Operativni sistemi

Upravljanje procesima

Prof. dr Dragan Stojanović

Katedra za računarstvo
Univerzitet u Nišu, Elektronski fakultet

Literatura

- *Operating Systems: Internals and Design Principles*, edition, W. Stallings, Pearson Education Inc., 7th – 2012, (5th -2005, 6th - 2008, 8th – 2014 , 9th – 2017)
  - http://williamstallings.com/OperatingSystems/
  - http://williamstallings.com/OperatingSystems/OS9e-Student/

- Poglavlje 3: Opis procesa i upravljanje

Koncept procesa

- Svi multprogramski OS zasnovani su na konceptu procesa
- Operativni sistem mora da obezbedi:
  - Kreiranje procesa
  - Preplitanje izvršenja više procesa u cilju maksimalnog iskorišćenja procesora
  - Dodelu resursa procesima u skladu sa određenom politikom,
  - Sinhronizaciju i uzajamno isključivanje procesa i zaštitu resursa svakog procesa od strane drugih procesa
  - Komunikaciju između procesa

Osnovni pojmovi i koncepti

- Računarska platforma se sastoji od skupa hardverskih resursa
- Računarske aplikacije se razvijaju kako bi izvršile neki zadatak; prihvataju ulaz spolja, vrše obradu i generišu izlaz
- Nije efikasno pisati aplikacije direktno za datu hardversku platformu
- OS obezbeđuje interfejs između aplikacije i hardvera računara
- OS pruža apstrakciju resursa koje aplikacije može tražiti ili pristupati i upravlja njihovom upotrebom

Šta je proces?

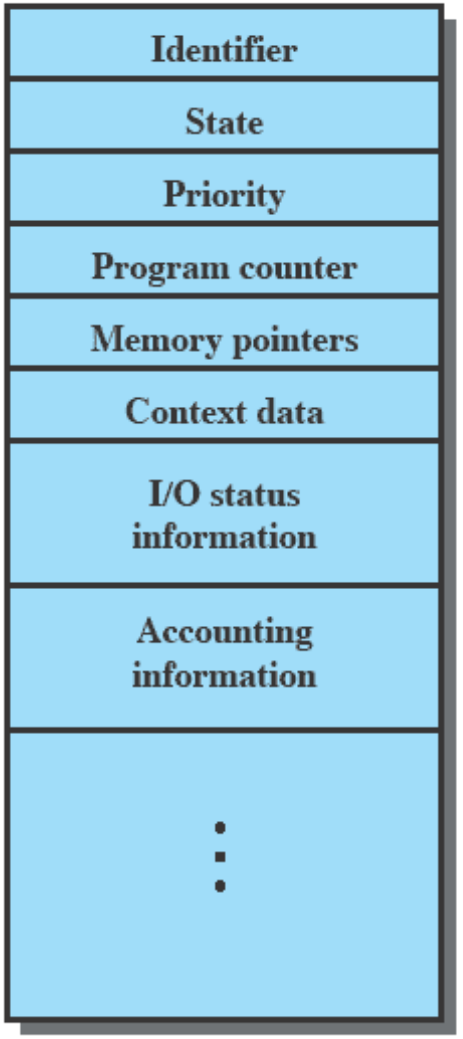
- Proces je osnovni koncept operativnog sistema i predstavlja program u izvršavanju na procesoru
  - Proces se ponekad naziva zadatak (task)
  - Svi multiprogramski OS su izgrađeni oko koncepta procesa
- Gledano sa strane OS-a, proces je osnovna jedinica izvršavanja i najmanji entitet koji se može planirati, dodeliti i izvršavati na procesoru
- Proces je 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
  - Steka
  - Atributa koji opisuju stanje procesa i koji su smešteni u Upravljački blok procesa:
    - Identifikator
    - Stanje
    - Prioritet
    - Programski brojač (*Program counter*)
    - Pokazivači na memoriju u koju je smešten proces
    - Kontekstni podaci (podaci smešteni u registrima procesora za vreme izvršenja procesa)
    - Informacije o U/I statusu
    - Informacije o obračunu korišćenja resursa
    - …

Upravljački blok procesa

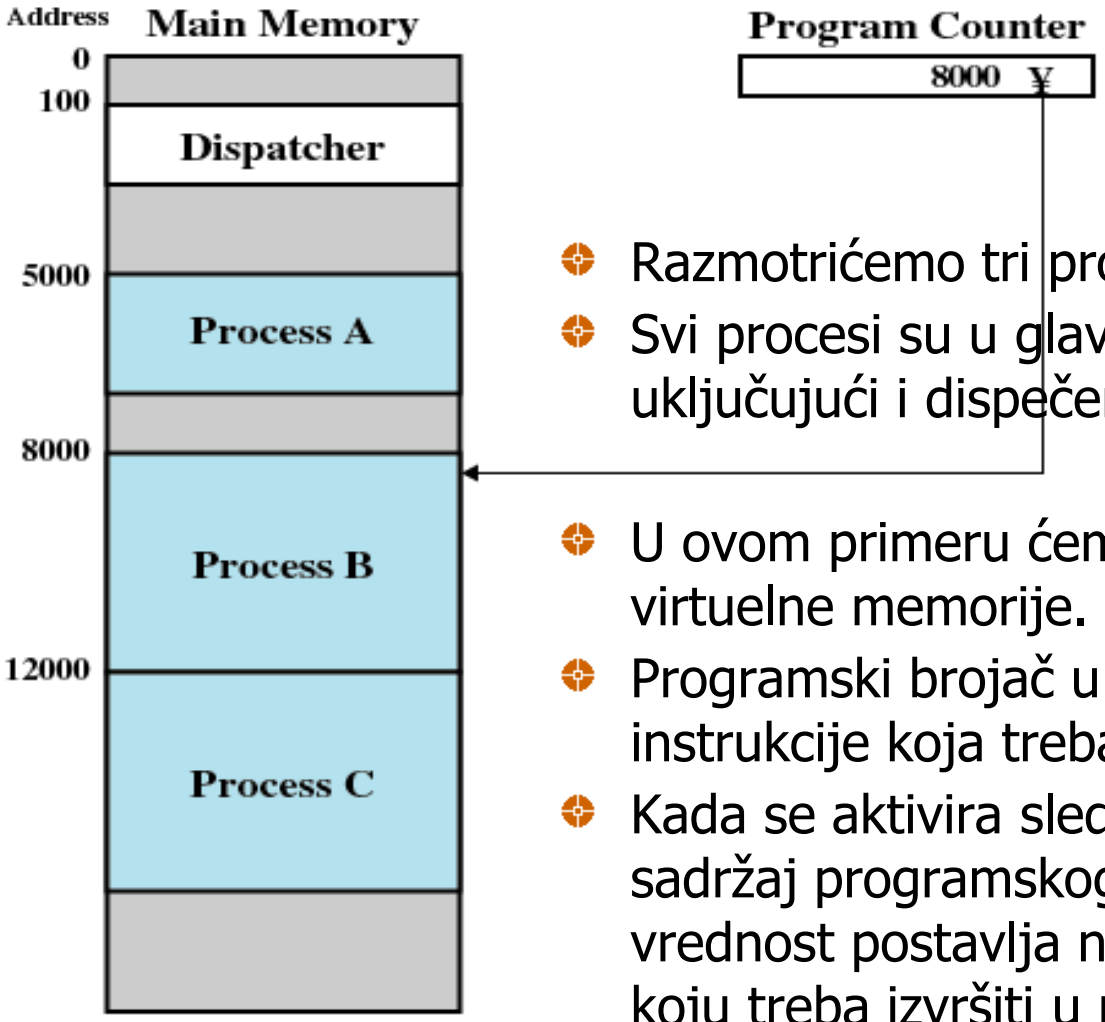
- *Process Control Block* - PCB
- Sadrži atribute procesa
- Kreiran i upravljan od strane OS
- Obezbeđuje podršku za višestruke procese
- Tabela procesa predstavlja skup PCB-ova svih procesa



Praćenje (tok, *trace*) procesa

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

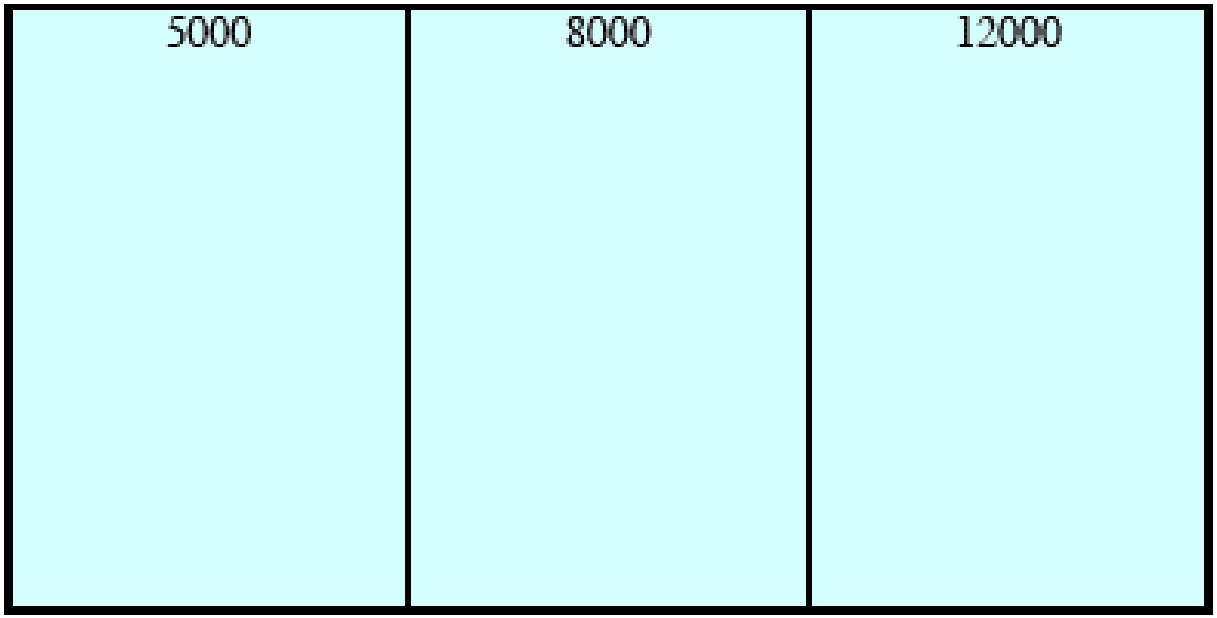
Izvršenje procesa



- 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

Praćenje procesa sa stanovišta procesa

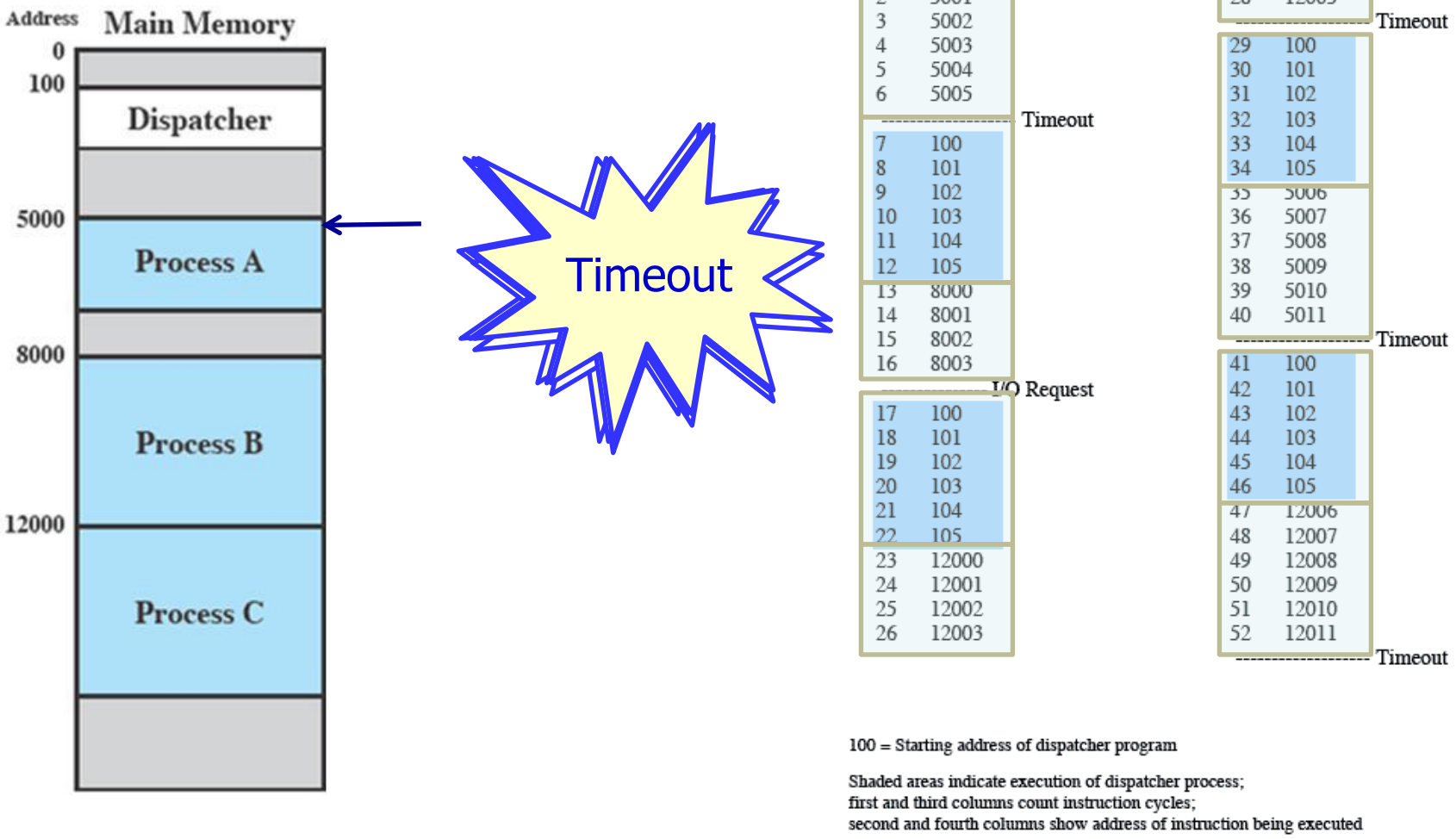
- Svaki proces se izvršava do svog završetka



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

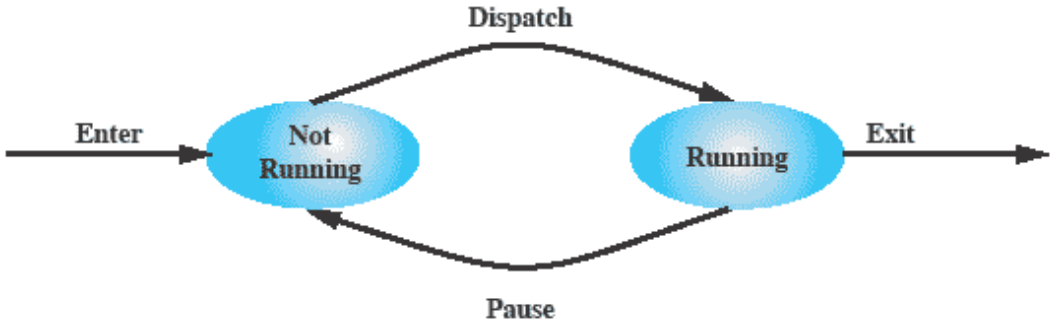
5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Praćenje procesa sa stanovišta procesora

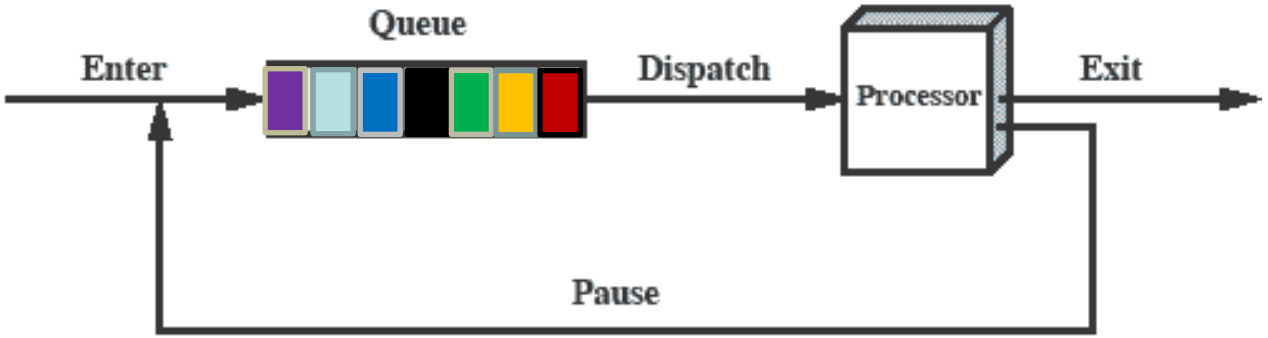


Model dva stanja procesa

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



- Red procesa
  - Procesi koji se ne izvršavaju se smeštaju u red (elementi su pokazivači na PCB), izvršavaju se podređeno vreme i vraćaju u red 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 (uzajamno blokiranje, samrtni zagrljaj, zastoj)

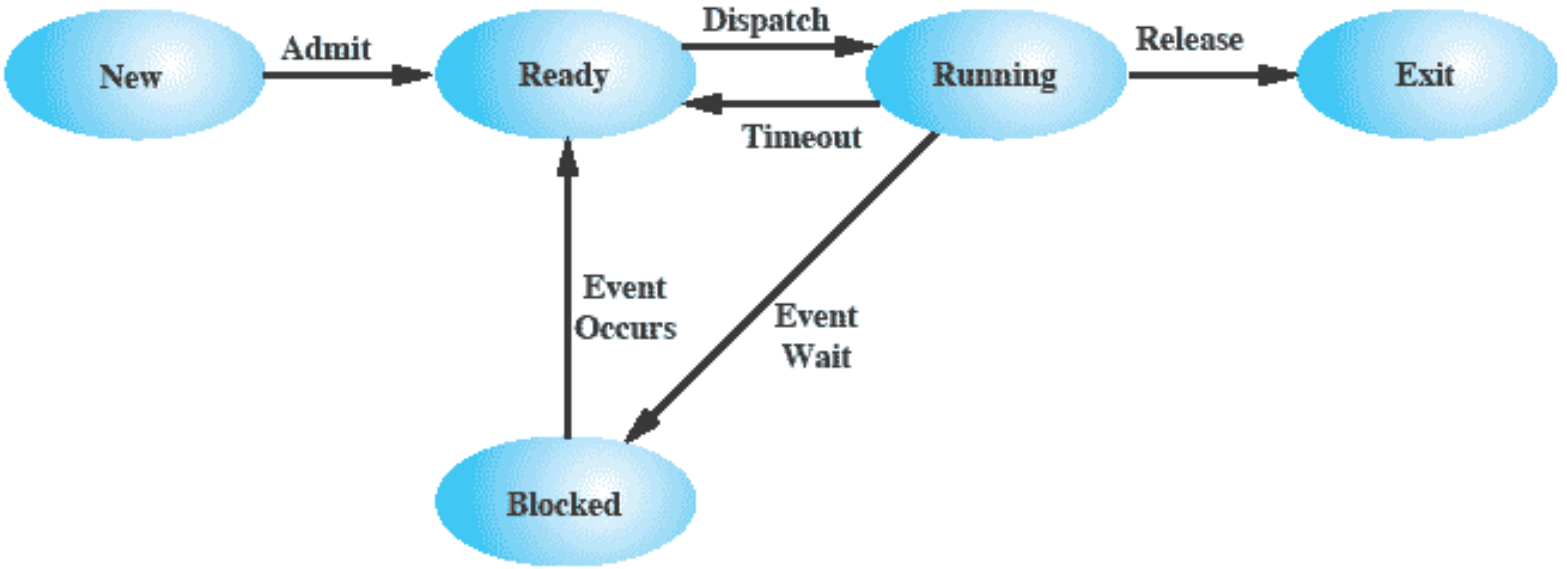
Kreiranje (stvaranje) procesa

- Glavni razlozi za kreiranje procesa
  1. Iniciranje nekog paketnog ("batch") posla
  2. Interaktivna prijava (login) korisnika
  3. Kreiran od strane OS za obezbeđenje nekog servisa
  4. Kreiran od strane postojećeg procesa
- Roditeljski (*parent*) i proces dete (*child,* potomak)
- Aktivnosti OS pri kreiranju procesa:
  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

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. Prekoračenje vremena čekanja na neki događaj
7. U/I greška, pogrešna instrukcija, privilegovana instrukcija, pogrešna upotreba podataka, itd.
8. "Ubijen" od OS ili operatera
9. Završetak roditeljskog procesa
10. Na zahtev roditeljskog procesa koji ima privilegije da završi procese decu

Model procesa sa pet stanja



- Izvršavanje (*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

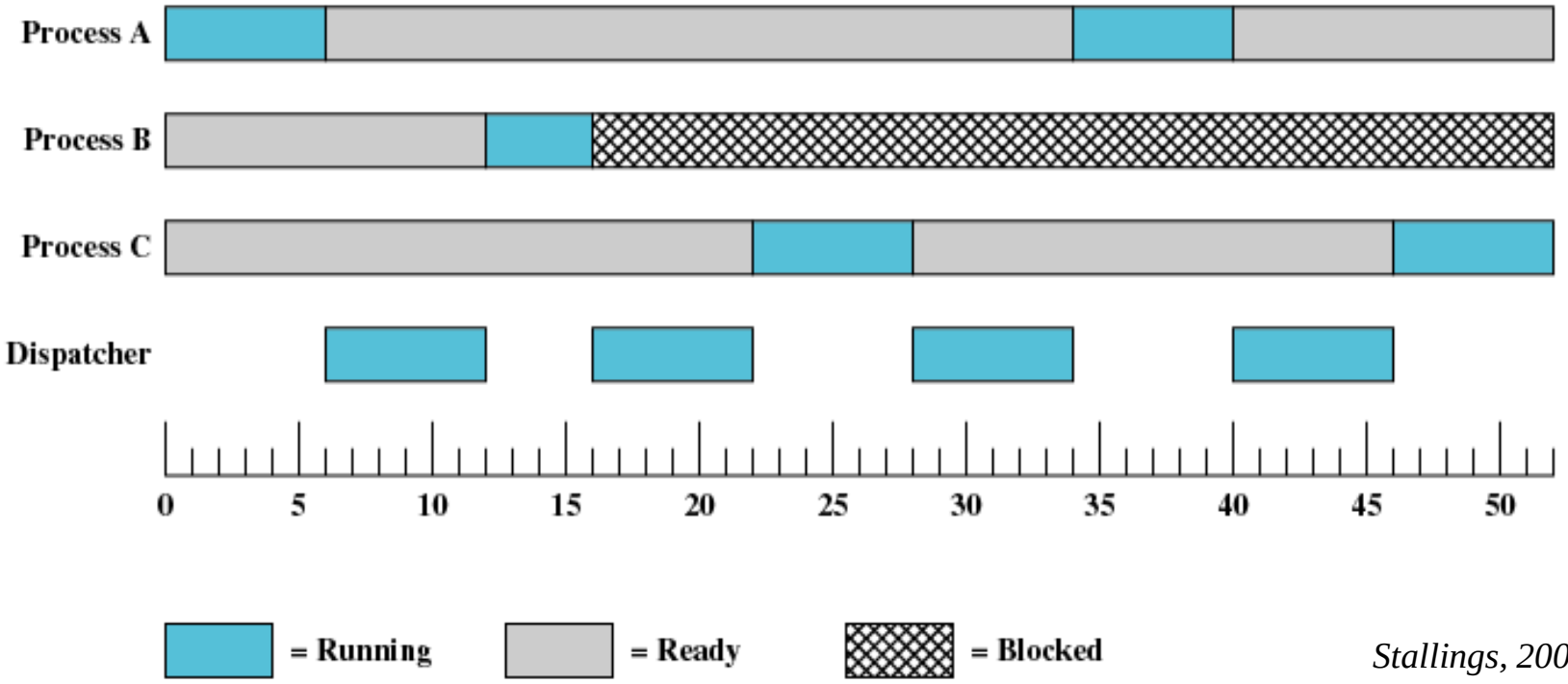
- Null → Novi
  - Kreiranje procesa za izvršenje programa
- Novi → Spreman
  - OS prebacuje proces kada ima dovoljno resursa za njegovo aktiviranje
- Spreman → Izvršavanje
  - OS bira jedan od spremnih procesa za izvršavanje na CPU
- Izvršavanje → Završen
  - Trenutno aktivan proces se završava od strane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

Prelazi između stanja procesa (2)

- Izvršavanje → 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

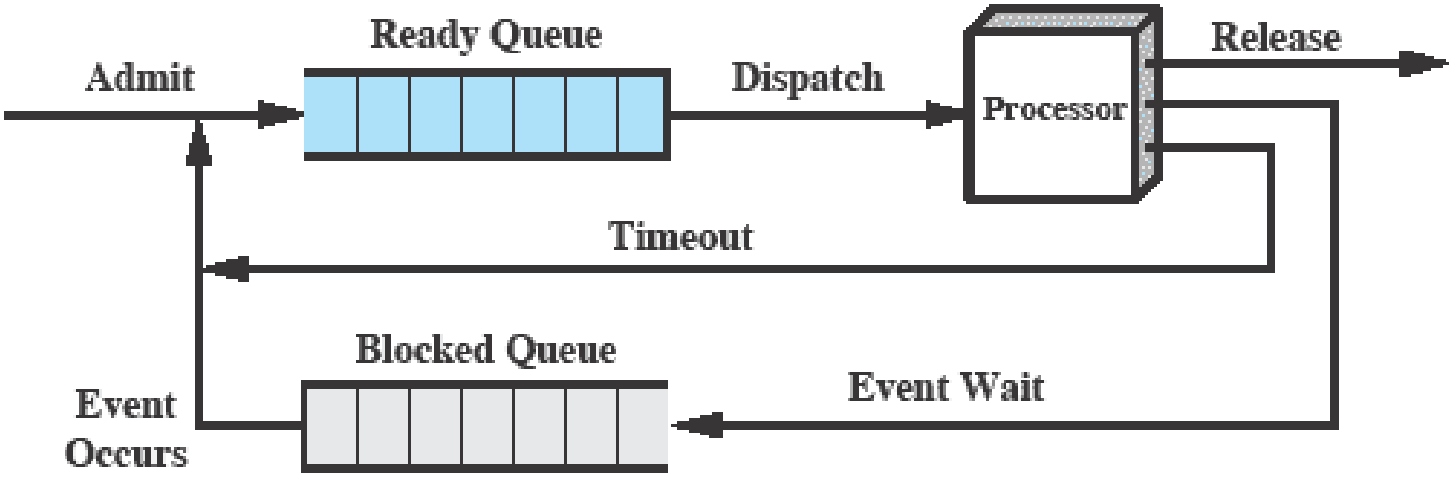
- Vremenski dijagram izvršenja procesa za prethodni primer



Stallings, 2005

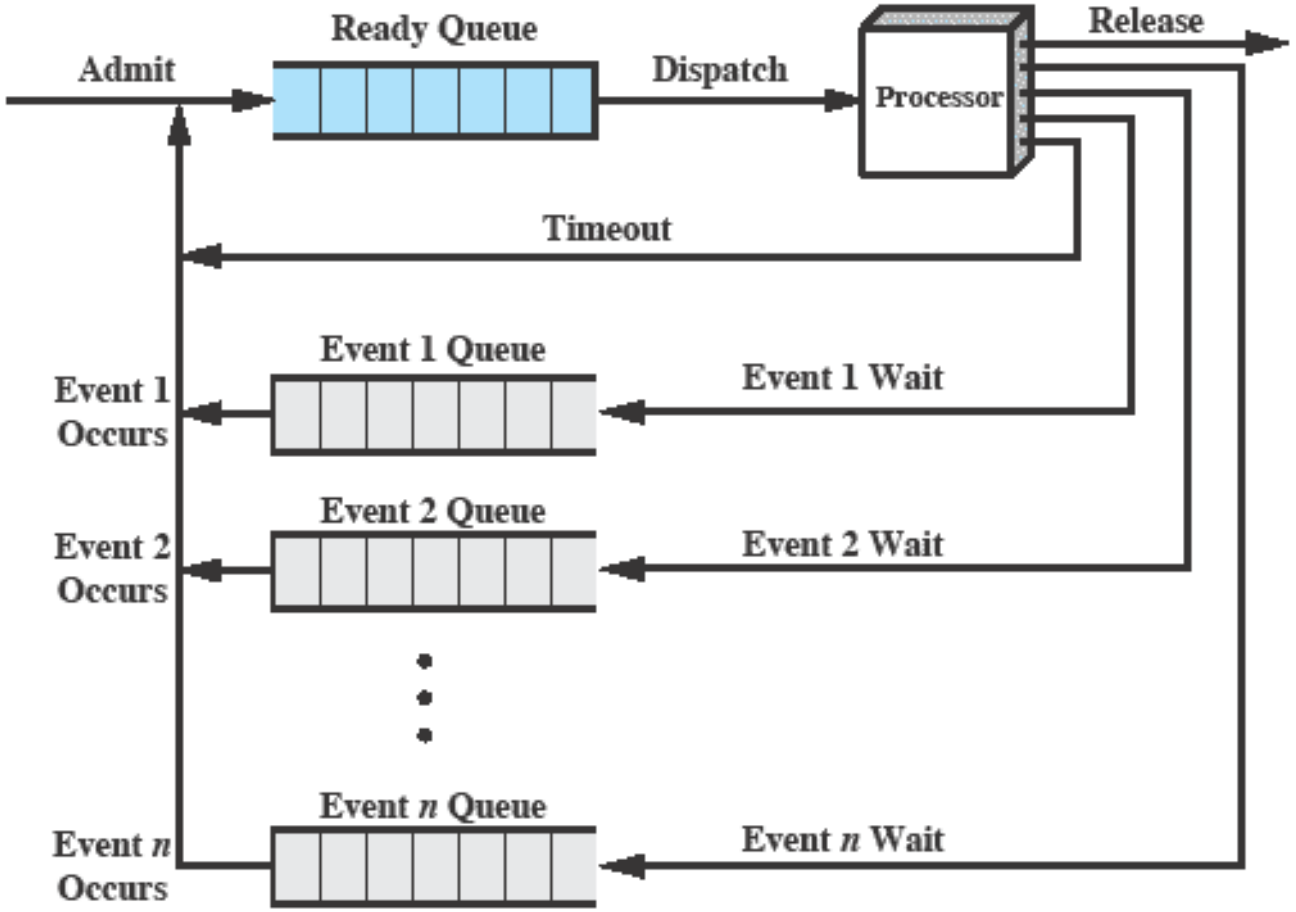
Dva reda procesa

- Jedinstveni red blokiranih procesa



(a) Single blocked queue

Višestruki redovi blokiranih



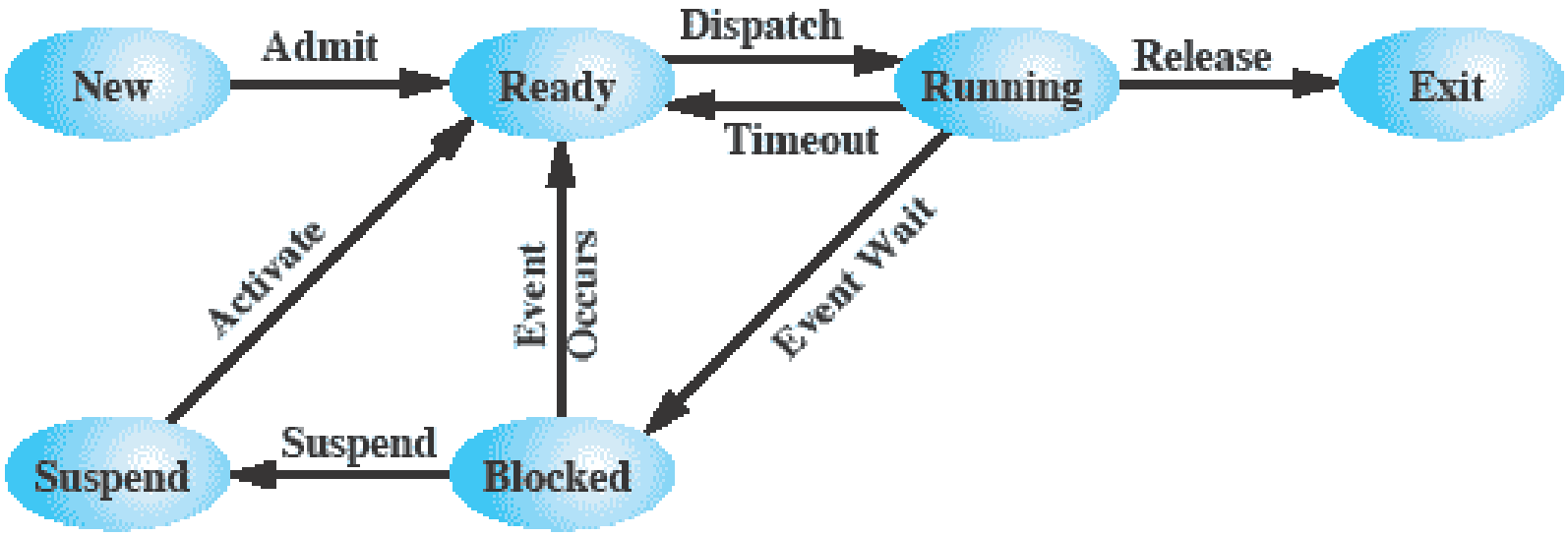
(b) Multiple blocked queues

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 iz memorije 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*)

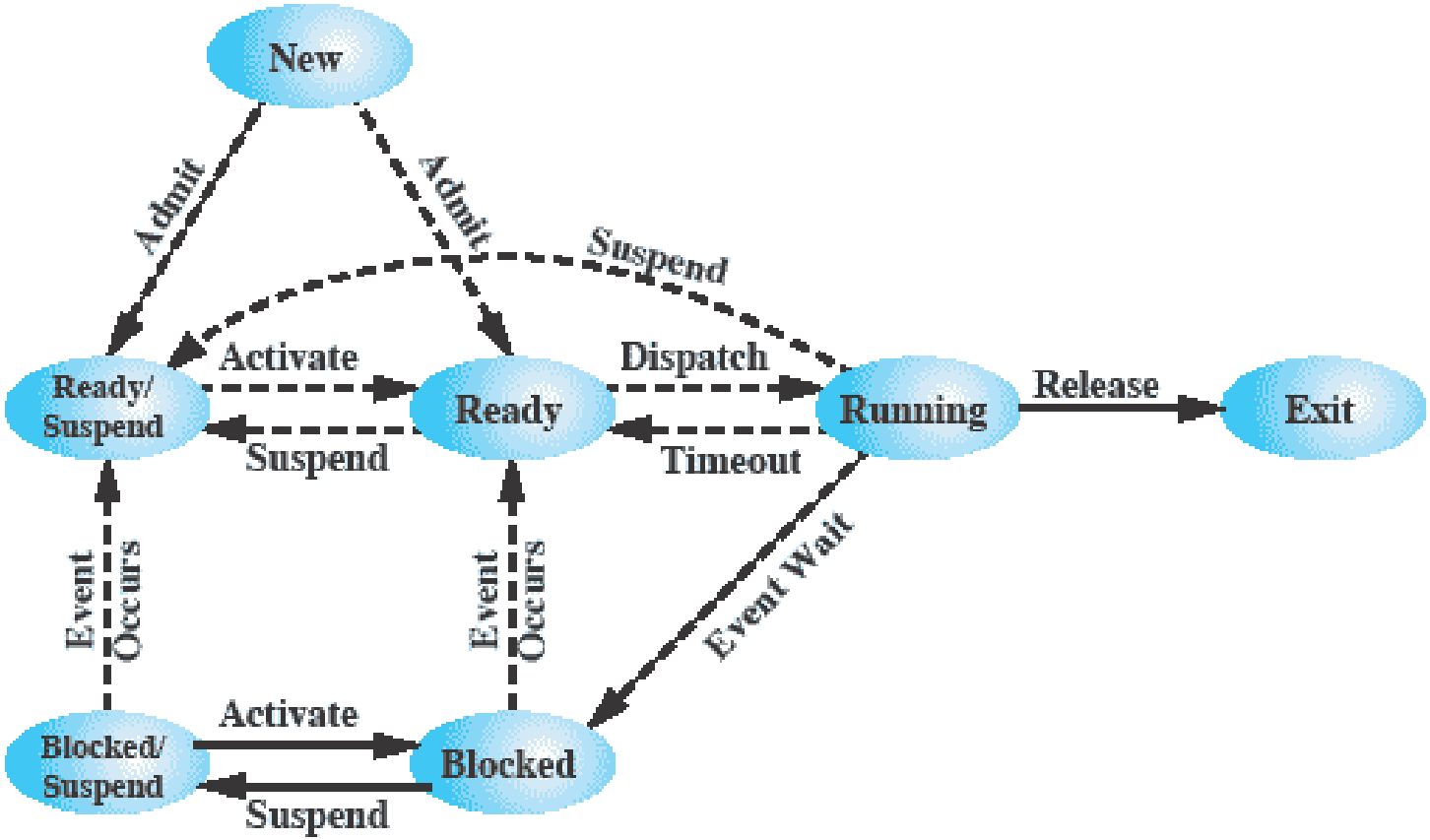
Dijagram prelaza stanja procesa

- Sa jednim suspendovanim stanjem



Dijagram prelaza stanja procesa

- Sa dva suspendovana stanja

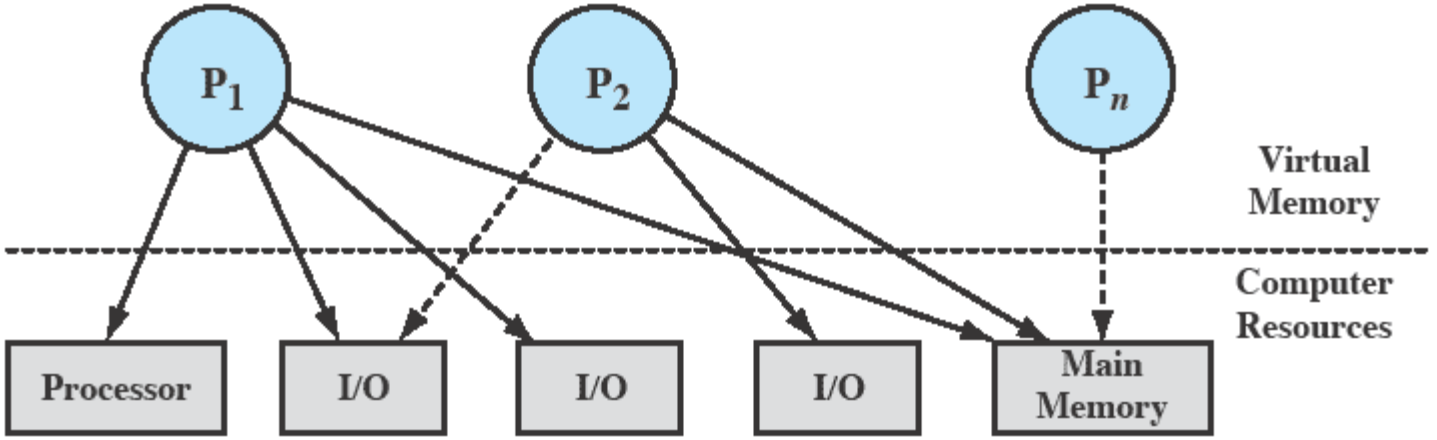


Razlozi za suspendovanje procesa

- 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 nekom 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 kreiraju se odgovarajuće tabele



Upravljačke tabele OS

- Generalna struktura upravljačkih tabela operativnog sistema

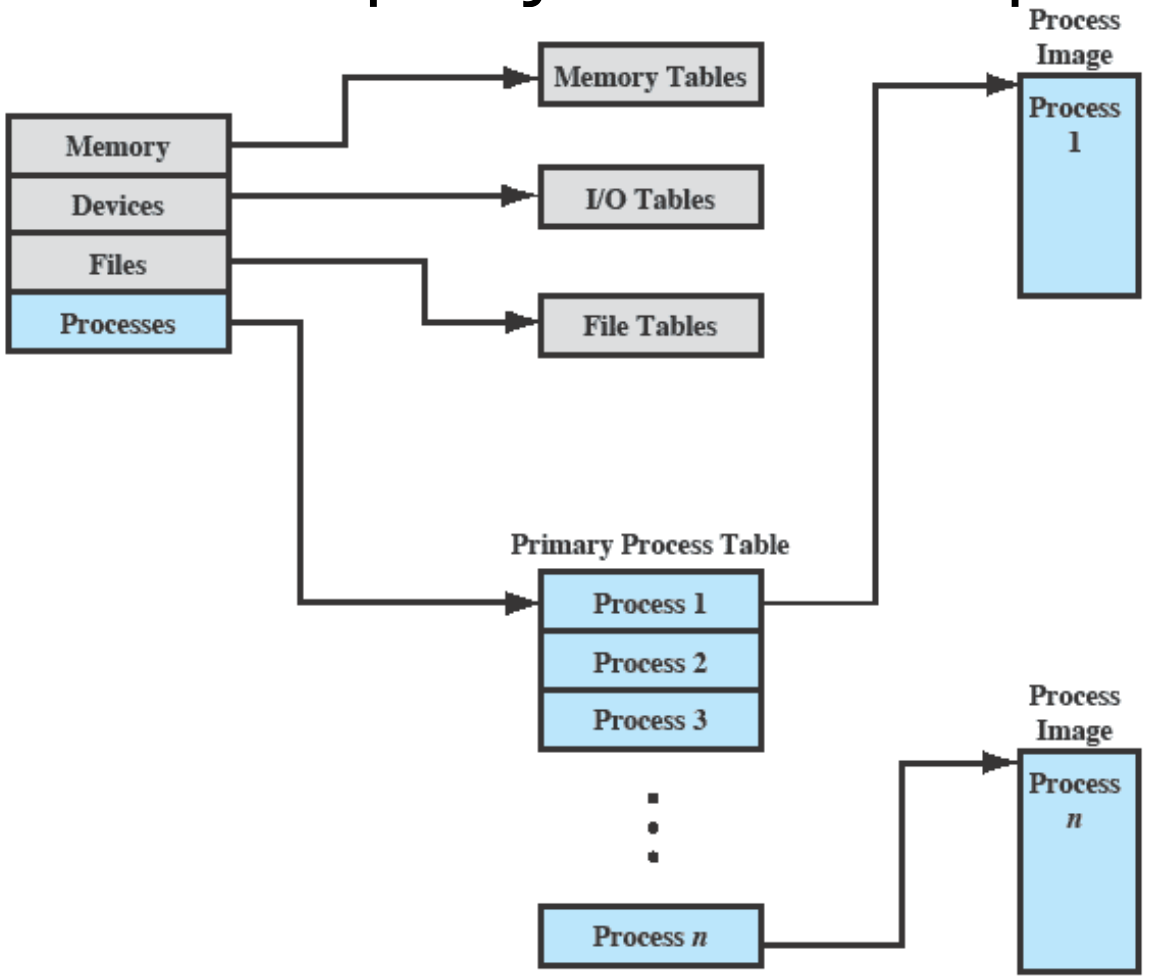


Tabela procesa

- Da bi upravljao procesima OS mora da u svakom trenutku 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, magacin (stek, *stack*), 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:
  - Identifikatori procesa
  - Informacije o stanju procesora
  - Informacije za upravljanje procesom

Identifikatori 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

Informacije o stanju 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
    - Itanium (IA-64) - psr registar ima polje od dva bita: cpl (0 najviše privilegija, 3 najmanje privilegija)

Informacije za upravljanje procesom

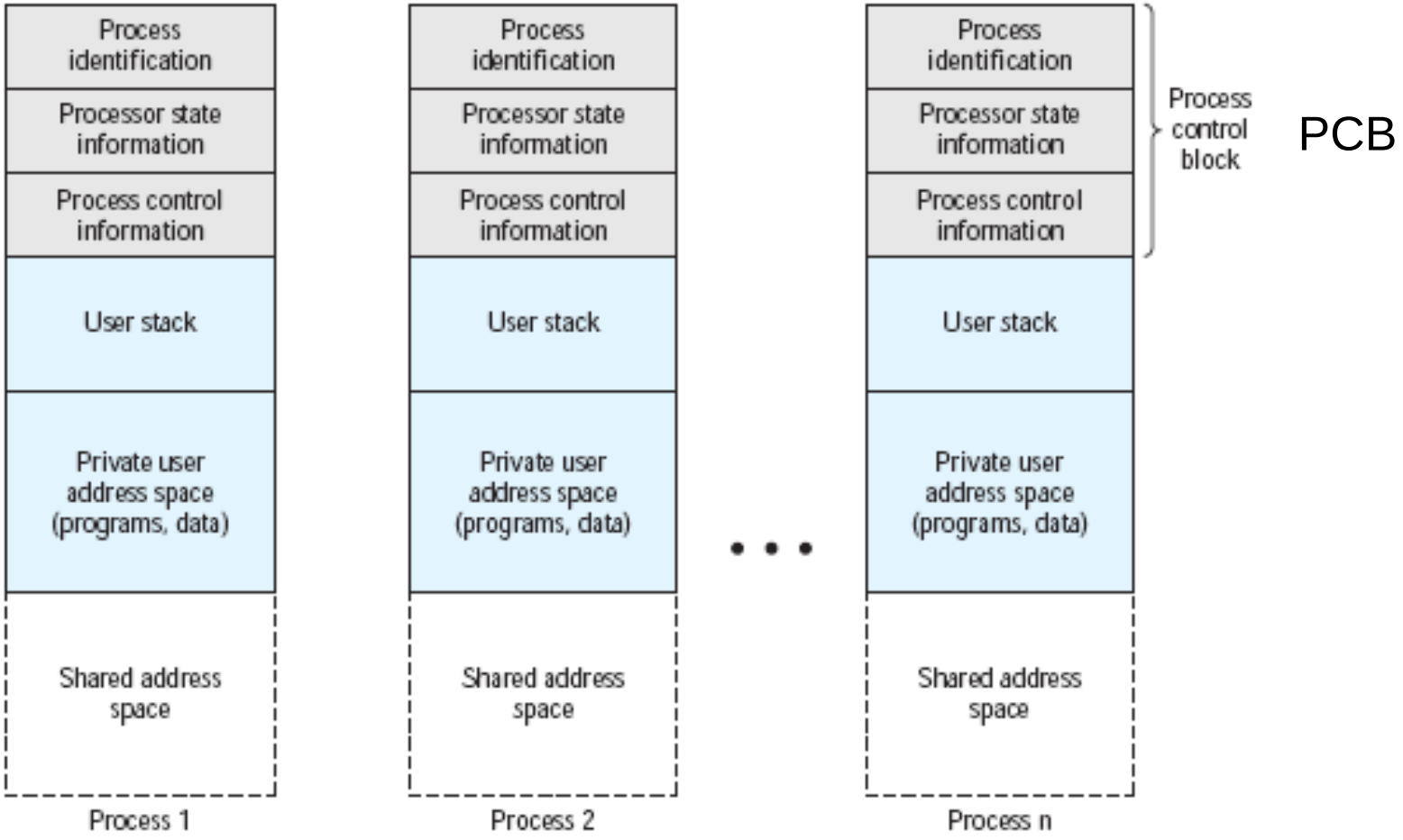
- Informacije o stanju i planiranju (*scheduling*)
  - Stanje procesa
  - Prioritet
  - Informacije vezane za planiranje – zavisno od algoritma planiranja
  - Događaj na koji proces čeka
- Struktuiranje (povezivanje, ulančavanje) PCB-a
  - Procesi (PCB) su povezani u različitim strukturama (redovima,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

Informacije za upravljanje procesom (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

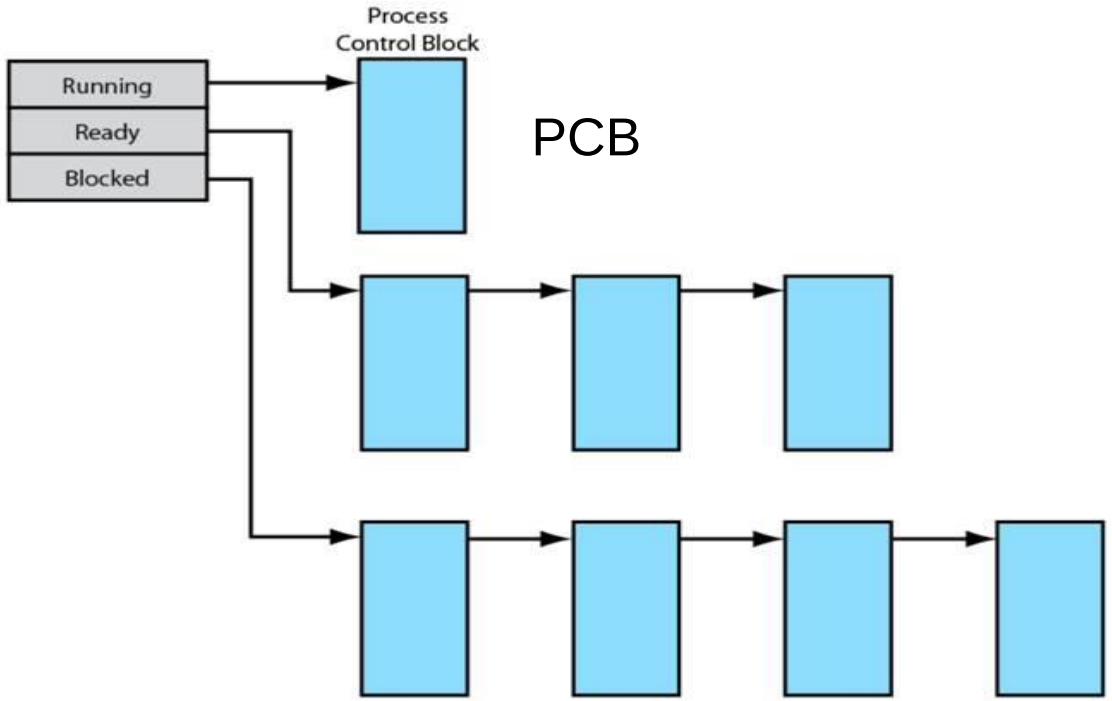
Slika procesa u virtuelnoj memoriji

- Struktura slike procesa u virtuelnoj memoriji

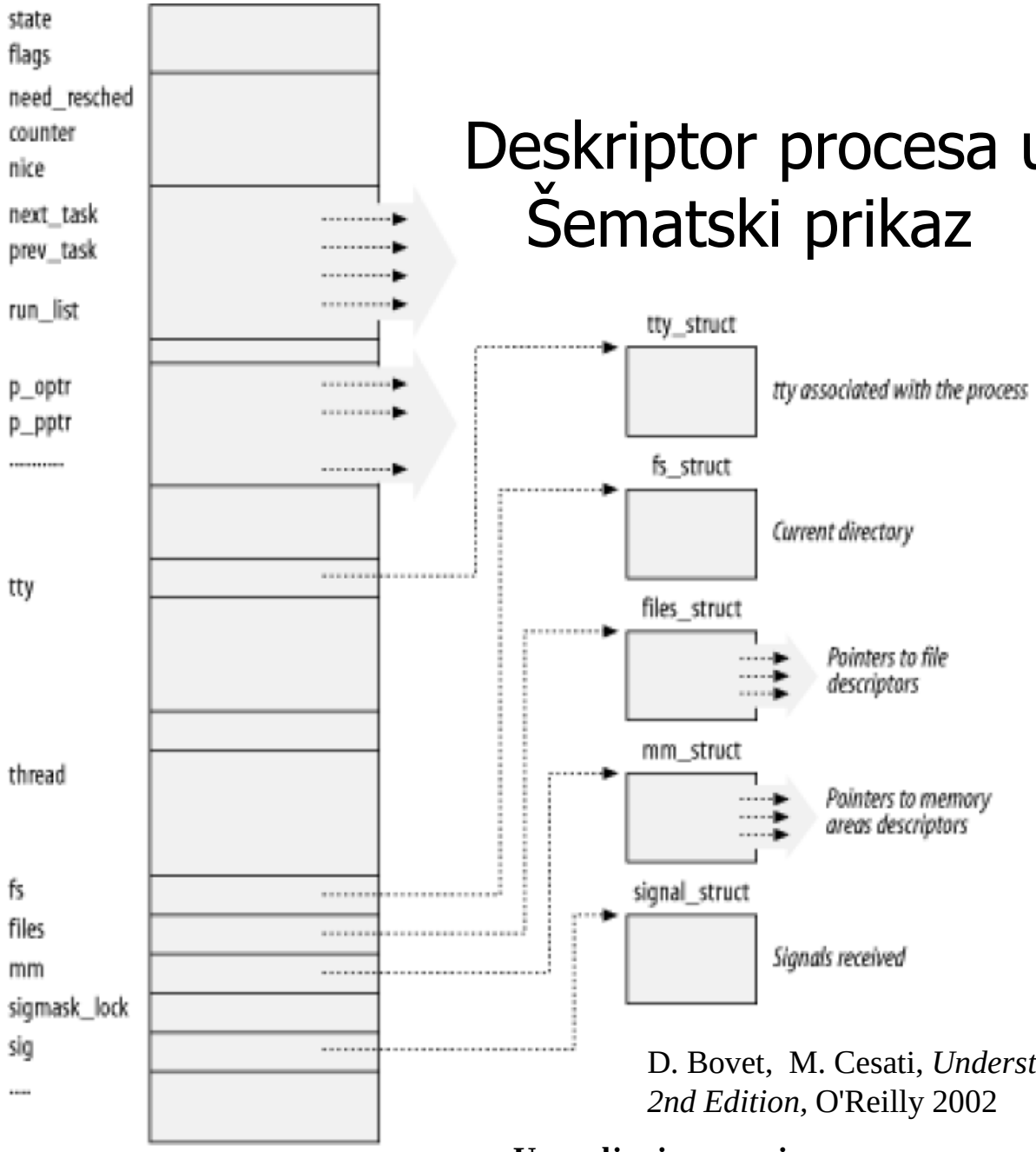


Struktura redova (listi) procesa

- PCB je najvažnija struktura podataka u operativnom sistemu (Deskriptor procesa)
- 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 tabelama 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 (2.4)
/include/linux/sched.h

```
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 (režima) izvršavanja
- Korisnički (*user*) mod (režim)
  - Režim izvršavanja sa manje privilegija – zabranjen pristup svim memorijskim adresama i izvršavanje privilegovanih (U/I) instrukcija
  - Korisnički programi se izvršavaju u ovom režimu (modu)
- Kernel (*system, supervisor*) mod (režim) – mod jezgra
  - Privilegovan režim izvršavanja
  - U ovom režimu (modu) se izvršava kernel operativnog sistema
  - Bit(-ovi) u PSW označavaju trenutni režim izvršenja
- Intel Itanium procesor IA-64 ima statusni registar procesora (*psr*) koji sadrži polje od dva bita cpl
  - Nivo 0 - najviše privilegija; nivo 3 - najmanje privilegija

Kreiranje procesa

- Prilikom kreiranja novog procesa OS treba 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 (*switch*) procesa predstavlja zamenu aktivnog procesa jednim od spremnih procesa
- Događaji koji zahtevaju izvršenje koda OS i mogu uzrokovati zamenu procesa
  - Prekid (*interrupt*) – eksterni događaj u odnosu na izvršenje procesa; prekid se javlja kao reakcija na eksterne, asinhrone događaje
    - Prekid generatora takta, U/I prekid, greška u referenci memorije
  - Trap – Odnosi se na izvršenje tekuće instrukcije procesa - ako nastane greška ili izuzetak koji je fatalan
  - Sistemski poziv (*supervisor call*) - Zahtev od strane procesa koji se izvršava za izvršenje U/I operacije (npr. pristup datoteci) kojim se aktivira odgovarajuća rutina OS.

Promena (*komutiranje*) režima

- Nakon izvršenja 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 režim, tako da rutina za obradu prekida može uključiti i privilegovane instrukcije
  - Kontekst prekinutog procesa se čuva u njegovom PCB
    - Informacije o stanju procesora
  - Izvršava se rutina za obradu prekida (*interrupt handler*)
  - Ukoliko nastanak prekida ne zahteva zamenu (*komutiranje*) procesa i promenu stanja procesa, obnavljaju se informacije o stanju procesora (kontekst) prekinutog procesa
    - U suprotnom vrši se komutiranje procesa i promena njegovog stanja
  - Prekid kloka ili poziv U/I operacije uzrokuje prebacivanje prekinutog procesa u novo stanje (Spreman, Blokiran)

Promena stanja procesa

- Zamena procesa uzrokuje promenu njegovog stanja
- Koraci koje obavlja OS pri zameni procesa:
  1. Snimiti sadržaj registara procesora, programski brojač i statusni registar procesora u PCB procesa
  2. Ažurirati PCB procesa koji je do tada bio u stanju Izvršavanje (*Running*)
  3. Premestiti PCB u odgovarajući red – Spreman, Blokiran, Spreman/suspendovan i promeniti mu stanje
  4. Izabrati sledeći proces za izvršavanje
  5. Ažurirati PCB procesa koji je izabran u stanje Izvršavanje
  6. Ažurirati stukture podataka za upravljanje memorijom
  7. Obnoviti sadržaj registara procesora na vrednosti koje su postojale u trenutku kad je proces promenjen iz stanja Izvršavanje

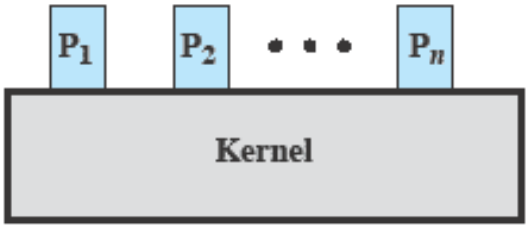
Izvršavanje operativnog sistema

- 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

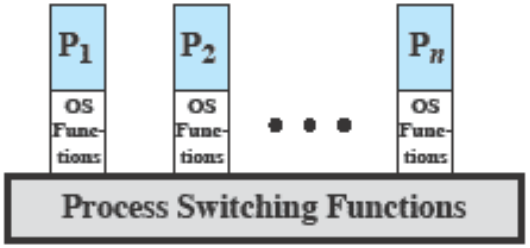
- Kernel bez procesa
  - Tradicionalni pristup u starijim OS
  - Izvršenje kernela van bilo kog procesa
  - OS kod se izvršava kao poseban entitet u glavnoj memoriji , sa svojim stekom, u kernel režimu (privilegovanom režimu)
- Izvršenje OS unutar korisničkih procesa
  - Uobičajeno u OS na PC i radnim 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 režimu 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



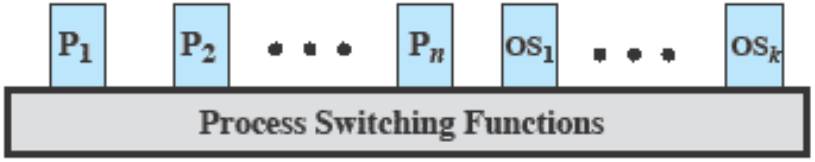
(a) Separate kernel

- Zasebno jezgro



(b) OS functions execute within user processes

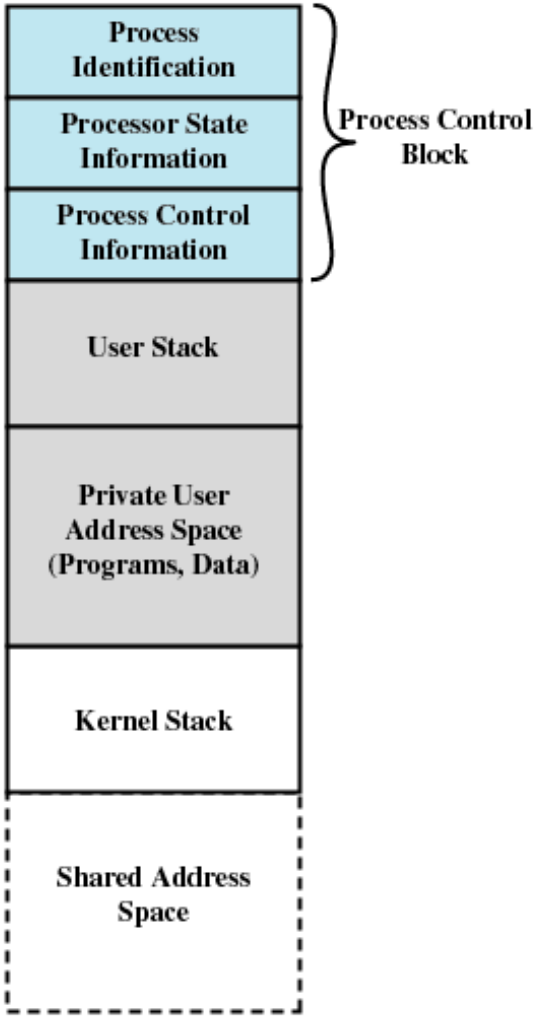
- Funkcije OS-a se izvršavaju unutar korisničkih procesa



(c) OS functions execute as separate processes

- Funkcije OS-a se izvršavaju kao zasebni procesi

Slika procesa kada se OS izvršava unutar korisničkog prostora



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

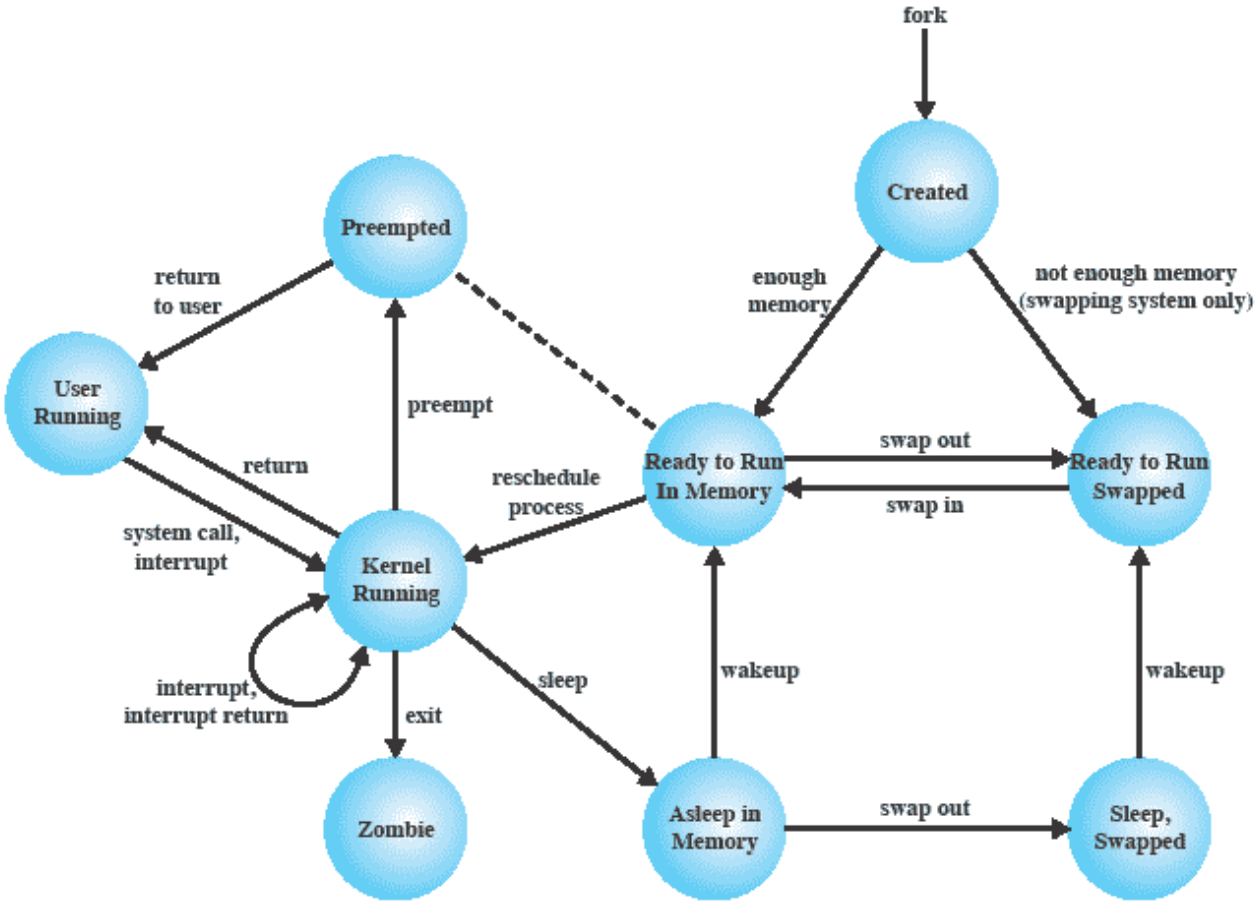
Unix SVR4

System V Release 4

- Veći deo operativnog sistema se izvršava u korisničkom procesu
- Sistemski procesi se izvršavaju isključivo u kernel režimu (modu)
- Korisnički procesi
  - Korisnički programi i funkcije se izvršavaju u korisničkom režimu.
  - U režimu kernela se izvršavaju funkcije koje pripadaju kernelu.

UNIX SVR4 - upravljanje procesima

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



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 procesa u UNIX-u 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 procesa
    - 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 (POSIX)
    - 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
  - Windows API
    - 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 (POSIX) - exit
  - Windows API - ExitProcess
- Sistemski poziv kojim jedan proces terminira drugi:
  - Unix (POSIX) - kill
  - Windows API - 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

Domaći zadatak

- The Linux Kernel archive & code reference
  - https://www.kernel.org/
  - http://lxr.free-electrons.com/
  - https://github.com/torvalds/linux
    - /include/linux/sched.h
- FreeBSD Code reference
  - https://svnweb.freebsd.org/base/stable/
  - https://github.com/freebsd/freebsd
    - /sys/sys/proc.h
- Android
  - https://source.android.com/source/index.html
  - http://androidxref.com/