Pregled operativnih sistema

Operativni sistemi 1

Institut za matematiku i informatiku
Prirodno-matematički fakultet, Kragujevac

doc. dr Miloš Ivanović
prof. dr Dragan Stojanović

Oktobar 2026. god.

O čemu će biti reči?

1 Ciljevi i funkcije OS-a

2 Istorijat razvoja OS-a

3 Glavna dostignuća razvoja OS-a

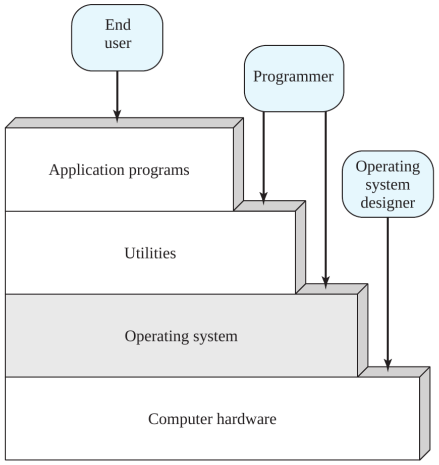
4 Glavni pravci razvoja savremenih OS

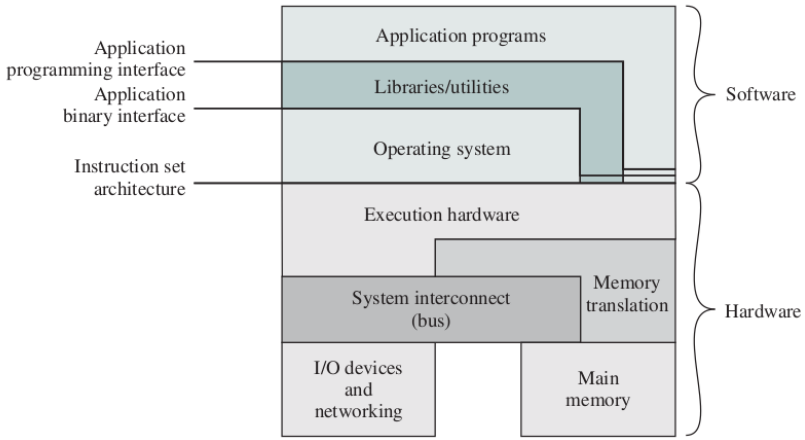
5 Primeri operativnih sistema

Ciljevi operativnog sistema

1. **Praktičnost** - OS čini računar udobnijim za korišćenje
2. **Efikasnost** - pomoću OS-a se računarski resursi efikasnije iskorišćavaju
3. **Mogućnost evolucije** - OS treba da je konstruisan na taj način da dozvoljava gradacijsko uvođenje novih usluga bez ometanja postojećih

Osnovni nivoi računarskog sistema

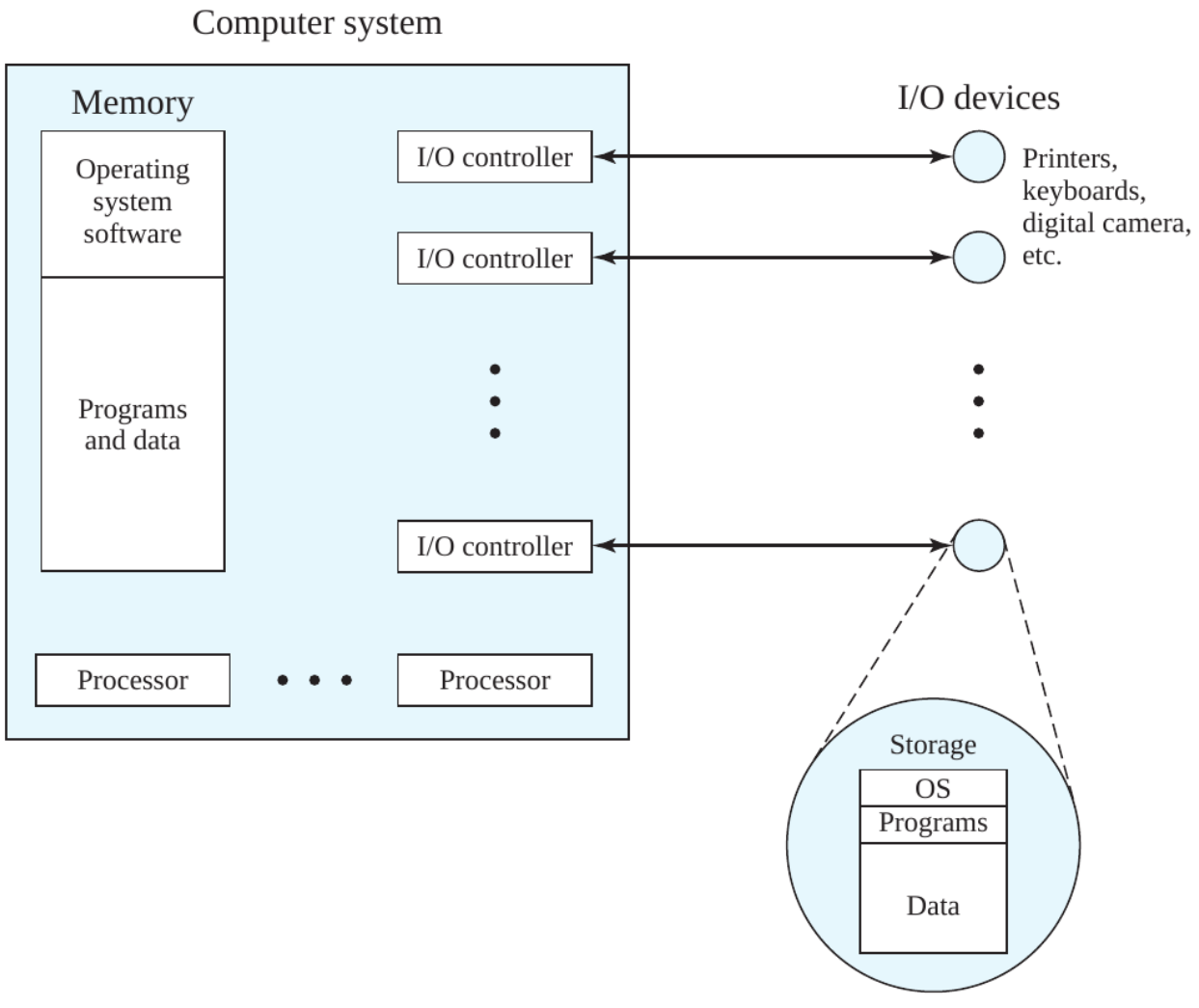




Usluge operativnog sistema

1. **Razvoj programa** - Obično pomoćni programi nisu deo jezgra, ali se isporučuju uz OS
2. **Izvršavanje programa** - zahteva izvođenje niza operacija
3. **Pristup U/I uređajima** - obezbeđivanje jednoobraznog interfejsa prema raznorodnim uređajima
4. **Kontrolisan pristup fajlovima** - priroda uređaja, organizacija fajlova, kontrola pristupa, ACL liste
5. **Pristup sistemu** - naročito bitan kod višekorisničkih sistema
6. **Obrada grešaka i odgovori** - softverske i hardverske greške, rešavanje izuzetaka
7. **Vođenje evidencije** - prikupljanje, logovi, statistička obrada radi predviđanja stanja isl.

OS kao osnovni upravljač resursima računarskog sistema



OS kao osnovni upravljač resursima računarskog sistema

Kernel

OS predaje upravljanje CPU da izvrši neki koristan rad, a zatim preuzima upravljanje dovoljno dugo da bi pripremio CPU za sledeći deo posla. U memoriji se uvek nalazi kernel, kao i drugi delovi OS-a koji se trenutno koriste.

Pokretači evolucije operativnih sistema

1. **Razvoj hardvera** - grafički terminali, prozori, straničenje, mobilni uređaji...
2. **Nove usluge** - zahtevi korisnika, novi trendovi (*cloud computing*) isl.
3. **Ispravke grešaka** - *patch* ispravlja *bug*, ali donosi nove *bug*-ove ;)

Serijska obrada

Period 1946.-1956.

- Ne postoji operativni sistem
- Programi u mašinskom jeziku se učitavaju pomoću ulaznog uređaja (čitač bušenih kartica)
- Svetlosni signal greške ili ispravan izlaz na štampaču

Glavni problemi serijske obrade

1. **Raspoređivanje vremena** - Operater je zakazivao vreme za računarom koje je moglo biti neiskorišćeno ili nedovoljno
2. **Vreme uspostavljanja posla** - obično duže od vremena pravog računanja

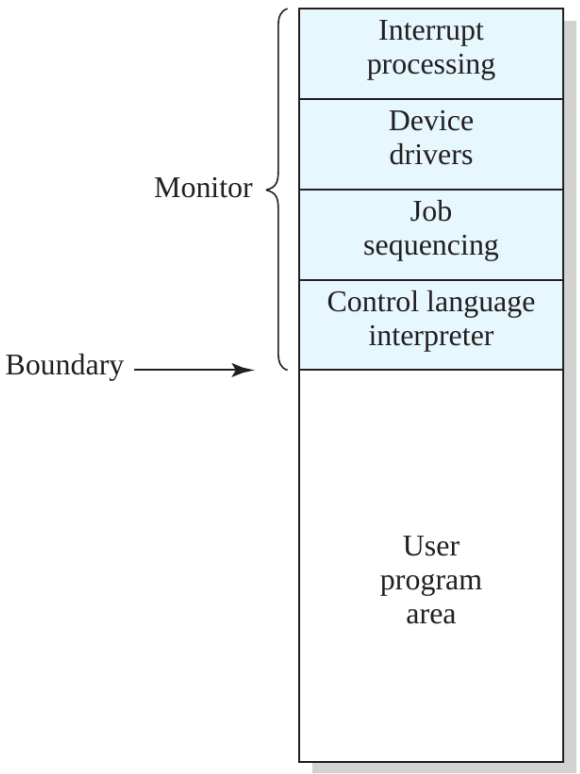
Prosti sistemi paketne obrade

Sredina 1950-ih

- Operativni sistem IBSYS na IBM 7090/7094 računarima
- Uvodi se poseban program, tzv. **Monitor**, pa korisnik više nema direktan pristup računaru
- Monitor se sastoji iz **rezidentnog monitora** i **pomoćnih programa**
- Na ulaz se postavlja ceo paket poslova
- Programe učitava jedan po jedan i sukcesivno izvršava
- Uvodi se poseban jezik za upravljanje monitorom **JCL (Job Control Language)**

Prosti sistemi paketne obrade

Slika: Levo: Raspored memorije rezidentnog monitora, Desno: Tipičan posao (*job*)



$JOB
$FTN
•
•
•
FORTRAN instructions
$LOAD
$RUN
•
•
•
Data
$END

Prosti sistemi paketne obrade

Monitor

Monitor je jednostavan program koji zavisi od sposobnosti CPU-a da izvršava instrukcije iz različitih delova memorije kako bi privremeno preuzeo ili predao upravljanje.

Poželjne osobine hardvera

1. **Zaštita memorije**
2. **Tajmer** - sprečavanje monopola nad CPU vremenom
3. **Privilegovane instrukcije** - primer su UI instrukcije koje recimo može da poziva samo monitora
4. **Prekidi (interrupts)** - u to vreme uglavnom nedostupni

Multiprogramirani sistemi paketne obrade

Glavni problem prostog sistema paketne obrade

Kako su UI uređaji uglavnom spori u odnosu na CPU, CPU je često besposlen.

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
Total	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Osnovna ideja je da se omogući **višeprocesna obrada podataka (multiprogramiranje)**, kada procesi mogu da se izvršavaju jednovremeno ukoliko se ne sukobljavaju oko računarskih resursa. Npr. ako proces čeka UI, procesor može da se preda drugom procesu.

Multiprogramirani sistemi paketne obrade

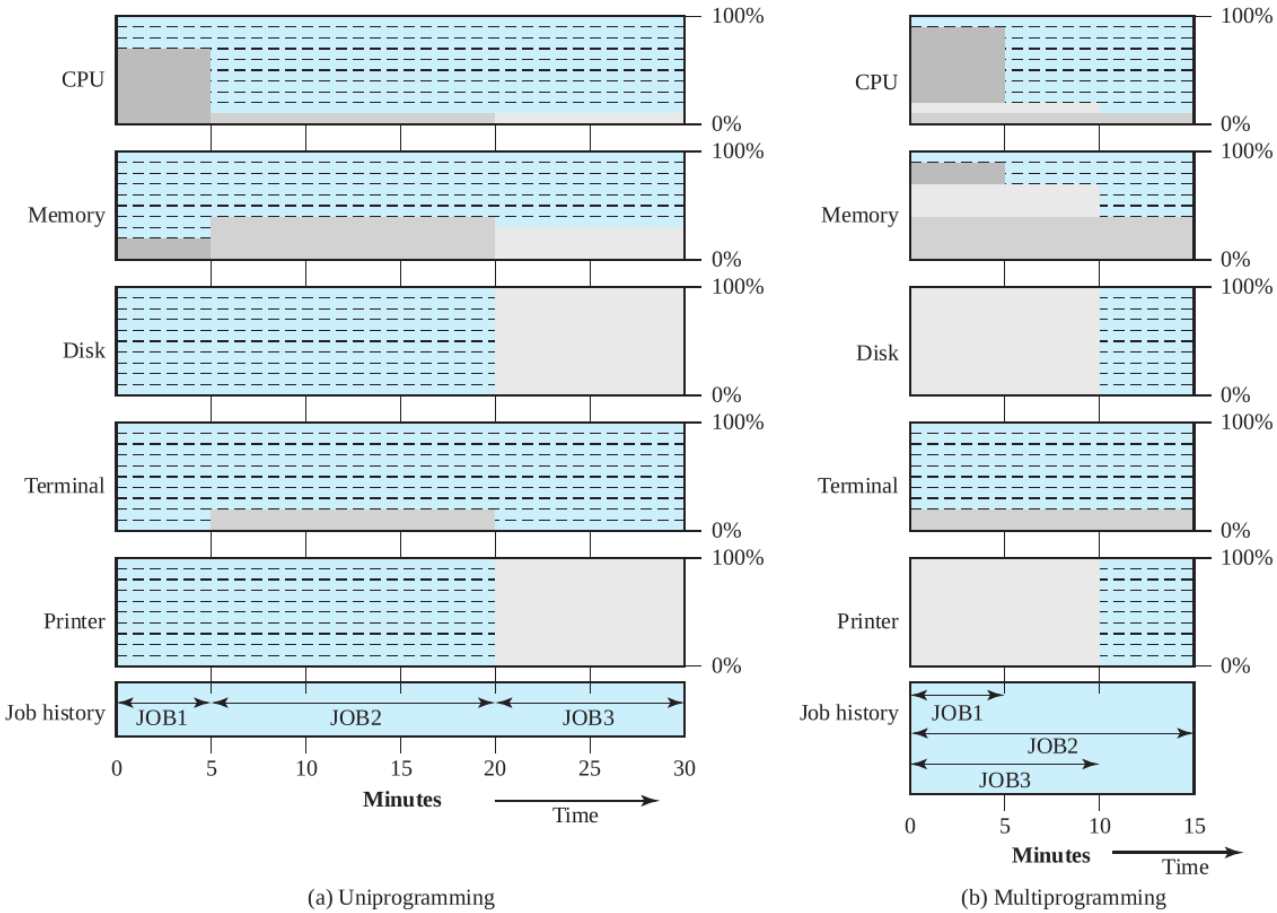
Primer

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Multiprogramirani sistemi paketne obrade

Histogram iskorišćenja računarskih resursa

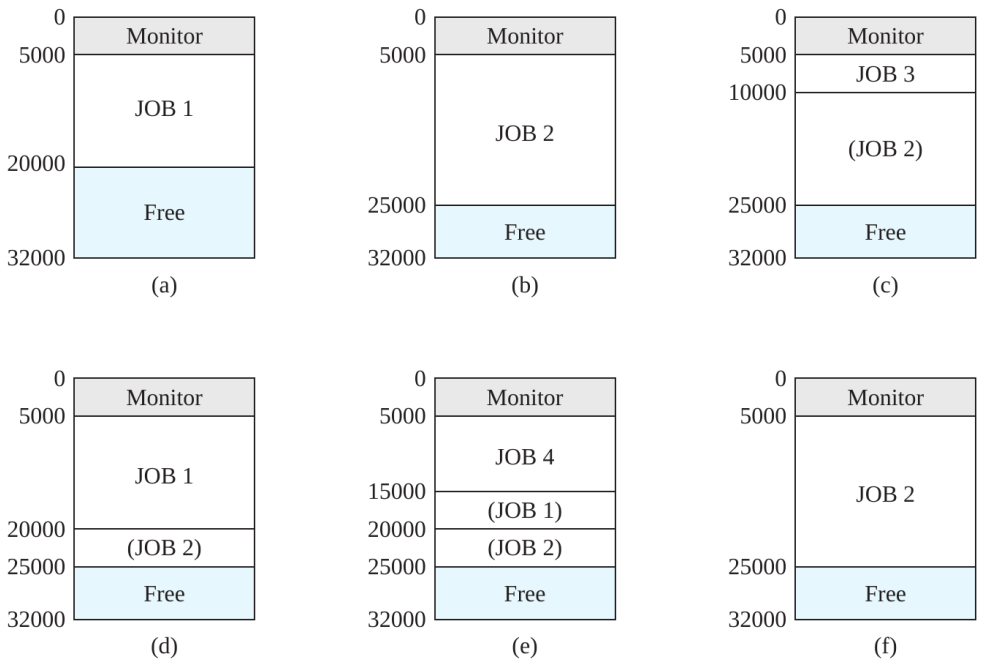


Sistemi sa deljenjem vremena

Primer sistema CCTS na IBM 709/7094

Motiv

Nastao sa uvođenjem **transakcionih sistema i interaktivnih terminala**.
Osnovni motiv je smanjenje vremena odgovora sistema korisniku.



Procesi

Definicija

Proces je osnova strukture OS-a i definiše se kao program u izvršavanju.

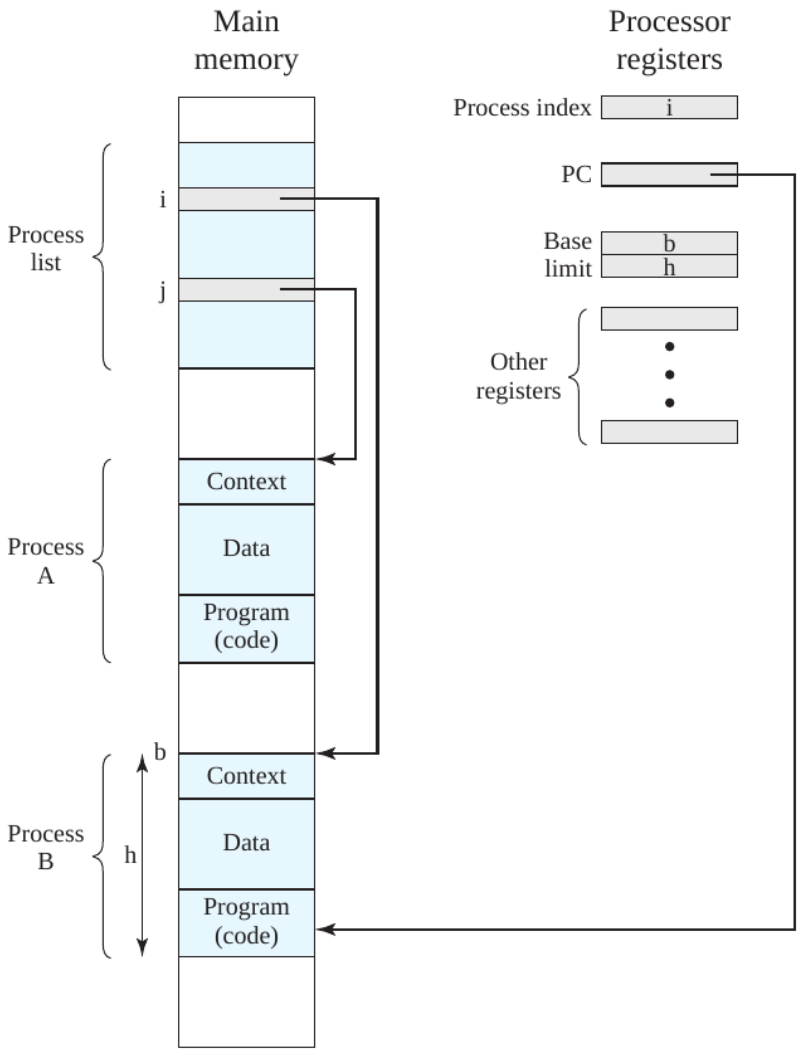
- **Prvi i osnovni alat** za razvoj prvih višekorisničkih i višeprogramskih sistema bili su **prekidi**
- Glavni problemi su: *loša sinhronizacija, otkaz međusobnog isključenja, neodređenost programskih operacija, uzajamno blokiranje (deadlock)*

Kontekst izvršenja–stanje procesa

Kontekst izvršenja su interni podaci kojima OS može da nadgleda stanje procesa. U kontekst procesa spadaju prioritet, sadržaj registara, da li proces čeka UI uređaj i koji isl.

Procesi

Tipična implementacija procesa



Upravljanje memorijom

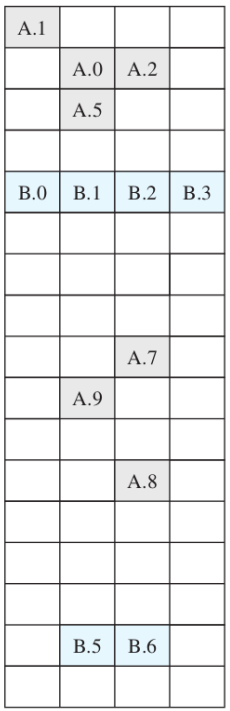
Osnovni zahtevi funkcionalnosti

1. **Izolacija procesa**
2. **Automatsko dodeljivanje i upravljanje** - programeru bi upravljanje memorijom moralo da bude transparentno
3. **Podrška modularnog programiranja**
4. **Zaštita i kontrola pristupa**
5. **Dugotrajna memorija** - upravljanje stalnim memorijama

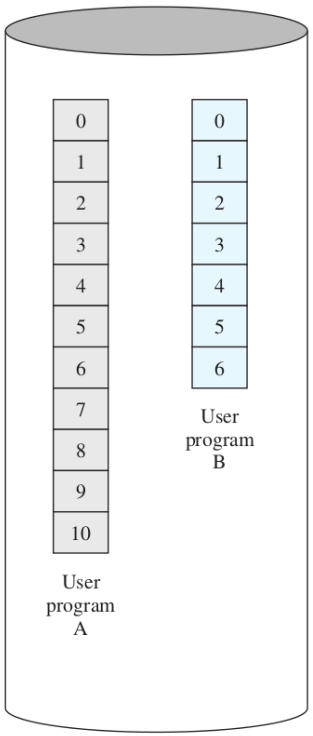
Virtuelna memorija

Uz podršku hardvera (MMU), dozvoljava se da se programi adresiraju logički, bez obzira na veličinu fizičke memorije. Organizaciona jedinica je **stranica**. **Virtuelna adresa** se sastoji iz broja stranice i pomeraja unutar nje.

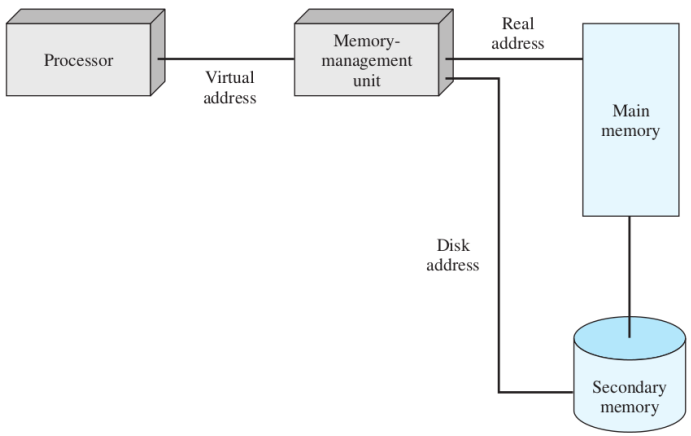
Upravljanje memorijom



Main memory



Disk



Zaštita informacija i bezbednost

Kategorije

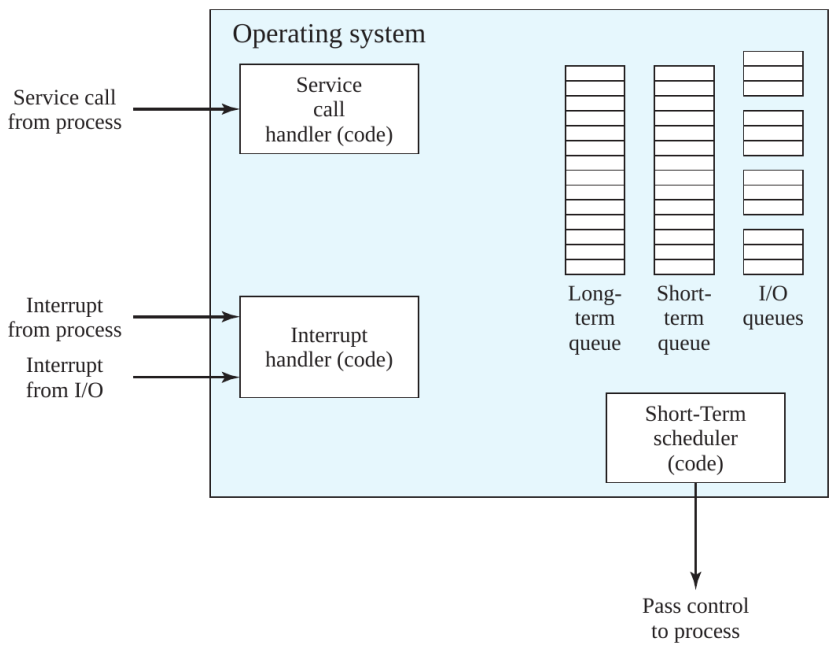
1. **Raspoloživost** - zaštita od prekida rada
2. **Poverljivost** - ko sme da čita podatke, a ko ne
3. **Integritet podataka** - zaštita od nedozvoljene izmene
4. **Autentičnost** - identitet korisnika i ispravnost podataka



Raspoređivanje i upravljanje resursima

Zahtevana funkcionalnost

1. **Nepristrasnost** - fer pristup resursima
2. **Distinkcija odgovora** - Npr. davanje prednosti procesima koji čekaju U/I
3. **Efikasnost** - Uvećati propusnu moć, smanjiti vreme odziva, prihvatiti što veći broj korisnika



Struktura ssitema

Nivoi funkcionalnosti hipotetičkog OS-a

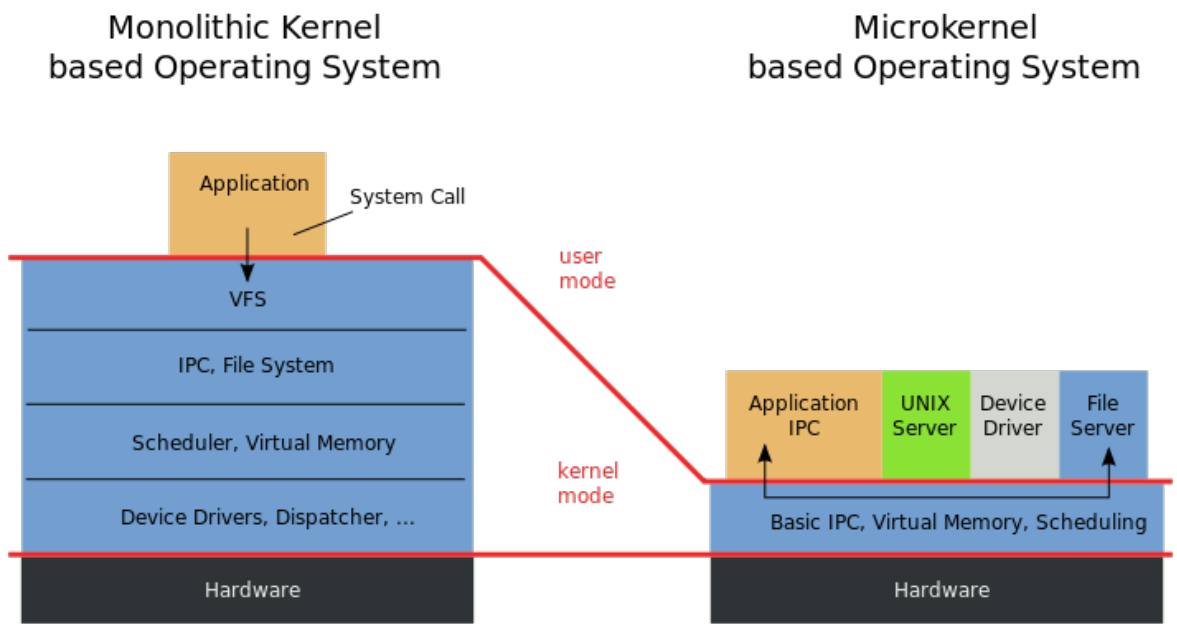
Level	Name	Objects	Example Operations
13	Shell	User programming environment	Statements in shell language
12	User processes	User processes	Quit, kill, suspend, resume
11	Directories	Directories	Create, destroy, attach, detach, search, list
10	Devices	External devices, such as printers, displays, and keyboards	Open, close, read, write
9	File system	Files	Create, destroy, open, close, read, write
8	Communications	Pipes	Create, destroy, open, close, read, write
7	Virtual memory	Segments, pages	Read, write, fetch
6	Local secondary store	Blocks of data, device channels	Read, write, allocate, free
5	Primitive processes	Primitive processes, semaphores, ready list	Suspend, resume, wait, signal
4	Interrupts	Interrupt-handling programs	Invoke, mask, unmask, retry
3	Procedures	Procedures, call stack, display	Mark stack, call, return
2	Instruction set	Evaluation stack, microprogram interpreter, scalar and array data	Load, store, add, subtract, branch
1	Electronic circuits	Registers, gates, buses, etc.	Clear, transfer, activate, complement

Mikrokernel (μ - kernel) arhitektura

Glavni pravci razvoja savremenih OS

Mikrokernel

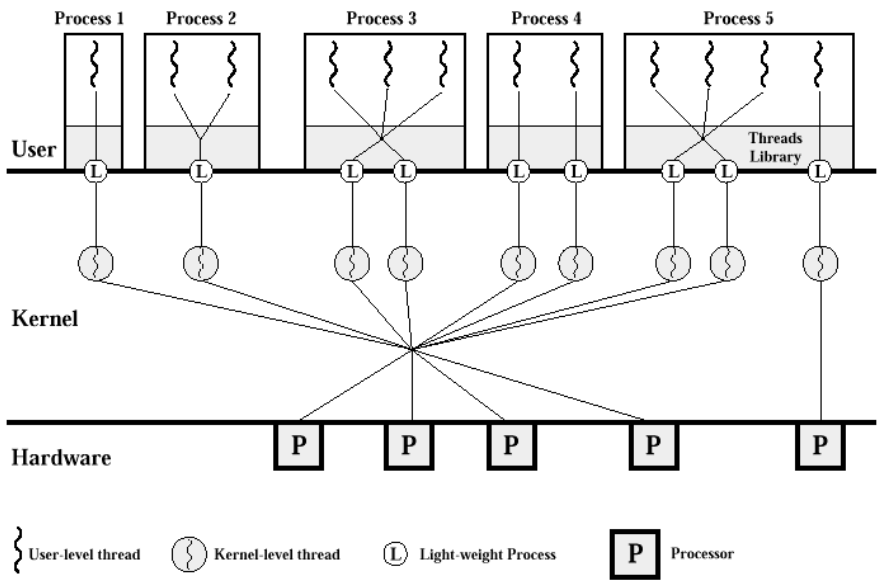
Umesto velikog **monolitnog kernela**, postavlja se **mikrokernel** sa najosnovnijim funkcionalnostima, a ostatak funkcionalnosti premešta u posebne *servere*. Portabilnost na račun performansi.



Višenitna obrada (*multithreading*)

Glavni pravci razvoja savremenih OS

- **Nit** je izvršna jedinica posla. Obuhvata sopstvenu ublast podataka za stek (da bi se omogućilo grananje)
- **Proces** je skup ≥ 1 niti i dodeljenih sistemskih resursa. Podelom jednog procesa u više niti programer postiže veću kontrolu.



SMP–*Symmetric MultiProcessing*

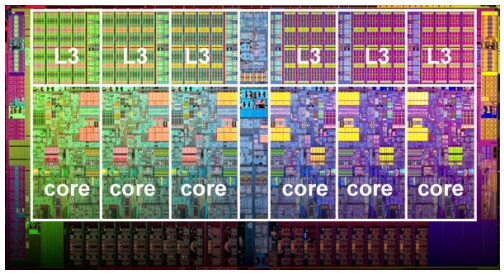
Glavni pravci razvoja savremenih OS

SMP obrada

U sistemu postoji više procesora, koji dele isti RAM i UI veze. Svi procesori mogu da vrše iste operacije.

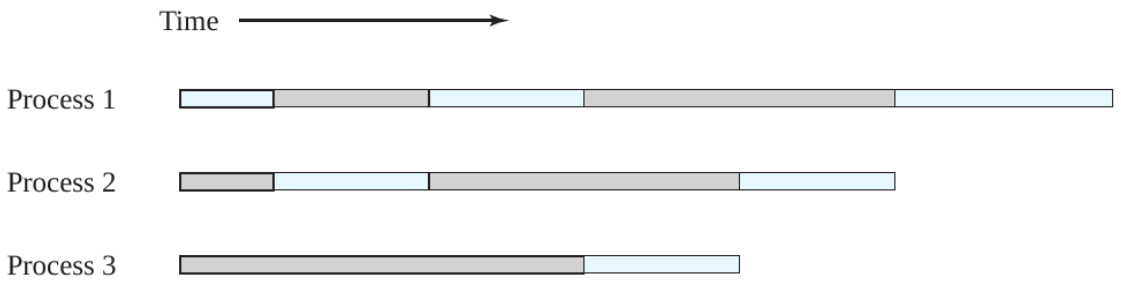
Prednosti u odnosu na jednoprocesorski sistem

1. Performanse
2. Raspoloživost
3. Skalabilnost

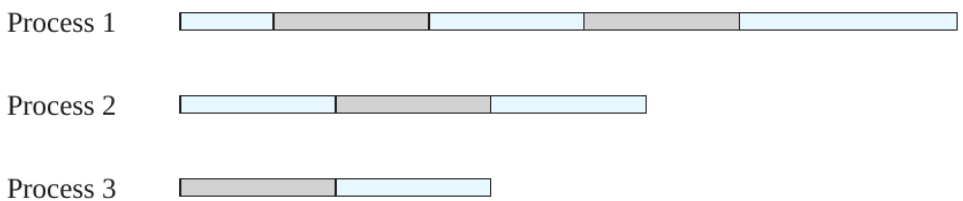


SMP–*Symmetric MultiProcessing*

Glavni pravci razvoja savremenih OS



(a) Interleaving (multiprogramming, one processor)

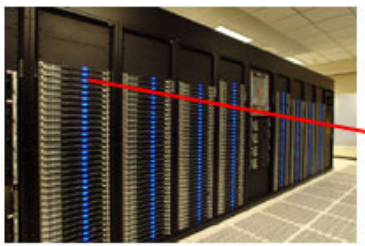


(b) Interleaving and overlapping (multiprocessing; two processors)



Distribuirani sistemi–klasteri

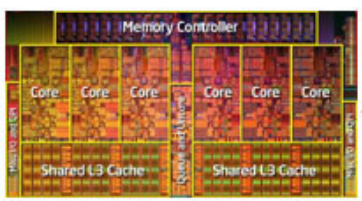
Glavni pravci razvoja savremenih OS

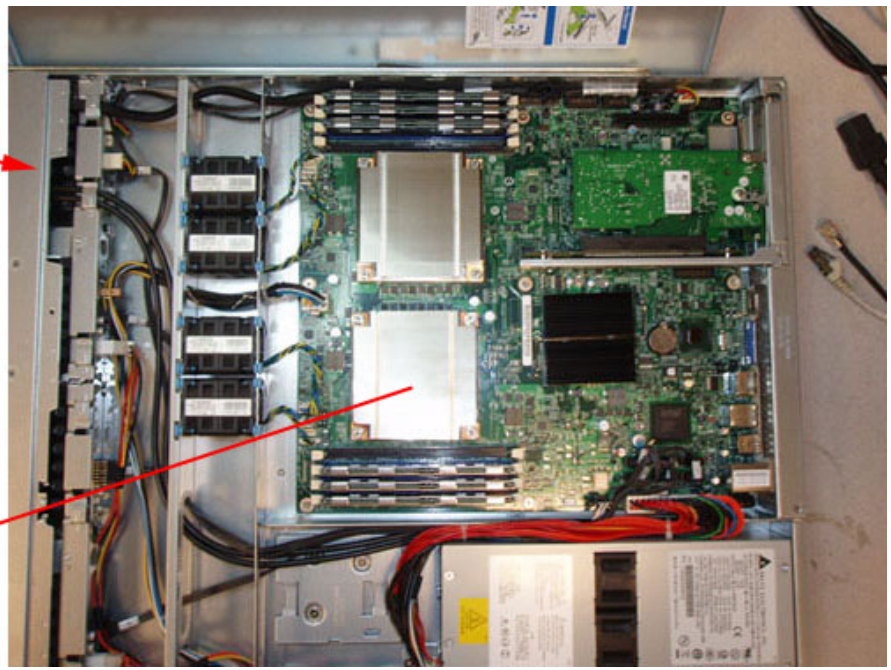


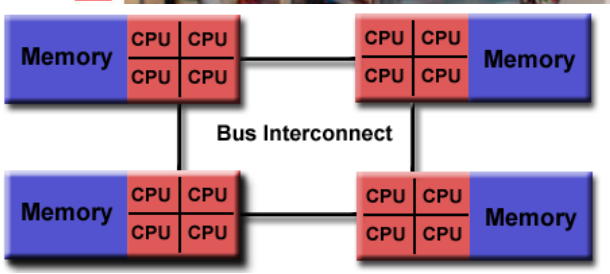
Supercomputer - each blue light is a node

Node - standalone Von Neumann computer

CPU / Processor / Socket - each has multiple cores / processors.

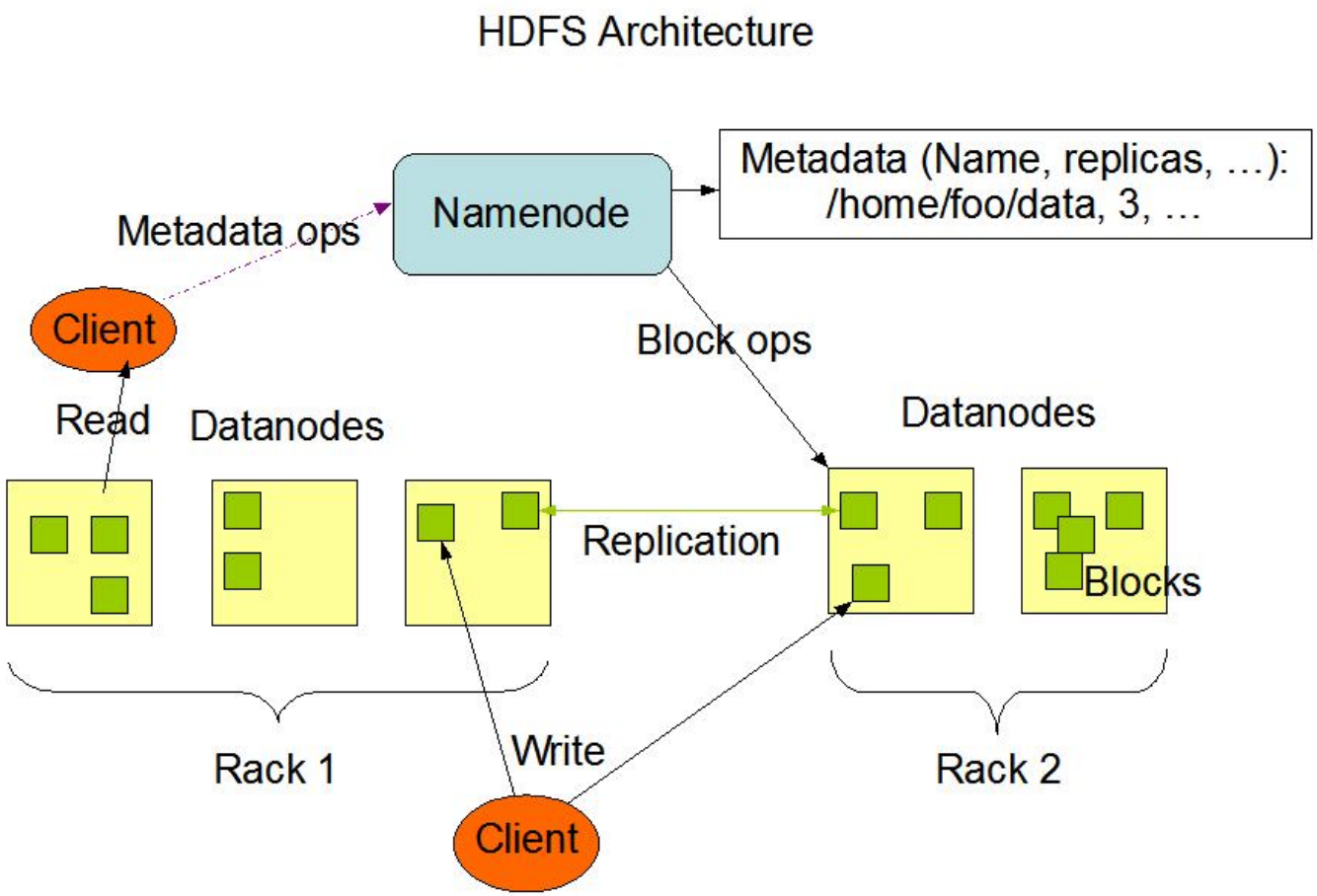






Primer distribuiranog fajl sistema - *Apache Hadoop*

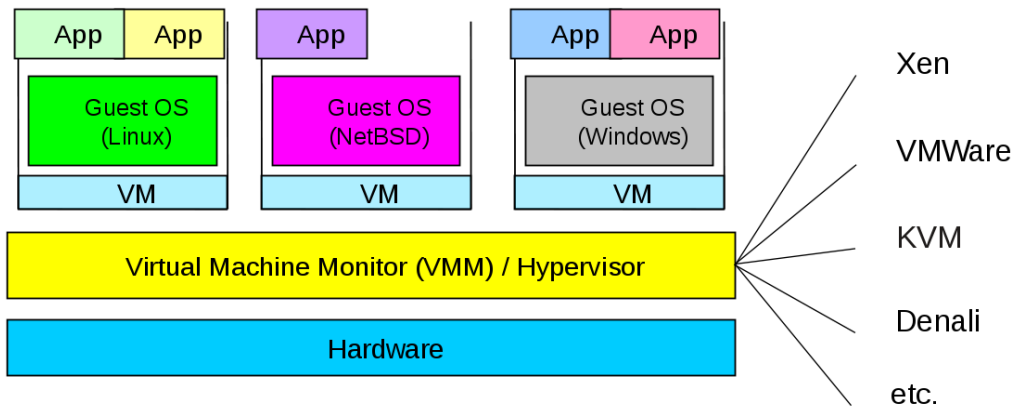
Hadoop Distributed File System



Virtuelizacija

Glavni pravci razvoja savremenih OS

1. Puna virtuelizacija
2. Paravirtuelizacija
3. Virtuelizacija novoa OS-a



Paraviruelizacijom se postižu performanse virtuelne mašine koje se približavaju performansama prave.

Microsoft Windows

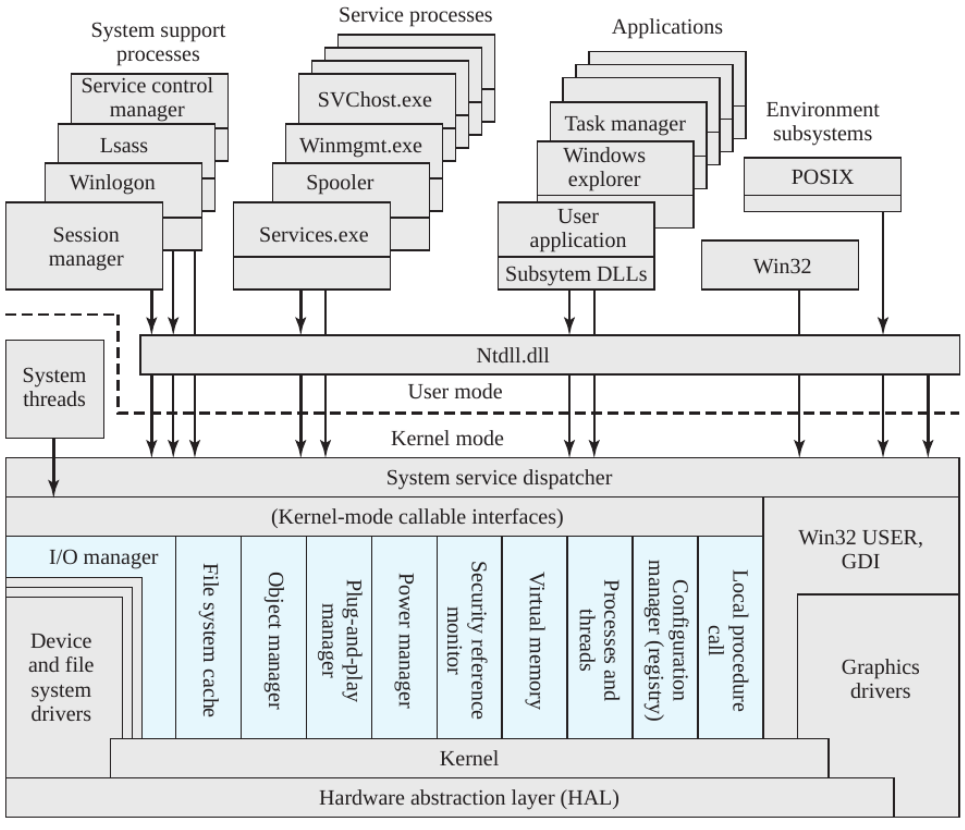
- Izmenjena arhitektura **mikrokernela**
- Zbog performansi, mnoge sistemske funkcije koje se nalaze izvan kernela se **izvršavaju u režimu kernela**
- Visoka modularnost

Komponente Windows-ovog kernel režima

1. **Executive** - upravljanje memorijom, procesi, niti, bezbednost, keš, PnP...
2. **Kernel** - jedini deo koji ne može da se prekida ili *swap*-uje
3. **Hardware Abstraction Layer (HAL)** - sloj apstrakcije hardvera
4. **GUI** - na Windows-u je deo kernela

Microsoft Windows

Arhitektura Windows-a 7



Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

UNIX sistemi

- Prva verzija napisana u asembleru Bell Labs 1970. godine za PDP-7
- Za PDP-11 ponovo napisan u C-u
- Tu je *Berkeley Software Distribution* - BSD
- UNIX System III i UNIX System V 1980-ih godina
- Monolitni kernel počeo da predstavlja problem
- BSD i Solaris 10 (komercijalni)

GNU Projekat, slobodni softver

Definicija slobodnog softvera

- You have the **freedom to run the program** as you wish, for any purpose.
- You have the **freedom to modify the program** to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.)
- You have the **freedom to redistribute copies**, either gratis or for a fee.
- You have the **freedom to distribute modified versions** of the program, so that the community can benefit from your improvements.

Slobodni softver - korisni linkovi

- GNU Project
- Why is open source good for business

Open Source Initiative, GNU Project

- Osnovana radi daljeg popularisanja koncepta programiranja sa otvorenim kodom
- Olakšava unapređenje za softverske proizvode
- Obezbeđuje da **bilo ko testira, ispravlja (debug) i unapređuje aplikacije**
- Povećava šanse da će skrivene greške biti pronađene i ispravljene
- Odlučujuće za ispravke **bezbedonosnih grešaka** koje treba da se brzo isprave
- Pojedinci i korporacije mogu da **menjaju izvorni kod**
- **Kreiranje prilagođenog softvera** koji zadovoljava potrebe određenog okruženja

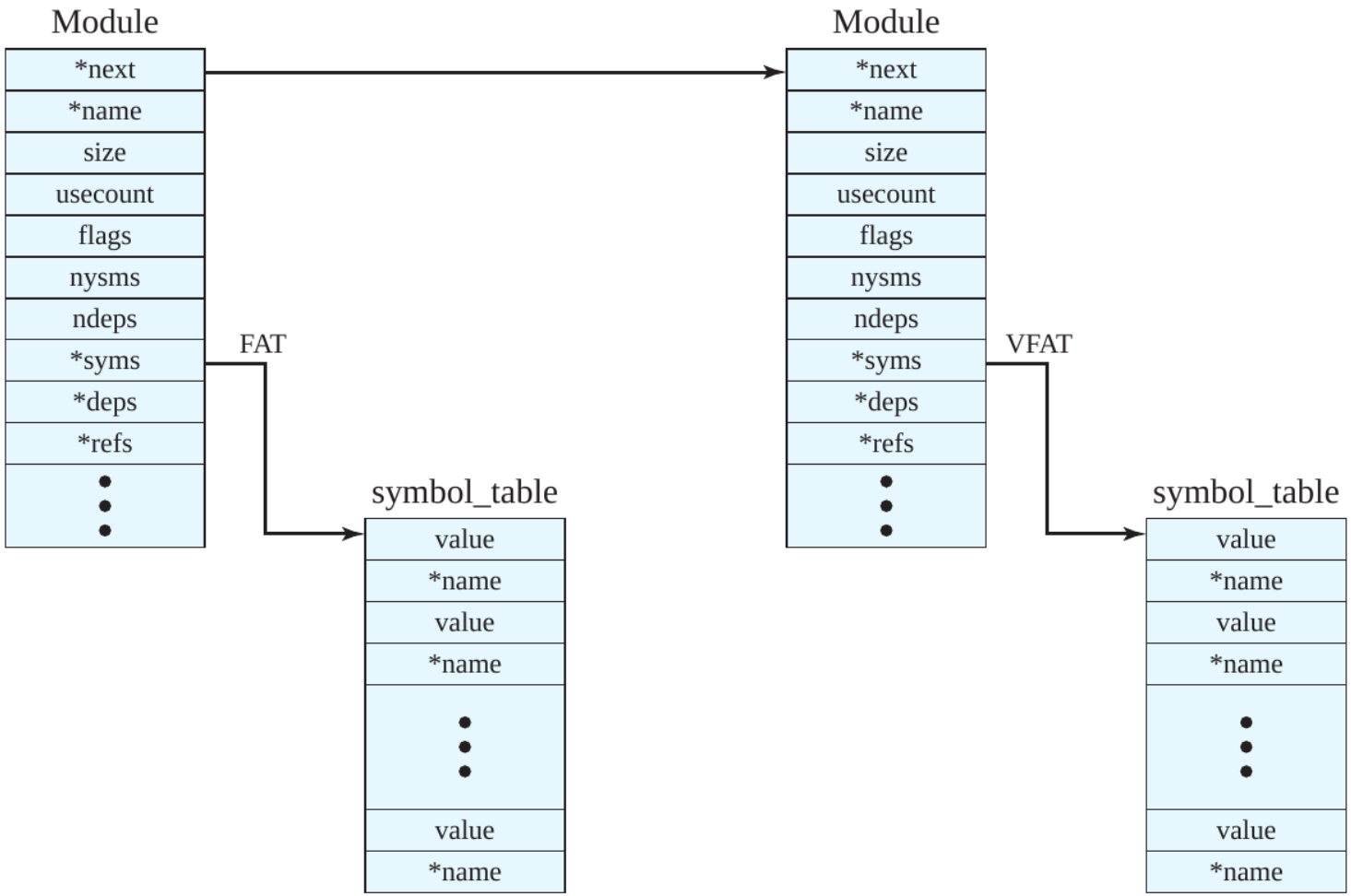
Linux

Modularna struktura

- Većina UNIX jezgara je monolitno, pa i Linux
- U monolitnoj arhitekturi ako se bilo šta promeni, jezgro mora da se prevede i sistem ponovo pokrene
- Međutim, Linux omogućava učitavanje nezavisnih kernl modula
- Moduli se mogu **vezivati dinamički**
- Moduli mogu posedovati **hijerarhijsku strukturu**, tj. pozivati jedni druge
- Svaki modul je određen dvema tabelama: **Tabelom modula** i **Tabelom simbola**
- **Tabela simbola** definiše simbole koje kontroliše dati modul, a koriste se na nekom drugom mestu

Linux

Modularna struktura



Linux

Komponente Linux jezgra

