STRUKTURA E SISTEMEVE OPERATIVE

KAPITULLI 3

Prof. Ass. Dr. Isak Shabani

Struktura e Sistemeve Operative

- Shërbimet e Sistemit Operativ
- Ndërfaqja e shfrytëzuesit të Sistemit Operativ
- Thirrjet e Sistemit
- Tipet e thirrjeve të Sistemit
- Programet e Sistemit
- Dizajni dhe Implementimi i Sistemit Operativ
- Struktura e Sistemit Operativ
- Makinat Virtuale
- Kontrolli i Gabimeve (Debugging) në Sistemin Operativ
- Gjenerimi i Sistemit Operativ
- Ngritja e Sistemit (System Boot)

Objektivat

- Të përshkruaj shërbimet që ofron sistemi operativ te shfrytëzuesit, proceset dhe sistemet tjera.
  - (Në ligjërata).
- Të diskutojmë mënyra të ndryshme të strukturimit të sistemit operativ.
  - (Në ligjërata).
- Të shpjegoj si instalohen sistemet operative, kostumizohen dhe si ngrihen (boot).
  - (Në ushtrime).

Shërbimet e Sistemeve Operativ

- Sistemet operative ofrojnë ambient ekzekutimi të programeve dhe të shërbimeve tek programet dhe shfrytëzuesit.
- Një bashkësi e shërbimeve të sistemit operativ ofron funksione që janë të dobishme për shfrytëzuesin:
  - **Ndërfaqja e shfrytëzuesit** – Gati të gjitha sistemet operative kanë ndërfaqe të shfrytëzuesit (User iInterface –UI)
    - Dallon në mes të:
      - Command-Line (CLI),
      - Ndërfaqe Grafike Graphics User Interface (GUI),
      - Grumbull instruksionesh - Batch
  - **Ekzekutimi i programit** – Sistemi duhet të jetë i gatshëm:
    - të vendos programin në memorie,
    - të ekzekutoj atë,
    - të përfundoj ekzekutimin, në mënyrë normale ose jo(në rast gabimi).
  - **Operacionet e I/O** - Programi në ekzekutim mund të kërkoj I/O, i cili mund të përfshijë një Fajll ose pajisje I/O.

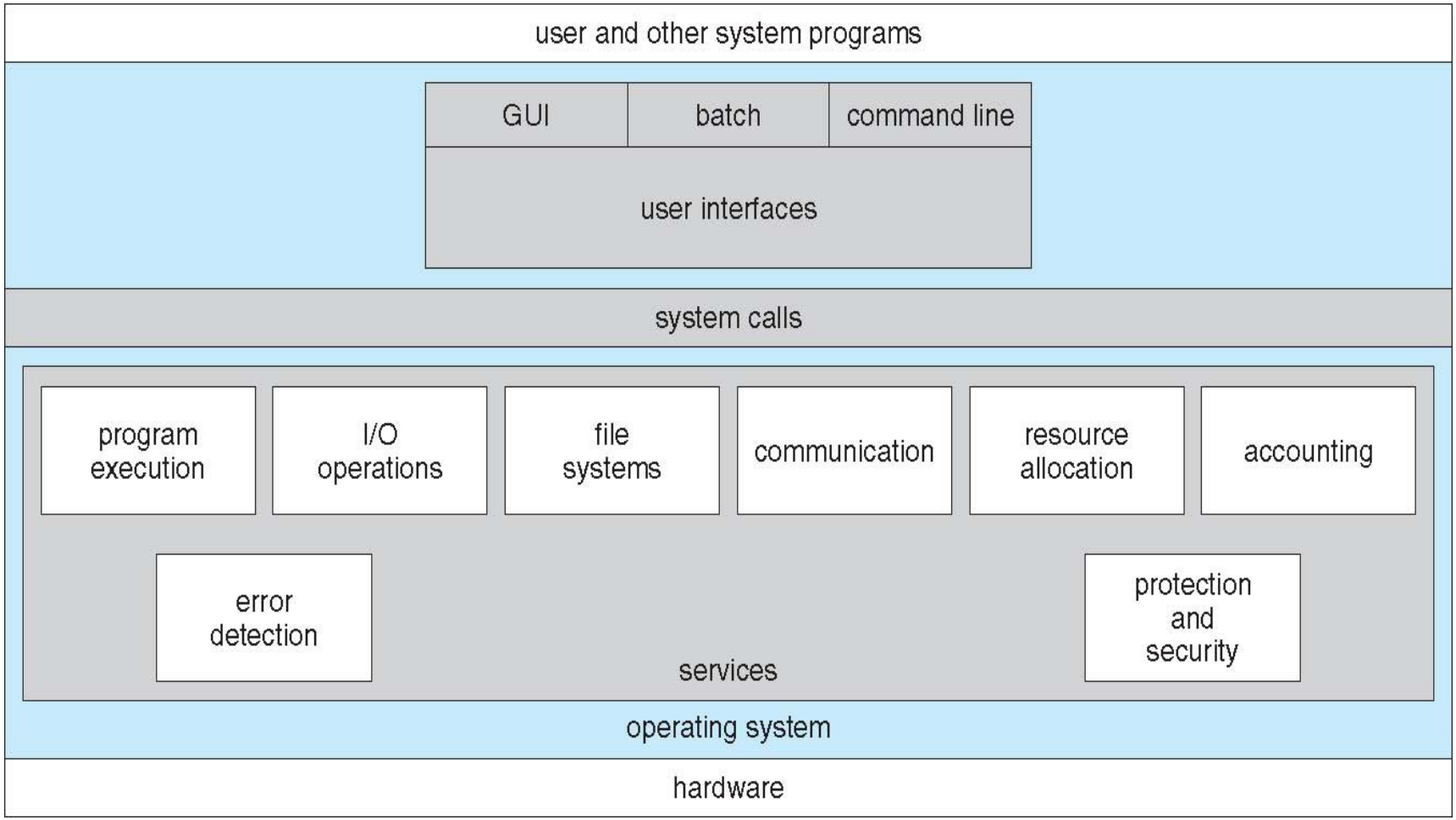
Shërbimet e Sistemeve Operativ (vazh.)

- **Manipulimi me fajlla të sistemit** - Sistemi i fajllave është i interesit të veçantë. Programet kanë nevojë të:
  - lexojnë dhe të shkruajnë fajllat dhe direktivat,
  - krijojnë dhe fshijnë,
  - kërkojnë,
  - të listojnë listë me informatat e fajllit,
  - menaxhimi i lejeve (permissions).
- **Komunikimet** – Proceset mund të shkëmbejnë informacione, në kompjuterin e njëjtë ose në rrjetë.
  - Komunikimi mund të jetë përmes:
    - memories së ndarë ose
    - përmes mesazheve (paketave të bartura nga SO).
- **Gjetja e gabimeve** – SO duhet të jetë i vetëdijshëm për gabimet e mundshme
  - Gabimet mund të shfaqen:
    - në harduer (CPU dhe memorie),
    - në pajisjet I/O dhe
    - në programin e shfrytëzuesit.
  - Për secilin tip të gabimit, SO duhet të ndërmarr veprimin e duhur për të siguruar llogaritje korrekte dhe të qëndrueshme.
  - Veglat për gjetjen e gabimeve mund të ngrisin mundësitë për shfrytëzuesit dhe për programet që ti shfrytëzojnë ato në mënyrë sa më efikase

Shërbimet e Sistemeve Operativ(vazh.)

- Një bashkësi tjetër e SO që ekziston për të siguruar operacione efikase të vet sistemit përmes ndarjes së resurseve
  - **Alokimi i resurseve –** kur shumë punë ose shfrytëzues lëshohen në mënyrë konkurrente, resurset duhet të alokohen te secili nga ata.
    - Shumë tipe të resurseve - disa (si ciklet e CPU, memoria kryesore, dhe storage-i i fajllave) mund të keni kode speciale të alokimit, të tjerat (si pajisjet I/O) mund të kenë kode të përgjithshme për kërkesë dhe lirim
  - **Llogaritë –** Të mbaj shënime se cilët shfrytëzues shfrytëzojnë cilin lloj të resurseve dhe sa e shfrytëzojnë atë
  - **Mbrojtja dhe siguria -** pronari i të dhënave të ruajtura në rrjetën me shumë shfrytëzues ose kompjuterin me shumë shfrytëzues dëshiron të kontrolloj përdorimin e të dhënave, proceset konkurrente nuk duhet të interferojnë me njëra tjetrën
    - **Mbrojtja** përfshinë sigurimin që të gjitha qasjet në resurset e sistemit janë të kontrolluara
    - **Siguria** e sistemit nga të jashtmit kërkon autentifikimin me shfrytëzues, zgjerohet në mbrojtjen e pajisjeve të jashtme I/O nga tentimet e qasjeve të gabuara
    - Nëse sistemi duhet të mbrohet dhe sigurohet, masat duhet të ndërmerren nga të gjitha anët e tij.

Pamje e Shërbimeve të Sistemit Operativ



Ndërfaqja e Shfrytëzuesit të Sistemit Operativ

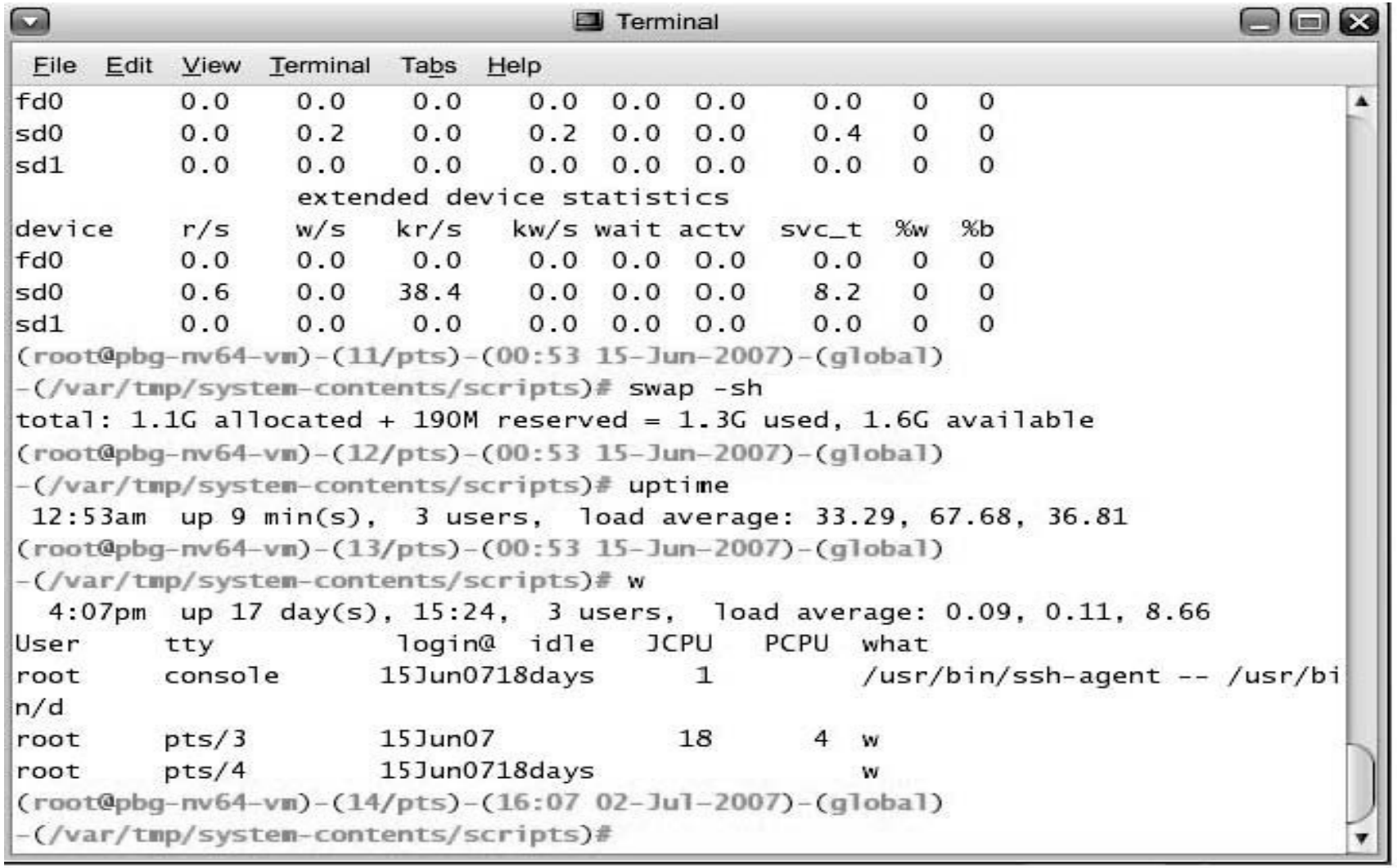
- Ndërfaqja me Linjë Komanduese (CLI) ose **interpretuesi i komandave** lejon vendosje direkte të komandave.
  - Nganjëherë i implementuar në kernell,
  - Nganjëherë nga programet e sistemit operativ,
  - Nganjëherë i implementuar në mënyra të ndryshme – **shells.**
  - Fillimisht e merr komanden nga shfrytëzuesi dhe e ekzekuton atë.
    - Nganjëherë komanda built-in, nganjëherë vetëm emrat e programeve
      - Për këtë të fundit, vendosja e mundësive të reja nuk kërkon modifikim të shell-it.

Ndërfaqja e Shfrytëzuesit të Sistemit Operativ - GUI

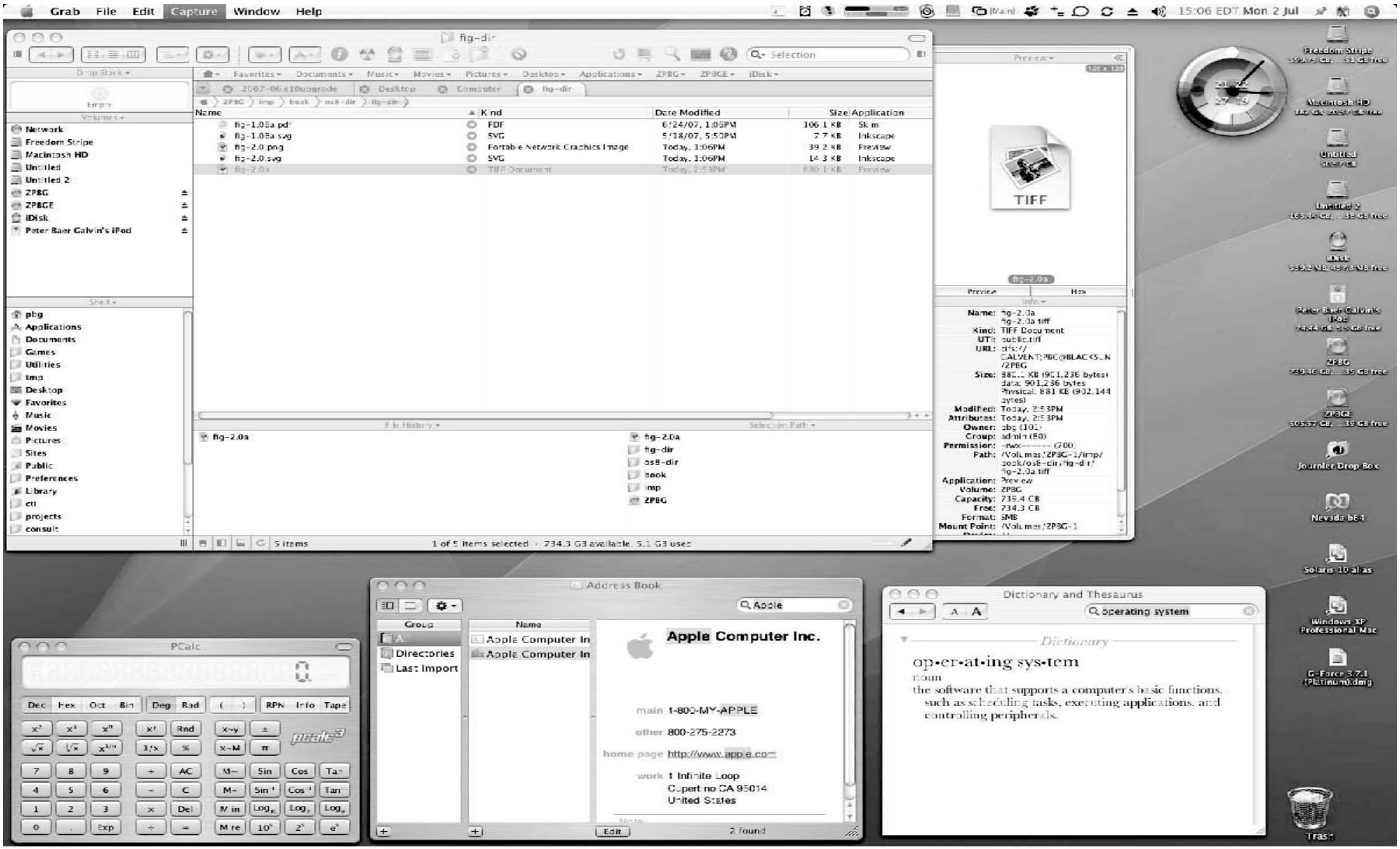
- I përafërt me shfrytëzuesin **desktop** me ndërfaqe metaforike
  - Zakonisht me mi, tasterë, dhe ekran.
  - **Ikona** përfaqëson fajlla, programe, veprime, etj
  - Lloje të ndryshme të butonave të miut mbi objektet e ndërfaqes shkaktojnë veprime të ndryshme (ofrojnë informacione, mundësi, ekzekutimin e funksioneve, hapjen e direktiva (të njohura si **folder**)
  - I zbuluar nga Xerox PARC.

- Shumë sisteme tani përfshijnë të dyja ndërfaqet atë me CLI dhe GUI
  - Microsoft Windows është GUI me CLI “command” shell
  - Apple Mac OS X si “Aqua” GUI ndërfaqe me UNIX kernel përfundi dhe shell të gatshëm.
  - Solaris është CLI me ndërfaqe GUI opsionale (Java Desktop, KDE)

Interpretuesi i Komandave të Bourne Shell



GUI i Mac OS X


Grab File Edit Capture Window Help
15:06 EDT Mon 2 Jul
fig-dir
TIFF
Address Book
Apple Computer Inc.
PCalc
Dictionary and Thesaurus
operating system
Trash

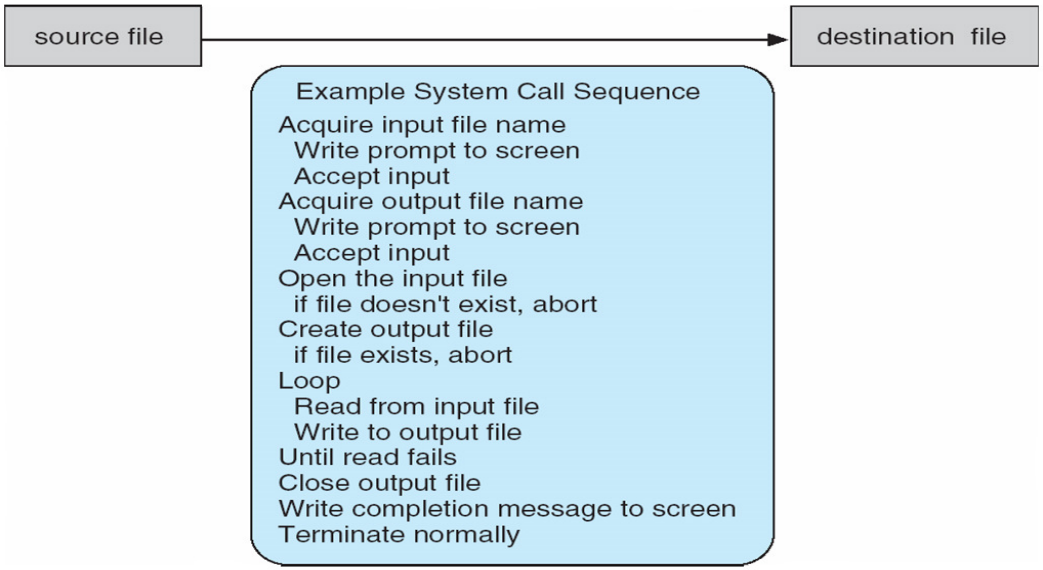
Thirrjet Sistemore

- Ndërfaqja ndërmjet SO dhe programeve shfrytëzuese është e definuar nga ana e grupeve të thirrjeve sistemore të cilat i prodhon SO.
- Ndërfaqe programore te shërbimet e ofruara nga SO.
- Zakonshëm të shkruara në gjuhët e programimit (C ose C++).
- Shumicen e rasteve të qasura nga programet përmes **Application Program Interface (API)** të nivelit të lartë sesa përmes thirrjeve direkte në sistem.
- Tre API-t më të shpeshta janë:
  - Win32 API për Windows,
  - POSIX API për sistemet me bazë-POSIX (që përfshinë virtualisht të gjitha versionet e UNIX, Linux, dhe Mac OS X) dhe
  - Java API për Java makinën virtuale (JVM).

Shembuj të Thirrjeve të Sistemit

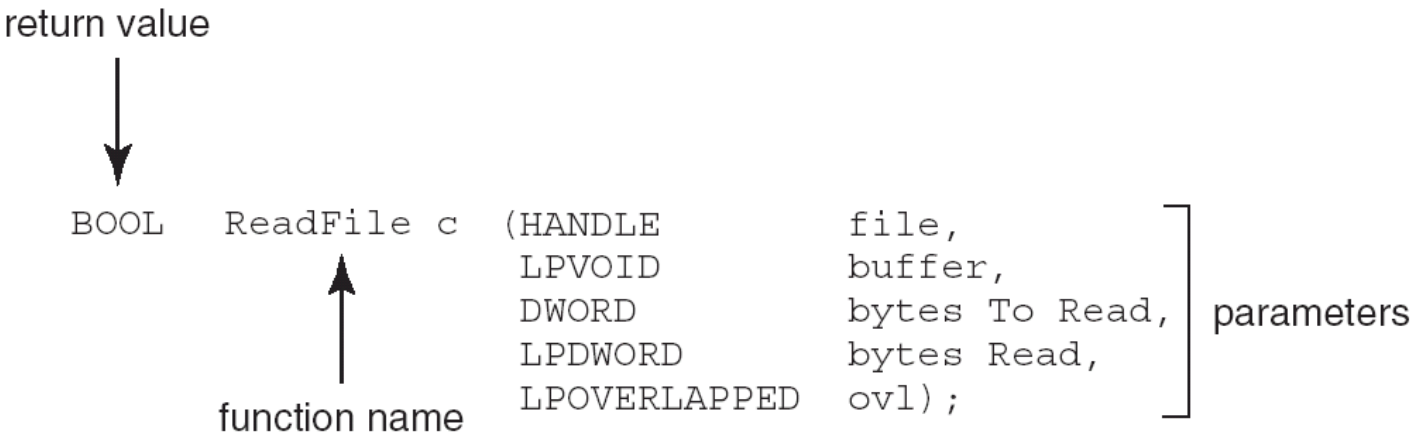
- Që të bëjmë më të qartë mekanizmin e thirrjes sistemore, le të shqyrtojmë thirrjen sistemore për lexim me tre parametra:
  - Parametri i parë përcakton skedarin,
  - Parametri i dytë tregon ku të vendosen të dhënat si referencë deri në përmbajtjen e baferit dhe
  - Parametri i tretë jep numrin e bajtava për lexim.
- Thirrja prej gjuhës së programimit C duket si më poshtë:

  count = read (fd, buffer, nbytes);
- Thirrja sistemore e kthen numrin e bajtave të lexuar në numrim.
- Në figurën e më psohtme është paraqitur sekuenca e thirrjes së Sistemit për të kopjuar përmbajtjen e një fajlli në një fajll tjetër.
- Emrat e thirrjeve-sistemore të shfrytëzuara në këtë tekst janë gjenerike.



Shembull i API Standard

- Mendojeni funksionin ReadFile() në Win32 API - një funksion për lexim nga fajlli.

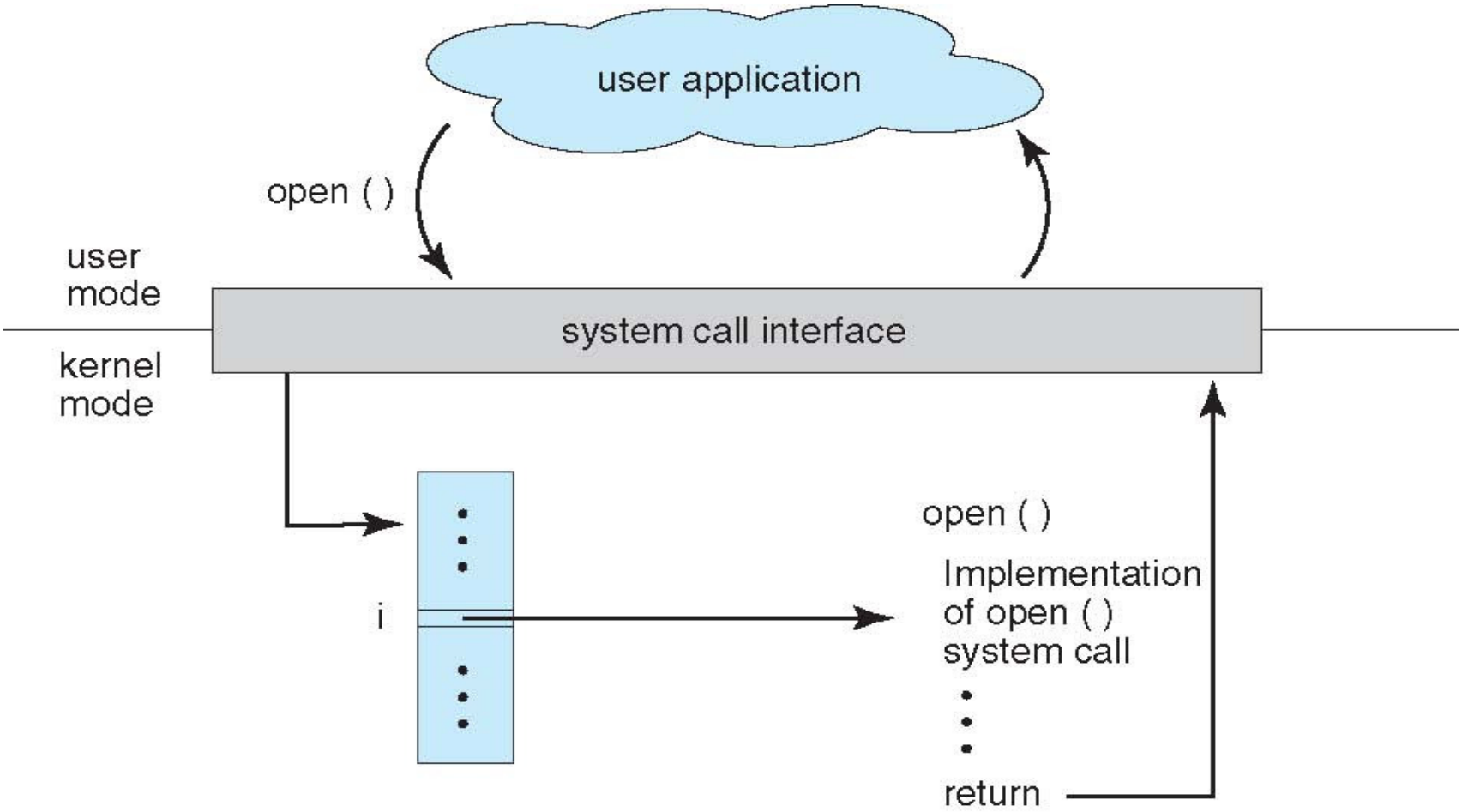


- Përshkrimi i parametrave të dhënë në ReadFile()
  - HANDLE file - fajlli për lexim,
  - LPVOID buffer - baferi ku të dhënat do të lexohen dhe të shkruhen,
  - DWORD bytes To Read - numri i bajtëve për lexim në bafer
  - LPDWORD bytes Read - numri i bajtëve për lexim gjatë leximit të fundit
  - LPOVERLAPPED ovl - tregon nëse kontrollimi(overlapping) i I/O është në shfrytëzim.

Implementimi i Thirrjes së Sistemit

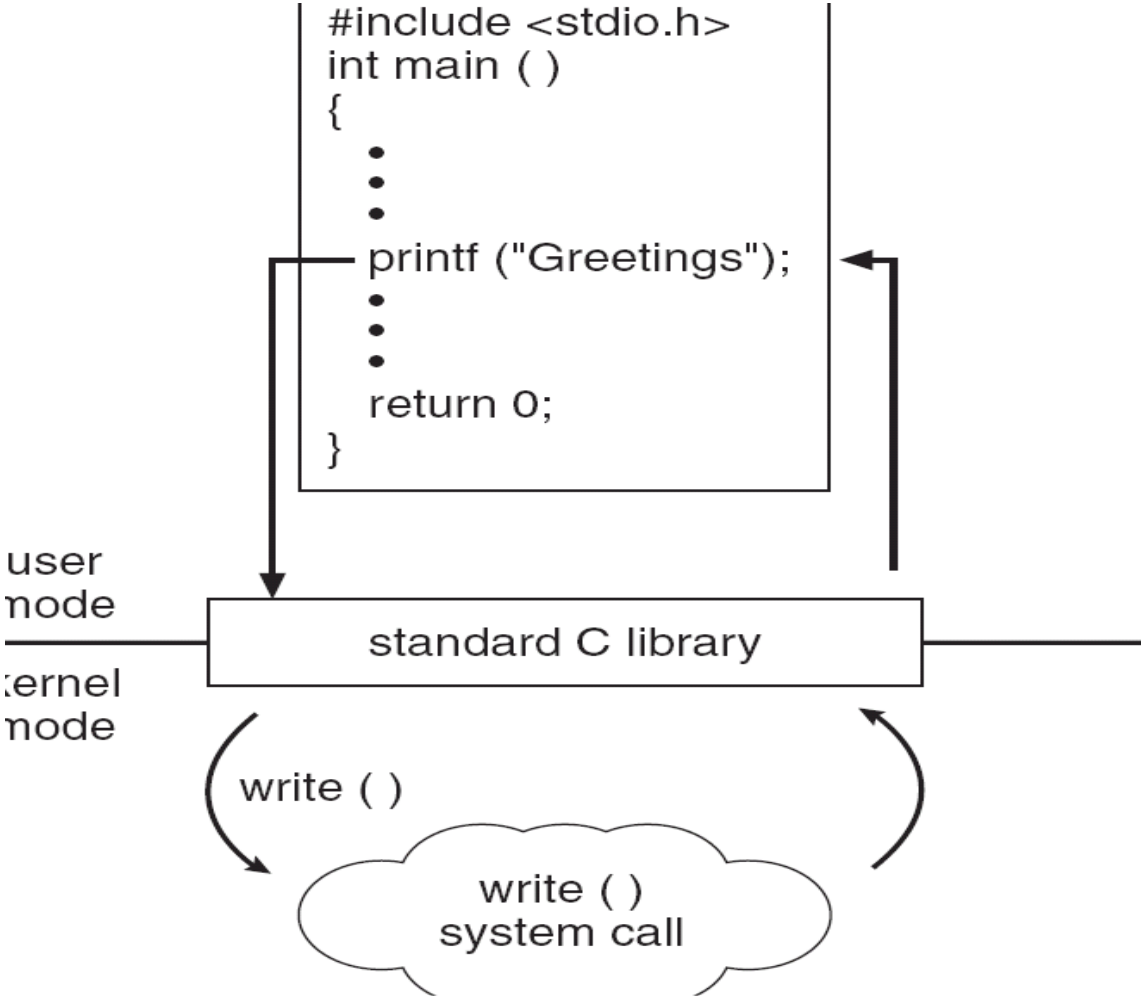
- Zakonshëm, një numër shoqëron secilën thirrje të sistemit
  - Ndërfaqja e thirrjes-së-sistemit përmban një tabelë të indeksuar në akordim me këta numra.
- Ndërfaqja e thirrjes së sistemit e thirr thirrjen e qëllimshme në SO kernelin dhe kthen statusin e thirrjes së sistemit dhe ndonjë vlerë kthyese.
- Thirrësi nuk duhet të di rreth asaj se si është implementuar thirrja e sistemit.
  - Vetëm duhet të i përmbahet API dhe të kuptoj se çka do të bëj SO si rezultat i thirrjes.
  - Shumica e detaleve të ndërfaqes së SO-së janë të fshehura nga programeri përmes API.
    - Menaxhuara nga libraria mbështetëse e run-time (bashkësi e funksioneve të ndërtuara në librari që janë të përfshira me kompajler).

API – Thirrja Sistemit – Relacioni SO



Shembull i Librarisë Standarde të C

- Programi C duke thirrur thirrjen e librarisë **printf()**, e cila thërret thirrjen sistemore **write().**

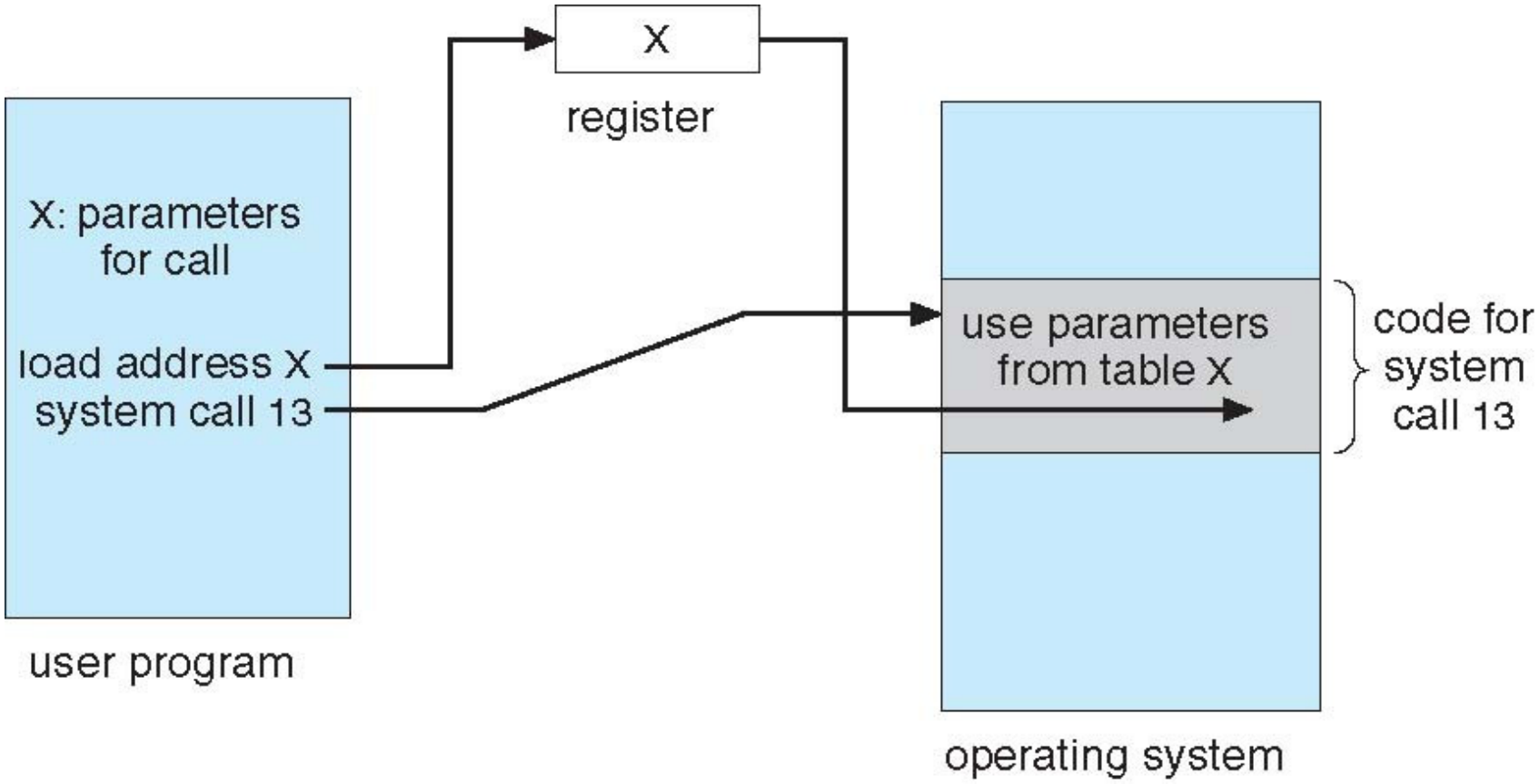


Pasimi i Parametrave në Thirrje të Sistemit

- Shpeshherë, kërkohen më shumë informacione se sa thjeshtë identifikimi i thirrjes së dëshiruar të sistemit:
  - Tipi i saktë dhe sasia e informatave varet varësisht prej SO dhe thirrjes.

- Tri metoda gjenerale përdoren për të pasuar parametrat tek SO
  - Më e thjeshta: paso parametrat në *regjistra*
    - Në disa raste, mund të ketë më shumë parametra sesa regjistra
  - Parametrat e ruajtur në *blloqe,* ose tabelë, në memorie, dhe adresa e bllokut të pasuar si parametër në regjistër
    - Kjo qasje është marrë nga Linux dhe Solaris
  - Parametrat e vendosur, ose të *shtyrë*, në *stek* nga programi dhe *nxjerrë (popped)* nga steku nga sistemi operativ
  - Metodat e bllokut dhe stekut nuk e limitojnë gjatësinë ose numrin e parametrave që pasohen.

Pasimi i Parametrave përmes Tabelës



Tipet e Thirrjeve të Sistemit

- Kontrolli i Procesit
  - mbaro, aborto
  - mbush, ekzekuto
  - krijo procesin, termino procesin
  - merr atributet e procesit, vendos atributet e procesit
  - prit për kohë të caktuar
  - prit ngjarjen, sinjalizo ngjarjen
  - aloko dhe liro memorjen
- Menaxhimi i fajllave
  - Krijo fajll, fshi fajll
  - hape, mbylle fajllin
  - lexo, shkruaj, ripoziciono
  - merr dhe vendos atributet e fajllit

Tipet e Thirrjeve të Sistemit (Vazh.)

- Menaxhimi i paisjeve
  - Kerko paisje, liro paisje
  - lexo, shkruaj, ripoziciono
  - merr atributet e paisjes, vendos atributet e paisjes
  - lidhi logjikisht ose shkëputi paisjet
- Mirëmbajtja e Informatave
  - merr kohën dhe datën, vendos kohën dhe datën
  - merr të dhëna nga sistemi, vendos të dhënat e sistemit
  - merr dhe vendos atributet e procesit, fajllit, të paisjes
- Komunikimet
  - krijo, fshi lidhjet komunikuëse
  - dërgo, prano mesazhe
  - transfero të dhënat e statutit
  - attach and detach remote devices

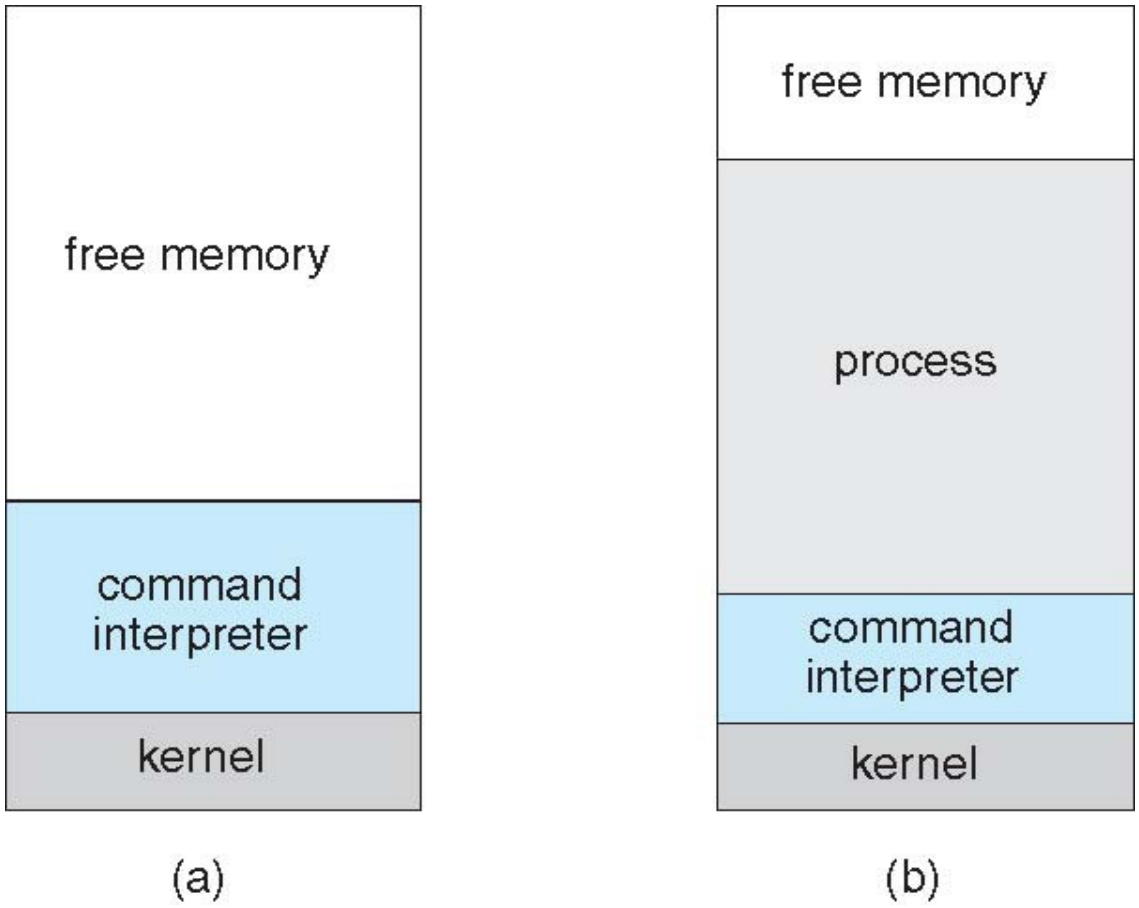
Shembuj të Thirrjeve të Sistemit të Windows dhe Unix

	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Shembull: MS-DOS

- Një-punë
- Thirrur nga Shell kur ngrihet sistemi
- Metodë e thjeshtë e lëshimit të programit
  - Nuk krijohet proces
- Hapësirë e vetme memorike
- E vendos programin në memorie, duke i mbishkruar të gjitha përveç kernelit
- Dalja nga programi -> rihapje e shell

Ekzekutimi në MS-DOS



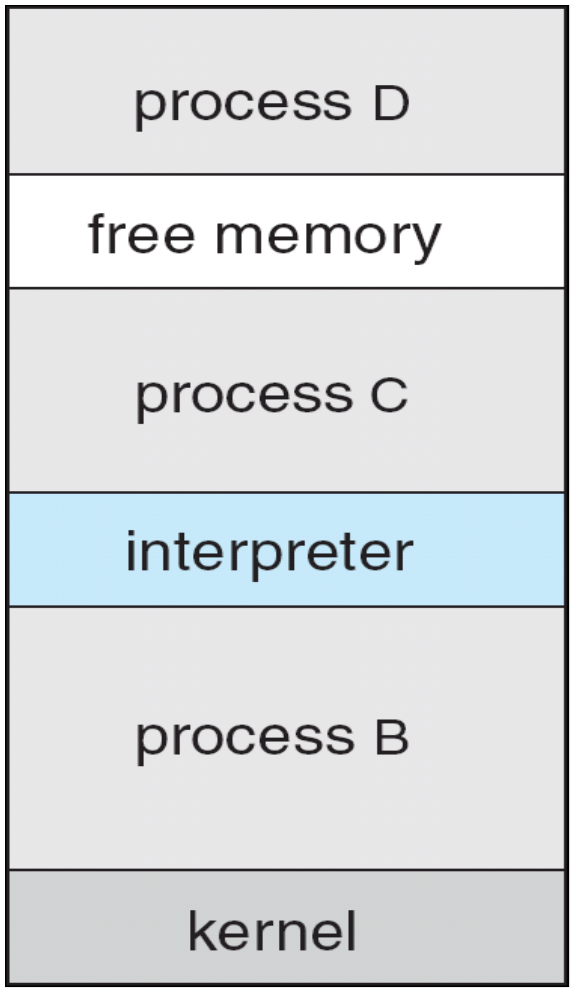
(a) Në startup të sistemit.

(b) Gjatë lëshimit të programit.

Shembull: FreeBSD

- Variant i Unix
- Multitasking (shumë punë përnjëherë)
- Qasja e Shfrytëzuesit -> thirr zgjedhjen e shfrytëzuesit të shell
- Shell ekzekuton thirrjen e sistemit fork() për të krijuar procesin
  - Ekzekuton exec() për të vendosur programin në proces
  - Shell pret që procesi të terminohet ose të vazhdoj me komandat e shfrytëzuesit
- Procesi bën dalje me kod 0 – pa gabime ose > 0 – kode për gabime

FreeBSD në lëshim të shumë programeve



Programet e Sistemit

- Programet e sistemit ofrojnë ambient të përshtatshëm për zhvillimin e programit dhe ekzekutimin.
- Mund të ndahen në:
  - Manipulim të Fajllave
  - Informatave mbi Statusin
  - Modifikimin e Fajllave
  - Mbështetje për gjuhët Programuese
  - Vendosja e Programit dhe Ekzekutimi
  - Komunikime
  - Programe aplikative
- Për shumicën e shfrytëzuesve SO definohet nga programet e sistemit, e jo përmes thirrjeve të sistemit.

Programet e Sistemit

- Ofrojnë ambient të përshtatshëm për zhvillimin e programit dhe ekzekutimin
  - Disa nga to janë vetëm ndërfaqe të shfrytëzuesit tek thirrjet e sistemit, të tjerat konsiderohet të jenë më komplekse.
- **Menaxhimi i fajllave** - Krijo, fshi, kopjo, riemro, shtyp, gjuaj, listë dhe maniupulimi i përgjithshëm i fajllave dhe folderëve.
- **Informata mbi Statusin**
  - Disa i kërkojnë sistemit për të dhënat: data, koha, shuma e memories së lirë, hapësira në disk, numri i shfrytëzuesve
  - Të tjerët ofrojnë detale mbi performancën, qasjet dhe informata për gjetjen e gabimeve
  - Zakonshisht, këto programe e formatizojnë dhe shtypin daljen në terminal ose pajisje tjera dalëse
  - Disa sisteme implementojnë regjistrin – i cili përdoret për regjistrimin dhe marrjen e të dhënave mbi konfiguracionin

Programet e Sistemit (Vazhd.)

- **Modifikimin e Fajllave**
  - Editorët e tekstit për të krijuar dhe modifikuar fajlla
  - Komanda të veçanta për kërkimin e përmbajtjes së fajllave ose për transformimin e tekstit
- **Mbështetje për gjuhët programuëse** - Kompajlerët, asamblerët, veglat për gjetjen e gabimeve dhe nganjëherë ofrohen edhe interpretuesit
- **Vendosja e programit dhe Ekzekutimi** - Vendosësit absolut, vendosësit e tabelave relokuëse, editorët e lidhjeve, mbulesë-ngarkues, sisteme për gjetjen e gabimeve për gjuhët e makinës me nivel të lartë
- **Komunikimet** – Ofrojnë mekanizmin për krijimin e lidhjeve virtuele në mes proceseve, shfrytëzuësve dhe sistemeve kompjuterike
  - Lejon shfrytëzuesit të dërgojnë mesazhe tek ekranet e njëri-tjetrit, shfletojnë ueb faqe, të dërgojnë e-mail mesazhe, të kyqen në largësi,të transferojnë fajlla nga një makinë në tjetrën.

Dizajni dhe Implementimi i Sistemit Operativ

- Dizajni dhe Implementimi i SO nuk është “i zgjidhshëm”, por disa qasje janë treguar të suksesshme
- Struktura e brendshme e Sistemeve të ndryshme Operative mund të varet shumë
- Fillon me definimin e qëllimeve dhe specifikacionet
- Të ndikuara nga zgjedhja e harduerit, tipit të sistemit
- *Qëllimet a Shfrytëzuesit* dhe *Qëllimet e Sistemit*
  - **Qëllimet e Shfrytëzuesit** – sistemi operativ duhet të jetë i përshtatshëm për shfrytëzim, i leht për tu mësuar, i qëndrueshëm, i sigurtë dhe i shpejtë
  - **Qëllimet e Sistemit** – sistemi operativ duhet të jetë i leht për t’u dizajnuar, implementuar, dhe mirëmbajtur, gjithashtu edhe fleksibil, i qëndrueshëm, pa gabime, dhe efikas

Dizajni dhe Implementimi i Sistemit Operativ(vazh.)

- Principet e rëndësishme që duhet ndarë

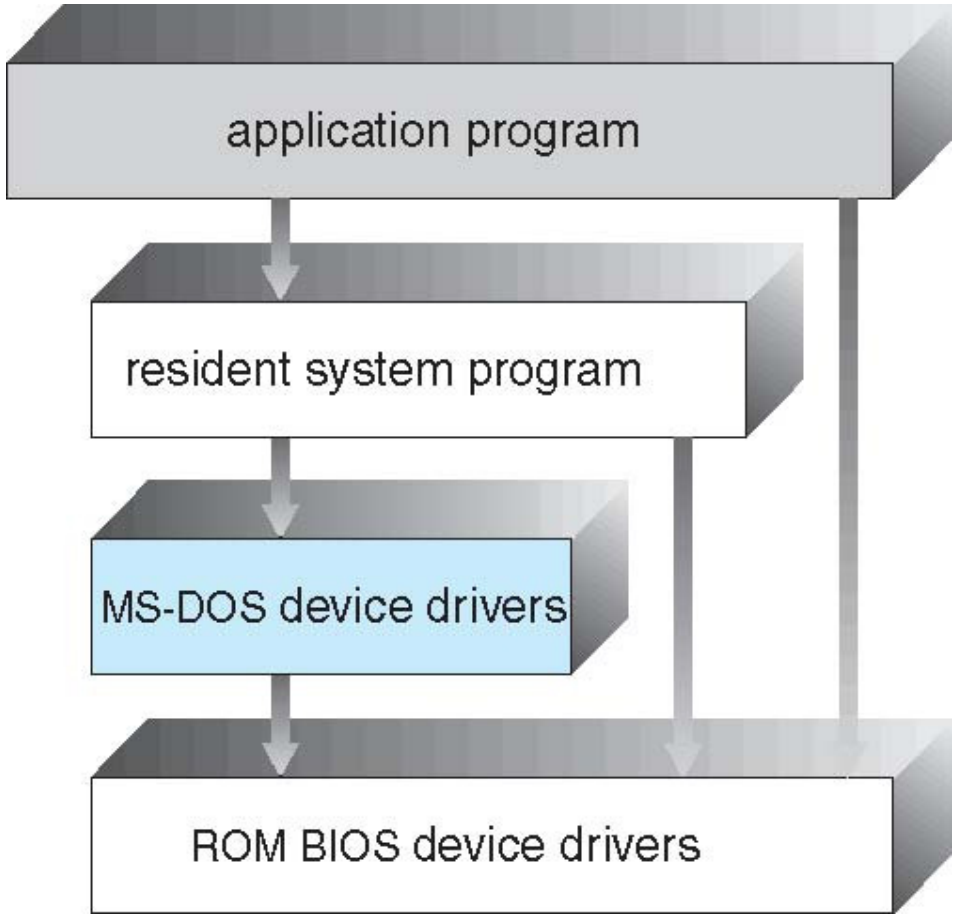
  Rregullat: Ç'farë do të bëhet?
  Mekanizmi: Si ta bëjmë?

- Mekanizmi përcakton se si të bëjmë diçka, ndërsa rregullat vendosin se çka do të bëhet
  - Ndarja e rregullave nga mekanizmi është princip shumë i rëndësishëm, ai lejon fleksibilitet maksimal nëse rregullat e vendimet duhet të ndryshohen më vonë

Strukturë të thjeshtë

- MS-DOS – i shkruar të ofroj të gjithë funksionalitetin në pak hapsirë
  - Nuk është i ndarë në module
  - Megjithatë MS-DOS ka disa struktura, ndërfaqet e tij dhe nivelet e funksionalitetit nuk janë aq mirë të ndara

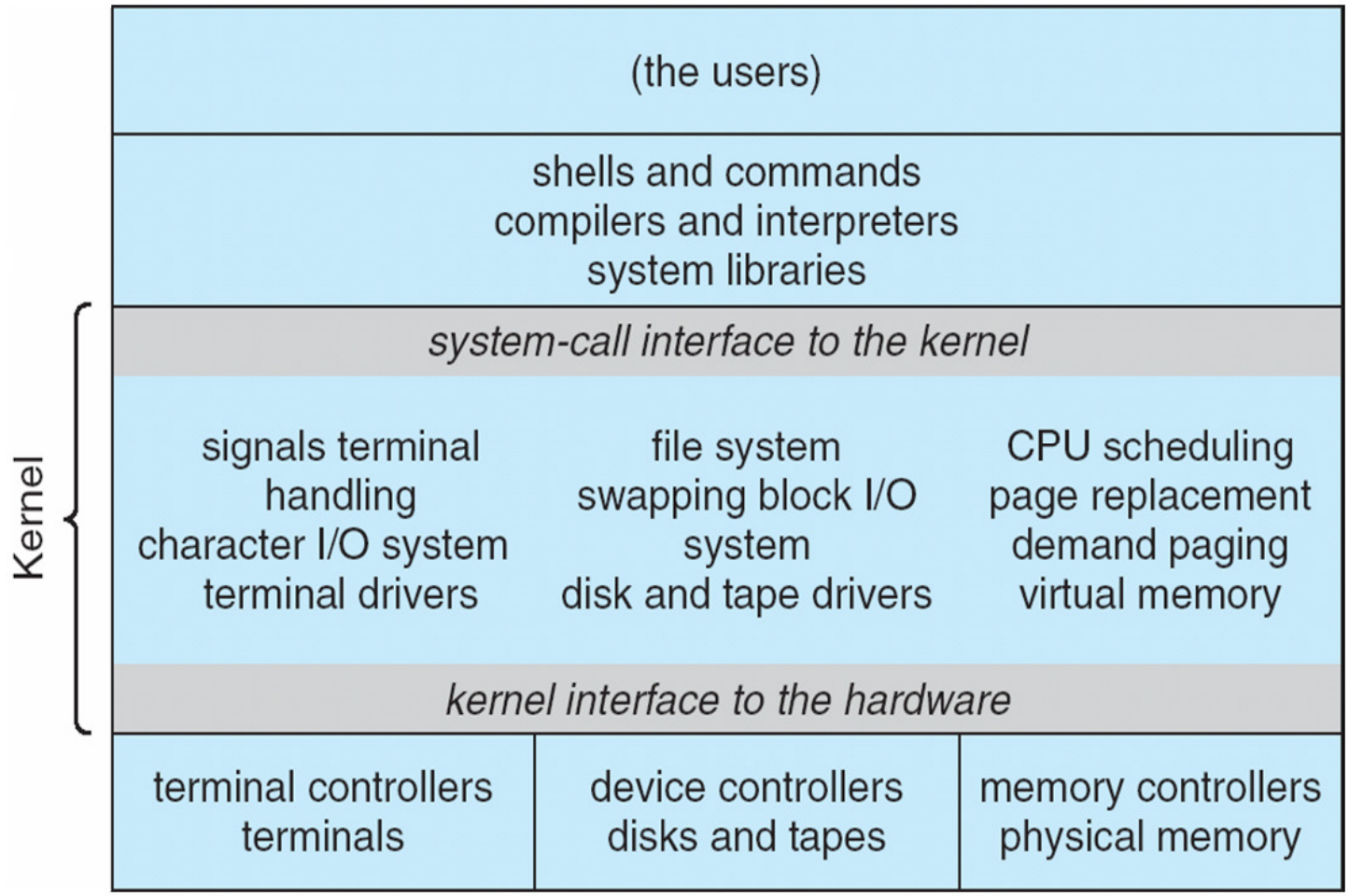
Struktura e Shtresave të MS-DOS



Qasja me Shtresa

- Sistemi operativ është i ndarë në një numrë të shtresave (niveleve), secili i ndërtuar në krye të nivelit më të ulët.
- Shtresa e fundit (shtresa 0), është hardueri;
- Shtresa më e larta (shtresa N) është ndërfaqja e shfrytëzuesit.
- Me modularitet, shtresat janë të zgjedhura në atë mënyrë që secila i shfrytëzon funksionet (operacionet) dhe shërbimet e shtresës më të ulët

Struktura Tradicionale e Sistemit UNIX

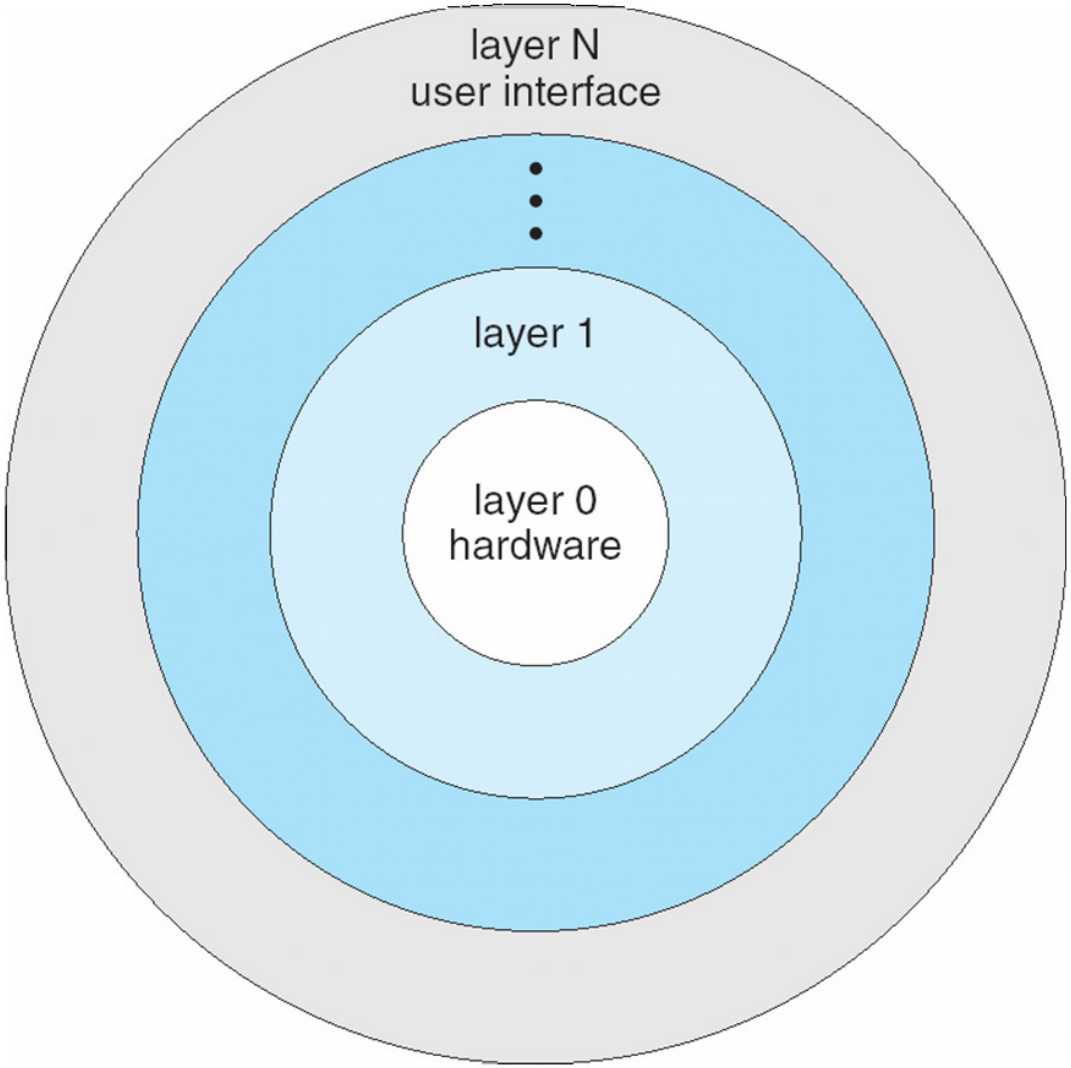


UNIX

- UNIX – I limituar nga funksionaliteti I harduerit, sistemi operativ origjinal i UNIX ka pasur strukturim te limituar.

- SO UNIX përmban dy pjesë të ndashme:
  - Programet e Sistemit dhe
  - Kernel-in
    - Përmban çdo gjë nën ndërfaqen e thirrjeve të sistemit dhe mbi harduerin fizik.
    - Ofron sistemin e fajllave,
    - Oraret e CPU,
    - Menaxhimin e memories dhe
    - Funksionet e SO, një numër i madh i funksioneve për një nivel.

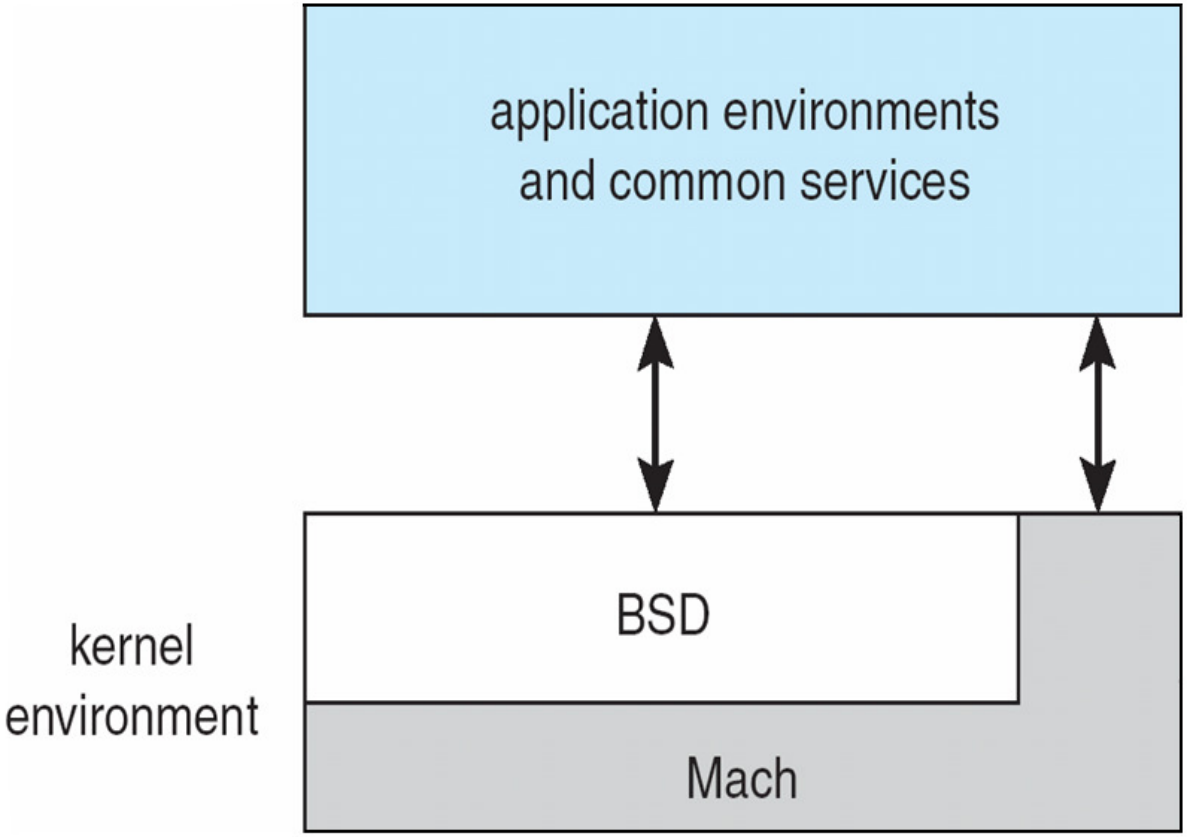
Sistemi Operativ në Shtresa



Struktura e Sistemit të Mikrokernelit

- Bartë çdo gjë të mundshme nga kerneli në hapsirën e "*shfrytëzuesit*".
- Komunikimi zë vend në mes të moduleve të shfrytëzuesve duke shfrytëzuar shkëmbimin e mesazheve.
- Benefitet:
  - Është lehtë për ta zgjeruar mikrokernelin
  - Është më e lehtë lidhja me arkitekturat e reja
  - Më i qëndrueshëm (më pak kod është duke u lëshuar në modin e kernelit)
  - Më e sigurt.
- Dëmet:
  - Mbingarkesa e performancës së hapësirës së shfrytëzuesit në komunikimet e hapësirës së kernelit.

Struktura e Mac OS X

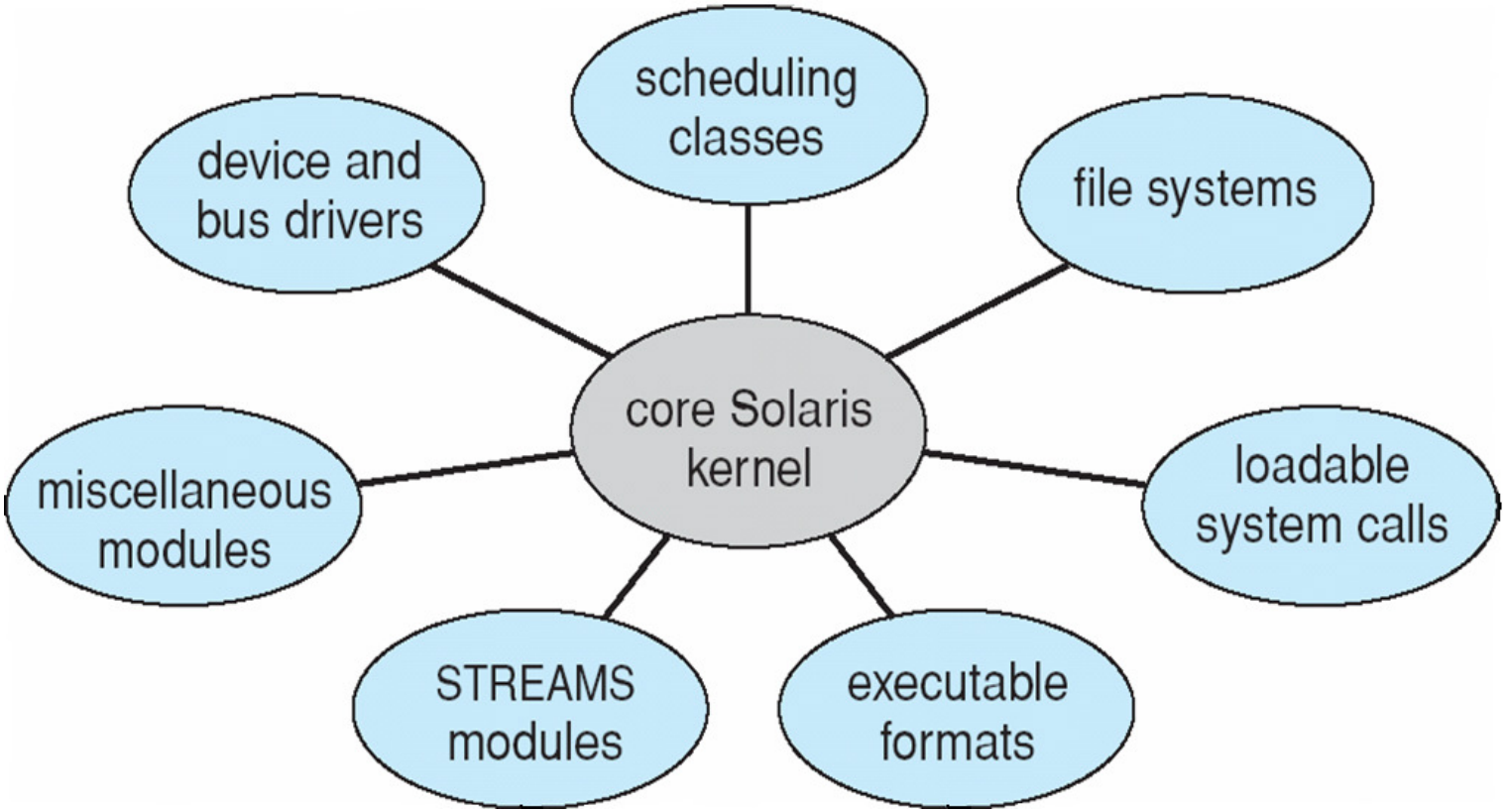


Modulet

- Shumica e SO moderne implementojnë modulet e kernelit
  - Përdor qasje të orientuar në objekte
  - Secila komponentë kryesore është e ndarë
  - Secila komunikon me njëra tjetrën përmes ndërfaqet e njohura
  - Secila mbushet sipas nevojes në kernel

- Mbi të gjitha, janë të ngjashme me shtresat por më fleksibile.

Trajtimi i Modularit të Solaris



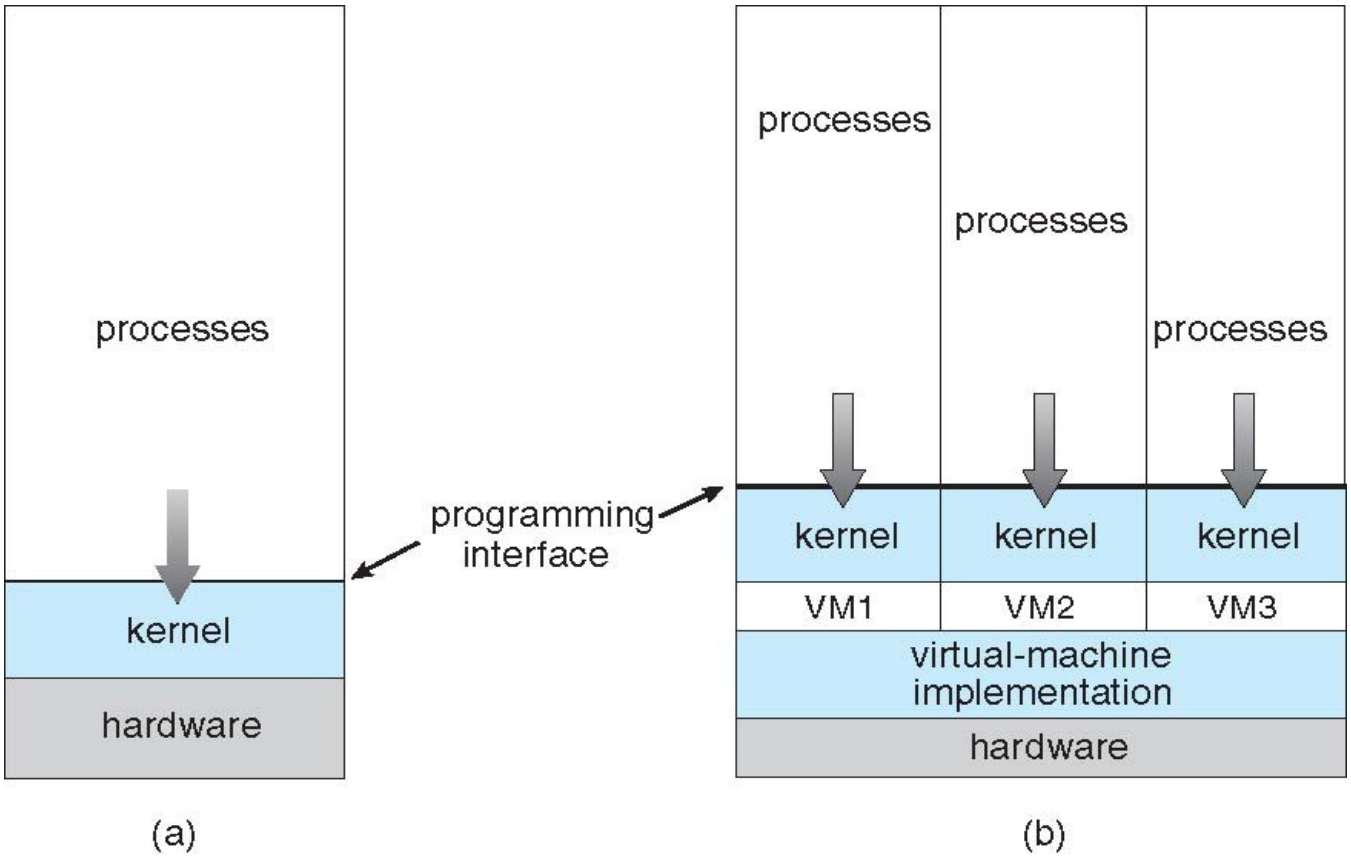
Makinat Virtuale

- **Makina virtuele** i trajton shtresat në konkluzione logjike.
-
- E trajton kernelin e SO sikur te ishte harduer.
- Makina virtuale ofron një ndërfaqe *identike* me themelin e harduerit.
- **Host-i** i sistemit operativ krijon iluzionin që procesi ka procesorin e tij të vetëm dhe memorien virtuale.
- Secili **musafir** i ofruar me (virtual) kopje të bazës së kompjuterit.

Historia e Makinave Virtuale dhe Benefitet

- Komercialisht së pari u shfaq nga IBM mainframe në vitin 1972
- Fillimisht, ambiente me shumë ekzekutime (SO të ndryshme) mund të ndajnë harduerin e njejtë
- Mbrohen prej njëra tjetrës
- Disa ndarje të fajllave mund t'u pengohet qasja, të kontrollohet
- Komutimi me njëra tjetrën, me sisteme të tjera fizike përmes rrjetës
- I përdorshëm për zhvillim, testim
- **Konsolidimi** i disa resurseve të ultë shfrytëzojnë sistemet më pak të ngarkuara
- "Open Virtual Machine Format", format standard i makinave virtuele, lejon VM të lëshohet brenda shumë platformave të makinave(host) virtuele.

Makinat Virtuale (vazhd.)



(a) Makina jo virtuale.

b) Makina virtuale

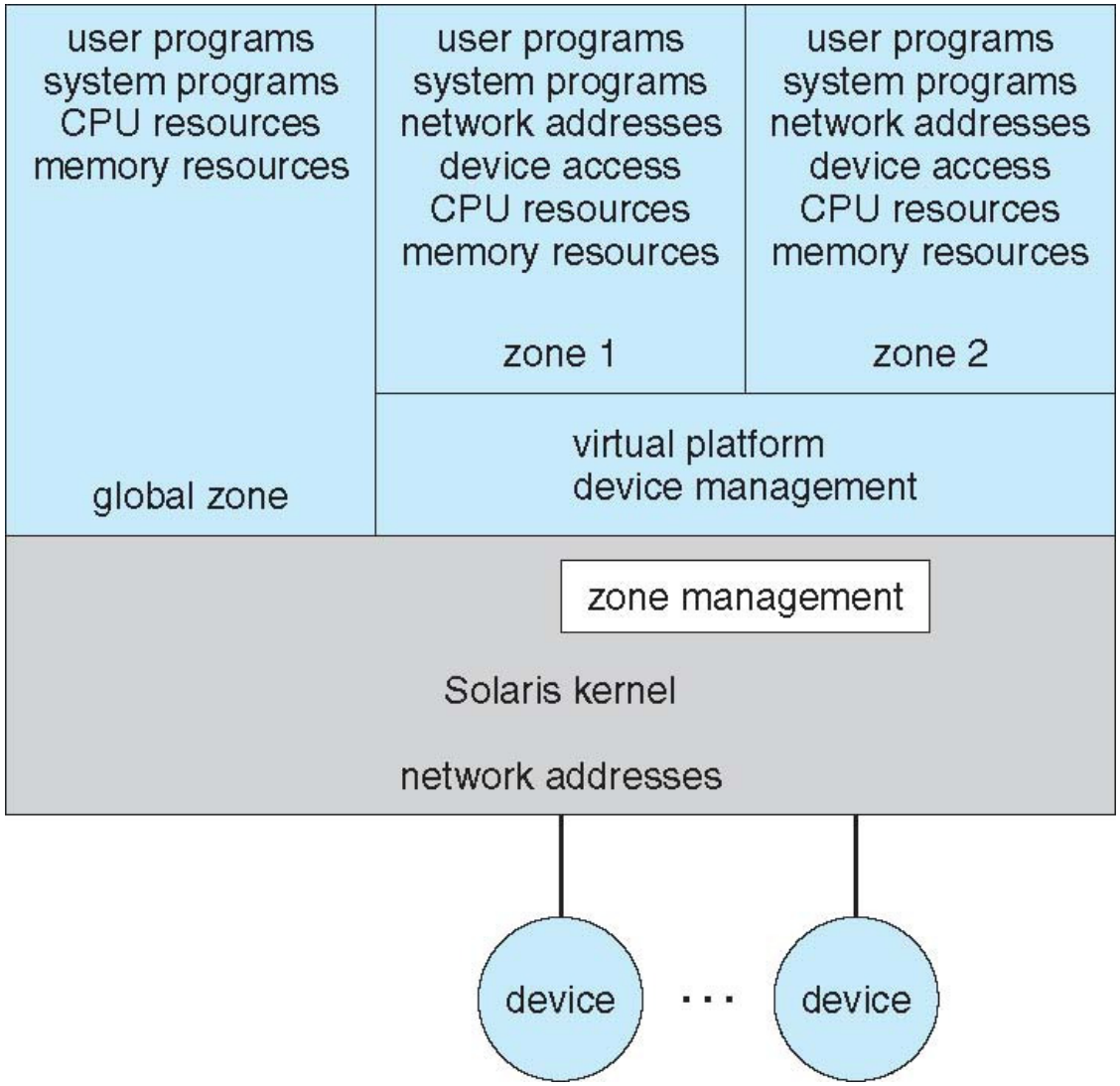
Para-virtualizimi

- U prezenton shfrytëzuesve sistem të ngjashëm por jo identik me harduerin.
- Shfrytëzuesi duhet të modifikohet që të lëshohet në harduerin e para-virtuelizuar.
- Shfrytëzuesi mund të jetë SO, ose si në rastin e Solaris 10 aplikacionet lëshohen në **kontejnerë.**

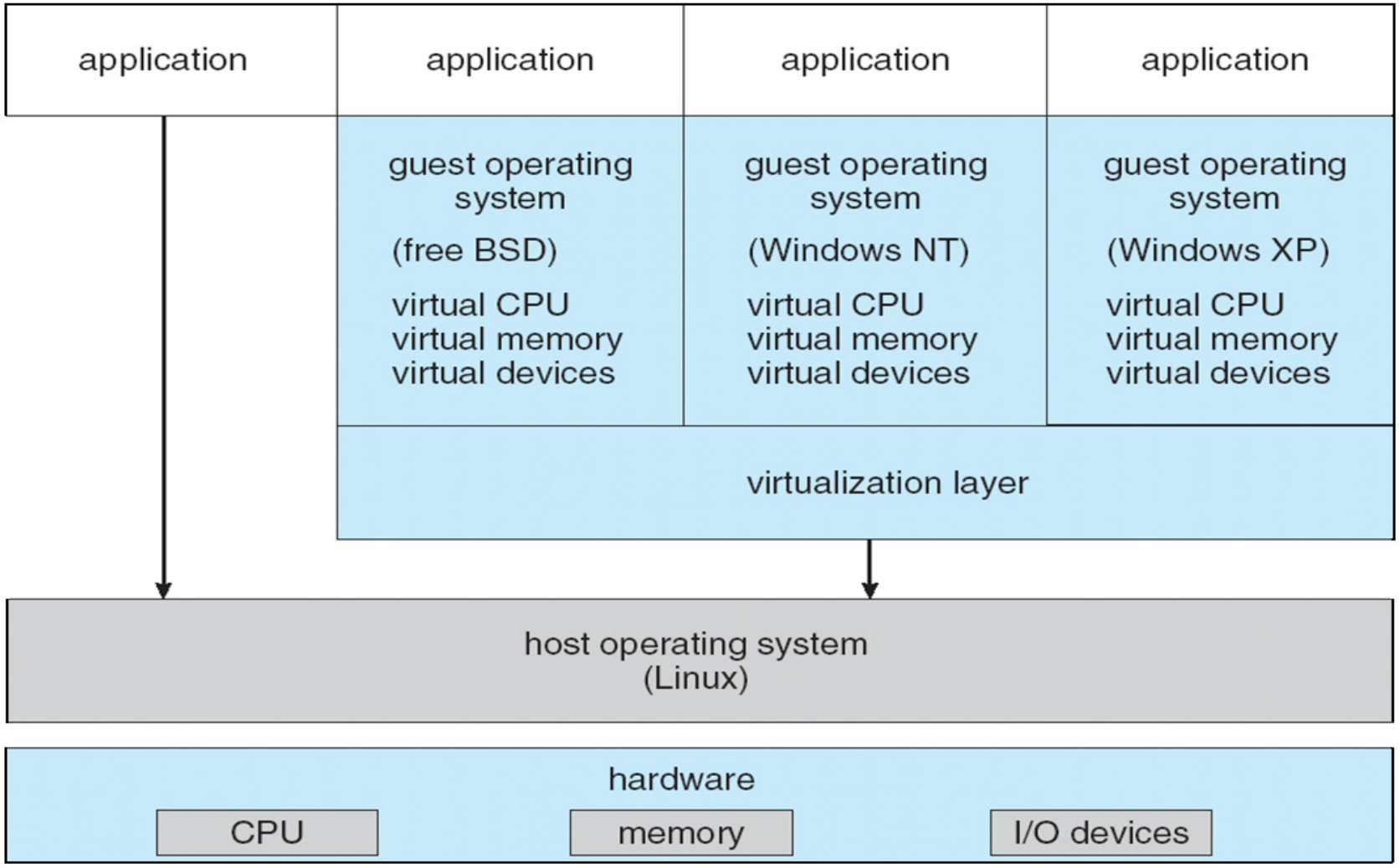
Implementimi i Virtuelizimit

- I vështirë për tu implementuar – duhet të ofroj një duplikat *ekzakt* të makinës
  - Zakonisht lëshohet në modin e shfrytëzuesit, krijon modin virtuel të shfrytëzuesit dhe modin virtuel të kernelit
- Koha mund të jetë problem – më e ngadalshme se makina reale
- Mbështetja harduerike është e nevojshme
  - Më shumë mbështetje -> virtuelizim më i mirë
  - p.sh. AMD ofron modet “host” dhe “guest”

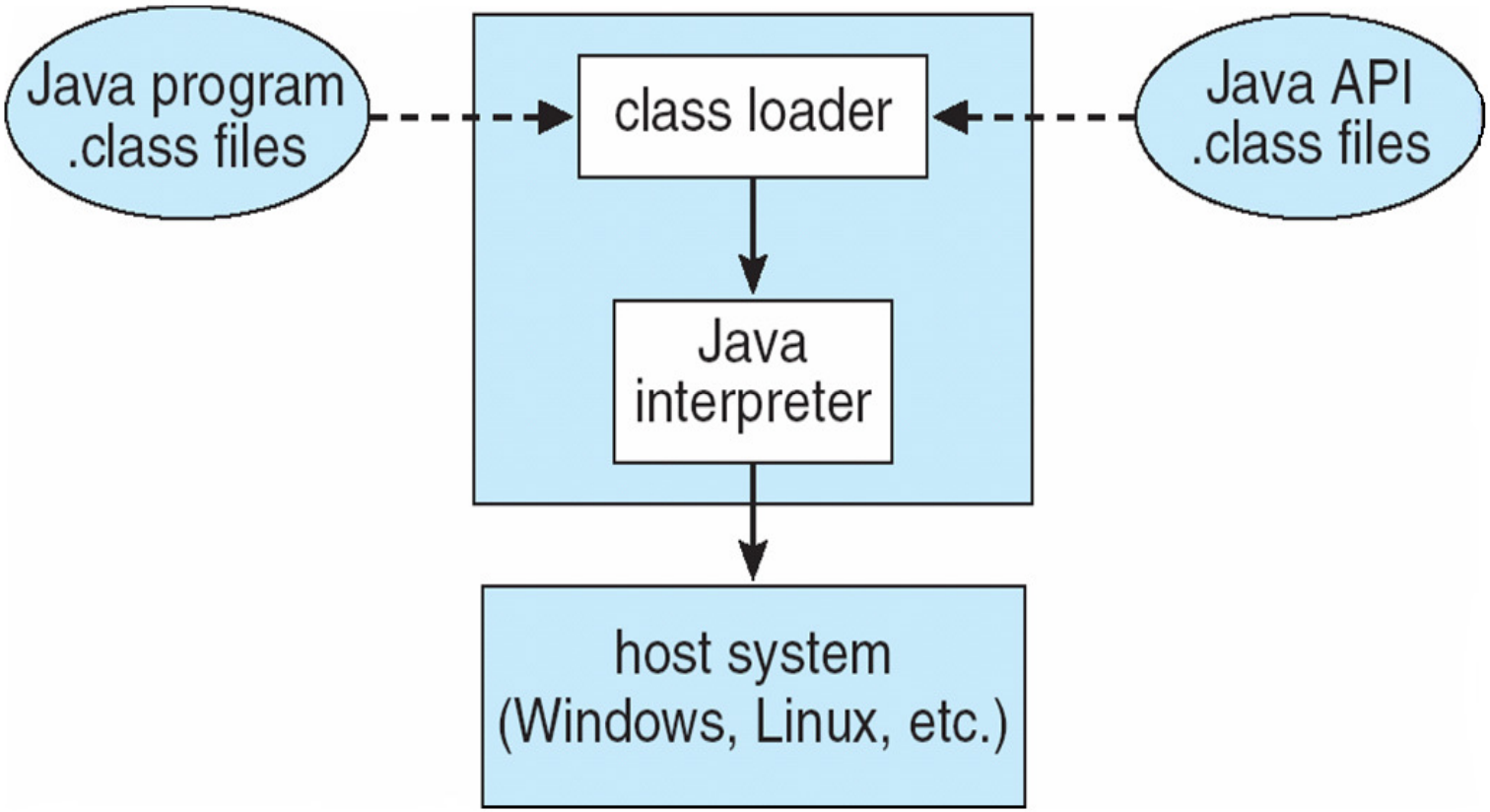
Solaris 10 me Dy Kontejnerë



Arkitektura e VMware



Makina Virtuale Java



Trajtimi i Gabimeve në Sistemin Operativ

- **Trajtimi i gabimeve (debugging)** është gjetja dhe përmirësimi i gabimeve, ose bagave (**bugs**)
- SO gjeneron **log files** që përmbajnë të dhëna për gabimin
- Prishja në aplikacion mund të gjeneroj fajllin **core dump** për kapjet e memories dhe proceseve
- Prishjet në sistemin operativ mund të gjenerojnë fajllin **crash dump** i cili përmban memorien e kernel
- Përveç bllokimeve/prishjeve, akordimi i performancës mund të optimizoj performancën e sistemit
- Ligji i Kernighan: “Gjetja e gabimeve është dyfish më e vështirë se shkrimi i kodit për herë të parë.
- Prandaj, nëse e shkruani kodin sa më mirë të jetë e mundur, ju jeni me definicion jo shumë i zgjuar për t’i gjetur gabimet.”
- DTrace vegla në Solaris, FreeBSD, Mac OS X u lejon instrumentit të drejtpërdrejtë në sistemin e prodhimit
  - **Sonda** ndizet kur të ekzekutohet kodi, duke kapur të dhëna për gjendjen dhe dërgimi i saj tek konsumatorët e këtyre sondave

Solaris 10 dtrace gjatë thirrjes së Sistemit

```
# ./all.d 'pgrep xclock' XEventsQueued
dtrace: script './all.d' matched 52377 probes
CPU FUNCTION
  0 -> XEventsQueued                       U
  0   -> _XEventsQueued                    U
  0     -> _X11TransBytesReadable          U
  0     <- _X11TransBytesReadable          U
  0     -> _X11TransSocketBytesReadable    U
  0     <- _X11TransSocketBytesreadable    U
  0     -> ioctl                           U
  0       -> ioctl                         K
  0         -> getf                        K
  0           -> set_active_fd             K
  0           <- set_active_fd             K
  0         <- getf                        K
  0         -> get_udatamodel              K
  0         <- get_udatamodel              K
...
  0         -> releasef                    K
  0           -> clear_active_fd           K
  0           <- clear_active_fd           K
  0           -> cv_broadcast              K
  0           <- cv_broadcast              K
  0         <- releasef                    K
  0       <- ioctl                         K
  0     <- ioctl                           U
  0   <- _XEventsQueued                    U
  0 <- XEventsQueued                       U
```

Gjenerimi i Sistemeve Operative

- Sistemet Operative janë dizajnuar të lëshohen në çfarëdo lloj klase të makinave.
- Ssistemi duhet të konfigurohet për secilin kompjuter specifik.
- Programi SYSGEN merr të dhëna të cilat lidhen me konfigurimin specifik të sistemit harduerik.
- Booting – startimi i kompjuterit duke e thirrur kernel-in.
- Programi Bootstrap– kod i ruajtur në ROM i cili është në gjendje të lokalizoj kernelin, të vendos atë në memorie, dhe të startoj ekzekutimin.

Ngritja e Sistemit (Boot)

- Sistemi Operativ duhet të jetë i gatshëm tek hardueri ashtu që hardueri të mund të startoj atë:
  - Një pjesë e vogël e kodit – **bootstrap loader**:
    - lokalizon kernelin,
    - e vendos në memorie dhe
    - e starton atë.
  - Nganjëherë proces dy-hapësh ku **boot bloku** në lokacione fikse e vendos (thirrë) bootstrap loader-in.
  - Kur të iniciohet furnizimi me rrymë, ekzekutimi fillon në një lokacion të caktuar memorik.
    - Firmware përdoret për të iniciuar boot kodin.