# Operativni sistemi - uvod

NASTAVNIK: VELJKO PETROVIĆ (PVELJKO@UNS.AC.RS)

# Operativni sistemi - Uvod

Nastavnici:

Veljko Petrović (pveljko@uns.ac.rs)

# Administrativni detalji

# O predavaču

- Veljko Petrović
- pveljko@uns.ac.rs
- NTP 330
- Preliminarni termin konsultacija: za sada samo po dogovoru, kada se raspored ustali, u fiksnom terminu. Verovatno elektronski, kao i ova predavanja: na Teams-u.

# Asistenti

- Gorana Gojić
- Radovan Turović

# Materijali

- Udžbenik se osvežava skoro svake godine— Operativni Sistemi, Problemi i Struktura prof. Hajdukovića
- Sav materijal će biti dostupan online na repozitorijumu
- Glavni izvor za predmet je, predvidivo udžbenik.
- Računaju se *i prezentacija i predavanja sama.*

# O predavanjima

- Evidencija prisustva postoji i automatizovana je kroz platformu.
- Studenti se mole da predavanja ne ometaju.
- Nikada nije pogrešno vreme da se postavi pitanje: najbolje je da se javite virtuelnim dizanjem ruke.

# O vežbama

- Automatska evidencija pristupa
- Samostalna izrada zadataka
- Apsolutno se očekuje da se *spremite za izradu zadataka* pre svake vežbe.
- Naravno, ne pre prve, ali svake sledeće.
- Vežbe se rade samo i isključivo *pod Linux operativnim sistemom.*
- Svake godine ljudi odluče da koriste Windows i svake godine to bude problem.

# O polaganju

- Ovde su navedena okvirno sva pravila polaganja.
- Pravila se mogu promeniti usled promene forme nastave, instrukcija uprave, ili drugih neočekivanih faktora.
- Studenti će o promenama biti blagovremeno obavešteni
- U slučaju nedoumice *uvek kontaktirajte predmetnog nastavnika.*

# Formiranje ocene

| Broj bodova | Ocena |
|---|---|
| 51-60 | 6 |
| 61-70 | 7 |
| 71-80 | 8 |
| 81-90 | 9 |
| 91-100 | 10 |

# Odakle bodovi?

- Predispitne obaveze (do 70 bodova)
- Ispitne obaveze (do 30 bodova)
- Bonus bodovi
  - Do 10 sa vežbi (bodovi primenjivi samo na onih 70 bodova predispitnih obaveza)
  - Do 10 sa predavanja (bodovi primenjivi samo na onih 30 bodova parcijalnih ispita)
  - Bonus bodovi *ne mogu* pomoći da se premaši maksimum.

# Bodovi predispitnih obaveza

| Obaveza | Opis | Bodovi |
|---|---|---|
| Test T12 + Test T34 | Osnove konkurentnog programiranja. | 36 |
| Složeni oblik vežbi SOV | Problemi konkurentnog programiranja. | 34 |

# Kako položiti?

- Prolaznu ocenu možete imati samo ako važi svaki od sledećih uslova:
  - T12 + T34 + SOV ≥ 36
  - I ≥ 16
  - T12 + T34 + SOV + I ≥ 51

# Kako pasti?

- Ako imate manje od 36 bodova sa predispitnih obaveza onda su svi bodovi koje imate nevažeći i morate predmet slušati opet iduće godine u potpunosti.
- Ako imate više od 36 bodova, možete polagati ispit do idućeg semestra kada ide predmet, kada vaši bodovi prestaju da važe.

# Integralni ispit

- Održava se u ispitnom roku
- Nosi najviše 30 bodova
- Namenjen je *isključivo* studentima koji na predispitnim obavezama imaju barem 36 bodova.
- Integralni ispit obuhvata celo gradivo.
- *Mora se prijaviti ispit.*

# Integralni ispit

- Ima 4 pitanja
  - Prvo nosi 6 bodova i na zaokruživanje je.
  - Drugo nosi 10 bodova i *garantovano* je iz pitanja iz udžbenika.
    - Lista pitanja za vas i detaljan primer ispita su u PDF fajlu koji se nalazi uz ovaj.
  - Treće i četvrto nose, zajedno, 14 bodova i mogu se odnositi na bilo koje gradivo pokriveno u udžbeniku, na predavanjima, ili bilo gde.

# Prepisivanje

- Ako se utvrdi da je neko prepisivao ili koristio bilo kakva nedozvoljena sredstva na proveri znanja:
  - Dobija se 0
  - Gubi se mogućnost izrade naknadnih zadataka.

# COVID-19 mere

Dok pandemija traje, moramo se držati svih mera tokom ispita.

Ovo nije opciono niti podložno pregovoru

# Potpis

- Uslov za potpis je **36 bodova ili više** na predispitnim obavezama.

# Važenje predispitnih obaveza

- Osvojeni bodovo za predispitne obaveze važe samo do ovo doba sledeće godine, zaključno sa oktobarskim rokom. Posle se vaše prisustvo zaboravlja, i ako hoćete da položite morate slušati ceo predmet opet.

# Ritam nastave (podložan promeni)

| Nedelja | Sadržaj računarskih vežbi |
|---|---|
| 1 | - |
| 2 | Vežbanje 1 |
| 3 | Vežbanje 2 |
| 4 | Vežbanje 3 |
| 5 | Vežbanje 4 |
| 6 | Vežbanje 5 |
| 7 | Vežbanje 6 |
| 8 | Test T34 |
| 9 | Vežbanje 7 |
| 10 | Vežbanje 8 |
| 11 | Vežbanje 9 |
| 12 | SOV |

# Arhitektura Računara

LAGAN UVOD

# Što ovo?

- Vi niste ranije imali arhitekturu računara.
- Nećemo vas previše mučiti ovim, ali je bitno da znate neke stvari.
- Dosta učimo kako idemo kroz predavanja
  - Ne zaboravite da možete uvek postavljati pitanja
- No, da bi vam život bio lakši, ovo je kratak uvod u neki osnovne termine.

# Čemu služi računar?

- Danas su računari ono što se popularno zove 'konvergentni uređaji' to jest, toliko prilagodivi da su preuzeli ulogu ogromnog broja uređaja za obradu podataka, komunikaciju, itd.
- Za nas, ovde, računar je samo programabilan kalkulator: to jest, želimo da napravimo mašinu koja nam omogućava da *obrađujemo podatke.*
- Iako ovo deluje zastarelo, na nivou arhitekture svaki moderan računar je baš ovo.

# Kako radi računar?

- Apstrakcija Tjuringove mašine
- Fon-Nojmanova Arhitektura
- U osnovi:
  - Neki deo vrši transformacije nad binarno kodiranim podacima koje su izomorfne Bulovoj algebri
  - Neki deo pamti:
    - Šta treba da se uradi
    - Nad čim

# Koji su delovi računara?

- Procesor
- Memorije
  - Radne
  - Masovne
- Magistrala
- Kontroleri
- I/O uređaji

# Šta čini procesor?

- Operativna kola
- Registri
- Keš
- Komunikacione linije
- Koprocesori?

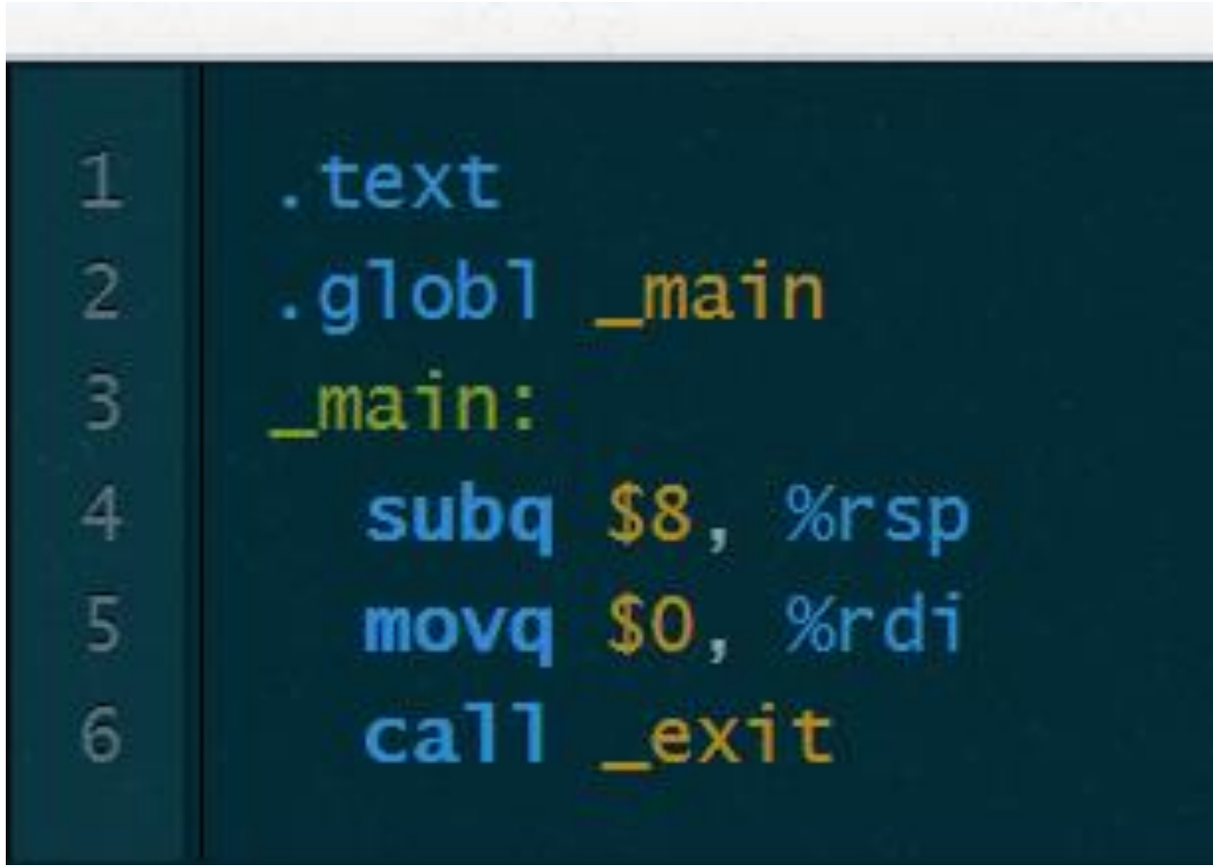
# Koji jezikom priča računar?

- Računar se na najnižem nivou programira *instrukcijama*
- Instrukcija kaže koju operaciju iz *seta instrukcija procesora* procesor treba da izvrši (opcode) i sa kojim parametrima. Na x86-64 arhitekturi to može da bude i do 15 bajta svega-i-svačega.
- Ovako je vrlo vrlo vrlo vrlo vrlo teško programirati budući da zahteva da kucamo potpuno nerazumljive brojeve.

# Asembler

- Korak iznad je programiranje gde se izdaju komande direktno procesoru, ali se one pišu u kodiranom jeziku koji kombinuje *mnemonike* za instrukcije sa kodiranim vrednostima koje specificiraju parametre.
- Asembler je program koji uzme ovakvo šifrovan kod i pretvori ga u mašinski kod pomenut ranije

# Asembler



```
1  .text
2  .globl _main
3  _main:
4     subq $8, %rsp
5     movq $0, %rdi
6     call _exit
```

# Jezici višeg nivoa

- Dugo vremena asembler je bio sve što su ljudi imali, no danas on se koristi isključivo za sistemske zadatke gde nema nikakvog drugog izbora.
- Za sve ostalo se koristi jezik koji na vrlo apstraktan način opiše operacije računara, a onda automatizovan proces (kompajler) prebaci to u mašinski jezik
- C i C++ su primeri takvih jezika
- Danas se još i koriste jezici koji se izvršavaju na tkzv. *virtuelnim mašinama* ali je princip isti.

# Linkovanje

- U praksi, kompajler samo *počne* naš posao i generiše mašinske komande (objektni kod)
- Da bi taj kod bio *izvršiv* mora da ga poveže sa svim ostalim kodom koji čini sistem funkcionalnim
- To se zove *linkovanje*
- Što nam treba sav taj drugi kod?
  - Koliko ima infrastrukture ispod softvera koji koristimo?

# Sistemski stek

- Bitna struktura za nas je nešto što se zove *sistemski stek*
- Šta je stek, u opšte? Koje operacije ima?
- Šta čine sistemski stek sistemskim?
  - %esp
  - Rast ka negativnim adresama

# Kako pričati sa spoljašnjim svetom?

- Magistrala
- I/O kroz instrukcije vs. I/O kroz mapiranje memorije
- Busy-wait vs. prekidi
- Kontroleri

# Hijerarhija vremena

- Koliko je jedna nanosekunda?
- Koje su skale vremena u jednom računaru? Koje su posledice toga?

# Uvod

UDŽBENIK — STRANICE 1-9

# Zadatak operativnog sistema

- Operativni sistem:
  - Objedinjuje raznorodne delove računara tako što *upravlja* ***procesorom, kontrolerima i RAM-om*****.**
  - Skriva od korisnika detalje funkcionisanja tako što *pretvara računar od mašine koja rukuje* ***bitima, bajtima i blokovima*** *u mašinu koja* ***rukuje datotekama i procesima***.

# Pojam datoteke

- Datoteka ima:
  - **Sadržaj** (korisničke podatke)
  - **Atribute** (npr. veličina ili vreme kreiranja) – u **deskriptoru datoteke**
- Uloga datoteke:
  - Trajno **čuvanje podataka**.
  - Pristup podacima je **čitanje** i **pisanje** (kojima predstoji **otvaranje** i nakon kojih sledi **zatvaranje** datoteke).
  - Što datoteke zatvaramo?

# Pojam procesa

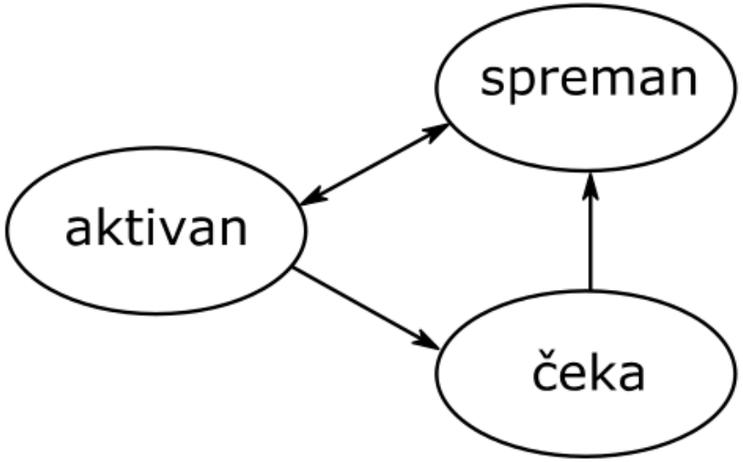
- **Aktivnost procesa** – angažovanje procesora na izvršavanju korisničkog programa.
- **Slika procesa** – adresni prostor procesa (**naredbe**, **stek** i **podaci**).
- **Atributi** – stanje, prioritet – čuvaju se u **deskriptoru procesa**.

# Stanje i prioritet procesa 1/2

- Tipična stanja procesa su: “**aktivan**”, “**čeka**” i “**spreman**”.
- Prioritet procesa određuje kada je proces aktivan:
  - Uvek je **aktivan proces** sa **najvišim** prioritetom.
  - Ako postoji **nekoliko** procesa sa **najvišim** prioritetom, vrši se **raspodela procesorskog vremena** između njih uz pomoć mehanizma **kvantuma**.
  - Isticanje kvantuma regulišu **prekidi sata**.

# Stanje i prioritet procesa 2/2

- Aktivan proces prelazi u stanje “**čeka**” kada je nemoguć nastavak njegove aktivnosti (npr. **UI** radnja). Nakon tog čekanja prelazi u stanje “**spreman”**.



# Stanje i prioritet procesa, praktična ilustracija

```
top - 00:50:33 up 3 min,  1 user,  load average: 0,54, 0,46, 0,20
Tasks: 199 total,   1 running, 198 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0,3 us,  0,3 sy,  0,0 ni, 99,4 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem :  16040,3 total,  13192,8 free,    285,7 used,   2561,8 buff/cache
MiB Swap:    947,2 total,    947,2 free,      0,0 used.  15445,7 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1984 veljko    20   0  572912  78812  34644 S   2,0   0,5   0:02.10 Xorg
 3052 veljko    20   0  599928  38388  28120 S   2,0   0,2   0:01.47 gnome-terminal-
16863 veljko    20   0   37616   4028   3300 R   0,7   0,0   0:00.17 top
    1 root      20   0  195116   9444   6660 S   0,3   0,1   0:06.71 systemd
 2119 veljko    20   0  118468   2204   1824 S   0,3   0,0   0:00.15 VBoxClient
    2 root      20   0       0      0      0 S   0,0   0,0   0:00.00 kthreadd
    3 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_gp
    4 root       0 -20       0      0      0 I   0,0   0,0   0:00.00 rcu_par_gp
```

# Uloga procesa

- Procesi omogućuju **bolje iskorišćenje** računara (procesora) i njegovu **bržu reakciju** na dešavanje spoljnih događaja (npr. unos teksta).
- Istovremeno postojanje više procesa omogućuje da se procesor preključi sa **aktivnog** procesa na **spreman** proces kada **aktivan** proces prelazi u stanje “**čeka**”.
- Dobar primer ovakvog ponašanja je čekanje **hitnog** procesa na spoljni događaj (npr. unošenje teksta sa tastature).

# Pojam niti

- Redosled naredbi programa (procesa) naziva se **trag (trace)** procesa.
- Proces je **sekvencijalan** ako je njegov trag poznat u **vreme programiranja**.
- Trag sekvencijalnog procesa naziva se **nit (thread)** koja **povezuje** izvršavane naredbe u **redosledu** njihovog izvršavanja.

# Mana sekvencijalnih procesa

- Mana sekvencijalnih procesa je da su **neosetljivi** na **spoljne** događaje (npr. editovanje teksta).
- Za editovanje su potrebne **dve** radnje (**interakcija** sa korisnikom i **čuvanje** unešenog teksta).
- Sekvencijalan editorski proces izvršava te dve radnje jednu za drugom:

```
...
for(;;) {
do_editor_command();
if(time_to_save_data())
save_data();
}
...
```

# Nesekvencijalan editorski proces (odvojene radnje)

```
...
for(;;) {
      do_editor_command_in_foreground();
}
...
for(;;) {
     if(time_to_save_data())
     save_data_in_background();
}
...
```

# Nesekvencijalan editorski proces (odvojene radnje)

- **Prioritetnija** radnja je posvećena **interakciji** sa korisnikom, a **manje prioritetna** pozadinska radnja je posvećena **čuvanju teksta**.
- Pod pretpostavkom da je **hitna** radnja zaustavljena, jer nema komandi od korisnika, **pozadinska** radnja može da se odvija sve dok, na primer, spoljni događaj poput **pritiska dirke** na tastaturi ne najavi početak interakcije sa korisnikom.

# Nesekvencijalan editorski proces (odvojene radnje)

- Tada se **zaustavlja pozadinska** radnja, radi nastavljanja **hitne** radnje.
- Kada se obavi korisnička komanda u okviru hitne radnje, a **hitna radnja se zaustavi** u očekivanju nove komande, **nastavlja se pozadinska radnja**.
- Zahvaljujući **preplitanju** hitne i pozadinske radnje, u toku editiranja **nema perioda bez odziva**.
- Podrazumeva se da opisanom nesekvencijalnom editorskom procesu odgovaraju **dve niti**.

# Nesekvencijalan editorski proces (odvojene radnje)

- Da bi opisano rukovanje nitima bilo moguće, **prioritet**, **stanje** i **stek** se ne vezuju za **proces**, nego za njegove **niti**.
- Znači svaka nit procesa ima svoj **prioritet**, svoje **stanje**, svoj **stek**, pa i svoj **deskriptor**.
- Za niti istog procesa se podrazumeva da **nisu potpuno nezavisne**, odnosno da **sarađuju razmenom podataka**.
- Tako, u slučaju **nesekvencijalnog** editorskog procesa, manje prioritetna nit se brine o **čuvanju** teksta koga **pripremi** prioritetna nit.

# Konkurentni procesi

- Procesi sa **više niti** nazivaju se **konkurentni** procesi (**konkurentni programi**).
- Podrazumeva se da samo jedna od niti može biti “**aktivna**”, dok su ostale u stanju “**spremna**” ili “**čeka**”.
- Preključivanje procesora sa jedne niti na drugu, uzrokuju **redosled izvšavanja koji nije određen u vreme programiranja**, što znači da je izvršavanje konkurentnih procesa u opštem slučaju **stohastično** zbog stohastične prirode dešavanja spoljnih događaja.

# Struktura operativnog sistema

- Zadatak operativnog sistema je da upravlja fizičkim i logičkim delovima računara u okviru svog **jezgra** (**kernela**).
- **Fizičkim** delovima upravljaju moduli za rukovanje:
  - Procesorom
  - Kontrolerima
  - Radnom memorijom
- **Logičkim** delovima upravljaju moduli za rukovanje:
  - Datotekama
  - Procesima

# Modul za rukovanje procesorom

- Zadatak ovog modula je **prekljućivanje** jedne niti na drugu.
- Moguće je prekljućivanje na niti u **istom procesu** ili u **različitim procesima**.
- Prekljućivanje procesora između niti **istog procesa** je **brže** nego prekljućivanje niti u okviru **različitih procesa** zato što se niti **istog procesa** nalaze u **istom adresnom prostoru**.
- Ovaj modul uvodi **operaciju prekljućivanja**.

# Modul za rukovanje kontrolerima

- Zadatak ovog modula je upravljanje **ulaznim** i **izlaznim** uređajima koji su zakačeni za kontrolere.
- Modul se sastoji od niza komponenti nazvanih **drajveri**.
- Cilj drajvera jeste da uređaje predstavi u **apstraktnom** obliku sa **jednoobraznim** i **pravilnim** načinom korišćenja (primer drajvera diska).
- Drajveri uvode operacije **ulaza** i **izlaza**, u okviru kojih se rukuje **preključivanjem (zaustavljanjem)** niti koja je zatražila operaciju.
- Drajveri takođe rukuju i **obradom prekida** kao reakcijom na javljanje uređaja putem mehanizma **prekida**.

# Modul za rukovanje radnom memorijom

- Zadatak ovog modula je da vodi evidenciju o **slobodnoj** radnoj memoriji radi **zauzimanja** i **oslobađanja**.
- U slučaju da podržava **virtuelnu memoriju**, ovaj modul se brine i o prebacivanju sadržaja između **radne** i **masovne memorije**.
- Ovaj modul uvodi operacije **zauzimanja** i **oslobađanja**.

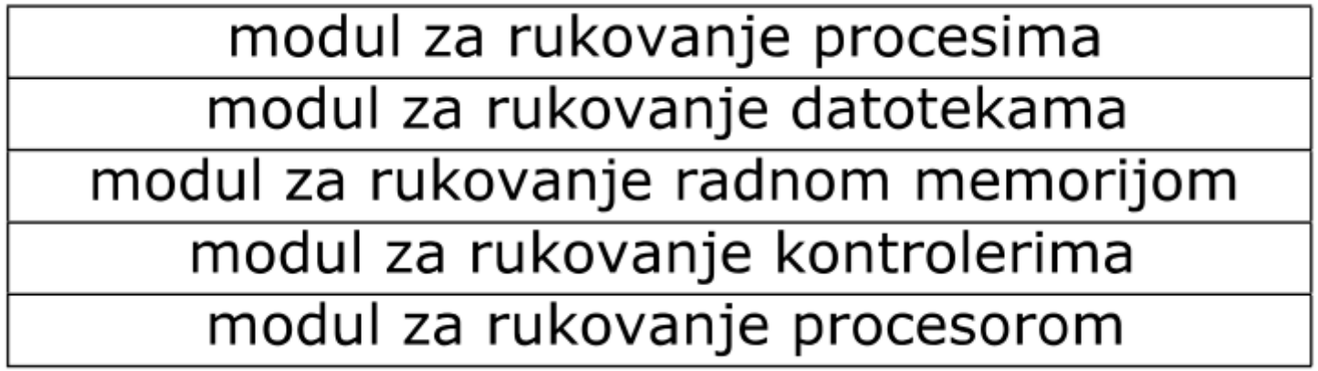
# Modul za rukovanje datotekama

- Zadatak modula za rukovanje datotekama je da omogući **otvaranje** i **zatvaranje** datoteka, kao i **čitanje** i **pisanje** njihovog sadržaja.
- Ovaj modul vodi evidenciju o **blokovima (HDD)** u kojima se nalaze sadržaji datoteka.
- Takođe ovaj modul vodi računa o **prebacivanju sadržaja između radne** i **masovne memorije** uz pomoću operacija **čitanja (ulaza)** i **pisanja (izlaza)**, kao i **bafera** potrebnih za smeštanje sadržaja (modul za rukovanje memorijom).
- Pored ovog modul uvodi i operacije **otvaranja** i **zatvaranja.**

# Modul za rukovanje procesima

- Zadatak ovog modula je da omogući **stvaranje** i **uništavanje procesa**, kao i **stvaranje** i **uništavanje** njihovih **niti**.
- Na ovaj način se uvodi **višeprocesni** i **višenitni** režim rada koji omogućava:
  - **Bolje iskorišćenje procesora**
  - **Podršku većeg broja korisnika**
  - **Bržu reakciju na spoljne događaje**
- Modul za rukovanje procesima poziva i operacije drugih modula (**za upravljanje datotekama i memorijom**).
- Modul za rukovanje procesima uvodi operacije **stvaranja** i **uništavanja** (**procesa** i **niti**).

# Slojeviti operativni sistem




- Za razliku od **slojevitog** operativnog sistema u praksi se uglavnom sreću **monolitni** operativni sistemi, koji **nemaju hijerarjhijsku strukturu** jer saradnja **nije ograničena** kao kod slojevitog OS.

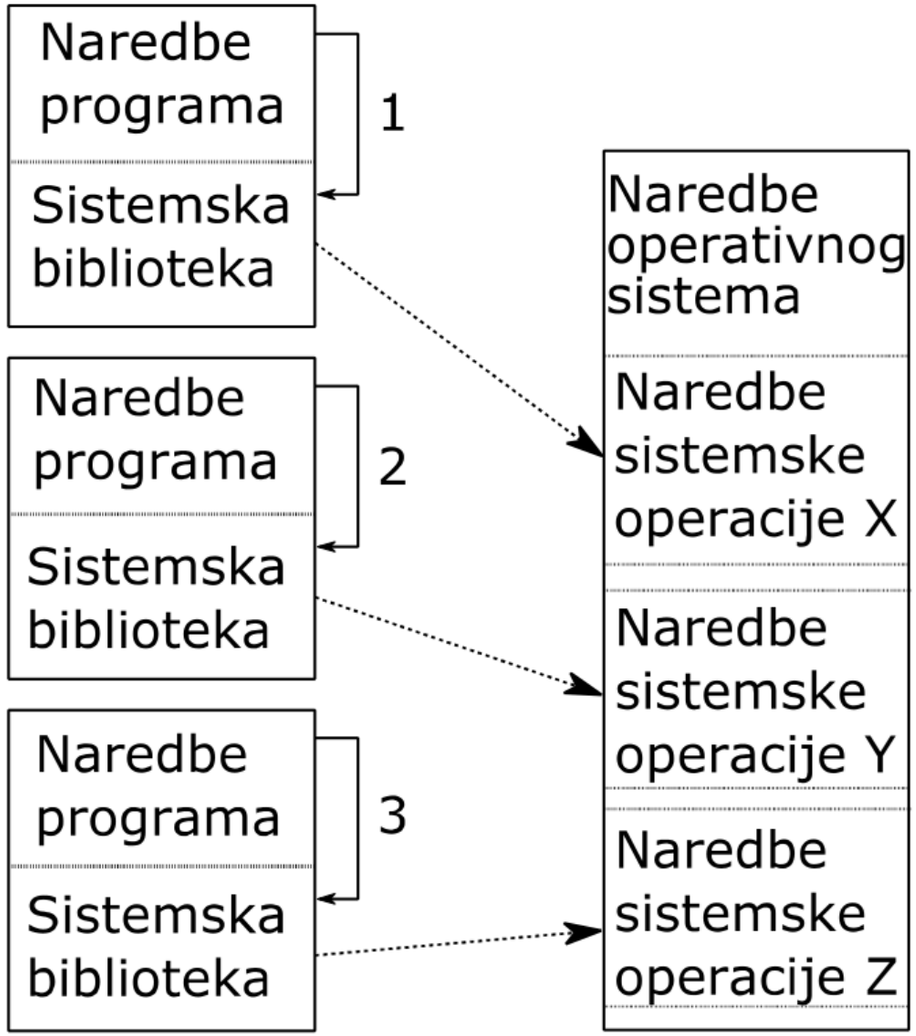
# Sistemski pozivi

- Svaki proces se nalazi u **korisničkom sloju** (gornjem sloju OS), i poseduje poseban adresni prostor koji se naziva **korisnički prostor** (**user space**).
- Operatvni sistem poseduje poseban adresni prostor koji se naziva **sistemski prostor** (**kernel space**).
- Zbog razdvojenosti ova dva prostora neophodno je uvođenje **sistemskih poziva, radi poziva operacija OS.**
- Sistemski pozivi zahtevaju korišćenje **ASM** naredbi i sakrivaju se unutar **sistemskih potprograma (sistemskih operacija)**.
- Svaki proces u sistemskom prostoru ima svoj **sistemski stek.**
- Sistemski potprogrami obrazuju **sistemsku biblioteku**.

# Sistemski pozivi

- Zahvaljujući **sistemskim potprogramima**, odnosno **sistemskoj biblioteci**, operativni sistem predstavlja **deo korisničkog programa**, iako za njega nije direktno linkovan.
- Istovremeno **postojanje više procesa** i **nepredvidivost preključivanja**, uzrokuje da je moguće da istovremeno postoji **više procesa**, koji su **započeli**, a **nisu završili** svoju aktivnost u okviru **operativnog sistema**, odnosno, čija aktivnost je **zaustavljena unutar sistemskih operacija** operativnog sistema.

# Preplitanje izvršavanja tri sistemske operacije



# Praktična ilustracija sistemskih poziva

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main(){
    unsigned char buff[32];
    ssize_t count = 0;
    int fd = open("open.c", O_RDONLY);
    while(count = read(fd, buff, 31)){
        buff[count] = '\0';
        printf("%s",buff);
    }
    close(fd);
}
```

# Interakcija korisnika i OS

- **Komande** komandnog jezika omogućavaju korišćenje OS na **interaktivnom** nivou.
- Za interpretiranje i preuzimanje komandi komandnog jezika zadužen je poseban proces iz korisničkog sloja koji se zove **interpreter komandnog jezika** (**shell**).
- Interpreter komandnog jezika koristi OS na programskom nivou, jer u toku svog rada poziva **sistemske operacije**.