Operativni sistemi

- Upravljanje procesima-

Prof. dr Dragan Stojanović

Katedra za računarstvo
Univerzitet u Nišu, Elektronski fakultet

Korišćen materijal

- *Operating Systems: Internals and Design Principles*, 5th edition, W. Stallings, Pearson Education Inc., 2005) – 6th edition 2008, 7th edition – 2012
  - http://williamstallings.com/OS/OS6e.html
  - http://williamstallings.com/OperatingSystems/
- Poglavlje 3: Opis procesa i upravljanje

Sadržaj

- Predstavljanje procesa u OS.
- Stanja procesa koja karakterišu ponašanje procesa.
- Strukture podataka koje se koriste za upravljanje procesima.
- Načini na koje OS koristi ove strukture podataka da upravlja izvršenjem procesa.
- Upravljanje procesima u UNIX SVR4.

Zahtevi operativnog sistema

- Osnovni zadatak: Upravljanje procesima
- Operativni sistem mora da obezbedi:
  - Mešanje (preplitanje) izvršenja više procesa
  - Dodelu (alokaciju) resursa procesima i zaštitu resursa svakog procesa od strane drugih procesa
  - Mogućnost deljenja i razmene podataka između procesa
  - Mogućnost sinhronizacije između procesa
- Računarska platforma se sastoji od kolekcije hardverskih resursa; OS obezbeđuje reprezentaciju resursa i interfejs za aplikacije u cilju korišćenja tih resursa

Šta je proces ?

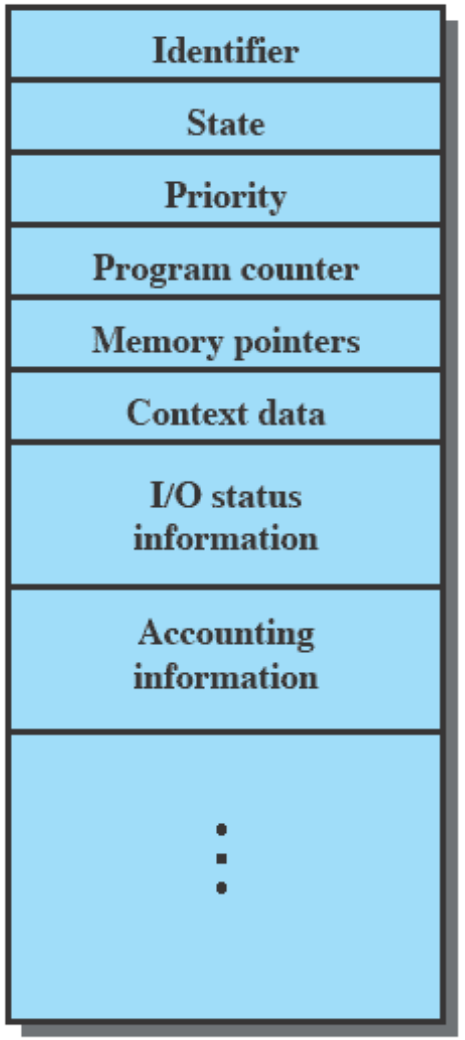
- Proces je osnovni koncept operativnog sistema i predstavlja program u izvršavanju na CPU
  - Proces se ponekad naziva zadatak (task)
  - Svi multiprogramski OS su izgrađeni oko koncepta procesa
- Gledano sa strane OS-a, proces je osnovna jedinica izvršavanja i najmanji entitet koji se može planirati, dodeliti i izvršavati na procesoru
- Jedinica aktivnosti koju karakteriše izvršenje sekvence instrukcija, trenutno stanje i pridružen skup sistemskih resursa

Elementi procesa

- Proces se sastoji od:
  - Programskog kôda
  - Skupa podataka nad kojima se izvršavaju instrukcije programa
  - Atributa koji opisuju stanje procesa dok se izvršava:
    - Identifikator
    - Stanje
    - Prioritet
    - Programski brojač (*Program counter*)
    - Pokazivači na memoriju koju zauzima proces
    - Kontekstni podaci
    - Informacije o U/I statusu
    - Informacije o obračunu korišćenja resursa

Upravljački blok procesa

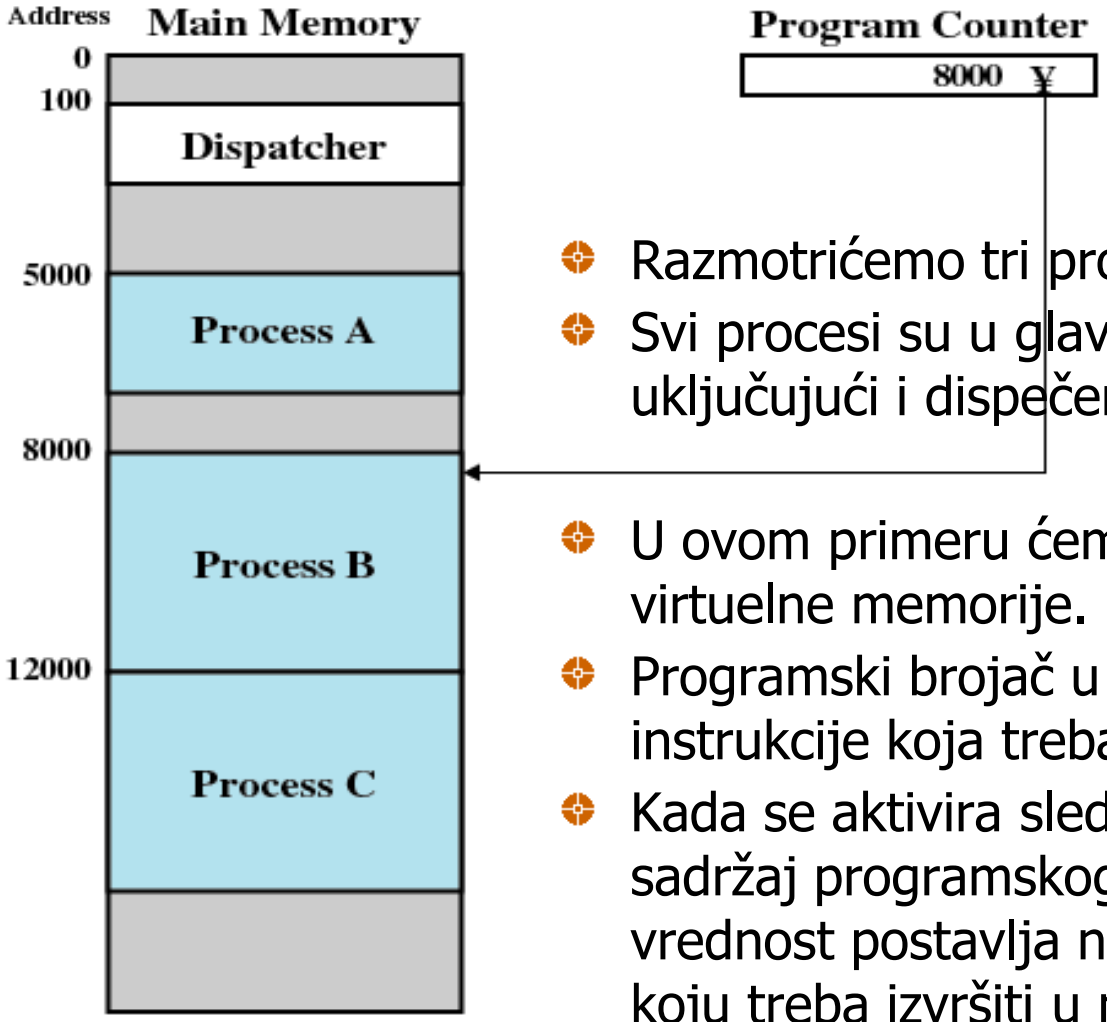
- *Process Control Block* - PCB
- Sadrži atribute procesa
- Kreiran i upravljan od strane OS
- Obezbeđuje podršku za višestruke procese
- PCB-i svih procesa čine Tabelu procesa



Trag (putanja, *trace*) procesa

- Ponašanje individualnog procesa je prikazano sekvencom instrukcija koje se izvršavaju
- Ova lista instrukcija se naziva **trag** (*trace*)
- ***Dispečer*** (*dispatcher*) je program koji prebacuje (*switch*) procesor od jednog procesa drugom, i vrši zamenu aktivnog procesa

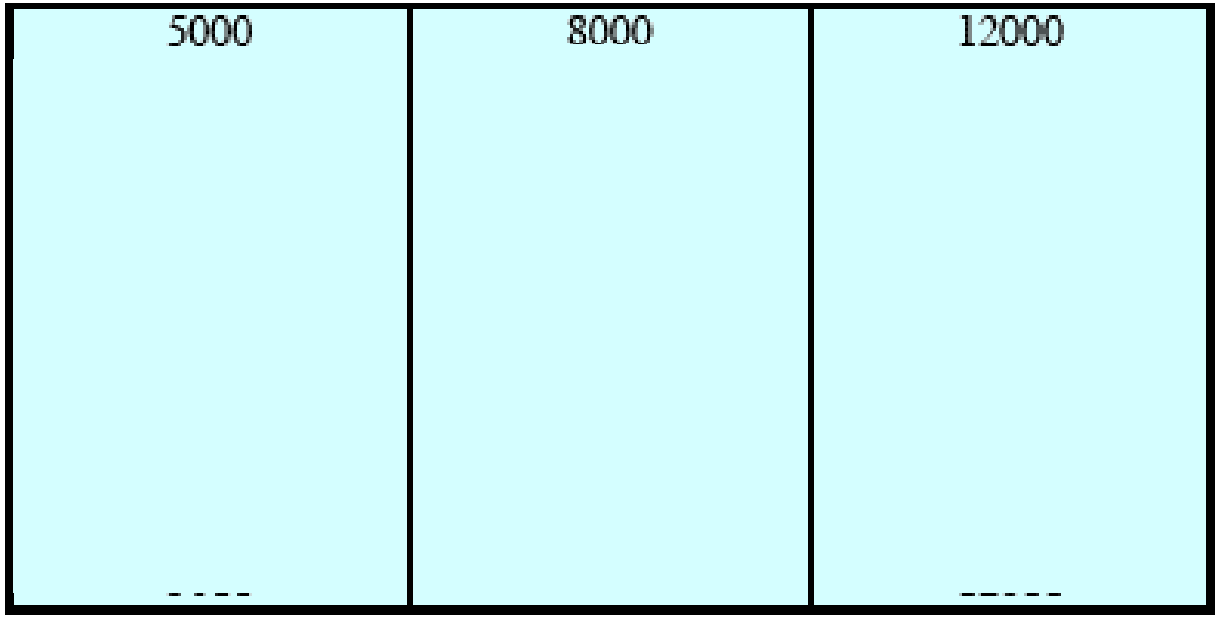
Izvršenje procesa



- Razmotrićemo tri procesa koji se izvršavaju
- Svi procesi su u glavnoj memoriji, uključujući i dispečer
- U ovom primeru ćemo ignorisati postojanje virtuelne memorije.
- Programski brojač u CPU sadrži adresu instrukcije koja treba da bude izvršena
- Kada se aktivira sledeći proces menja se sadržaj programskog brojača i njegova vrednost postavlja na adresu instrukcije koju treba izvršiti u novom procesu

Trag procesa sa stanovišta procesa

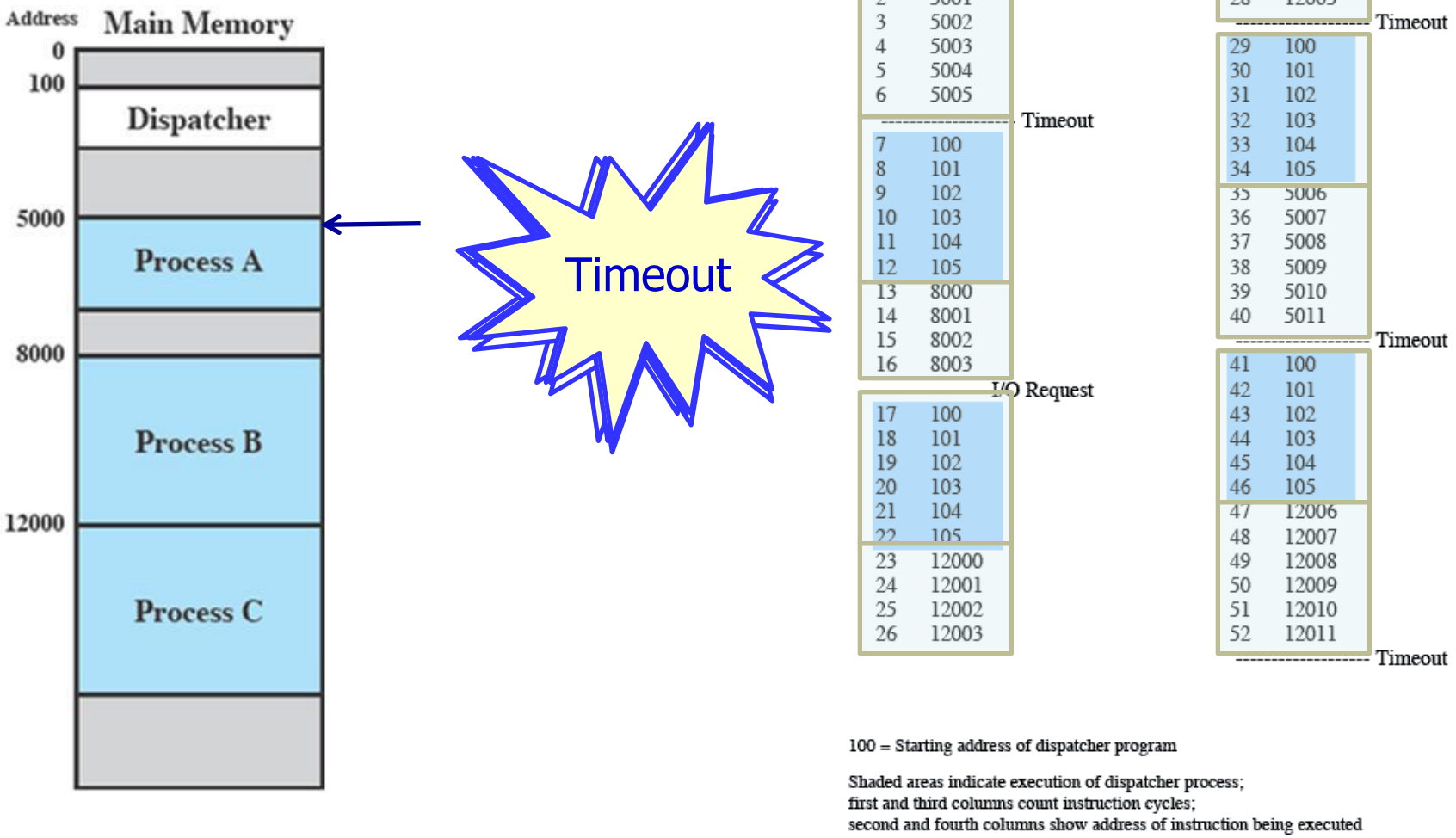
- Svaki proces se izvršava do svog završetka



(a) Trace of Process A (b) Trace of Process B (c) Trace of Process C

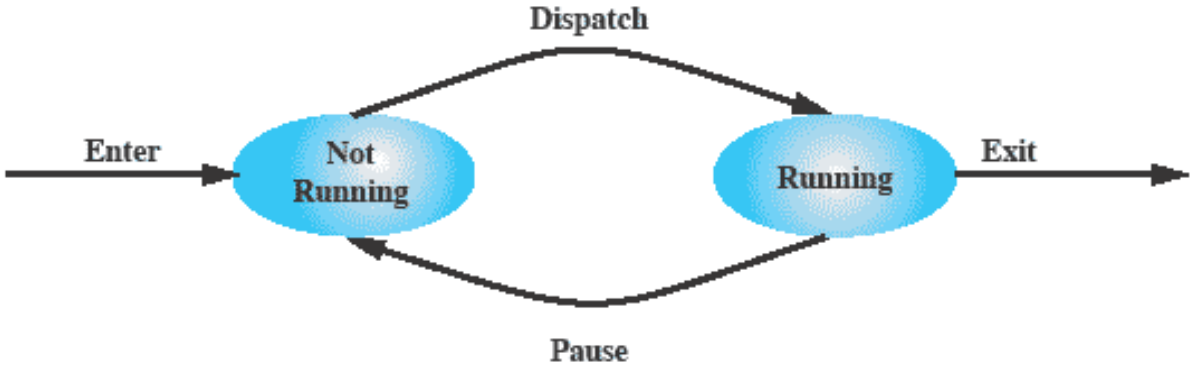
5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Trag procesa sa stanovišta procesora

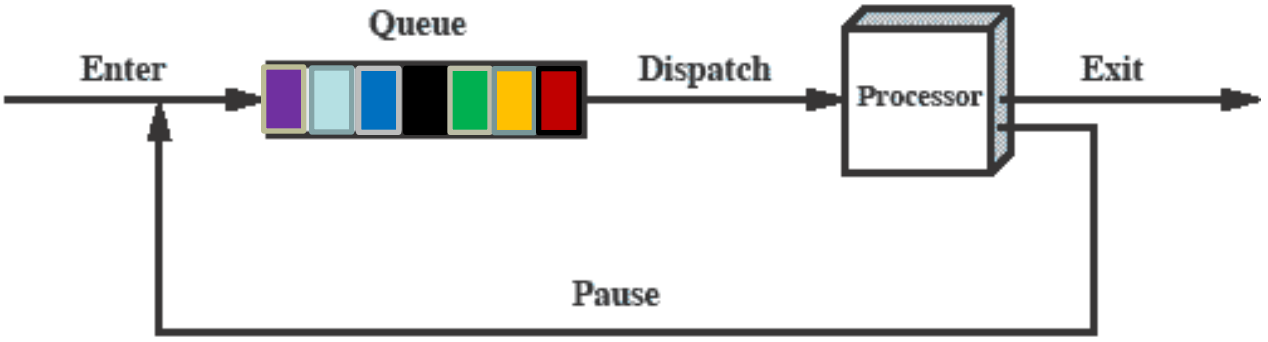


Model dva stanja procesa

- Proces može biti u jednom od dva stanja
  - Izvršava se
  - Ne izvršava se



- Red procesa
  - Procesi se pomeraju iz reda do i nazad sve dok se ne završe



Upravljanje procesima u OS

- Osnovne funkcije:
  - Kreiranje i završavanje procesa
  - Suspendovanje i ponovo aktiviranje
  - Planiranje izvršenja procesa i upravljanje procesorom/procesorima
  - Obezbeđenje mehanizama za sinhronizaciju i komunikaciju između procesa
  - Obezbeđenje mehanizama za upravljanje deadlock-om (samrtni zagrljaj, uzajamno blokiranje, zastoj)

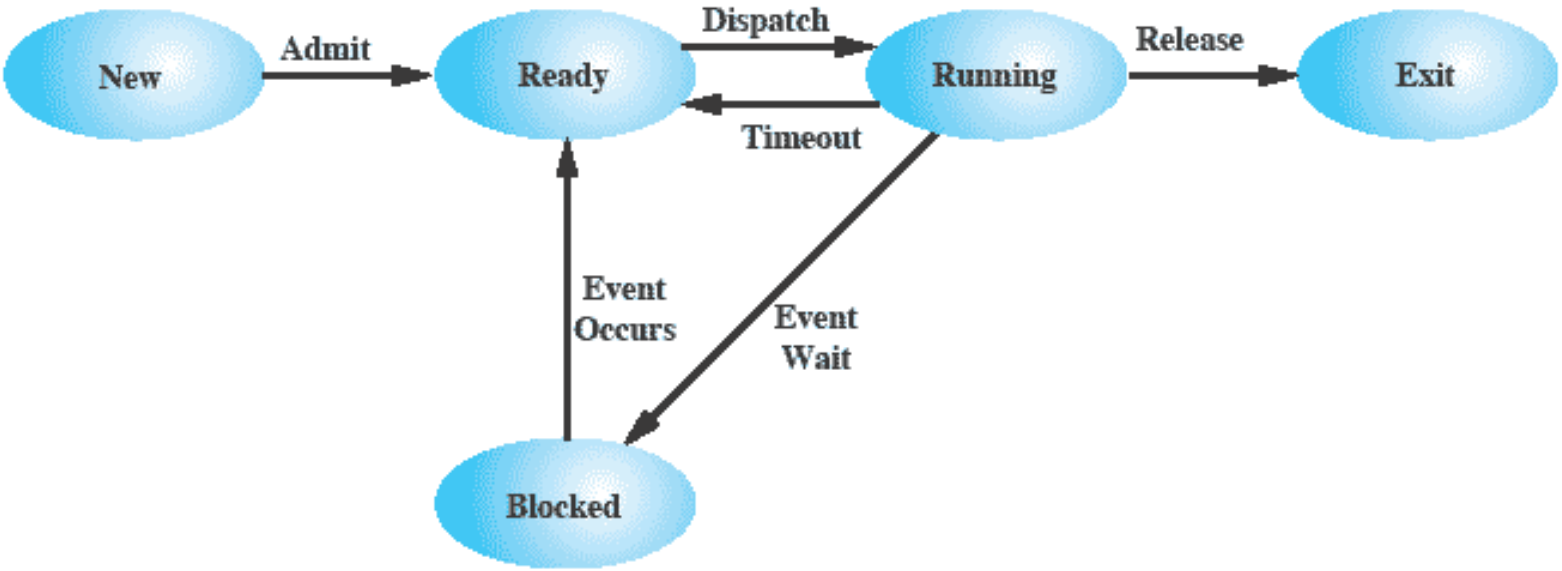
Kreiranje procesa

- Glavni razlozi za kreiranje procesa
  1. Iniciranje nekog "batch" posla
  2. Interaktivni login
  3. Kreiran od strane operativnog sistema za obezbeđenje nekog servisa
  4. Kreiran od strane postojećeg procesa
- Roditeljski (*parent*) i proces dete (*child*)
- Aktivnosti:
  1. Dodela identifikatora procesu
  2. Dodela inicijalnog prioriteta procesu
  3. Kreiranje i unos PCB novog procesa u Tabelu procesa
  4. Dodela početnih resursa procesu (memorija, otvorene datoteke, itd.)

Završetak (terminiranje) procesa

- Razlozi za završetak procesa
  1. Normalni završetak
  2. Isteklo dodeljeno vreme za izvršavanje
  3. Nedovoljno raspoložive memorije
  4. Greška usled narušavanja zaštite (*protection error*) u pristupu memoriji ili drugim resursima bez autorizacije
  5. Aritmetička greška
  6. U/I greška
  7. Ubijen od OS ili operatera
  8. Završetak roditeljskog procesa
  9. Na zahtev roditeljskog procesa koji ima privilegije da završi procese decu

Model procesa sa pet stanja



- Izvršava se (*Running*) - Proces koji se trenutno izvršava
- Spreman (*Ready*) - Proces koji je spreman za izvršavanje kada mu se pruži prilika
- Blokiran (*Blocked*)- Proces ne može da se izvršava dok se ne desi neki događaj, poput završetka U/I operacije
- Novi (*New*) – Proces tek kreiran, ali još nije primljen od strane OS u skup izvršnih procesa
- Završen (Exit) – proces izbačen iz skupa izvršnih procesa od strane OS

Prelazi između stanja procesa

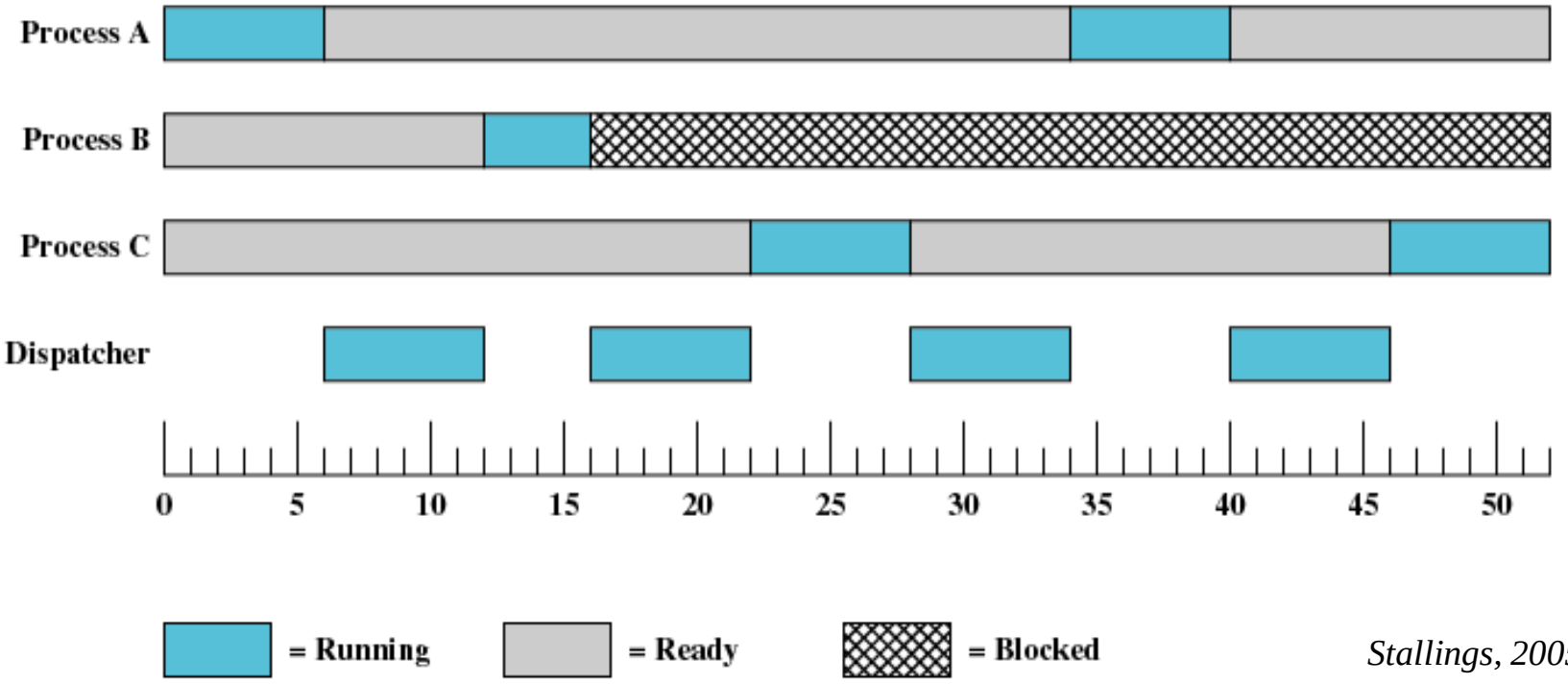
- Null → Novi
  - Kreiranje procesa za izvršenje programa
- Novi → Spreman
  - OS prebacuje proces kada ima dovoljno resursa za njegovo aktiviranje
- Spreman → Izvršava se
  - OS bira jedan od spremnih procesa za izvršavanje na CPU
- Izvršava se → Završen
  - Trenutno aktivan proces se završava od strane OS iz nekog od razloga
- Izvršava se → Spreman
  - Isteklo je vreme dodeljeno procesu za izvršavanje, ili je proces višeg prioriteta postao spreman

Prelazi između stanja procesa (2)

- Izvršava se → Blokiran
  - Proces je zahtevao resurs zbog koga mora da čeka (datoteka, memorijska sekcija, poruka od drugog procesa) ili U/I operaciju koja mora biti završena pre nastavka procesa
- Blokiran → Spreman
  - Nastao je događaj na koji je proces čekao
- Spreman → Završen
  - Nije prikazano na dijagramu; u nekim sistemima roditeljski proces može završiti proces decu u bilo kom trenutku, pa i kad su u stanju spreman, a takođe završetkom roditeljskog procesa završavaju se i sva njegova deca procesi
- Blokiran → Završen

Stanja procesa (1)

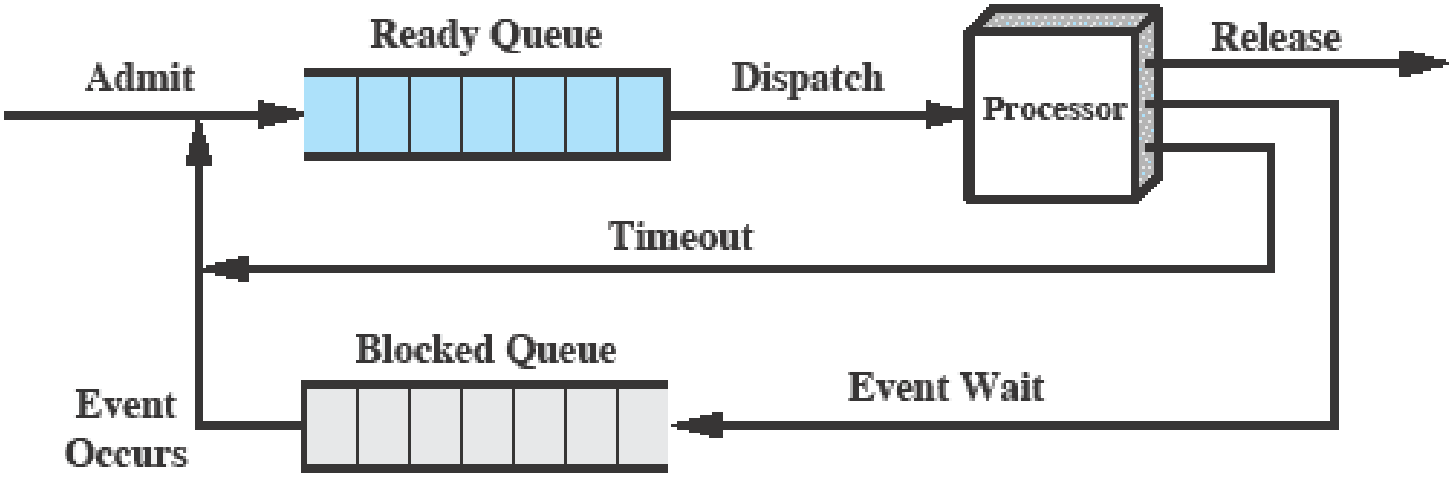
- Vremenski dijagram izvršenja procesa za prethodni primer



Stallings, 2005

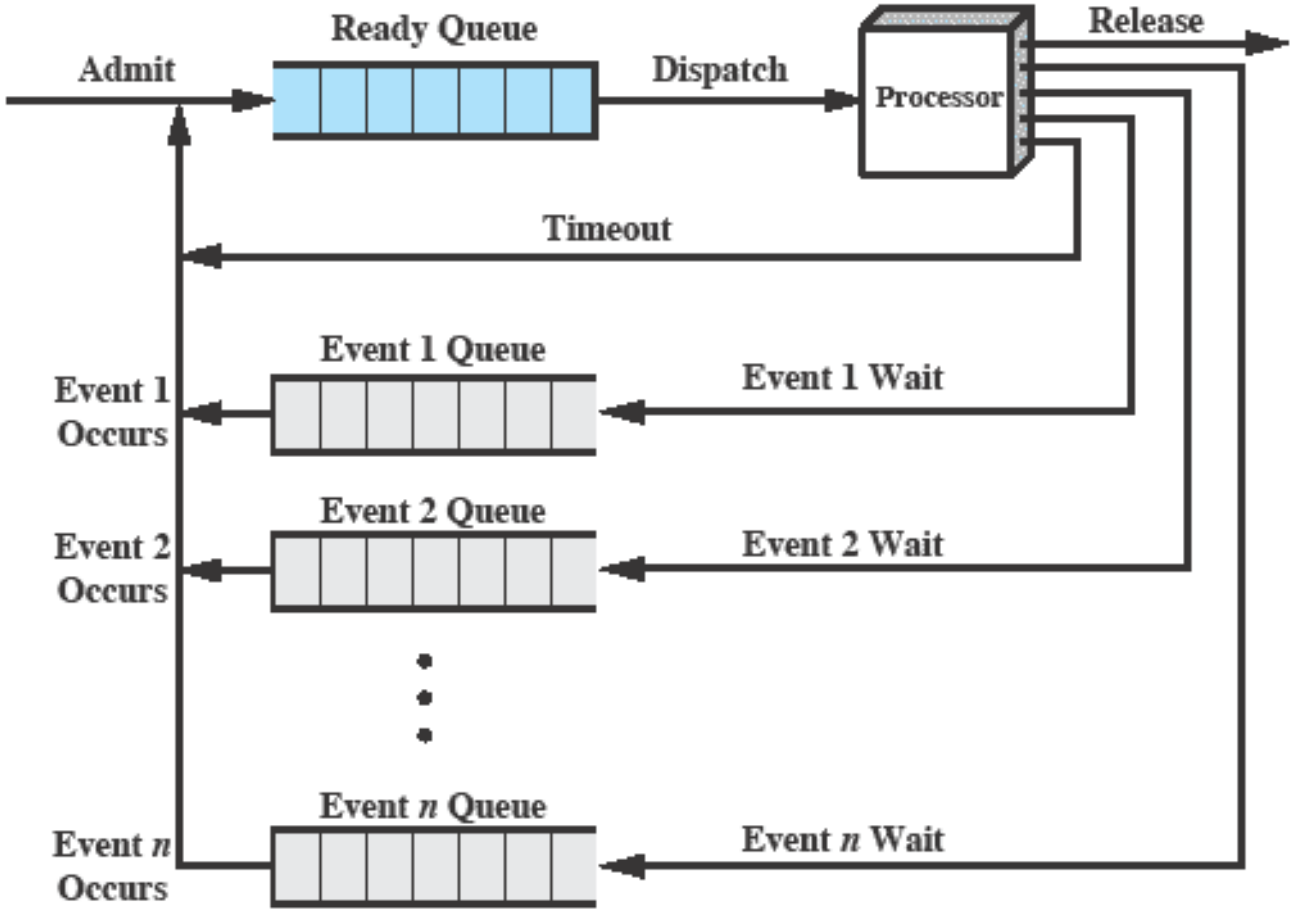
Dva reda procesa

- Jedinstveni red blokiranih procesa



(a) Single blocked queue

Višestruki redovi blokiranih



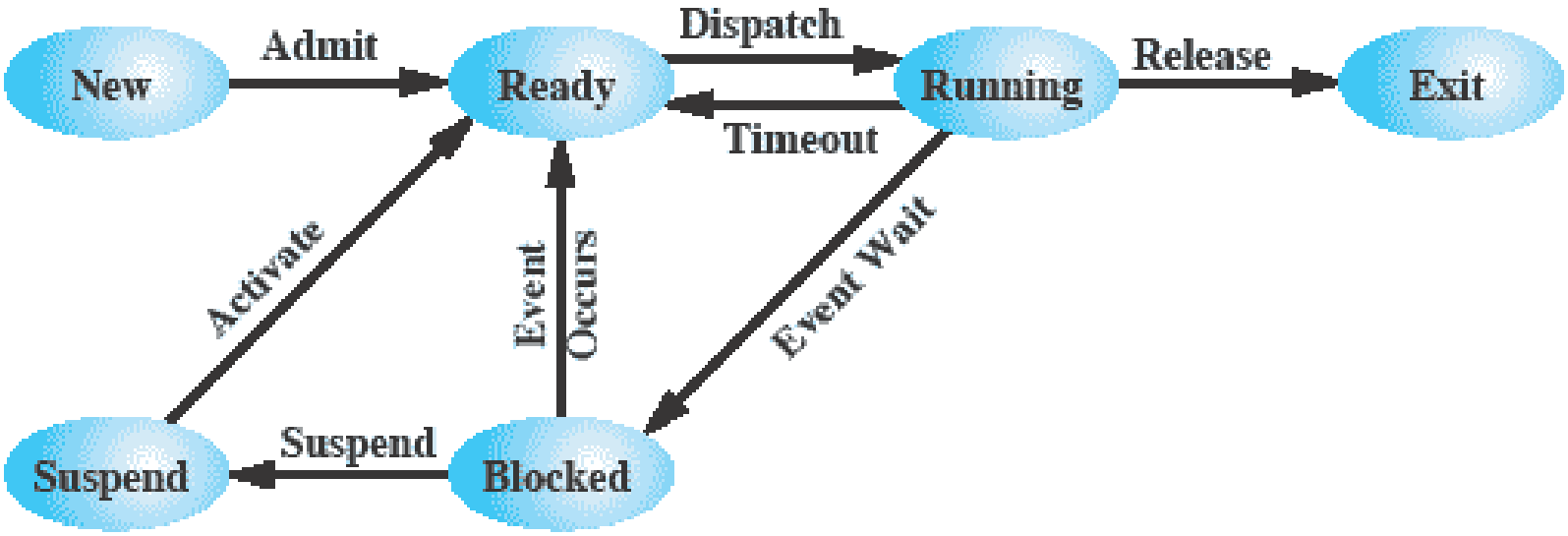
(b) Multiple blocked queues

Suspendovani procesi

- Procesor je mnogo brži od U/I uređaja, tako da može da se desi da mnogi (svi) procesi čekaju na završetak U/I operacija
  - Prebaciti (*Swap*) ove procese na disk radi oslobađanja više memorije i koristiti procesor za aktiviranje novih procesa ili prethodno suspendovanih procesa
- Proces u stanju blokiran prelazi u stanje ***suspendovan*** kada se prebaci na disk
- Dva nova stanja
  - Blokiran/Suspendovan (*Blocked/Suspend*)
  - Spreman/Suspendovan (*Ready/Suspend*)

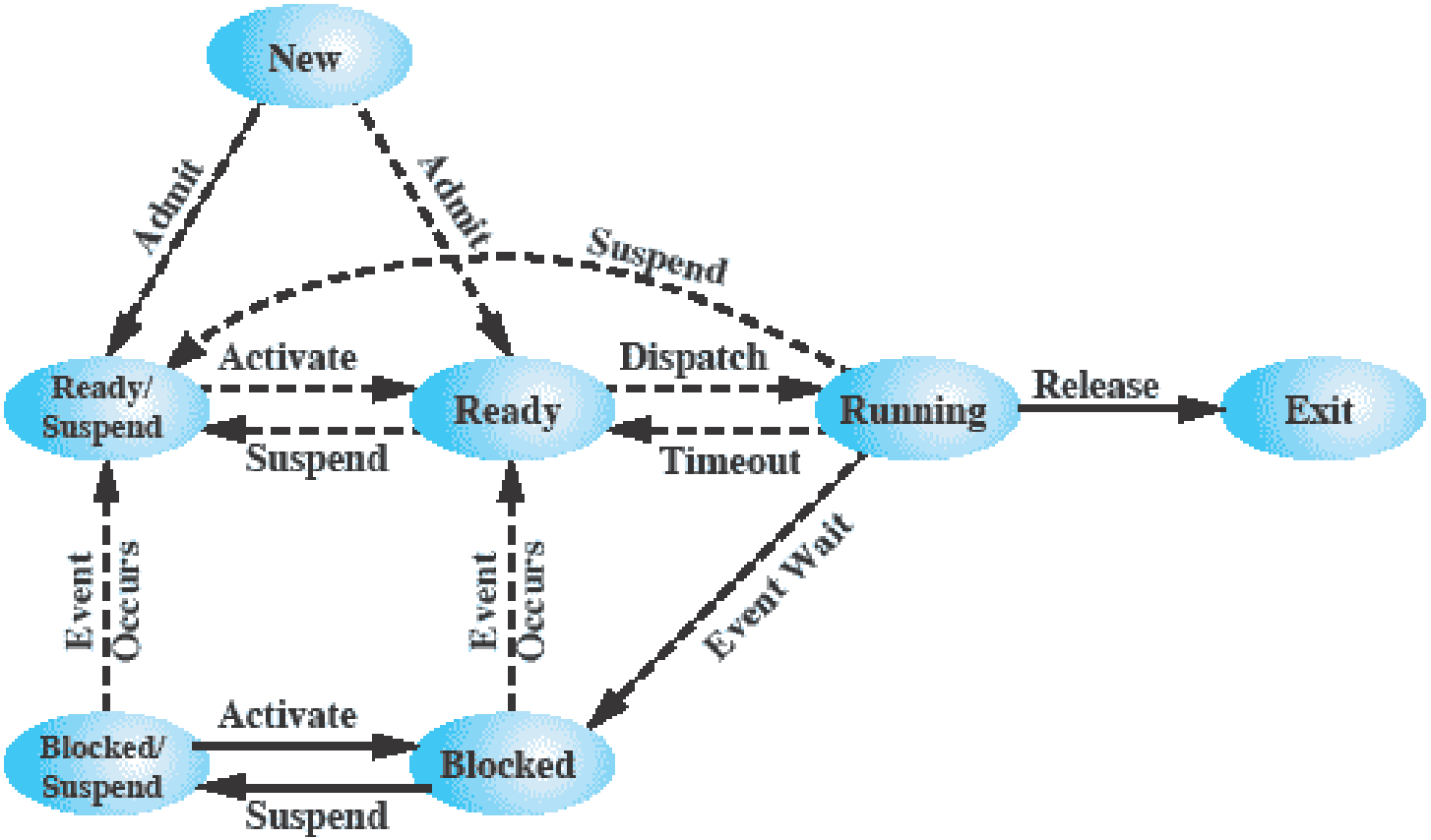
Dijagram prelaza stanja procesa

- Sa jednim suspendovanim stanjem



Dijagram prelaza stanja procesa

- Sa dva suspendovana stanja

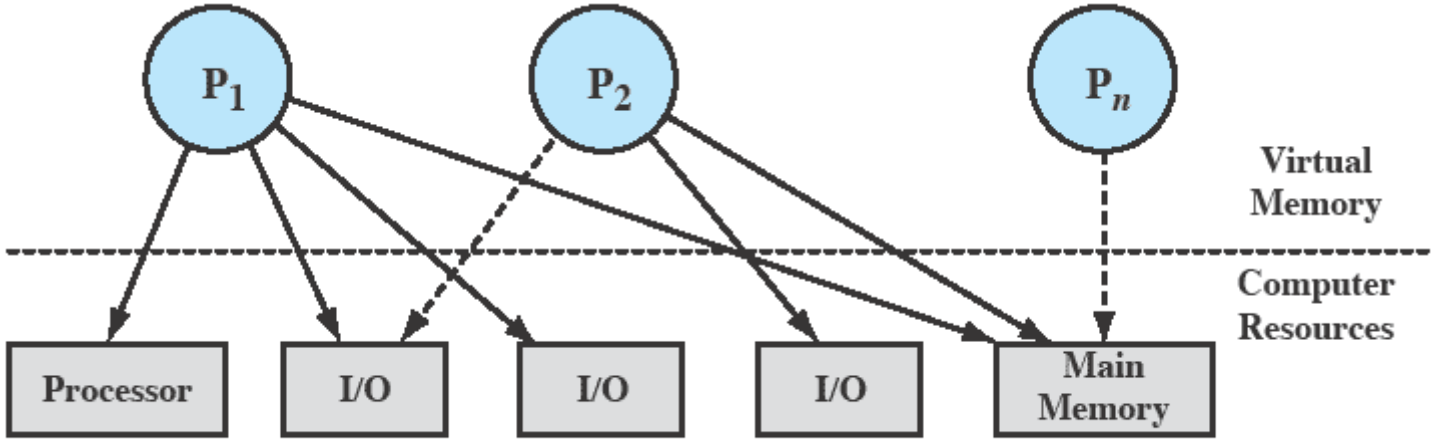


Razlozi za suspendovanje procesa

- Prebacivanje (swapping) – OS mora da oslobodi dovoljno glavne memorije da aktivira proces koji je spreman za izvršenje
- Drugi OS razlozi – OS može suspendovati proces za koji se sumnja da izaziva "probleme" tokom izvršavanja
- Zahtev interaktivnog korisnika – Suspendovanje procesa u toku debagiranja
- Vremenski – proces se izvršava periodično; između perioda može biti suspendovan
- Roditeljski proces može suspendovati izvršenje procesa-dece

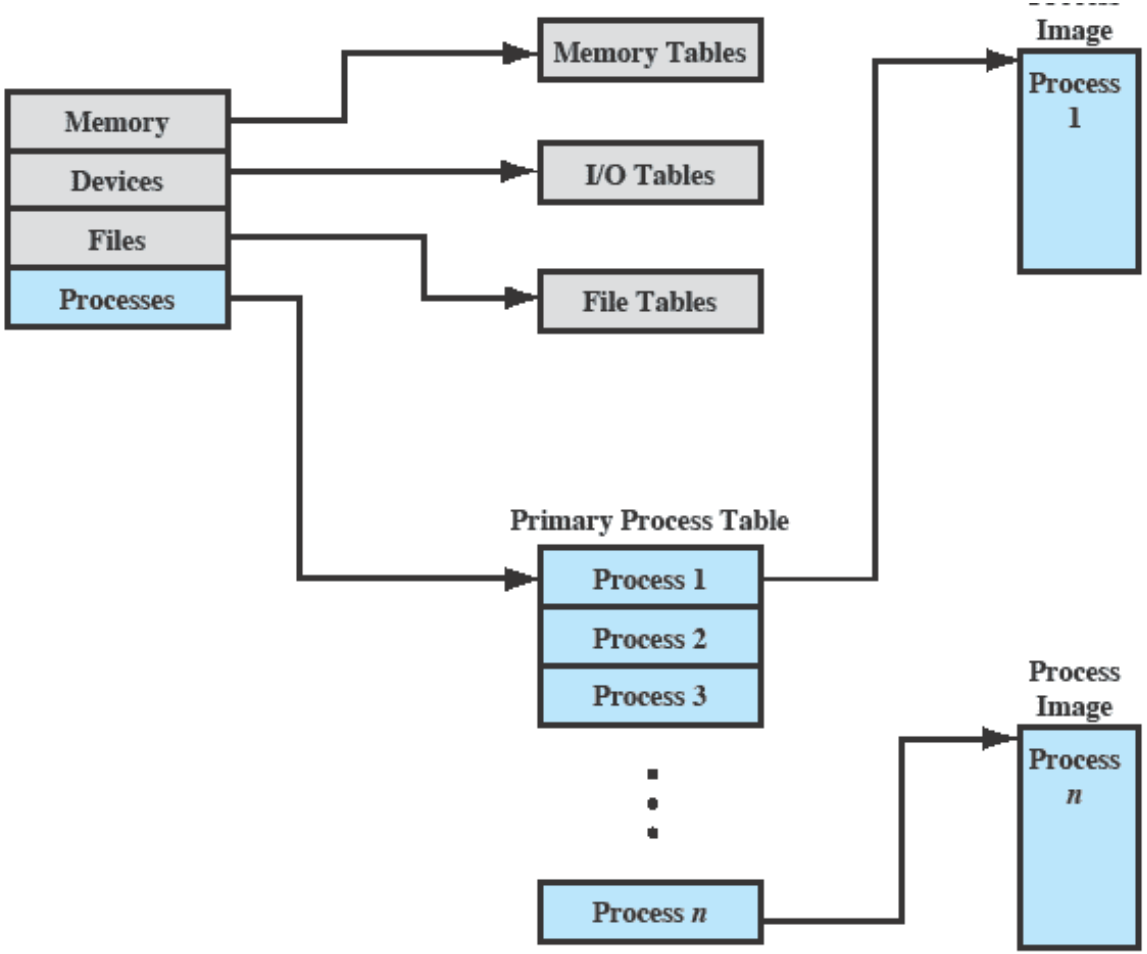
Procesi i resursi

- Procesi i njima dodeljeni (alocirani) resursi u jednom vremenskom trenutku
- Da bi OS upravljao procesima i resursima on mora imati informacije o trenutnom stanju svakog procesa i resursa.
- Za svaki entitet kojim upravlja OS konstruišu se odgovarajuće tabele



Upravljačke tabele OS

- Generalna struktura upravljačkih tabela operativnog sistema



Tabela(-e) procesa

- Da bi upravljao procesima OS mora da zna detalje o svakom procesu
  - Trenutno stanje
  - Identifikator procesa (Process ID)
  - Lokacija u memoriji
  - Ostale atribute procesa
- **Upravljački blok procesa** (*Process Control Block* – PCB) – kolekcija atributa procesa (deskriptor procesa)
  - Najvažnija struktura podataka u OS; sadrži sve informacije o procesima neophodne OS – skup PCB-a definiše stanje OS
- **Slika procesa** *(Process image)* predstavlja skup koji čine program, podaci, magacini (*stack*) - korisnički i sistemski, i atributi procesa (PCB) smešteni u glavnu memoriju (delom i na disku)

Atributi procesa

- Informacije u okviru upravljačkog bloka procesa (PCB) mogu se grupisati u tri kategorije:
  - Identifikacija procesa
  - Statusne informacije procesora
  - Upravljačke informacije procesa

Identifikacija procesa

- Svakom procesu je dodeljen jedinstveni numerički identifikator - (Process ID, PID)
- Svaki proces može da poseduje identifikatore roditeljskog procesa, grupe procesa korisnika koji je kreirao proces, itd.
- Sve druge tabele kojima upravlja OS mogu da koriste identifikator procesa da bi kros-referencirale PCB procesa

Statusne informacije procesora

- Sastoje se od sadržaja registara procesora.
  - Registara opšte namene vidljivih od strane korisnika
  - Upravljačkih i statusnih registara (PC, uslovni kodovi, flagovi za dozvoljen/nedozvoljen prekid, mod izvršenja,...)
  - Stack pointera - LIFO strukture za smeštanje parametara, povratnih adresa i lokalnih promenljivih pri pozivima procedura i sistemskim pozivima
- *Program Status Word* (PSW)
  - Sadrži statusne informacije
  - Primer : EFLAGS registar na Pentium procesorima

Upravljačke informacije procesa

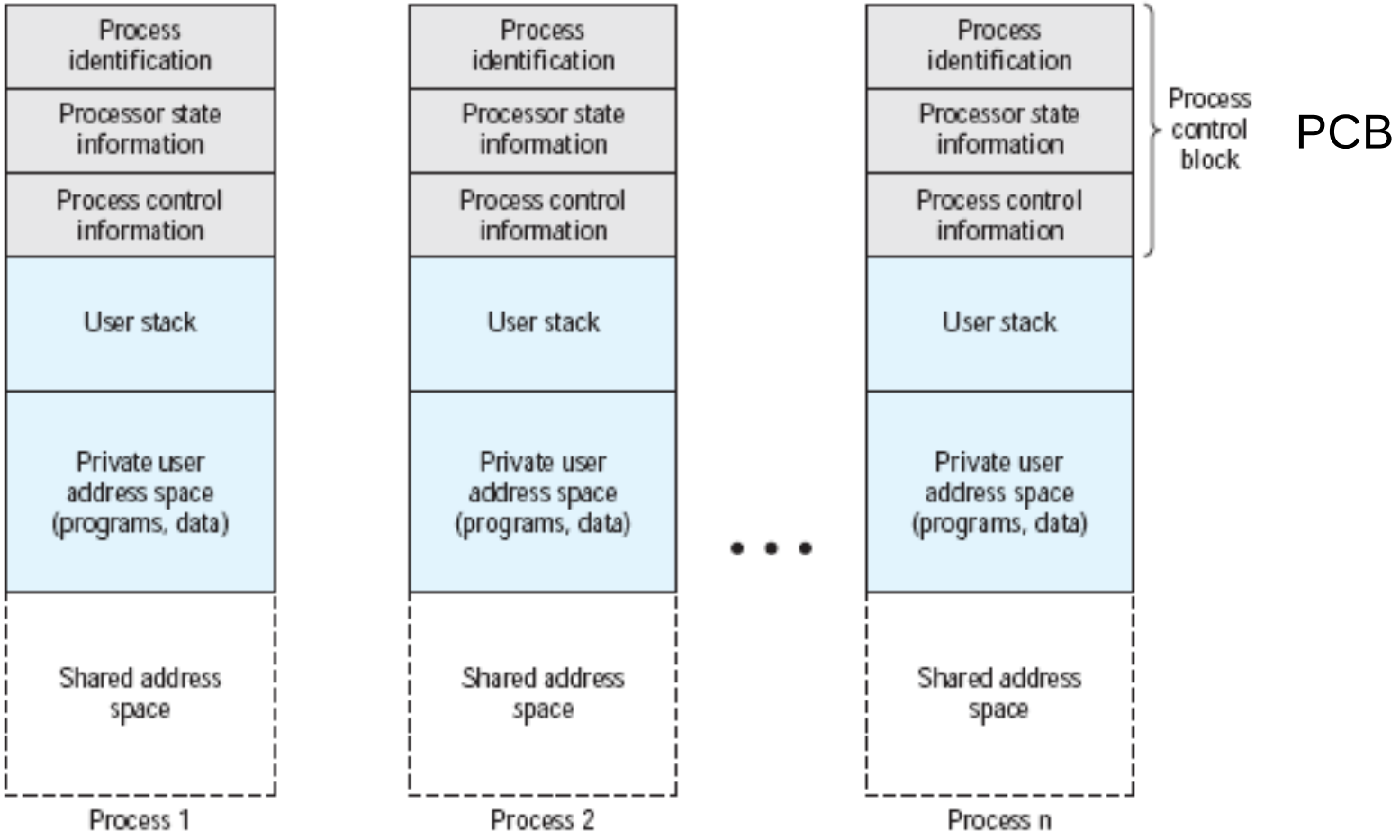
- Informacije o stanju i planiranju (*scheduling*)
  - Stanje procesa
  - Prioritet
  - Informacije vezane za planiranje – zavisno od algoritma planiranja
  - Događaj na koji proces čeka
- Struktuiranje (povezivanje, ulančavanje) PCB-a
  - Procesi (PCB) su povezani u različitim strukturama (redovima, listama, itd.): roditelj-dete, red čekanja na U/I, itd.
- Međuprocesna komunikacija
  - Različiti flag-ovi, signali, poruke između procesa

Upravljačke informacije procesa (2)

- Privilegije procesa
  - Privilegije za pristup memoriji, izvršenje određenih tipova instrukcija, sistemskih servisa
- Upravljanje memorijom
  - Pokazivači (adrese) na tabele stranica/segmenata koje opisuju virtuelnu memoriju dodeljenu procesu
- Vlasništvo i korišćenje resursa
  - Resursi kojima upravlja proces, poput otvorenih datoteka
  - Takođe može biti uključena i istorija korišćenja procesora ili ostalih resursa za potrebe planiranja procesa

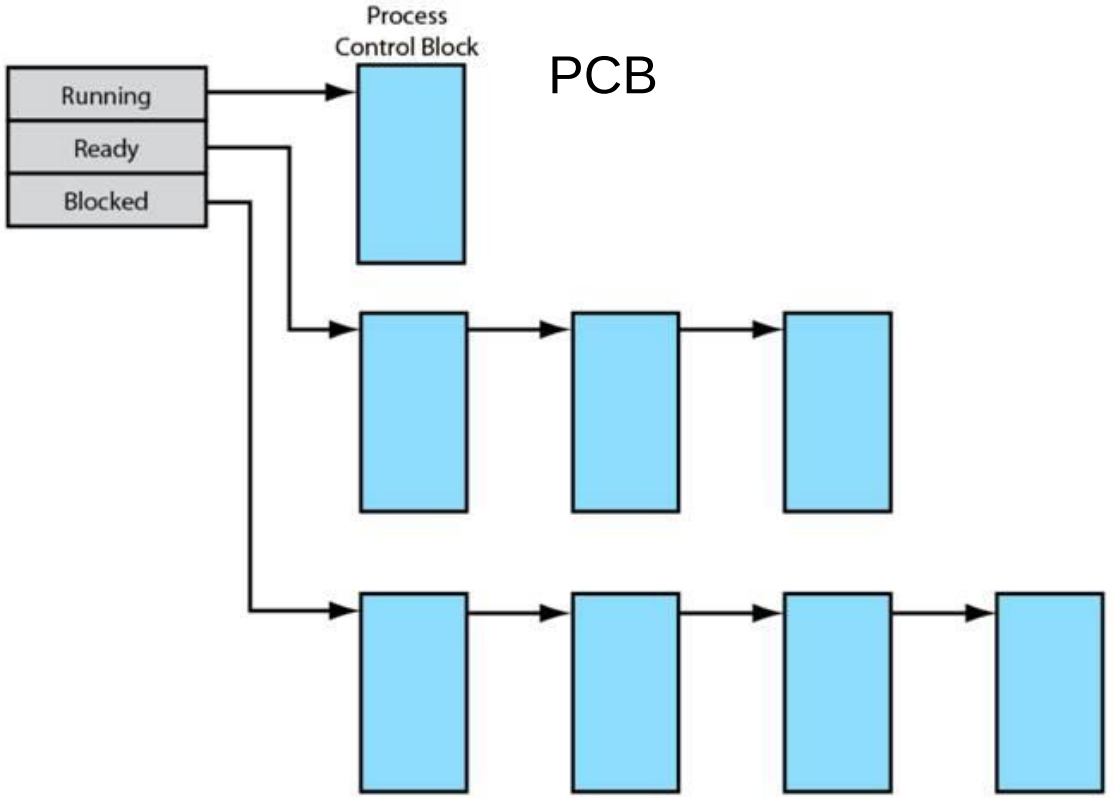
Slika procesa u virtuelnoj memoriji

- Struktura slike korisničkih procesa u virtuelnoj memoriji

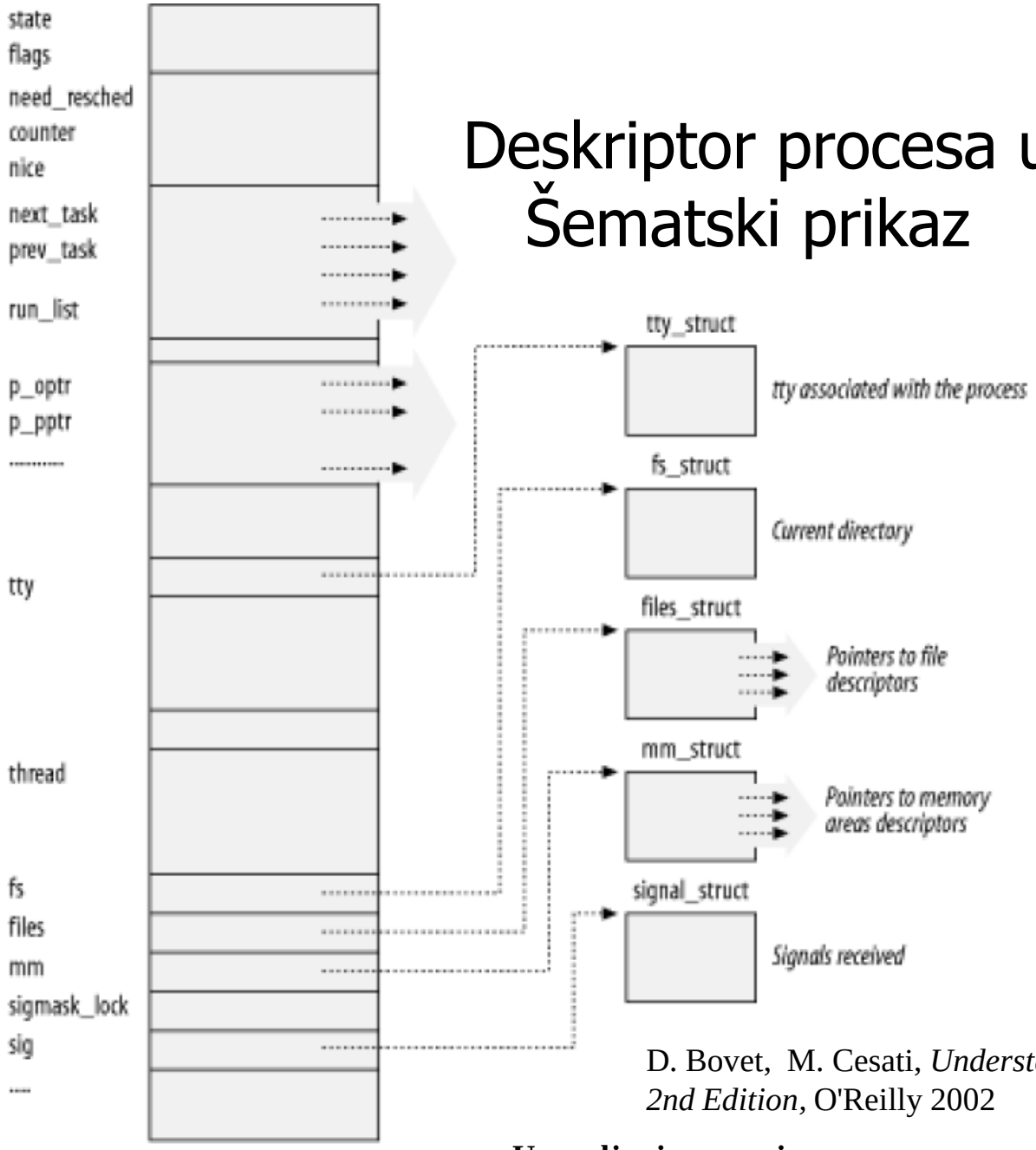


Struktura redova (listi) procesa

- PCB je najvažnija struktura podataka u operativnom sistemu
- OS moduli (planiranje, alokacija resursa, obrada prekida, nadgledanje i analiza performansi, itd.) čitaju i/ili modifikuju PCB-ove
- PCB su povezani u različitim redovima i tablicama koji se često implementiraju kao lančane liste



Deskriptor procesa u Linux-u - Šematski prikaz



D. Bovet, M. Cesati, *Understanding the Linux Kernel. 2nd Edition*, O'Reilly 2002

Deskriptor procesa u Linux-u (1)

```
struct task_struct {
    volatile long       state;  /*TASK_RUNNING, TASK_INTERRUPTIBLE, TASK_UNINTERRUPTIBLE,
      TASK_STOPPED, TASK_ZOMBIE */
    long                counter; //brojač vremenskih taktova
    long                priority; //prioritet procesa
    unsigned            long signal;
    unsigned            long blocked;   /* bitmap of masked signals */
    unsigned            long flags;     /* per process flags, defined below */

    struct task_struct   *next_task, *prev_task; //pokazivač na naredni/prethodni proces u
      listi
    unsigned long       saved_kernel_stack;
    unsigned long       kernel_stack_page;

    int                 pid;

  /*  pointers to (original) parent process, youngest child, younger sibling,
     * older sibling, respectively.  (p->father can be replaced with
     * p->p_pptr->pid) */
  //pokazivači na roditelja, decu i braću tekućeg procesa
    struct task_struct   *p_opptr, *p_pptr, *p_cptr,  *p_ysptr, *p_osptr;
  ...
```

Deskriptor procesa u Linux-u (2)

```
...
  struct wait_queue    *wait_chldexit;
  unsigned short       uid,euid,suid,fsuid;
  unsigned short       gid,egid,sgid,fsgid;
  unsigned long        timeout, policy, rt_priority;
  long                 utime, stime, cutime, cstime, start_time;
  struct fs_struct     *fs;
  struct files_struct  *files;
  struct mm_struct     *mm;
  struct signal_struct *sig;
  ...
} //end task_struct
```

Upravljanje izvršenjem procesa

- Većina procesora podržava najmanje dva moda izvršavanja
- Korisnički (*user*) mod
  - Mod sa manje privilegija – zabranjen pristup svim memorijskim adresama i izvršavanje privilegovanih (U/I) instrukcija
  - Korisnički programi se izvršavaju u ovom modu
- Kernel (*system, control, supervisor*) mod – mod jezgra
  - Privilegovan mod
  - U ovom modu se izvršava kernel operativnog sistema
  - Bit(-ovi) u PSW označavaju trenutni mod izvršenja

Kreiranje procesa

- Prilikom kreiranja novog procesa OS mora da:
  - Dodeli jedinstveni identifikator procesa
  - Dodeli (alocira) memorijski prostor za proces, tačnije za sve elemente slike procesa
  - Inicijalizuje upravljački blok procesa (PCB)
  - Uspostavi odgovarajuće veze novog procesa ulančavanjem njegovog PCB u odgovarajuće redove/liste
  - Kreira ili proširi ostale strukture podataka
    - Na primer, za obračun korišćenja resursa, ili ocenu i analizu performansi

Zamena (*komutiranje*) procesa

- Zamena (*komutiranje, switching*) procesa predstavlja zamenu aktivnog procesa od strane OS
- Kada nastaje zamena procesa:
  - Prekid (*interrupt*) – eksterni događaj u odnosu na izvršenje procesa; prekid se javlja kao reakcija na eksterne, asinhrone događaje
    - Clock prekid, U/I prekid, greška memorije (*memory/page fault*)
  - Trap – Odnosi se na izvršenje tekuće instrukcije procesa - ako nastane greška ili izuzetak koji je fatalan
  - Sistemski (*supervisor*) poziv - Zahtev od strane procesa koji se izvršava za izvršenje U/I operacije (npr. pristup datoteci) kojim se aktivira odgovarajuća procedura u kodu OS.

Promena (*komutiranje*) moda

- Nakon izvršetka svake instrukcije, procesor proverava da li je nastao prekid (*interrupt* signal)
- Ukoliko je prekid nastao, procesor izvršava sledeće korake:
  - Postavlja programski brojač na početnu adresu rutine za obradu prekida (*interrupt handler*)
  - Prebacuje se iz korisničkog u kernel mod, tako da rutina za obradu prekida može uključiti i privilegovane instrukcije
  - Kontekst prekinutog procesa se čuva u njegovom PCB
    - Statusne informacije procesora
  - Izvršava se rutina za obradu prekida
  - Ukoliko nastanak prekida ne zahteva zamenu (*switching*) procesa i promenu stanja procesa, restauriraju se statusne informacije prekinutog procesa
  - Prekid kloka ili zbog U/I operacije zahteva prebacivanje prekinutog procesa u novo stanje (Spreman, Blokiran)

Promena stanja procesa

- Promena stanja procesa (konteksta) – *process/context switch*
- Koraci u promeni procesa
  1. Sačuvati kontekst procesa uključujući programski brojač i registre procesora
  2. Ažurirati PCB procesa koji je trenutno u stanju Izvršava se (*Running*)
  3. Premestiti PCB u odgovarajući red – spreman, blokiran, spreman/suspendovan
  4. Izabrati sledeći proces za izvršavanje
  5. Ažurirati PCB procesa koji je selektovan
  6. Ažurirati stukture podataka za upravljanje memorijom
  7. Obnoviti (rekonstruisati) kontekst selektovanog procesa

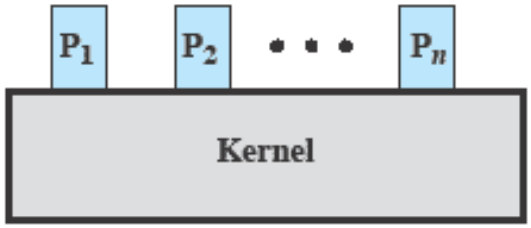
Izvršavanje operativnog sistema

- OS radi na isti način kao i ostali računarski softver
- Ako je OS samo kolekcija programa i ako se ovi programi izvršavaju od strane procesora baš kao i bilo koji drugi programi, da li je OS proces?
- Ako jeste, kako se njime upravlja?
  - Ko (šta) upravlja njime?

Izvršenje OS-a

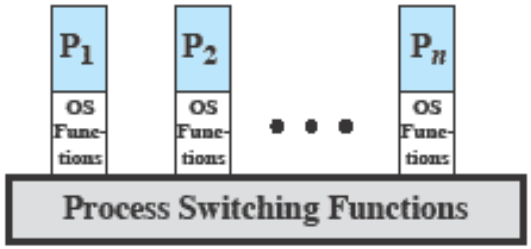
- Kernel bez procesa
  - Tradicionalni pristup u starijim OS
  - Izvršenje kernela van bilo kog procesa
  - OS kod se izvršava kao poseban entitet u glavnoj memoriji , sa svojim stekom, u kernel modu (privilegovanom režimu)
- Izvršenje OS unutar korisničkih procesa
  - Uobičajeno u OS na PC i radnim stanicama
  - OS softver predstavlja kolekciju procedura koje korisnički proces može pozvati za izvršenje različitih funkcija i koje se izvršavaju unutar okruženja korisničkog procesa
  - Slika procesa sadrži i delove za program, podatke i stek kernel programa
  - Proces se izvršava u kernel modu kada se izvršava kod OS-a
- OS zasnovan na procesima
  - OS implementiran kao kolekcija sistemskih procesa
  - Korisno je u višeprocesorskom ili višeračunarskom okruženju

Odnos između OS i korisničkih procesa



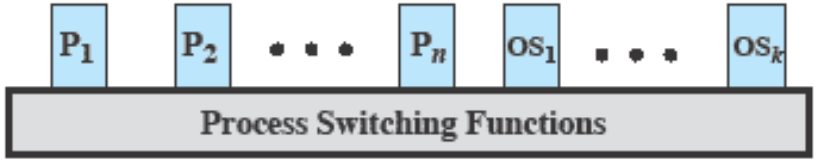
(a) Separate kernel

- Zasebno jezgro



(b) OS functions execute within user processes

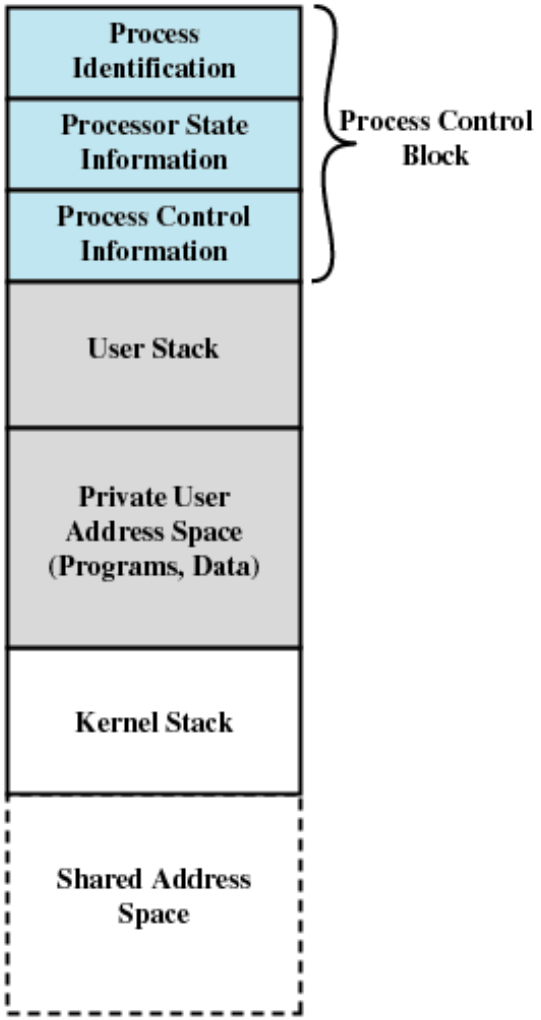
- Funkcije OS-a se izvršavaju unutar korisničkih procesa



(c) OS functions execute as separate processes

- Funkcije OS-a se izvršavaju kao zasebni procesi

Slika procesa



- OS se izvršava unutar korisničkog procesa (adresnog prostora)
- Kernel stek se koristi za upravljanje pozivima procedura dok je proces u kernel modu
- OS kod i podaci su u deljenom adresnom prostoru (deljeni su između svih korisničkih procesa)

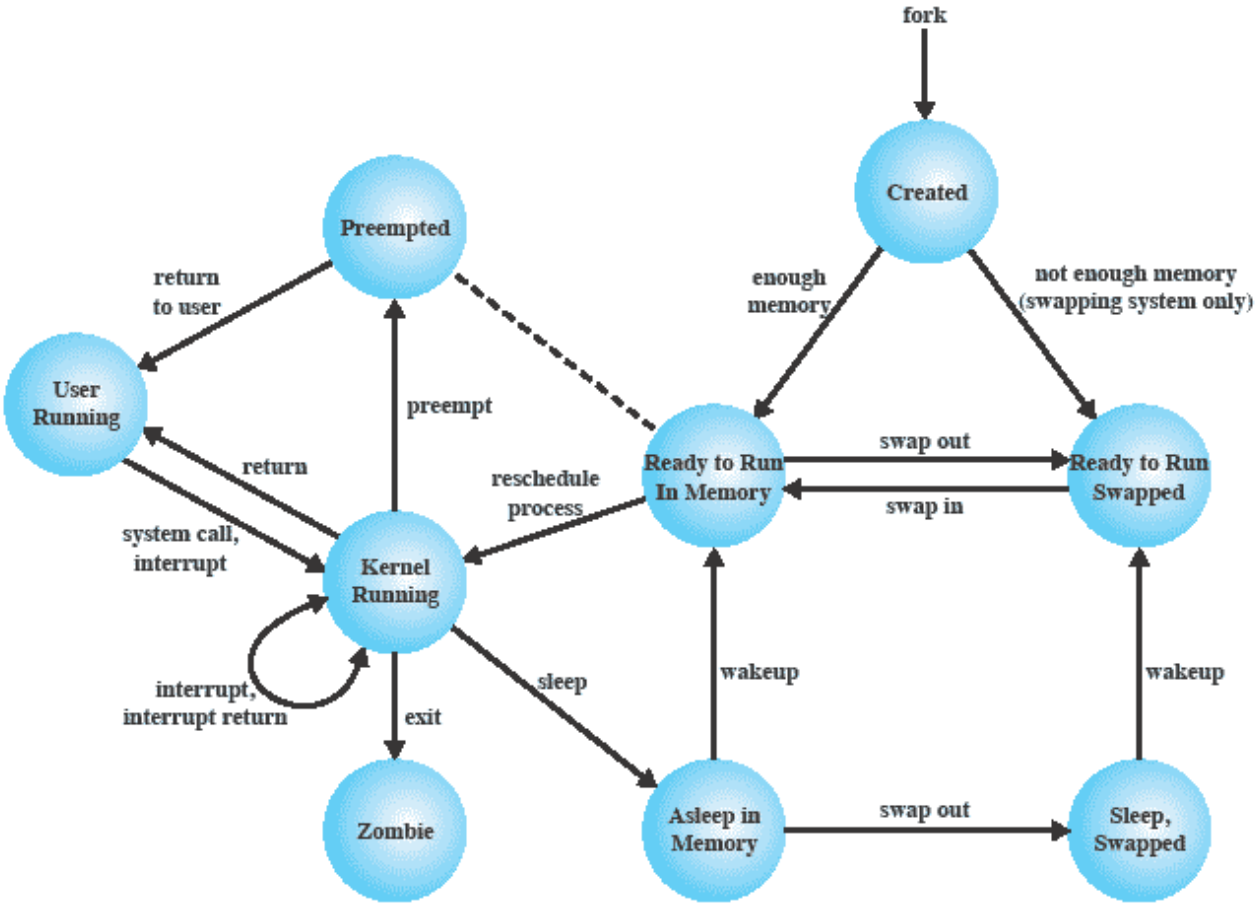
Unix SVR4

System V Release 4

- Veći deo operatinvog sistema se izvršava u korisničkom procesu (koristi model b) sa slajda 46)
- Sistemski procesi se izvršavaju isključivo u kernel modu
- Korisnički procesi
  - Korisnički popgrami i funkcije se izvršavaju u korisničkom modu.
  - U kernel modu se izvršavju funkcije koje pripadaju kernelu.

UNIX SVR4 - upravljanje procesima

- Stanja procesa i promene stanja (9)
- Dva stanja *Running* (izvršavanje u korisničkom i kernel modu)



Upravljanje procesima

UNIX proces

- Proces u UNIX-u je kolekcija struktura podataka koje obezbeđuju OS-u sve informacije neophodne za upravljanje i raspoređivanje procesa.
- Slika UNIX procesa sadrži sledeće elemente:
  - Kontekst korisničkog nivoa
    - Tekst programa, podaci, korisnički stek, deljena memorija
  - Kontekst registara
    - Programski brojač, statusni registar procesora, stek pointeri, registri opšte namene
  - Kontekst sistemskog nivoa
    - Ulaz (*entry*) u tabelu procesa (PCB)
    - U (*user*) oblast – neophodne kernelu kada se izvršava u kontekstu procesa
    - Tabela regiona – definiše mapiranje između virtuelnih i fizičkih adresa, kao i prava pristupa određenim regionima – koristi se od strane sistema za upravljanje memorijom
    - Kernel stek – za pozive/povratke iz kernel procedura

Kreiranje procesa

- Prednji (*foreground*) procesi
- Pozadinski (*background*) procesi – demoni, servisi
- Kako videti procese u sistemu?
  - Unix komanda ps
  - Windows - TaskManager (Ctrl+Alt+Del)
- Kako kreirati proces?
  - Proces koji se izvršava poziva sistemski poziv za kreiranje procesa
  - Unix
    - fork – proces kreira svoj klon; pravi se kopija slike procesa roditelja
    - exec (*execv, execve, execvp, execl, execle, execlp)* - definiše program koji će se izvršavati u novom procesu i okolinu novog procesa
  - Win32
    - CreateProcess - kreira novi proces i puni ga novim programom
- Nakon kreiranja oba procesa imaju sopstvene slike u memoriji i adresne prostore

Sistemski pozivi za terminiranje

- Sistemski poziv kojim proces terminira sam sebe:
  - Unix exit
  - Windows API (Win32) ExitProcess
- Sistemski poziv kojim jedan proces terminira drugi:
  - Unix kill
  - Windows API (Win32) TerminateProcess

Sistemski pozivi

- POSIX
- UNIX/Linux
- Windows API

Procesi u UNIX-u

Kreiranje procesa u UNIX-u

```
pid = fork( );                  /* if the fork succeeds, pid > 0 in the parent */
if (pid < 0) {
      handle_error( );          /* fork failed (e.g., memory or some table is full) */
} else if (pid > 0) {
                                /* parent code goes here. /*/
} else {
                                /* child code goes here. /*/
}
```

Sistemski pozivi za upravljanje procesima u Unix-u

System call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, opts)	Wait for a child to terminate
s = execve(name, argv, envp)	Replace a process' core image
exit(status)	Terminate process execution and return status
s = sigaction(sig, &act, &oldact)	Define action to take on signals
s = sigreturn(&context)	Return from a signal
s = sigprocmask(how, &set, &old)	Examine or change the signal mask
s = sigpending(set)	Get the set of blocked signals
s = sigsuspend(sigmask)	Replace the signal mask and suspend the process
s = kill(pid, sig)	Send a signal to a process
residual = alarm(seconds)	Set the alarm clock
s = pause( )	Suspend the caller until the next signal

s je kod greške
pid je ID procesa
residual je preostalo vreme od prethodnog alarma

Sistemski pozivi za upravljanje poslovima, procesima, nitima i fiberima u Windows API

Windows API Function	Description
CreateProcess	Create a new process
CreateThread	Create a new thread in an existing process
CreateFiber	Create a new fiber
ExitProcess	Terminate current process and all its threads
ExitThread	Terminate this thread
ExitFiber	Terminate this fiber
SetPriorityClass	Set the priority class for a process
SetThreadPriority	Set the priority for one thread
CreateSemaphore	Create a new semapahore
CreateMutex	Create a new mutex
OpenSemaphore	Open an existing semaphore
OpenMutex	Open an existing mutex
WaitForSingleObject	Block on a single semaphore, mutex, etc.
WaitForMultipleObjects	Block on a set of objects whose handles are given
PulseEvent	Set an event to signaled then to nonsignaled
ReleaseMutex	Release a mutex to allow another thread to acquire it
ReleaseSemaphore	Increase the semaphore count by 1
EnterCriticalSection	Acquire the lock on a critical section
LeaveCriticalSection	Release the lock on a critical section