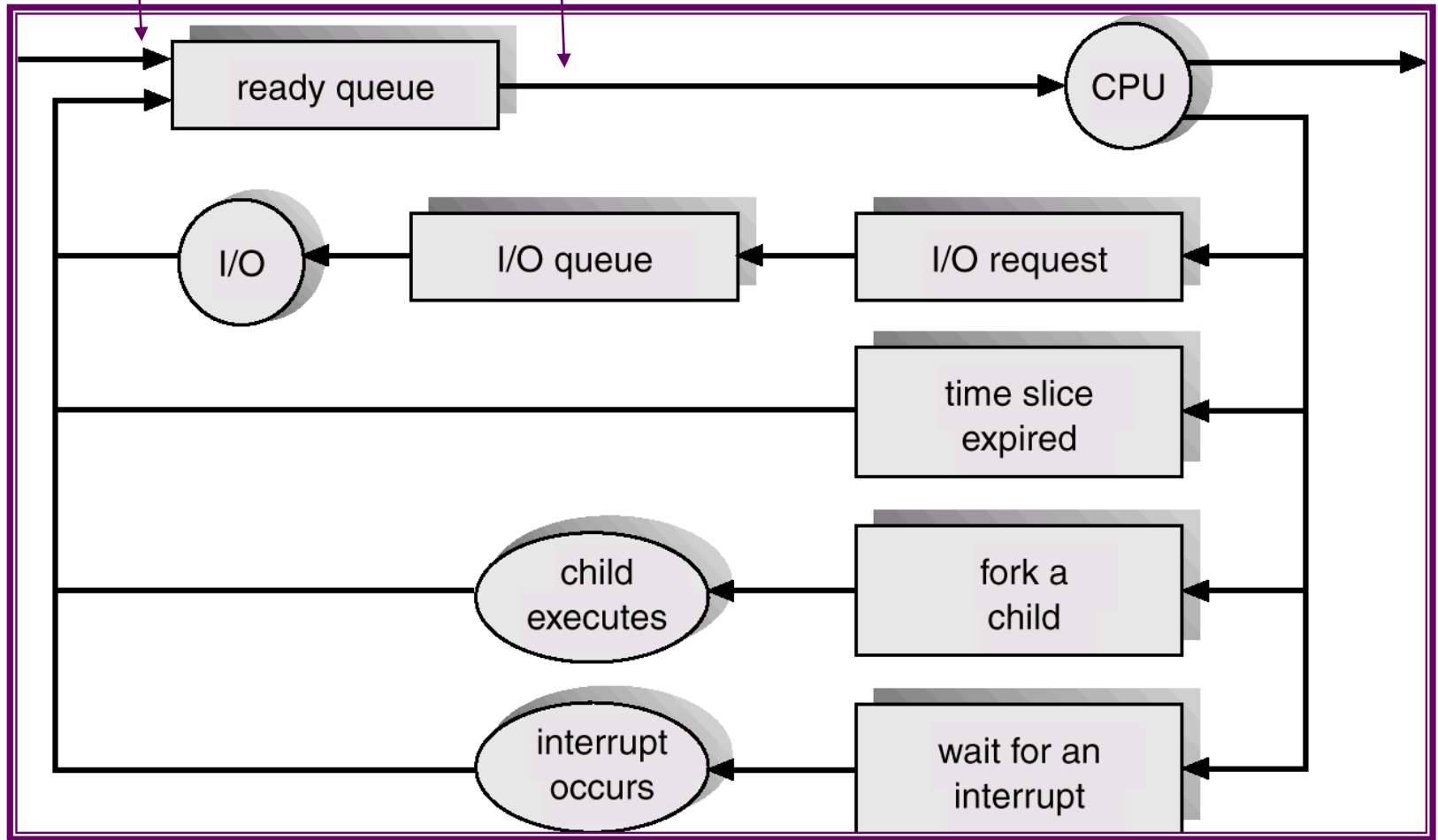
Ready Queue and Various I/O Device Queues



Prikaz raspoređivanja procesa

long-term (dugoročni) raspoređivač

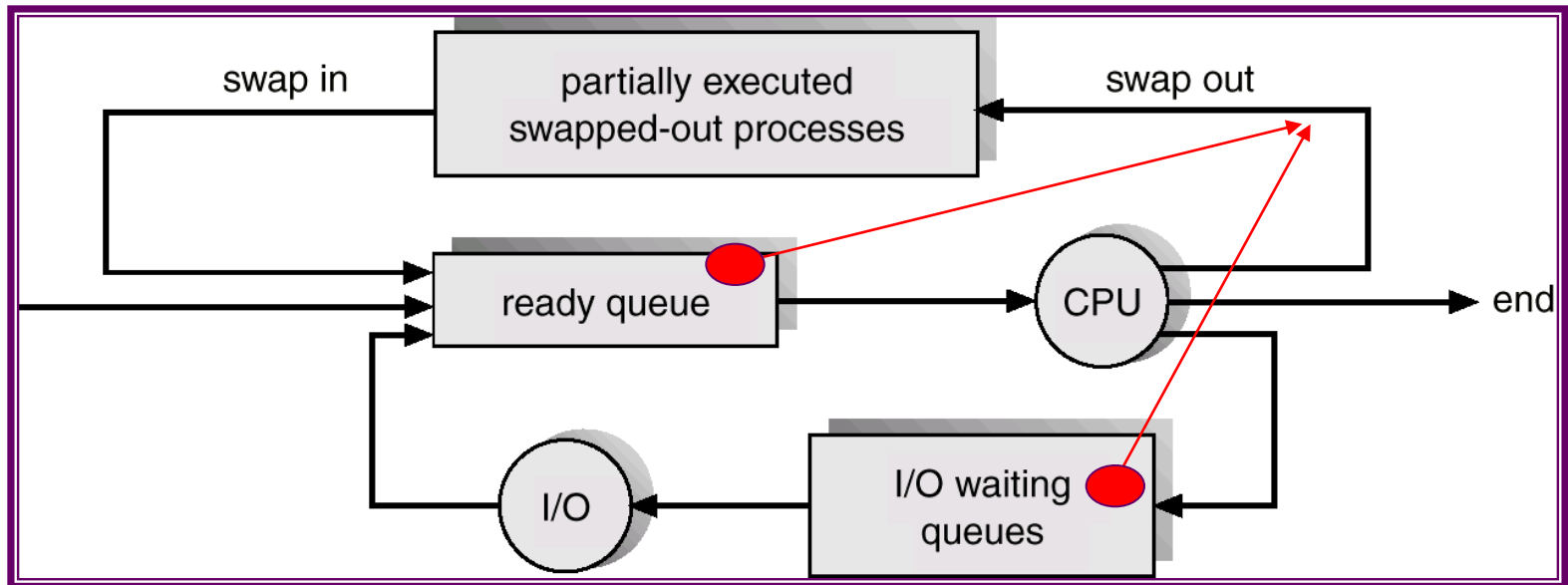
short-term (kratkoročni) raspoređivač



Raspoređivači (*Schedulers*)

- **Long-term (dugi, dugoročni) raspoređivač**
- (ili raspoređivač poslova):
 - ☞ selektuje procese
 - ☞ i dovodi ih
 - ☞ u red čekanja za spremne poslove (*Ready queue*)
- **Short-term (kratki, kratkoročni) raspoređivač**
- (ili **CPU raspoređivač**):
 - ☞ seletuje proces
 - ☞ koji će se sledeći izvršiti
 - ☞ i **dodeljuje mu CPU**

Blok dijagram za medium-term (srednji) raspoređivač



Raspoređivači (*Schedulers*)

- **kratkoročni raspoređivač** se poziva **veoma često**
 - ☞ (milisekunde) ⇒ (mora biti brz).
- **dugoročni raspoređivač** se ne poziva često
 - ☞ (sekunde, minuti) ⇒ (može biti spor).
- dugoročni raspoređivač reguliše stepen **multiprogramiranja**
- **Procese možemo podeliti:**
- **I/O intenzivne procese (*I/O bound process*):**
 - 📄 najveći deo njihovog vremena izvršavanja otpada na I/O cikluse,
 - 📄 relativno mala upotreba CPU
- **CPU intenzivne procese (*CPU bound process*):**
 - 📄 najveći deo svog vremena korsi na CPU izračunavanja,
 - 📄 veoma velika upotreba CPU

Prebacivanje konteksta (*Context Switch*)

- Kada se CPU prebacuje na drugi proces
 - ☞ sistem mora sačuvati stanje starog procesa
 - ☞ i
 - ☞ učitati sačuvano stanje za novi proces
- Prebacivanje konteksta (*Context switch*) =
 - ☞ pamti stanje starog procesa
 - ☞ +
 - ☞ učitava sačuvano stanje novog procesa
- Prebacivanje konteksta je gubitak vremena (*overhead*);
 - ☞ sistem ne može raditi koristan posao dok prebacuje
- Vreme zavisi od hardverskih performansi

Kreiranje procesa (*Process Creation*)

■ **Kreiranje:**

- ➡ **Roditelj** proces kreira **dete** proces
- ➡ koji, dalje kreira druge procese
- ➡ i formira stablo procesa

■ **Deljenje resursa (*Resource sharing*)**

- ➡ Roditelj i dete procesi **dele sve resurse**
- ➡ Dete proces **deli podskup** od roditeljskih resursa
- ➡ Roditelj i dete proces **ne dele resurse**

■ **Izvršavanje (*Execution*)**

- ➡ Roditelj i dete procesi se **izvršavaju konkurentno**
- ➡ Roditelj proces **čeka da dete proces završi** aktivnosti

Kreiranje procesa

■ Memorijski adresni prostor

- ➡ Dete proces kopira adresni prostor roditelja
- ➡ Dete proces dobija program koji puni njegov adresni prostor.

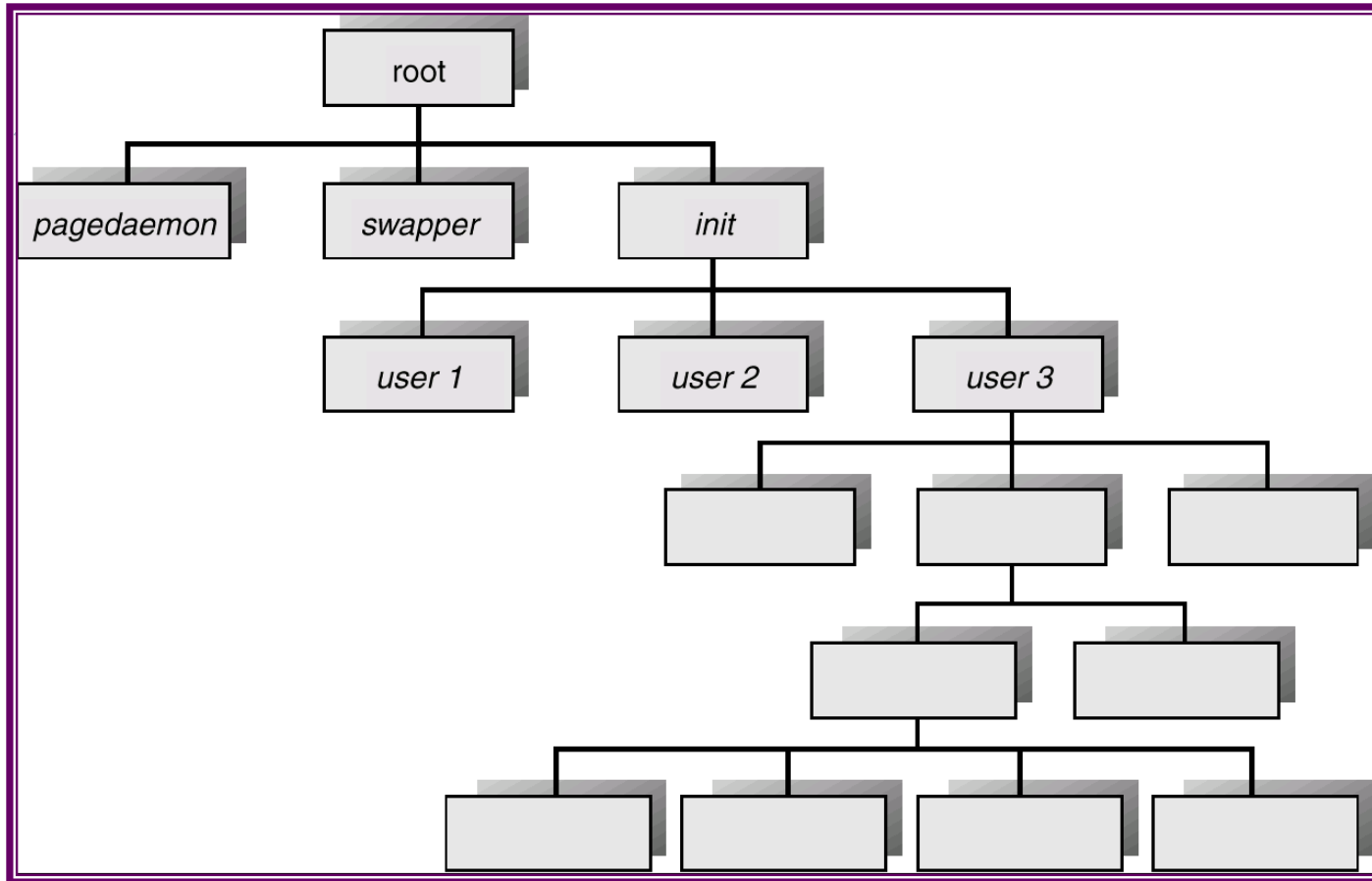
■ UNIX primeri

- ➡ **fork** sistemski poziv **kreira novi proces**
- ➡ **exec** sistemski poziv **se koristi posle fork**
- ➡ koji **puni adresni prostor novim programom**

Kreiranje procesa (fork)

- `/* fork drugi proces*/`
- `pid = fork ();`
- `if (pid == 0)`
- `{` `/*dete proces*/`
- `execvp(“/bin/ls”, “ls”, NULL);`
- `}`
- `else` `/*roditelj proces*/`
- `/*roditelj će čekati da dete završi*/`
- `wait(NULL);`
- `printf(“Dete je završilo”);`
- `exit (0);`
- `}`
- `}`

Stablo procesa na UNIX sistemu



Završetak procesa (*Process Termination*)

- **(exit) = normal termination**
- Proces izvršava poslednju aktivnost i
- **traži** od operativnog sistema da ga obriše (**exit**)
 - ☞ Dete proces vraća izlazne podatke roditelju (preko **wait**)
 - ☞ **Resursi koji su pripadali procesu detetu se oslobađaju**
- **(abort) = abnormal termination**
- **Roditelj proces** može **prekinuti izvršavanje dece procesa**, ako:
 - ☞ Dete proces je **prekoračilo dodeljene resurse**
 - ☞ **Aktivnost** koju obavlja dete proces **nije više potrebna**
- **Ako proces roditelj završi aktivnosti:**
 - 📄 neki operativni sistemi ne dozvoljavaju da dete proces nastavi sa izvršavanjem **ako je roditelj završio**
 - 📄 **Nasilno prekidanje (*Cascading termination*)**

Saradnja među procesima

- **Nezavisni** procesi:
 - ☞ nemogu uticati ili ne trpe uticaj
 - ☞ od strane drugih procesa.
 - ☞ (ne dele podatke, ili resurse)

- **Kooperativni** procesi:
 - ☞ mogu uticati ili da se na njih utiče
 - ☞ od strane drugih procesa.
 - ☞ (dele podatke, ili resurse)

- **Prednosti** saradnje među procesima:
 - ☞ Deljenje informacija
 - ☞ Ubrzavanje rada
 - ☞ Modularnost
 - ☞ Pogodnost

Problem Proizvođač-Potrošač

- Paradigma za kooperativne procese,
- **proizvođač** proces proizvodi informacije
- **potrošač** koji troši informacije
- **Bafer (buffer)**
 - ☞ **bafer beskonačnog kapaciteta** (*unbounded-buffer*) nema ograničenje u veličini bafera.
 - ☞ **ograničeni bafer** (*bounded-buffer*) ima ograničenu veličinu bafera
- **Sinhronizacija procesa:**
 - ☞ turn
 - ☞ flags
 - ☞ semaphores
 - ☞ monitor
 - ☞ IPC (sistem poruka)

Bounded-Buffer – Rešenje za deljenje memorije

- Deljeni podaci

```
#define BUFFER_SIZE N
typedef struct
{
    ...
} item;
item buffer[BUFFER_SIZE];
int in = 0;           #first free buffer position
int out = 0;         #first full buffer position
```

- Rešenje je tačno, ali može koristiti samo `BUFFER_SIZE-1` elemenata

Bounded-Buffer – Proces proizvođač

```
item nextProduced;  
int in = 0;           #in ->first free buffer position  
int out = 0;        #out->first full buffer position  
while (1)  
  {  
    while (((in + 1) % BUFFER_SIZE) == out)  
  
      ; /* buffer is full, do nothing */  
  
    buffer[in] = nextProduced;  
  
    in = (in + 1) % BUFFER_SIZE;  
  
  }
```

Bounded-Buffer – Proces potrošač

item nextConsumed;

int in = 0; #first free buffer position

int out = 0; #first full buffer position

while (1)

{

while (**in == out**) #empty
 ; /* do nothing */

nextConsumed = buffer[out];

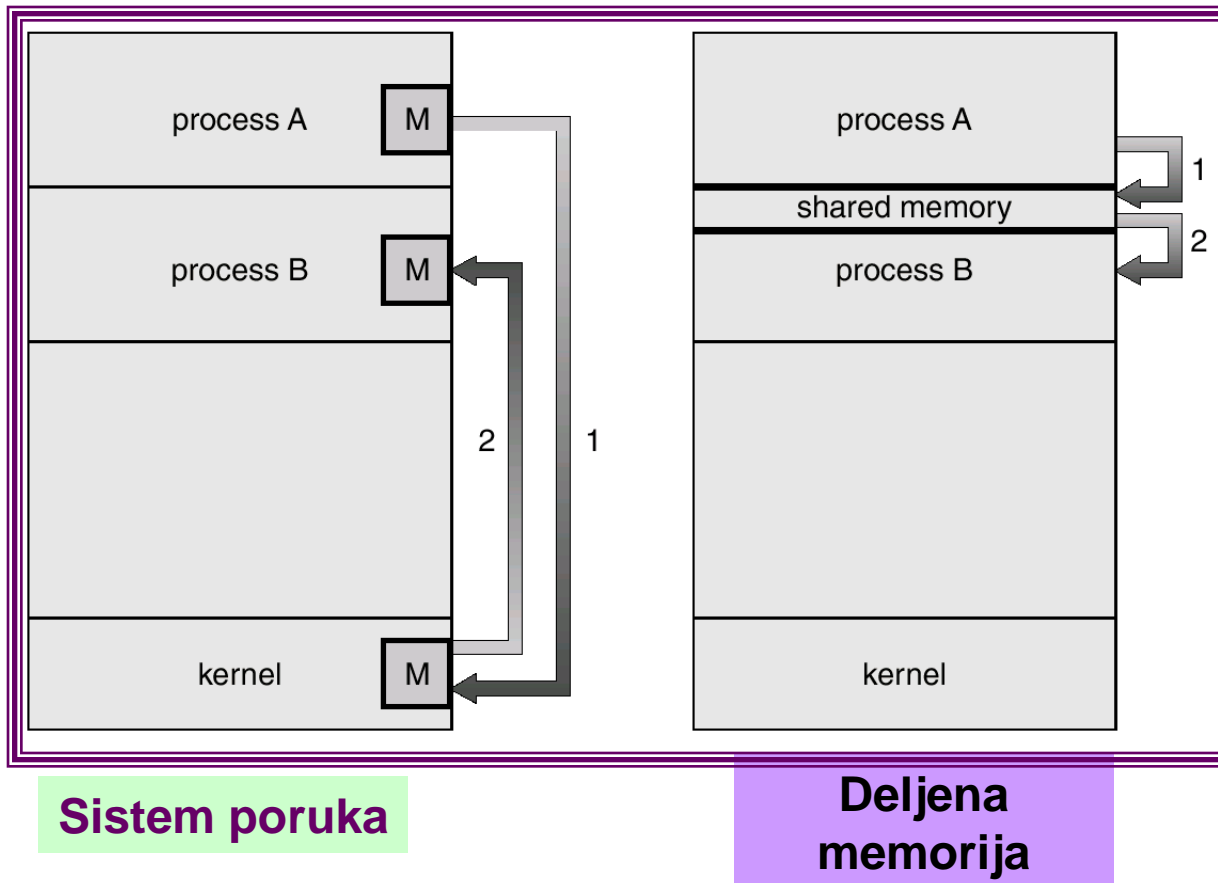
out = (out + 1) % BUFFER_SIZE;
}

Interprocesna komunikacija (IPC)

- **Mehanizam** za **procese** koji omogućava da:
 - ☞ **komuniciraju**
 - ☞ **sinhronizuju** njihove akcije
- **Tri metode:**
 - ☞ **signali (UNIX)**
 - ☞ **deljena memorija**
 - ☞ **sistem poruka**

Komunikacioni model

- Komunikacija se može realizovati:
 - ☞ sistemom poruka (*message passing*)
 - ☞ ili
 - ☞ deljenjem memorije



(IPC): poruke

- **Sistem poruka:**
 - ☞ procesi razmenjuju poruke
 - ☞ procesi komuniciraju sa ostalim procesima
 - ☞ bez deljenja memorije
- **IPC sistem** obezbeđuje **dve operacije:**
 - ☞ **slanje poruke** (*send message*) – poruke fiksne ili promenljive veličine
 - ☞ **prijem poruke** (*receive message*)
- **Ako procesi P i Q žele da komuniciraju**, oni trebaju da:
 - ☞ uspostave **komunikacioni link** između njih
 - ☞ poruke razmenjuju preko **send/receive** operacija
- **Implementacija komunikacionog linka:**
 - ☞ **fizička** (npr., deljena memorija, hardverska magistrala)
 - ☞ **logička** (npr., logičke osobine)

Pitanja vezana za implementaciju

- **Kako** su linkovi uspostavljeni?
- Da li se linku **mogu** pridružiti **više od dva** procesa?
- **Koliko** linkova **može** postojati
 ☞ između **svakog para** komunikacionih procesa?
- Šta je **kapacitet** linka?
- Da li je **veličina poruke** koju link **može** preneti: **fiksna** ili **promenljiva**?
- Da li je link **unidirectional** ili **bi-directional**?

Direktna komunikacija

- **Procesi** trebaju da imaju **ime** i to svaki različito:

- ☞ **send (P, message)** – poslaće poruku procesu P

- ☞ **receive (Q, message)** – primiće poruku od procesa Q

- **Osobine direktne komunikacije:**

- ☞ **Veze** se uspostavljaju **automatski**

- ☞ **Veza** je udružena sa **tačno jednim parom** komunikacionih procesa

- ☞ **Između svakog para** postoji **tačno jedna veza**

- ☞ Veza **može biti unidirectional**, ali se **koristi** i **bi-directional**

Indirektna komunikacija

- **Poruke se šalju i primaju preko poštanskih sandučića (*mailboxes*)**
 - ☞ (ili preko portova)
 - ☞ Svako sanduče ima **jedinstven id**
 - ☞ Procesi **moгу komunicirati** samo **ako dele sanduče**
- **Osobine indirektnе komunikacije:**
 - ☞ **Veza se uspostavlja samo** ako procesi **dele zajedničko sanduče**
 - ☞ **Vezi se mogu pridružiti više procesa**
 - ☞ **Između svakog para može postojati više različitih veza**
 - ☞ Veza može biti **unidirectional ili bi-directional**.

Indirektna komunikacija

■ Operacije

- ✎ kreiranje novog sandučeta
- ✎ slanje i primanje poruka kroz sanduče
- ✎ brisanje sandučeta

■ Naredbe za slanje:

- ✎ **send(A, message)** – poslaće poruku u sanduče A
- ✎ **receive(A, message)** – primiće poruku iz sanduče A

Indirektna komunikacija

■ Deljenje sandučeta

- ☞ P_1 , P_2 , i P_3 dele sandučče A.
- ☞ P_1 , šalje; P_2 i P_3 primaju.
- ☞ **Ko je dobio poruku?**

■ Rešenje:

- ☞ 1. Dozvoliti da se link uspostavlja **između najviše dva procesa**
- ☞ 2. Dozvoliti da **samo jedan proces može obaviti prijem u jednom trenutku**
- ☞ 3. Dozvoliti da sistem **proizvoljno bira primaoca**
 - 📄 Pošiljaoc je obavestio ko je bio primaoc.

Sinhronizacija

- **Sistem poruka** može biti **blokirajući** ili **neblokirajući**
- **blokirajući** funkcioniše **sinhrono**
- **neblokirajući** funkcioniše **asinhrono**
- **slanje i primanje** poruka može biti:
 - ☞ **blokirajuće**
 - ☞ ili
 - ☞ **neblokirajuće**

Baferisanje

- **Message Buffer** = Red u kom poruka čeka ;
- **realizacija na jedan od 3 načina:**
 1. **Nulti kapacitet (*Zero capacity*):** 0 poruka se čeka
Pošiljalac mora čekati primaoca (rendezvous).
 2. **Ograničen kapacitet (*Bounded capacity*):** ograničena dužina na n poruka
Pošiljalac mora da sačeka ako je bafer pun
 3. **Neograničen kapacitet (*Unbounded capacity*)** – neograničena dužina
Pošiljalac nikad ne čeka
- **Potvrda:** (P proces šalje poruku procesu Q)

<ul style="list-style-type: none">☞ Proces P☞ send(Q, message)☞ receive(Q, message)	<ul style="list-style-type: none">Proces Qreceive(P, messages)send(P, “acknowledgment”)
--	---

Osobine poruka

- Postoje **tri vrste poruka** koje razmenjuju procesi:
 1. **Fiksne veličine**
 2. **Promenljive veličine**
 3. **Tipske poruke**
- Druge osobine (**waiting**):
 - ☞ Nikad ne čeka
 - ☞ Čeka za odgovor (moguć deadlock)
- Izgubljene poruke (**lost messages**)
 - ☞ otkrivanje
 - ☞ ponovno prenošenje

Komunikacija u klijent-server sistemima

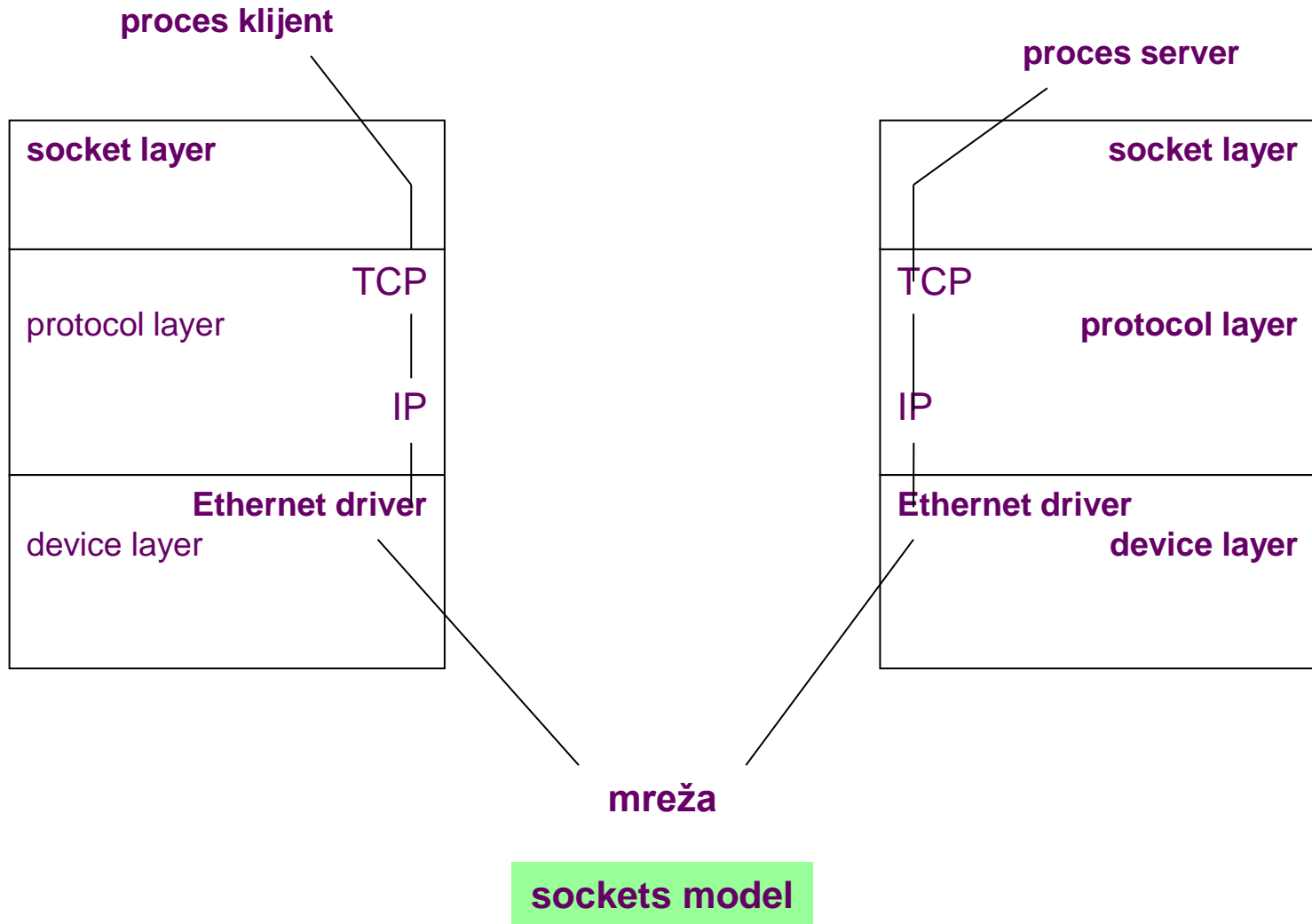
- **Sockets**

- Poziv udaljene procedure (*Remote Procedure Calls -RPC*)

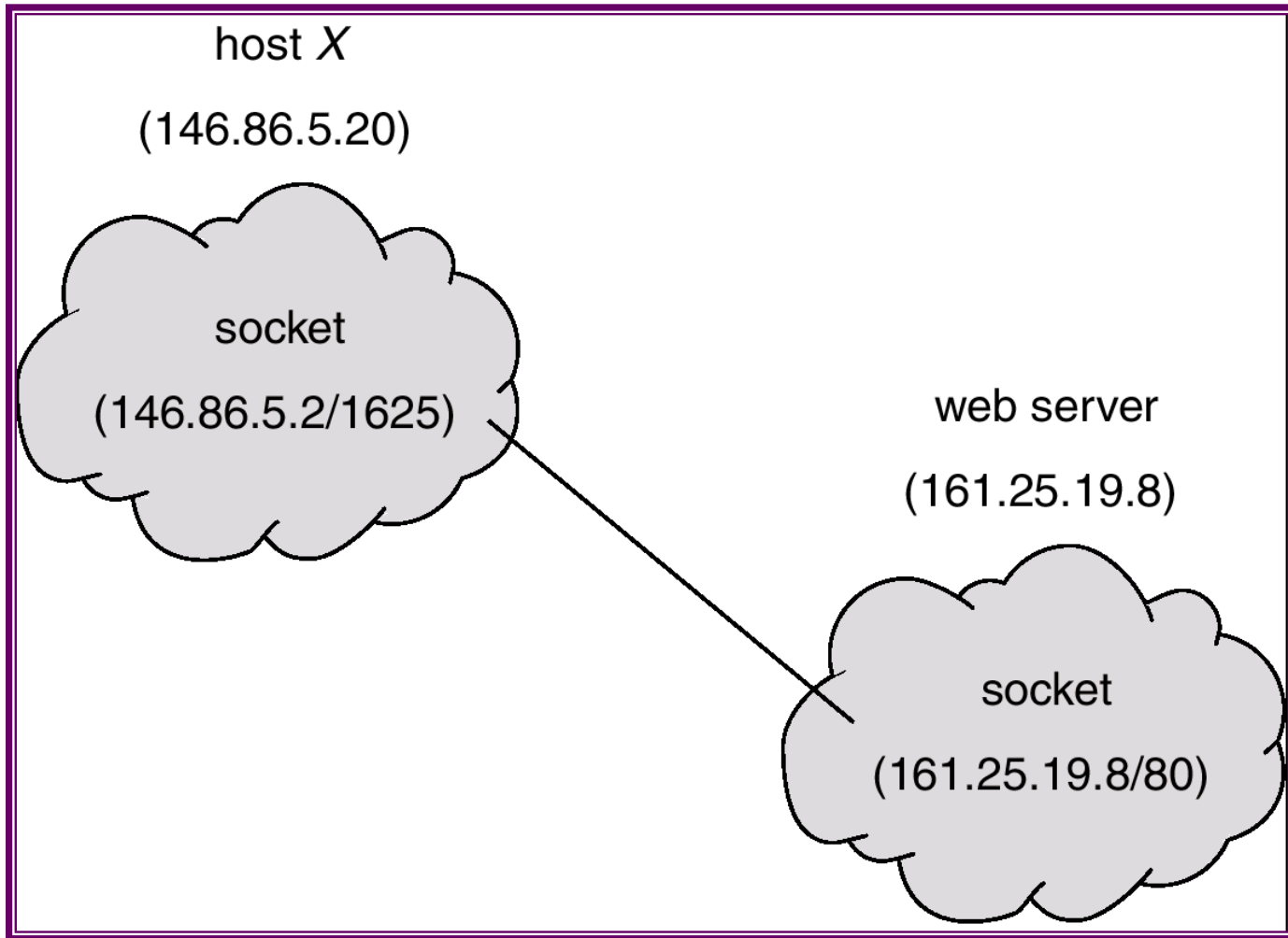
Sockets

- **BSD način** za IPC između različitih mašina
- **Socket** se definiše kao **krajnja tačka komunikacije**
- **Socket** = Povezuje **IP adresu** i **port**
- **Socket 161.25.19.8:1625** se obraća
 - ☞ portu **1625**
 - ☞ na hostu **161.25.19.8**
- **Komunikacija se odvija između para socketa**

Socket model



Socket komunikacija



Poziv udaljene procedure (RPC)

- **Poziv udaljene procedure (RPC)** je abstrakcija poziva procedure
- između procesa na mrežnom sistemu

- **Stubovi:** klijentska strana je **proxy** za aktuelnu proceduru na serveru

- **Stub klijentske strane:**
 - ☞ pronalazi server
 - ☞ šalje parametre

- **Stub serverske strane:**
 - ☞ prima ovu poruku
 - ☞ raspakuje parametre
 - ☞ izvršava proceduru na serveru.

Izvršavanje RPC

