OPERATIVNI SISTEMI

Slajdovi su kreirani na osnovu knjige “Operativni sistemi, principi unutrašnje organizacije i dizajna, 7. izdanje“, William Stallings, CET, Beograd, 2013.

Osnovni koncepti

Pojam OS

- Operativni sistem je program koji:
  - upravlja izvršavanjem aplikacionih programa

Pojam OS

- Operativni sistem je program koji:
  - služi kao interfejs između programa i hardvera računara

Ciljevi OS

- Operativni sistem treba da obezbedi:
  - Pogodnost
    - da računar bude korisniku pogodniji za korišćenje
  - Efikasnost
    - da se resursi računarskog sistema koriste na efikasan način
  - Mogućnost izvršavanja novih korisničkih aplikacija
    - da je omogućeno dodavanje novih funkcionalnosti sistema nezavisno od ugrađenih osnovnih servisa

OS kao interfejs između korisnika i računara

- Krajnji korisnik vidi računarski sistem kao skup aplikacija
- Aplikacija može biti implementirana kao skup mašinskih instrukcija koji je u potpunosti odgovoran za upravljanje hardverom računara
- Vrlo složeno za implementaciju
- Zato se obezbeđuje skup sistemskih programa, koji obavlja upravljanje hardverom
- Aplikacija se obraća sistemskom programu da obavi određene funkcije
- Najvažniji sistemski program je operativni sistem

OS kao interfejs između korisnika i računara

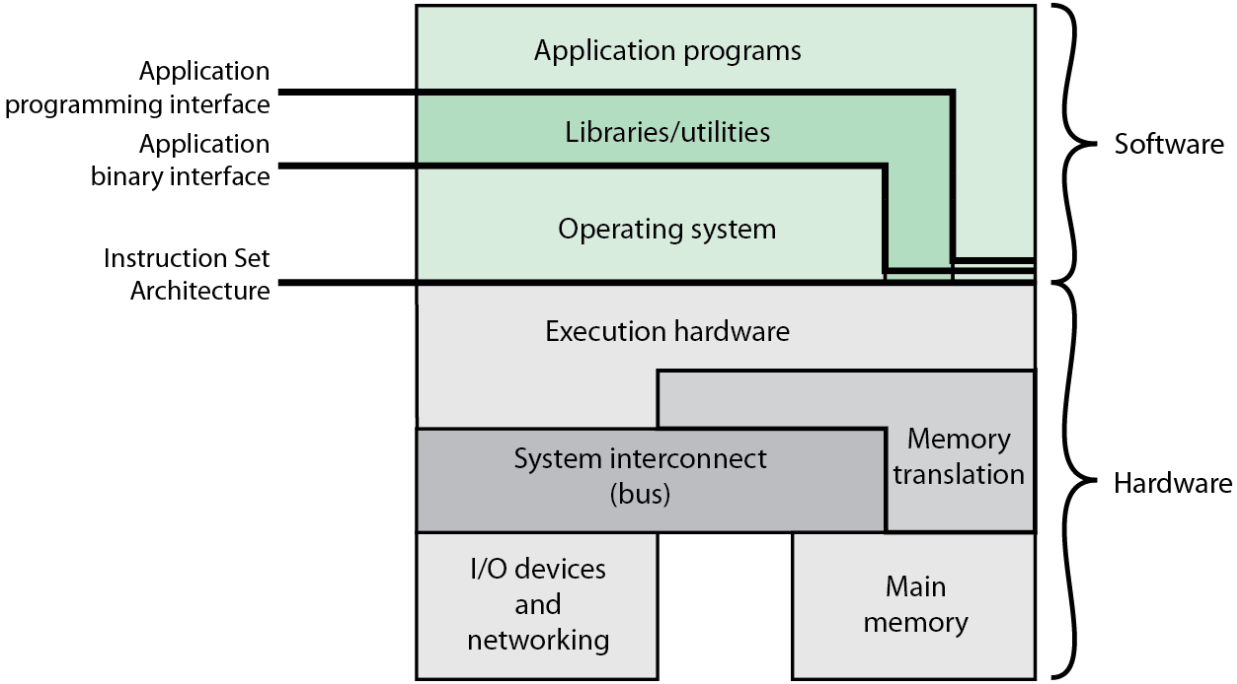


Figure 2.1 Computer Hardware and Software Infrastructure

Programski interfejsi u računarskom sistemu

- Interfejs aplikativnog programiranja (API)
  - specifikacija servisa koje aplikacija može da poziva
  - npr. funkcije u sistemskoj biblioteci
- Binarni interfejs aplikacije (ABI)
  - kao API, ali nižeg nivoa – veza sa operativnim sistemom
  - definiše format komunikacije sa OS (sistemski pozivi, predstavljanja tipova podataka, način poziva funkcija, ...)
- Arhitektura skupa instrukcija (ISA)
  - repertoar instrukcija mašinskog jezika koje hardver podržava

OS kao interfejs između korisnika i računara

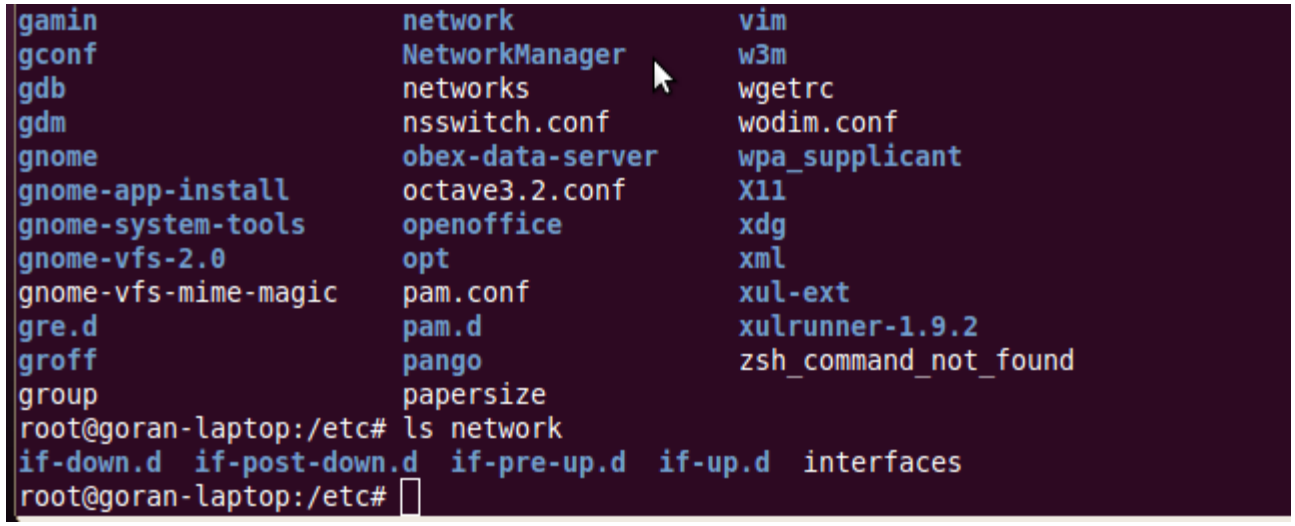
- Dva osnovna načina interakcije korisnika sa operativnim sistemom
  - Komandna linija
  - Grafički interfejs

Komandna linija kao veza ka operativnom sistemu

- Interakcija se vrši unosom tekstualnih komandi

Komandna linija kao veza ka operativnom sistemu

- Interpreter komandi je
  - Ugrađen u kernel ili
  - Poseban program u okviru OS (*shell*)
    - Ovo je varijanta kod Windows i UNIX-baziranih OS



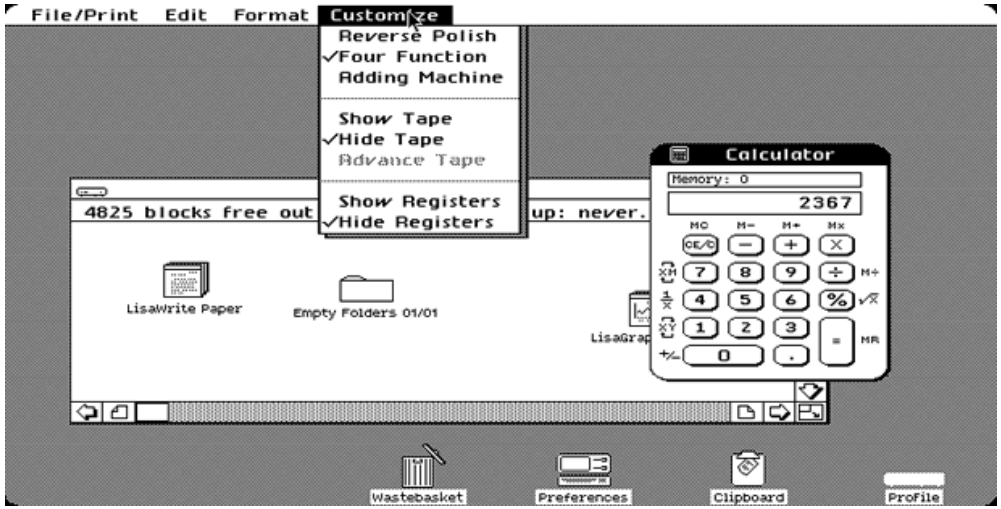
Komandna linija kao veza ka operativnom sistemu

- Način implementacije komandi
  1. Interpreter komandi u sebi sadrži kod za izvršenje komande
  2. Komande se izvršavaju pozivom sistemskih programa
     - Češća varijanta
     - Komandna linija „ne razume“ komande
     - Komandna linija služi samo za identifikaciju fajla koji će biti učitan i izvršen
     - Npr. Linux komanda **`ls test_folder`** startuje program **`ls`** i prosleđuje mu parametar **`test_folder`**

Grafički interfejs kao veza sa operativnim sistemom

- Korisnik korišćenjem miša ili prsta manipuliše grafičkim elementima koji predstavljaju programe, fajlove, direktorijume i sistemske funkcije

Grafički interfejs kao veza sa operativnim sistemom



1983.

Sada

Standardne usluge OS

- Izvršavanje programa
  - OS upravlja resursima koje program koristi (učitavanje instrukcija i podataka u glavnu memoriju, inicijalizacija U/I uređaja, ...)
- Pristup U/I uređajima
  - OS pruža interfejs za pristup U/I uređajima
  - Programer zahteva U/I operaciju kroz interfejs, a OS je zadužen za dalju komunikaciju sa U/I uređajem
- Pristup fajlovima
  - OS vodi računa o internoj strukturi uređaja za skladištenje i načinu skladištenja i obezbeđuje mehanizme kontrole pristupa fajlovima

Standardne usluge OS

- Deljeni pristup sistemu
  - OS obezbeđuje zaštitu resursa od neovlašćenog pristupa i rešava konflikte u takmičenju za resurse
- Upravljanje greškama
  - OS obezbeđuje reakciju na hardverske i softverske greške (prekid programa, ponavljanje operacije, prijava greške)
- Nadzor sistema
  - OS prikuplja statistiku upotrebe resursa i nadgleda performanse

OS kao upravljač resursa

- Računar je skup resursa za prenos, skladištenje i obradu podataka
- OS je odgovoran za upravljanje ovim resursima
- OS funkcioniše na isti način kao i ostali softver
- OS je program ili skup programa koje izvršava procesor
- OS se često odriče upravljanja i zavisi od procesora koji će mu dozvoliti da ponovo preuzme upravljanje
- Razlika u odnosu na drugi softver je namena
  - Softver OS sadrži instrukcije čijim izvršavanjem se upravlja resursima sistema (raspoređivanje aplikacija, zauzimanje memorije, ...)

OS kao upravljač resursa

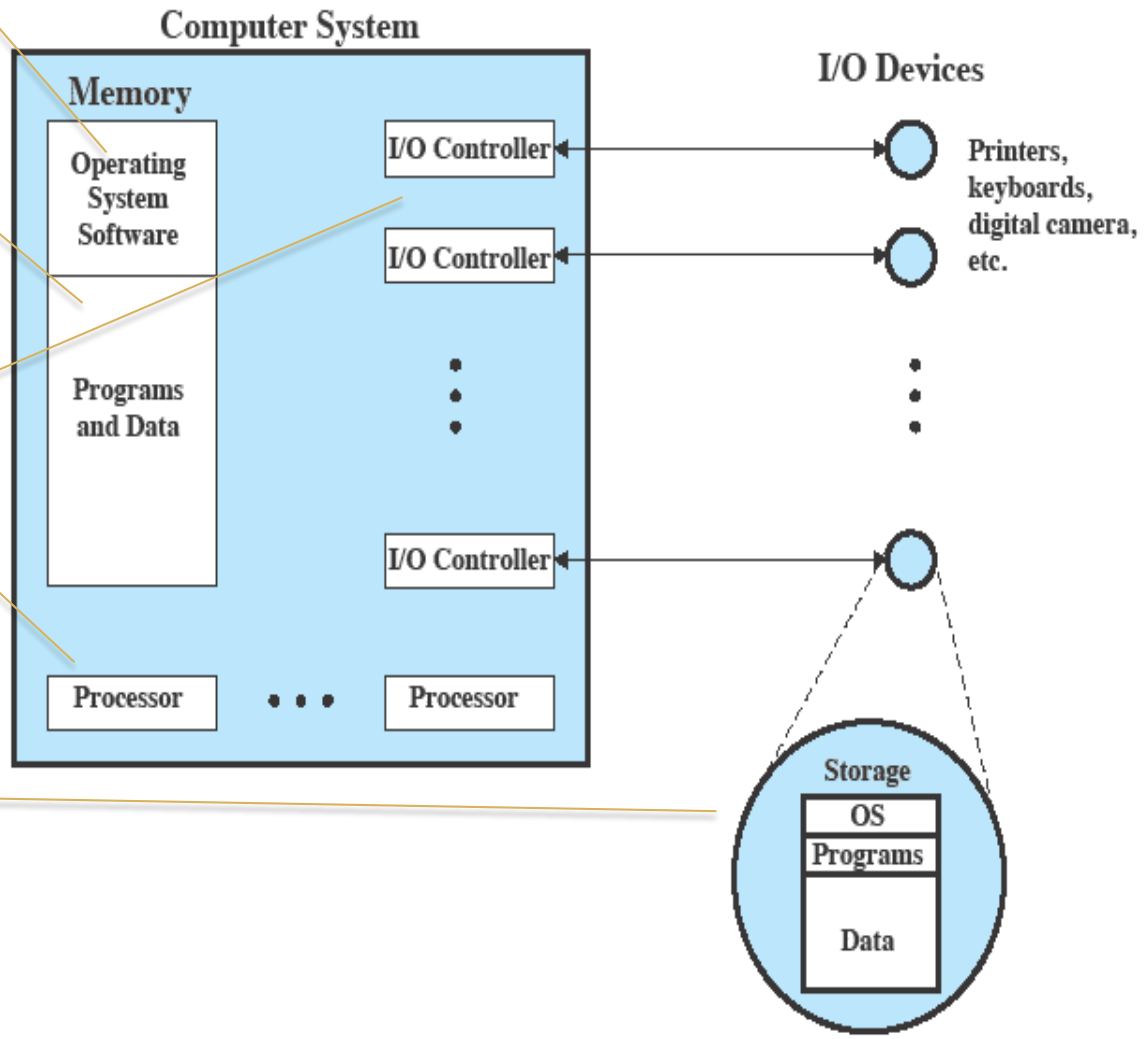
Sadrži jezgro (kernel) OS (najčešće korišćene funkcije) i druge delove OS koji se trenutno koriste

OS i hardver upravljaju dodelom ovog dela memorije

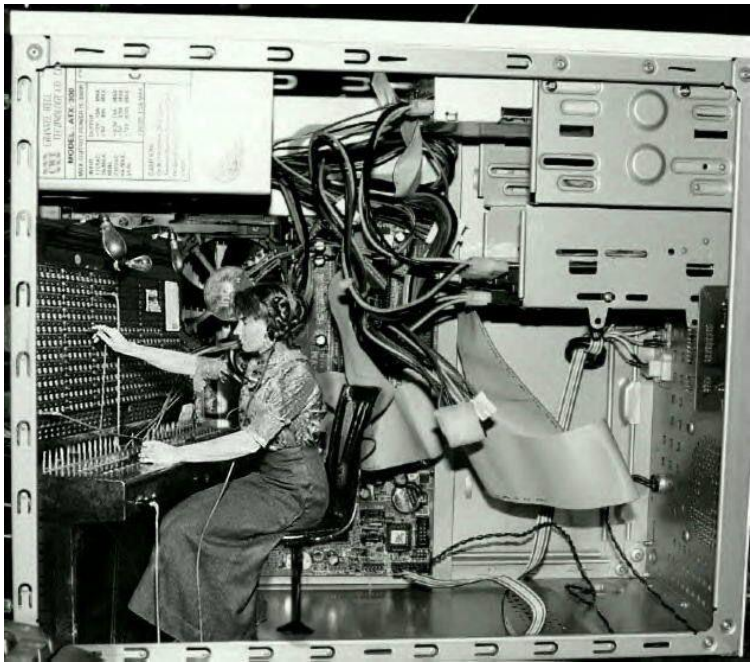
OS odlučuje kada program može da koristi U/I uređaj

OS odlučuje koliko će vremena procesor posvetiti izvršavanju određenog programa

OS upravlja pristupom i upotrebom fajlova



Istorija razvoja OS



- Serijska obrada
- Jednostavni sistemi paketne obrade
- Multiprogramirani sistemi paketne obrade
- Sistemi sa deljenjem vremena

Serijska obrada

- U početku razvoja računara:
  - računari nisu imali OS
    - korisnici su direktno pristupali hardveru
  - samo jedan korisnik je u jednom trenutku imao pristup računaru

Serijska obrada

- Problemi:
  - korisnici se upisuju na spisak za rezervisanje računara
    - teško organizovati raspored ako izvršavanje traje duže od planiranog
    - neracionalno korišćenje ako traje kraće od planiranog
  - postavljanje programa – značajan deo vremena se trošio na postavljanje programa da se izvrši
  - Direktno se upravlja hardverskim resursima
  - Nema istovremenog izvršavanja više programa

Jednostavni sistemi paketne obrade

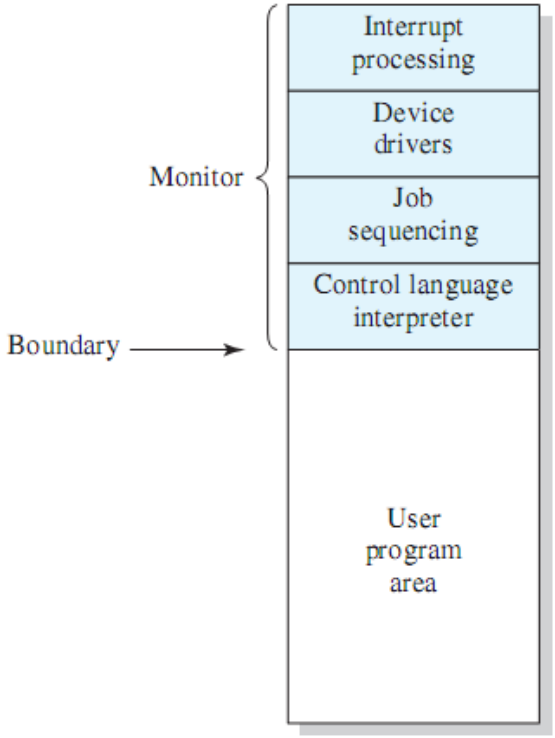
- Korak ka efikasnijem korišćenju računara
- Korisnik više ne pristupa računaru direktno
- Korisnici predaju poslove operatoru, koji ih kao skup programa postavlja na računar

Jednostavni sistemi paketne obrade

- Pripremljen skup programa izvršava poseban softver – **monitor**
  - monitor je program sa posebnom namenom da upravlja izvršenjem drugih programa
  - monitor izvršava jedan po jedan program
  - svaki program je kreiran tako da se vrati na monitor nakon što se završi
  - nakon toga, monitor automatski učitava naredni program

Funkcije monitora

- Monitor kontroliše sekvencu programa
- Poseban deo memorije je namenjen smeštanju programa monitora
- Monitor učitava korisnički program i predaje mu kontrolu
- Korisnički program vraća kontrolu monitoru



Funkcije procesora

- Procesor izvršava instrukcije iz dela memorije u kojoj je smešten monitor
- Kada monitor učita korisnički program, procesor će izvršiti instrukciju grananja koja određuje da procesor nastavi izvršavanje instrukcija sa početka korisničkog programa
- Procesor izvršava instrukcije u korisničkom programu dok ne završi sve ili do pojave greške
- „*Predati kontrolu korisničkom programu*“ znači da procesor prelazi da izvršava instrukcije u delu memorije gde je smešten korisnički program
- „*Vraćanje kontrole monitoru*“ znači da procesor narednu instrukciju izvršava iz programa monitora

Hardverska podrška za monitor

- Zaštita memorije
  - korisnički program ne sme da pristupa memoriji u kojoj je monitor. Hardver procesora mora da detektuje i spreči takve instrukcije
- Vremenski brojač
  - zaštita da jedan program ne koristi procesor predugo
  - nakon isteka brojača, program se zaustavlja i kontrola predaje monitoru
- Privilegovane instrukcije
  - deo instrukcija može da izvrši samo monitor
- Prekidi
  - daju više fleksibilnosti jer omogućuju da se upravljanje predaje i oduzima programima

Režim izvršavanja

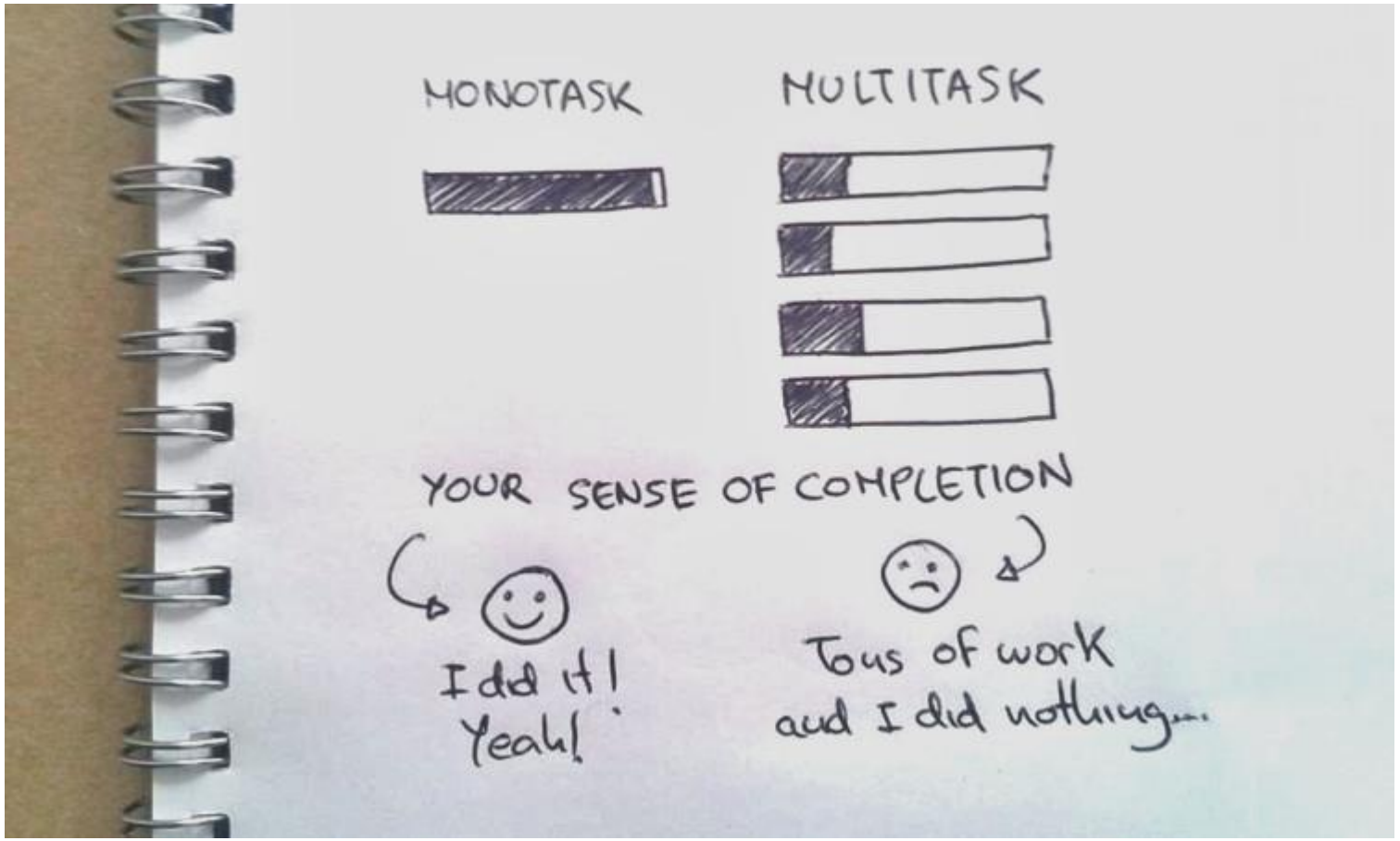
- Da bi se sproveli koncepti zaštite memorije i privilegovanih instrukcija, u OS su uvedena dva režima izvršavanja:
  - korisnički režim
    - korisnički programi se izvršavaju u ovom režimu
    - nije moguće pristupiti određenim delovima memorije
    - nije moguće izvršiti određene instrukcije
  - režim kernela
    - monitor se izvršava u ovom režimu
    - može da se pristupa zaštićenoj memoriji
    - mogu da se izvršavaju privilegovane instrukcije

Multiprogramirani sistemi paketne obrade

- Ideja je da se programi više ne delegiraju monitoru sekvencijalno
  - Istovremeno je više programa postavljeno za izvršavanje
  - I dalje u jednom trenutku procesor može da izvršava samo jedan program

Multiprogramirani sistemi paketne obrade

- Da li dobijamo ovo?



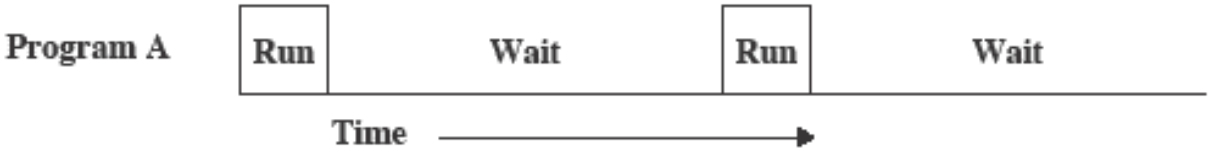
Multiprogramirani sistemi paketne obrade

- Kada je procesor posvećen samo jednom programu najveći deo vremena je besposlen
- Dok traje izvršavanje U/I operacije, procesor je slobodan
- U/I uređaji su puno sporiji od procesora
- Primer trajanja operacija programa

Čitanje reda iz fajla	15 μs
Izvršavanje 100 instrukcija	1 μs
Upis reda u fajl	15 μs
Ukupno:	31 μs

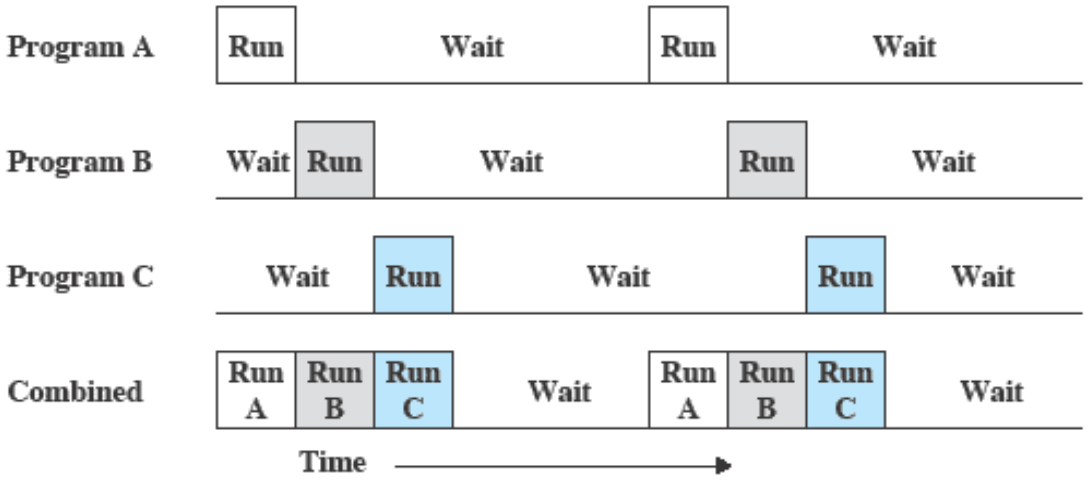
- Iskorišćenje procesora: $\frac{1}{31} = 0.032 = 3.2\%$

Jednoprogramiranje



- Procesor izvršava instrukcije dok ne dođe do U/I instrukcije
- Zatim mora da sačeka završetak U/I operacije da bi nastavio rad

Multiprogramiranje



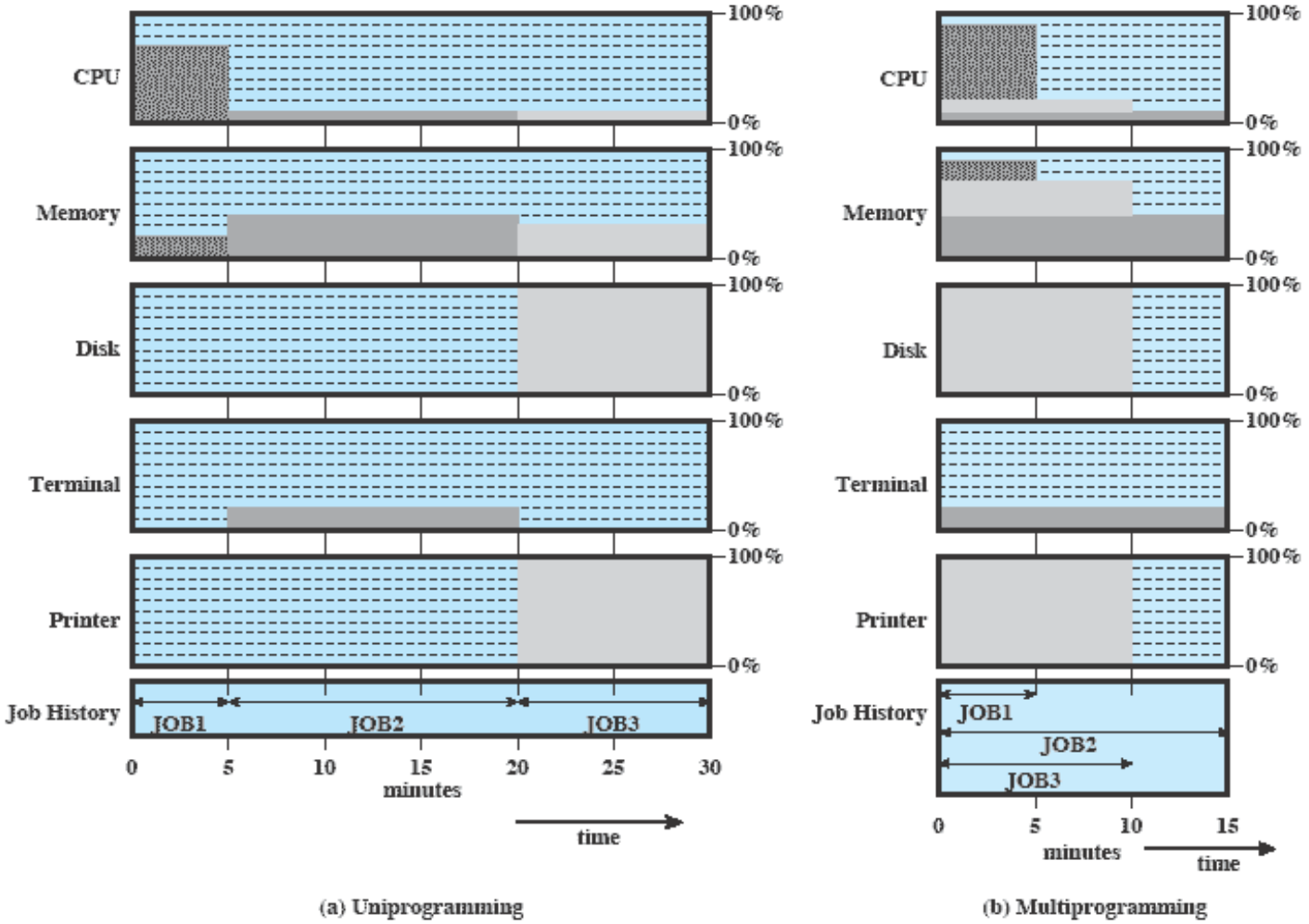
- Višeprocesna obrada podataka
- Pored OS, u memoriju se smešta više korisničkih programa
- Kada jedan program čeka na završetak U/I operacije, procesor može da pređe da izvršava drugi program

Poređenje performansi
Jednoprogramiranje naspram multiprogramiranja

- Primer izvršavanja tri programa

	Program 1	Program 2	Program 3
Vrsta posla	veliko izračunavanje	veliki U/I	veliki U/I
Trajanje	5 minuta	15 minuta	10 minuta
Potrebna memorija	50 MB	100 MB	75 MB
Pristupa disku?	Ne	Ne	Da
Pristupa terminalu?	Ne	Da	Ne
Pristupa štampaču?	Ne	Ne	Da

Histogram iskorišćenja



Efekti multiprogramiranja na iskorišćenost resursa

	Jednoprogramiranje	Multiprogramiranje
Upotreba procesora	20%	40%
Upotreba memorije	33%	67%
Upotreba diska	33%	67%
Upotreba štampača	33%	67%
Ukupno vreme	30 minuta	15 minuta
Propusna moć	6 poslova/sat	12 poslova/sat
Prosečno vreme odziva	18 minuta	10 minuta

Hardverska podrška za multiprogramiranje

- U/I prekidi
- DMA (direktan pristup memoriji)
- Omogućuju da procesor izda naredbu za U/I operaciju i nastavi da izvršava drugi posao, dok U/I operaciju vrši kontroler uređaja
- Po završetku U/I operacije, kontroler uređaja postavlja prekid i kontrola se predaje programu za obradu prekida u OS
- OS će predati kontrolu drugom poslu

OS podrška za multiprogramiranje

- Da bi se realizovalo multiprogramiranje, OS treba da omogući:
  - Upravljanje memorijom
    - poslovi (ili njihovi delovi) koji se izvršavaju trebaju biti u memoriji istovremeno
  - Raspoređivanje poslova
    - ako je više poslova spremno za izvršavanje, potreban je algoritam raspoređivanja koji odlučuje koji posao se sledeći izvršava i koliko dugo

Sistemi sa deljenjem vremena

- Razvijeni 1960-ih godina za podršku višekorisničkom pristupu računarskim resursima
- Deljenje vremena
  - procesor se deli između više korisnika
- Korisnici istovremeno pristupaju sistemu preko terminala, a OS prepliće izvršavanje korisničkih programa
- Pošto je vreme ljudske reakcije relativno sporo, vreme odziva je blisko onom u jednokorisničkom režimu

Multiprogramiranje naspram deljenja vremena

	Multiprogramirana paketna obrada	Deljenje vremena
Šta obezbeđuje	Multiprogramiranje	Višekorisnički rad
Glavni cilj	Što veća iskorišćenost procesora	Smanjenje vremena odgovora
Izvor naredbi za OS	Naredbe jezika za upravljanje poslovima dobijaju se uz posao	Naredbe se unose na terminalu

Realizacija sistema sa deljenjem vremena

- CTSS (*Compatible Time Sharing System)*
  - Razvijen 1961.
  - Učitava i izvršava jedan po jedan korisnički program
- Raspodela vremenskih isečaka
  - Sistemski generator takta pravi prekid na svake 0.2 sekunde
  - Pri svakom prekidu OS preuzima upravljanje i može da dodeli procesor drugom korisniku

Zahtevi za savremeni OS

- Deljenje vremena i multprogramiranje su uveli standardne zahteve za dizajn savremenih OS
- Multiprogramiranje
  - uvelo zahtev deljenja resursa
  - konkurentni pristup, takmičenje za resurse zaštita od štetnog preplitanja, ...
- Deljenje vremena
  - uvelo zahtev višekorisničkog pristupa resursima
  - zaštita i raspoređivanje
- Današnji OS su dizajnirani da ispune ove zahteve

Savremeni OS

Linux

Windows

Mac

Savremeni OS

- GNU/Linux
  - Linus Torvalds ga razvio 1991.
  - Baziran na AT&T Unix
  - Otvorenog koda
  - Različite distribucije
    - Sadrže Linux kernel i kolekciju dodatnog softvera
    - Debian, Fedora, Ubuntu, CentOS, ...

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix –

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

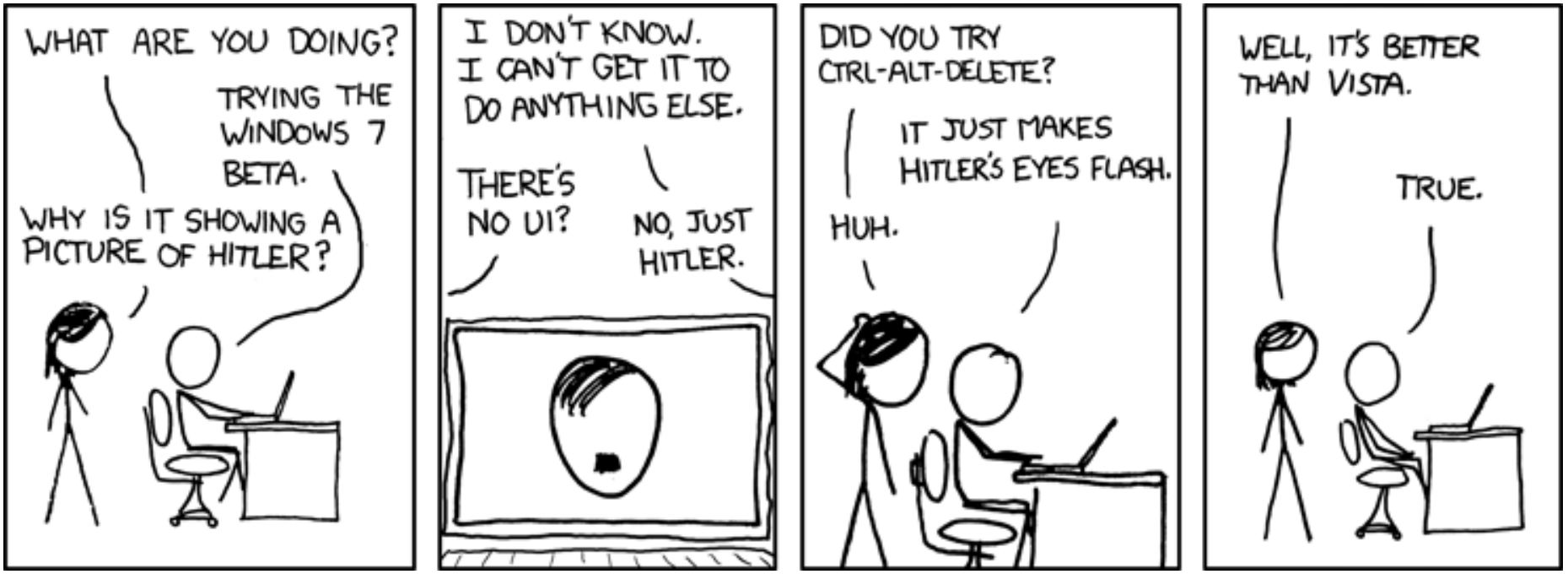
I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them 🙂

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes – it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Savremeni OS

- Windows
  - Razvijen od strane Microsoft korporacije
  - Globalno najzastupljeniji OS na Desktop računarima (> 90%)
  - Nije dostupan izvorni kod

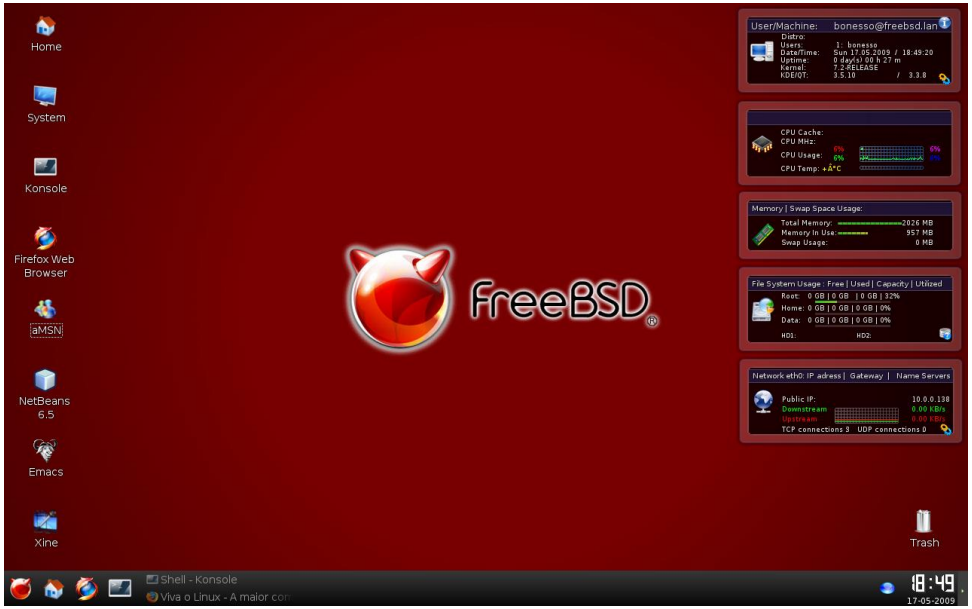

WHAT ARE YOU DOING?
TRYING THE WINDOWS 7 BETA.
WHY IS IT SHOWING A PICTURE OF HITLER?
I DON'T KNOW. I CAN'T GET IT TO DO ANYTHING ELSE.
THERE'S NO UI?
NO, JUST HITLER.
DID YOU TRY CTRL-ALT-DELETE?
IT JUST MAKES HITLER'S EYES FLASH.
HUH.
WELL, IT'S BETTER THAN VISTA.
TRUE.

Savremeni OS

- MacOS
  - Apple razvio za rad na svojim Macintosh računarima
  - Unix-baziran

BSD Unix

- BSD Unix
  - Razvijen po uzoru na Unix 1978.
  - Otvoren kod od 1994.
  - Darwin, glavna komponenta kernela Mac OS X je bazirana na BSD Unixu
  - Distrubucije
    - FreeBSD
    - NetBSD
    - OpenBSD
    - DragonflyBSD



Savremeni OS

- Solaris
  - Razvijen 1991. od strane Sun Microsystems po uzoru na Unix
  - OpenSolaris
    - Solaris otvorenog koda publikovan 2005.
  - Otkako je 2009. Oracle kupio Sun, ne distribuira se sa otvorenim kodom
  - Aktuelna verzija je Solaris 11

Savremeni OS

- OS za mobilne platforme
  - Android
    - Google razvio kao OS otvorenog koda
    - Izvršava se na različitim mobilnim platformama
    - ≈ 50% globalna zastupljenost među svim uređajima
  - iOS
    - Apple razvio za svoje iPhone i iPad uređaje
    - Razvijen na bazi Mac OS

Implementacija OS

- Prvi operativni sistemi pisani u asembleru
- Danas je većina pisana u jezicima višeg nivoa
- Različiti delovi mogu biti u različitim jezicima
  - Niži nivoi kernela u jezicima nižeg nivoa
  - Sistemski programi u jezicima višeg nivoa

Implementacija OS

- MS-DOS
  - Pisan u Intel 8088 asembleru
- Linux
  - Pisan najvećim delom u C (71%)
  - Jedan deo funkcija najnižeg nivoa u asembleru
  - Delovi implementirani korišćenjem C++, Lisp, Perl, Python, Fortran jezika
- Windows
  - Implementiran u C-u, ali se oslanja na koncepte objektno-orijentisanog dizajna

Implementacija OS

- Koliko je komplikovano?
- Red Hat Linux 7.1 distribucija sadržala 30 miliona linija koda
  - Od toga kernel sadrži 2.4 miliona linija koda (8%)
- Kada bi se ponovo implementiralo
  - Procenjeno na 8000 čovek-godina
  - Koštalo bi 1.5 milijardu dolara

You need to master many details – as with any big project – but the differences are quantitive, not qualitative.

Robert Love, Linux Kernel Development Book

Glavna dostignuća

- Osnovna dostignuća na koja se oslanjaju savremeni OS:
  - **Procesi**
  - Upravljanje memorijom
  - Zaštita informacija i bezbednost
  - Raspoređivanje i upravljanje resursima

Pojam procesa

- Program u izvršavanju
- Primerak programa koji se izvršava na računaru
- Entitet koji se može dodeliti i izvršavati na procesoru
- Jedinica aktivnosti koju karakterišu naredbe za izvršavanje, tekuće stanje i dodeljeni skup sistemskih resursa

Komponente procesa

- Proces se sastoji iz tri dela:
  - Izvršni program
    - instrukcije koje procesor treba da izvrši
  - Podaci
    - pridruženi podaci koji su potrebni programu pri izvršavanju (promenljive)
  - Kontekst izvršenja (stanje) procesa
    - interni podaci pomoću kojih OS može da upravlja procesom
    - kompletno stanje procesa u bilo kom trenutku
    - tu je sačuvan sadržaj registara procesa (programski brojač, registri podataka) da bi proces mogao da nastavi izvršavanje nakon što je prekinut
    - Sadrži prioritet procesa, informaciju da li proces čeka na U/I, ...

Upravljanje procesom (pojednostavljen model)

Lista procesa sadrži jedan zapis za svaki proces koji sadrži pokazivač na lokaciju bloka memorije koji sadrži taj proces

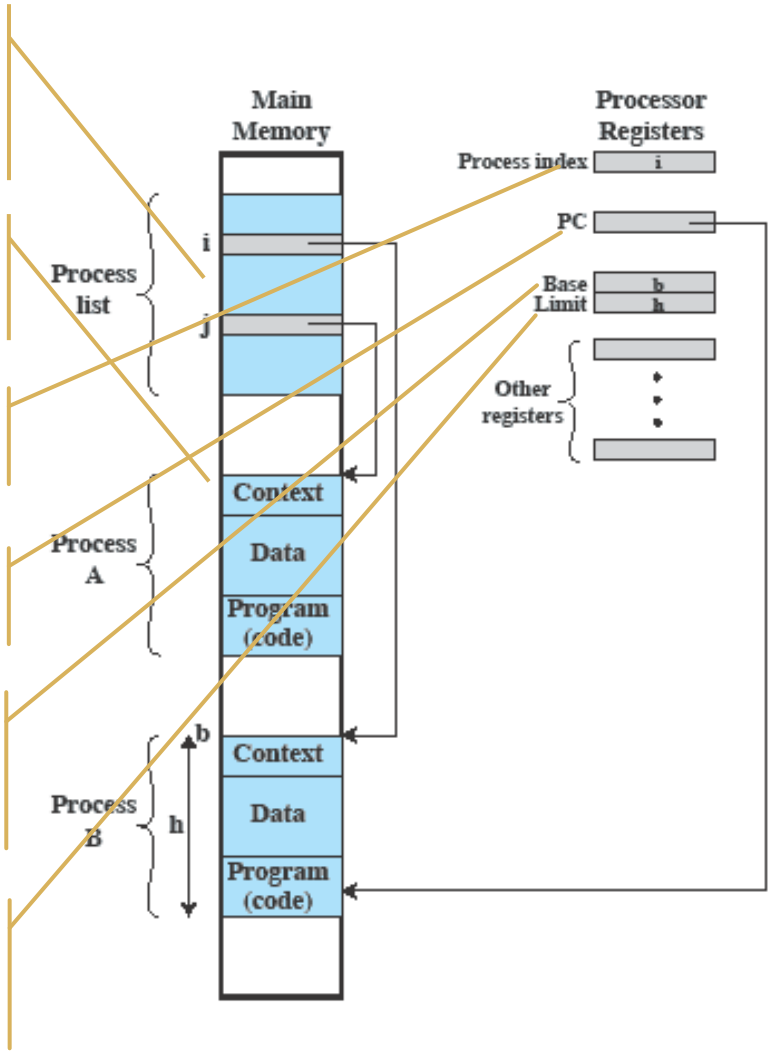
Kontekst procesa može biti smešten sa samim procesom ili odvojeno

Proces koji se trenutno izvršava na procesoru

Naredna instrukcija u aktivnom procesu

Početna adresa bloka u kojem je proces. Služi za relativno referenciranje programskog brojača i svih podataka

Dužina bloka u kojem je proces. Služi za zaštitu pristupa memoriji. Ne smeju se referencirati lokacije izvan granice



Višenitna obrada

- Proces je podeljen na niti koje se izvršavaju konkurentno
- Nit
  - izvršna jedinica posla
  - sadrži kontekst i sopstvenu oblast podataka
  - izvršava se sekvencijalno i može se prekinuti
- Proces
  - skup jedne ili više niti i dodeljenog sistemskog resursa
  - kreiranjem niti programer ima veću kontrolu nad modularnošću aplikacije i vremenskim usklađivanjem događaja u njoj

Glavna dostignuća

- Osnovna dostignuća na koja se oslanjaju savremeni OS:
  - Procesi
  - **Upravljanje memorijom**
  - Zaštita informacija i bezbednost
  - Raspoređivanje i upravljanje resursima

Upravljanje memorijom

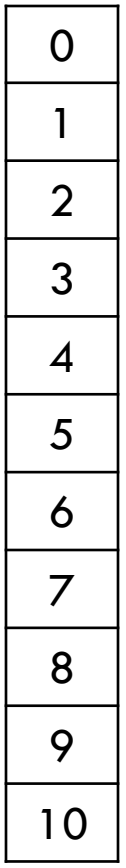
- Odgovornosti OS pri upravljanju memorijom
  - Izolacija procesa
    - Proces ne sme da pristupi delu memorije rezervisanom za drugi proces
  - Automatsko dodeljivanje i upravljanje
    - Korisnička aplikacija ne mora direktno da vodi računa o upravljanju memorijom
  - Zaštita i kontrola pristupa
    - Delovima memorije mogu na različite načine da pristupe različiti korisnici
  - Dugotrajna memorija
    - OS mora da omogući programima da skladište informacije na duže vreme, po isključenju računara
    - Fajl sistem je implementacija dugotrajne memorije

Virtuelno adresiranje

- Mehanizam koji omogućuje da programi vrše adresiranje sa logičke tačke gledišta
  - program ne mora da vodi računa o količini fizički raspoložive glavne memorije
  - program ne mora da vodi računa o stvarnoj adresi u glavnoj memoriji u koju je smešten
    - stvarna adresa se određuje u trenutku izvršavanja a ne u trenutku pisanja ili prevođenja programa

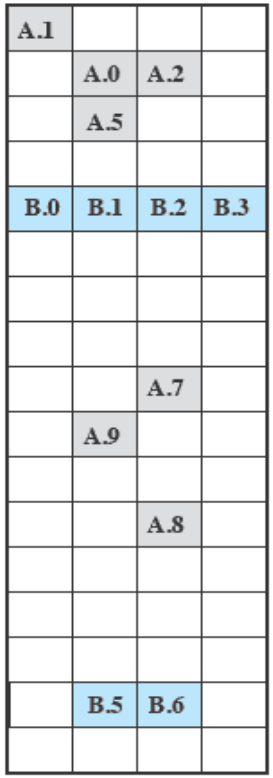
Straničenje

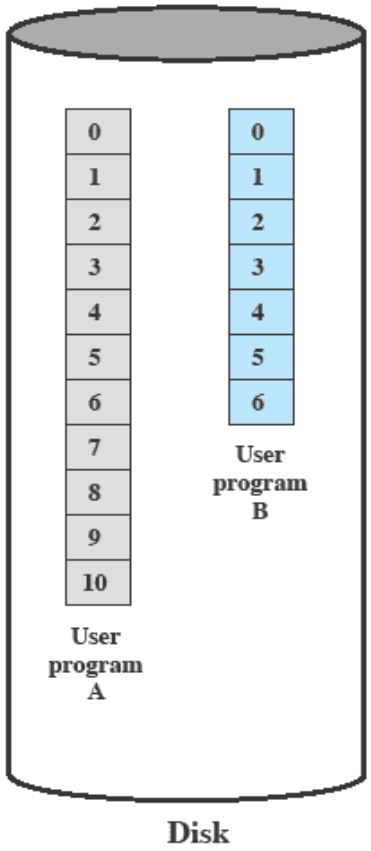
- Proces je podeljen u određeni broj blokova fiksne dužine, koji se nazivaju stranice
- Program adresira reč pomoću virtuelne adrese
  - sastoji se od broja stranice i pomeraja unutar nje
  - stranica može biti smeštena bilo gde u glavnoj memoriji
- Sistem straničenja pruža dinamičko mapiranje virtuelne adrese koja se koristi u programu na realnu adresu (fizička adresa u glavnoj memoriji)



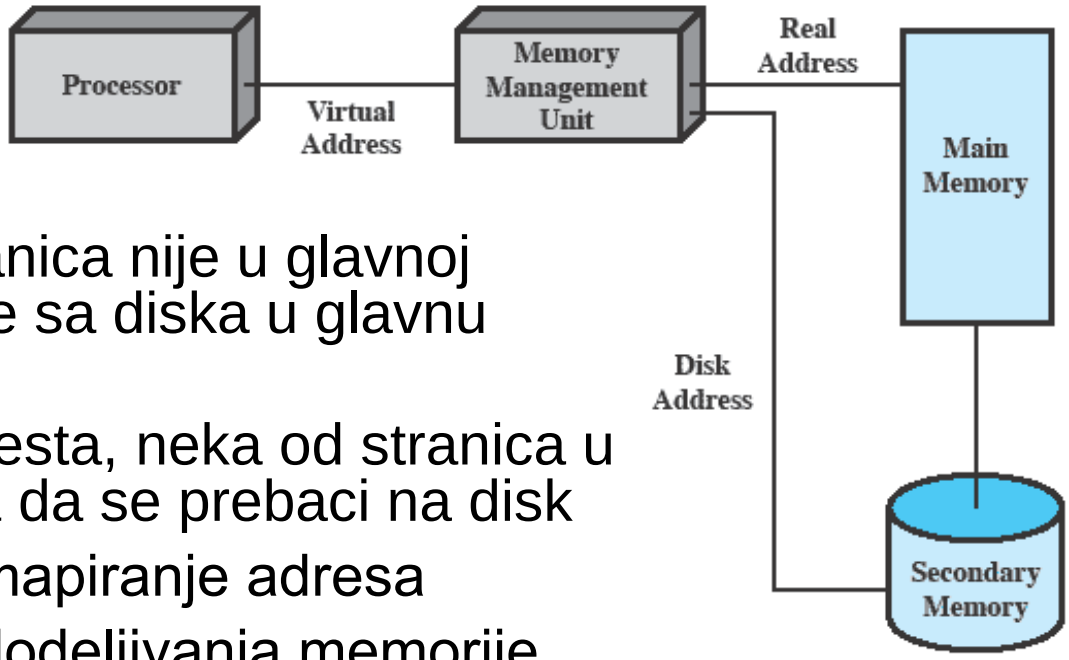
Virtuelna memorija

- Ne moraju sve stranice procesa da budu celo vreme u glavnoj memoriji
- Sve stranice se nalaze na disku, a samo neke u glavnoj memoriji
- Ukoliko se adresira stranica koja nije u glavnoj memoriji, hardver za upravljanje memorijom to otkriva i učitava u memoriju potrebnu stranicu





Adresiranje virtuelne memorije



- Ako referencirana stranica nije u glavnoj memoriji, prebacuje se sa diska u glavnu memoriju
- Ako nema dovoljno mesta, neka od stranica u glavnoj memoriji mora da se prebaci na disk
- Hardver obezbeđuje mapiranje adresa
- OS određuje politiku dodeljivanja memorije
  - koje i koliko stranica će se naći u memoriji
  - algoritam zamene stranica u glavnoj memoriji
  - algoritam prebacivanja stranice iz glavne memorije na disk
  - ...

Glavna dostignuća

- Osnovna dostignuća na koja se oslanjaju savremeni OS:
  - Procesi
  - Upravljanje memorijom
  - **Zaštita informacija i bezbednost**
  - Raspoređivanje i upravljanje resursima

Zaštita informacija i bezbednost

- Raspoloživost
  - zaštita sistema od prekida rada
- Poverljivost
  - korisnici ne smeju da čitaju podatke za koje nemaju pravo pristupa
- Integritet podataka
  - zaštita podataka od nedozvoljene izmene
- Autentičnost
  - provera identiteta korisnika i validnost poruka ili podataka

Glavna dostignuća

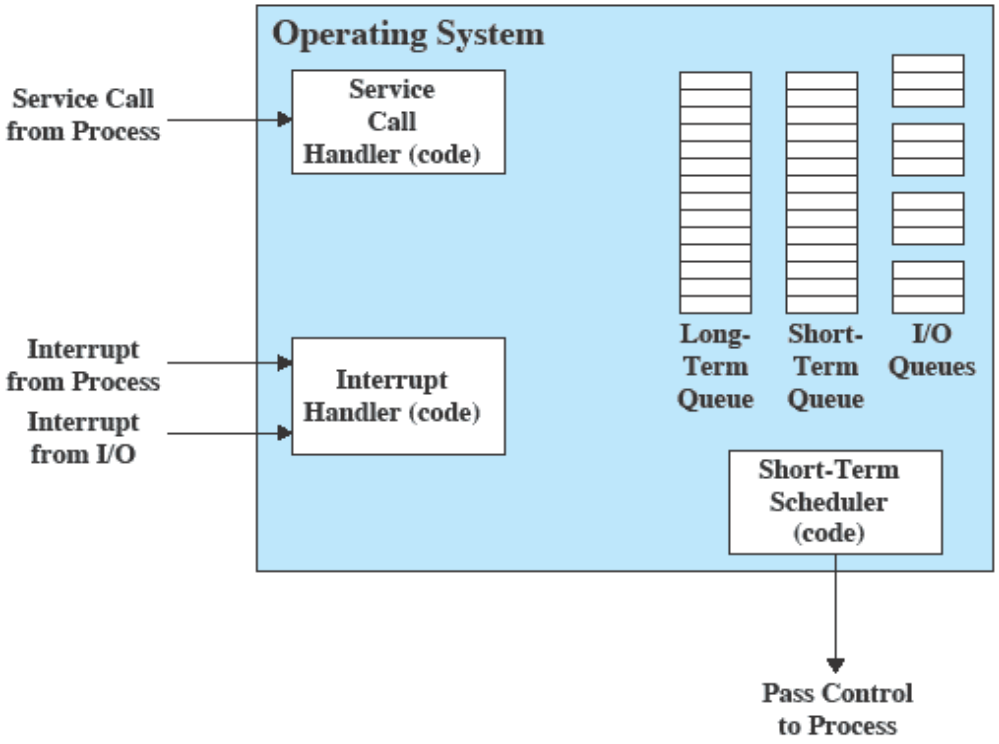
- Osnovna dostignuća na koja se oslanjaju savremeni OS:
  - Procesi
  - Upravljanje memorijom
  - Zaštita informacija i bezbednost
  - **Raspoređivanje i upravljanje resursima**

Upravljanje resursima

- OS upravlja resursima (glavna memorija, procesor, U/I uređaji) i raspoređuje njihovu upotrebu prema procesima, pri čemu mora da obezbedi:
  - Nepristrasnost
    - poslovi sličnog tipa koji se takmiče za jedan resurs, treba da dobiju približno isti i fer pristup tom resursu
  - Različitost odziva
    - namerna pristrasnost dodelom resursa određenom procesu, što će omogućiti da se bolje ispuni ukupan skup zahteva
  - Efikasnost
    - OS pokušava da uveća propusnu moć, a smanji vreme odziva
    - ovo su međusobno suprostavljeni kriterijumi, OS treba da nađe balans

Elementi OS za raspoređivanje

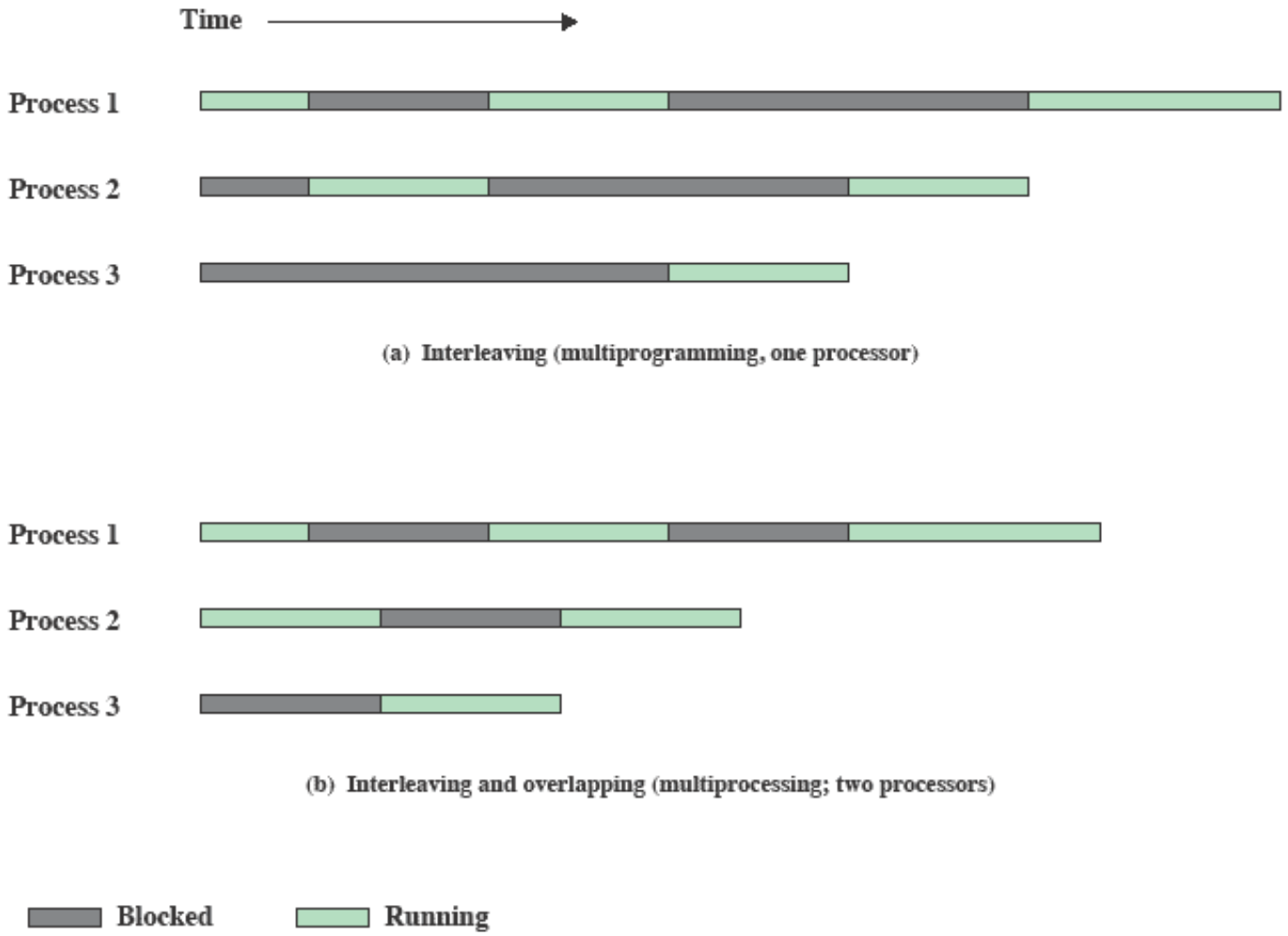
- U kratkoročnom redu su procesi spremni za izvršavanje
  - kružno dodeljivanje
  - prema prioritetu
- U dugoročnom redu su novi poslovi koji čekaju da se ubace u kratkoročni red
- Za svaki U/I uređaj formira se red procesa koji čekaju na upotrebu uređaja
- Operativni sistem opslužuje prekide i sistemske pozive
- Po opsluživanju prekida ili sistemskog poziva, kratkoročni raspoređivač određuje proces koji će naredni da se izvrši



Simetrična višeprocesna obrada (SMP)

- Bavi se raspoređivanjem procesa na hardveru sa simetričnim multiprocesorima
- Procesori dele istu glavnu memoriju i U/I uređaje
- Svi procesori mogu da obavljaju istu funkciju
- Više procesa može paralelno da se izvršava
- OS vodi računa o raspoređivanju niti ili procesa na pojedinačne procesore i sinhronizaciji između njih

Multiprogramiranje i višeprocesna obrada



Prednosti višeprocesne obrade

- Performanse
  - ako se delovi posla mogu odraditi paralelno, sistem sa više procesora će postići bolje performanse
- Raspoloživost
  - Otkaz jednog procesora neće zaustaviti sistem
  - Pošto svi procesori mogu da obavljaju iste funkcije, sistem će nastaviti da radi sa smanjenim performansama
- Postepeno poboljšanje sistema
  - moguće je poboljšati performanse dodavanjem procesora
- Skaliranje
  - proizvođači mogu ponuditi opseg proizvoda različite cene i performansi na osnovu broja procesora

Dizajn OS za SMP

- Simultani konkurentni procesi ili niti
  - strukture kernela treba da podrže mogućnost konkurentnog pristupa od strane više procesa koji se istovremeno izvršavaju na različitim procesorima
- Raspoređivanje
  - Svaki procesor može da sprovodi raspoređivanje, što komplikuje politiku raspoređivanja
- Sinhronizacija
  - Međusobna isključivost i signaliziranje više aktivnih procesa istovremeno koji imaju pristup deljenim adresnim prostorima i U/I resursima

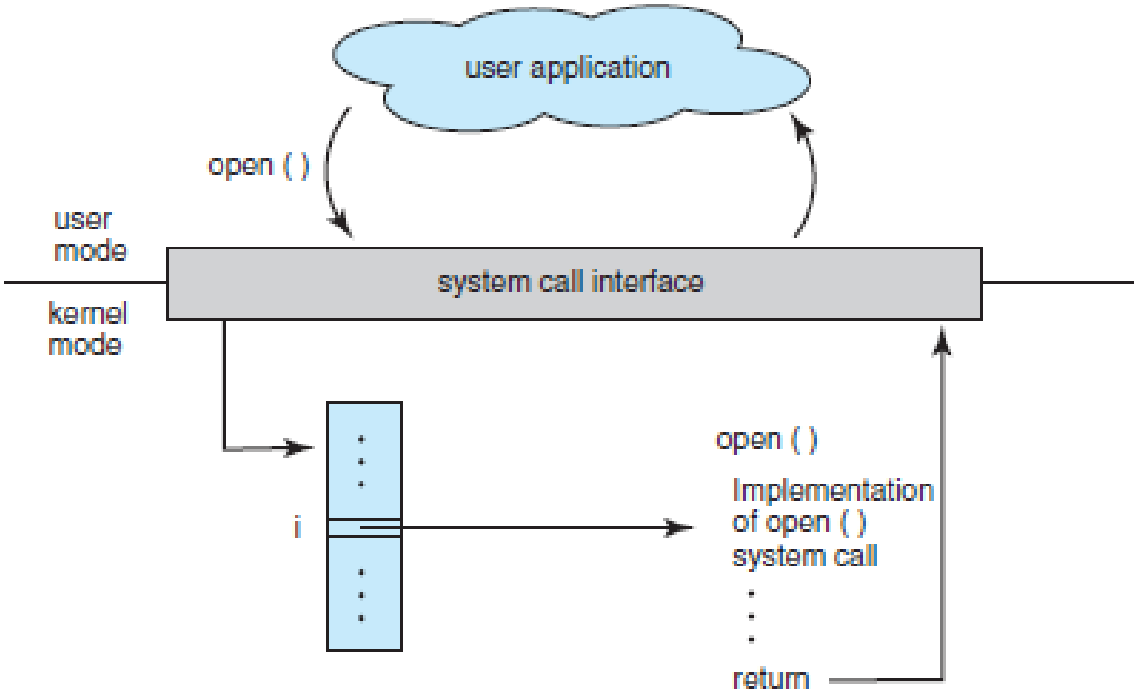
Dizajn OS za SMP

- Upravljanje memorijom
  - iskoristiti paralelizam hardvera za bolju performansu
  - koordinirati procesore pri straničenju, kako bi bili konzistentni kada više procesora deli istu stranicu
- Pouzdanost i otpornost na greške
  - OS treba da reaguje na gubitak procesora smanjenjem performanse i ponovnim raspoređivanjem

Komunikacija korisničke aplikacije sa OS

- Sistemski pozivi
  - OS pruža skup servisa koji se mogu pozvati iz sloja korisničkih aplikacija i koji obavljaju određenu funkcionalnost OS
  - Mogu se pozivati direktno ili
  - Indirektno korišćenjem sistemske biblioteke

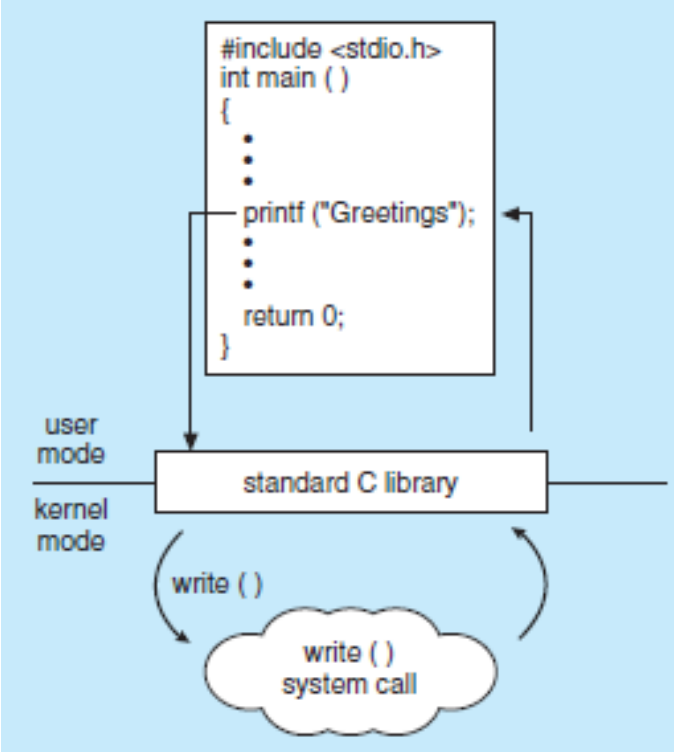
Direktan poziv sistemskih poziva



- Korisnička aplikacija zahteva uslugu OS slanjem sistemskog poziva
- OS u režimu kernela izvršava traženi zadatak i vraća odgovor aplikaciji

Sistemska biblioteka

- Sadrži programski kod za upućivanje sistemskih poziva OS
- Služi kao srednji sloj između korisničke aplikacije i OS
- Omotač oko sistemskih poziva
- Korisnička aplikacija ne mora da vodi računa o specifikaciji sistemskog poziva



Tipovi sistemskih poziva

- Upravljanje procesima
  - Završi, prekini
  - Učitaj, izvrši
  - Kreiraj
  - Preuzmi i postavi atribute procesa
  - Čekaj određeno vreme
  - Čekaj na događaj
  - Signaliziraj događaj
  - Zauzmi i oslobodi memoriju

Tipovi sistemskih poziva

- Upravljanje fajlovima
  - Kreiraj i obriši
  - Otvori i zatvori
  - Pročitaj, upiši i pozicioniraj se
  - Preuzmi i postavi atribute fajla

Tipovi sistemskih poziva

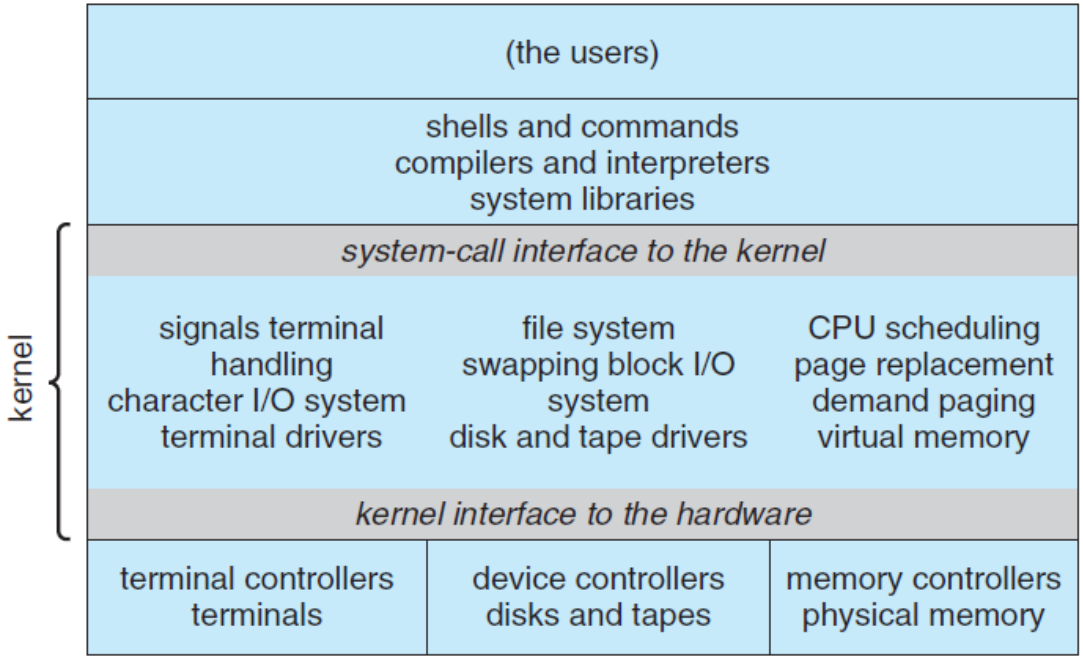
- Upravljanje uređajima
  - Zatraži i otpusti uređaj
  - Pročitaj i upiši
  - Preuzmi i postavi atribute uređaja
  - Logički dodaj i izbaci uređaj iz sistema

Tipovi sistemskih poziva

- Evidencija informacija
  - Preuzmi i postavi sistemsko vreme i datum
  - Uzmi i postavi podatke iz sistema
- Komunikacija
  - Kreiraj i obriši komunikacionu vezu
  - Pošalji i primi poruku

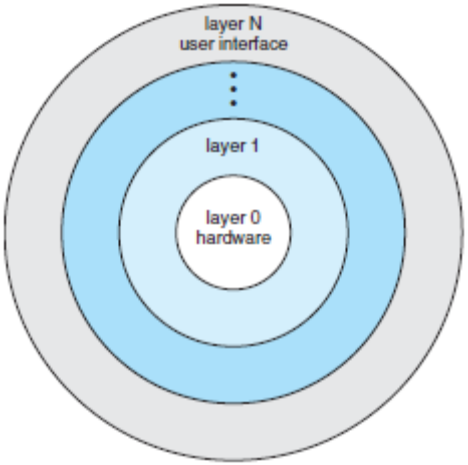
Struktura kernela OS

- Monolitno jezgro
  - u jednom velikom jezgru sve funkcionalnosti OS
    - raspoređivanje, fajl sistem, upravljanje memorijom, ...
    - Primer strukture UNIX sistema



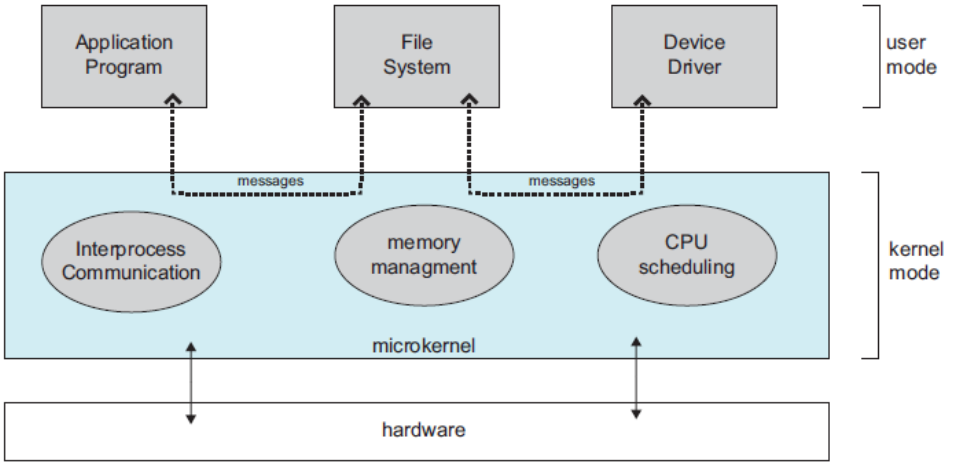
Struktura kernela OS

- Slojevita struktura
  - Funkcionalnosti podeljene u hijerarhijski organizovane slojeve
  - Svaki sloj
    - Poziva operacije koje pruža niži nivo
    - Pruža operacije koje mogu biti pozvane od višeg nivoa
  - Problem je
    - što određeni delovi komuniciraju dvosmerno, pa nije moguće odrediti koji deo treba da bude na višem hijerarhijskom nivou
    - Sporo jer se prolazi kroz niz sistemskih poziva kroz slojeve



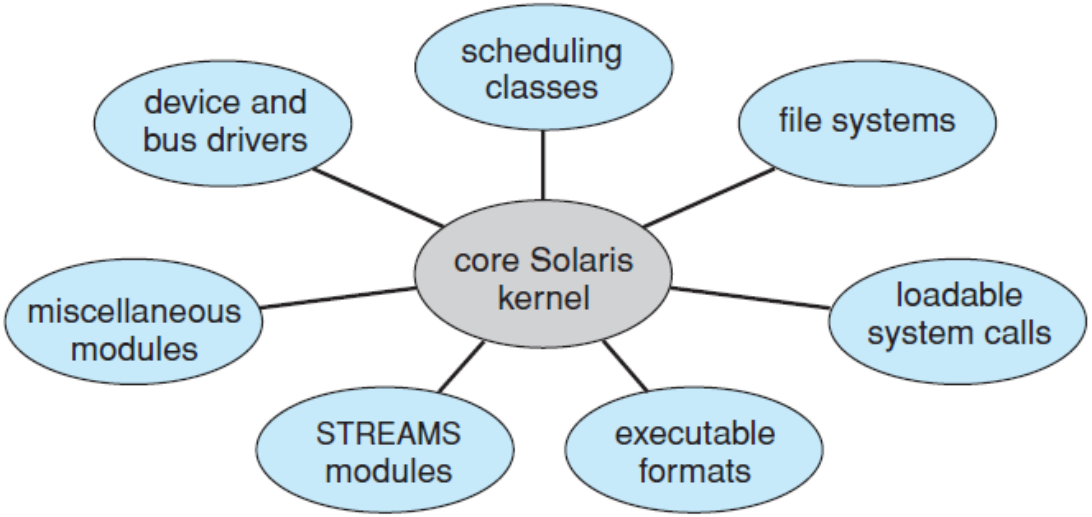
Struktura kernela OS

- Mikrokernel
  - Jezgro sadrži samo nekoliko osnovnih funkcija
  - Ostali servisi se obezbeđuju preko procesa koji rade u korisničkom režimu
  - Pojednostavljuje implementaciju, pruža fleksibilnost, dobro prilagođen za distribuirana okruženja
  - Slabe performanse zbog toga što servisi iz korisničkog moda moraju preko sistemskih poziva da obavljaju operacije OS



Struktura kernela OS

- Modularna struktura
  - Funkcionalnosti podeljene u module koji se dinamički učitavaju
  - Kernel uvek sadrži ključne funkcije, a ostale se učitavaju dinamički u toku izvršavanja
  - Sistem podeljen u nezavisne celine, kao kod slojevite strukture, ali svaka celina može da komunicira sa svakom
  - Linux, Windows, Mac OS X, Solaris ovako strukturirani



Struktura kernela OS

- Hibridna struktura
  - Većina sistema u sebi kombinuje različite strukture
  - Linux i Solaris su monolitni, ali i modularni
  - Windows je najvećim delom monolitan, ali ima deo funkcija koje se izvršavaju u korisničkom režimu kao kod mikrokernela

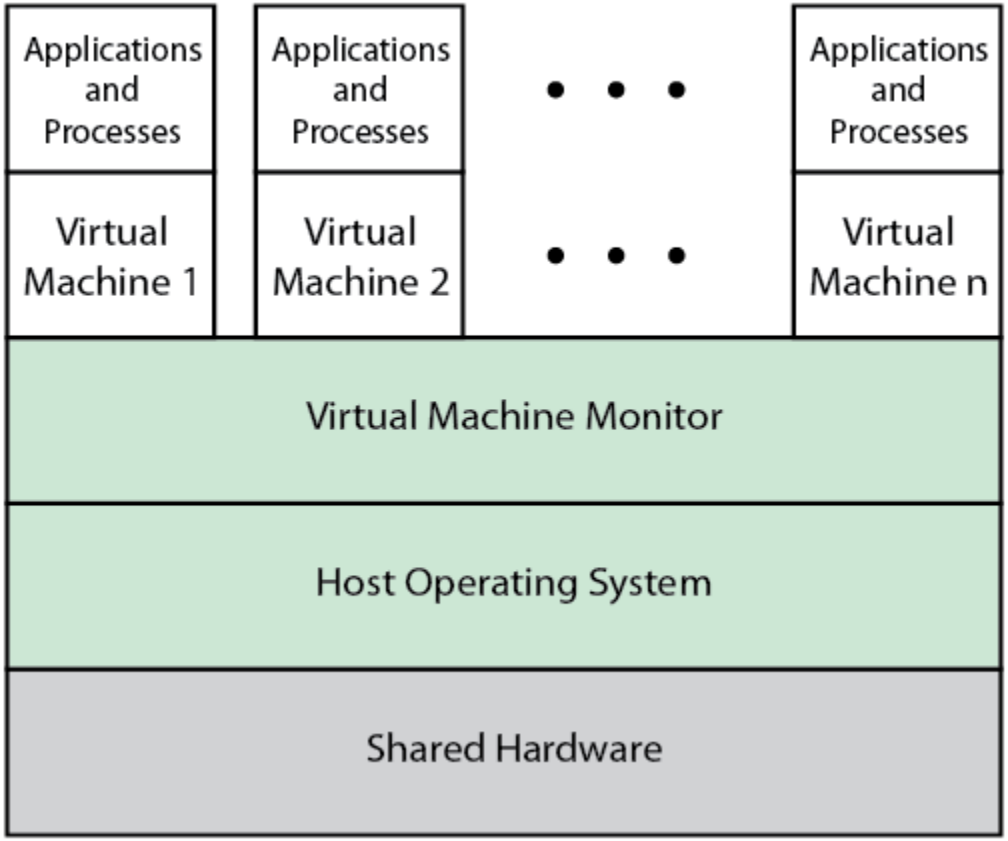
Pokretanje OS

- Kako pri pokretanju računara, hardver zna gde je kod OS da bi krenuo da ga izvršava?
- Bootstrap loader
  - Program koji se prvi pokreće i koji je odgovoran za učitavanje OS u memoriju i pokretanje
  - Učitavanje realizovano u dva nivoa
    - Jednostavni *bootstrap loader* zapisan u ROM memoriji koji se prvi pokreće (BIOS, UEFI)
    - Ovaj *loader* je odgovoran da sa predefinisane lokacije na disku učita kompleksniji *loader* koji učitava OS

Virtuelizacija

- Omogućuje da se na jednom računaru istovremeno izvršava više operativnih sistema ili više instanci jednog OS
- Na taj način moguće je na istoj platformi izvršavati više aplikacija koje rade na različitim OS
- *Host* operativni sistem može da podrži više virtuelnih mašina
  - svaka mašina ima karakteristike posebnog OS i, u nekim verzijama virtuelizacije, posebne hardverske platforme

Koncept virtuelne mašine



- Svaka VM izvršava svoj OS
- VMM obezbeđuje vezu ovih virtuelnih OS sa hardverom (preko *host* OS)

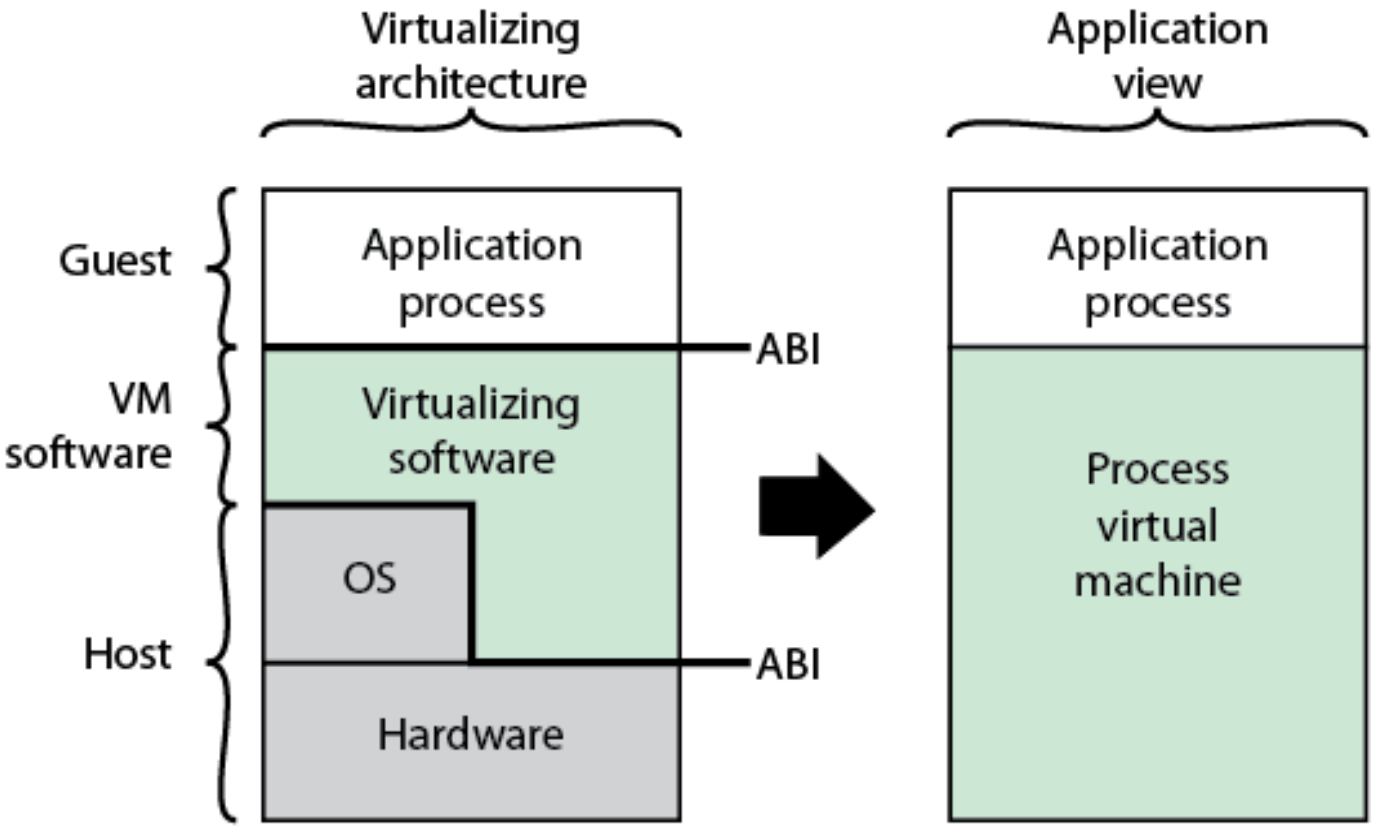
Pojam „mašine“

- Iz perspektive aplikacije, mašinu definišu:
  - svojstva jezika visokog nivoa i sistemski bibliotečki pozivi
  - API definiše mašinu onako kako je vidi aplikacija
- Iz perspektive procesa, koji izvršava prevedeni aplikativni program, mašina je:
  - virtuelni memorijski prostora dodeljen procesu
  - procesorski registri koje može da koristi
  - Skup mašinskih instrukcija korisničkog nivoa koje može da izvršava
  - Skup sistemskih poziva OS koje može da poziva za U/I
  - ABI definiše mašinu onako kako je vidi proces
- Iz perspektive OS, mašina je:
  - fajl sistem i U/I resursi kojima procesi pristupaju
  - stvarna memorija i U/I resursi koje sistem adresira za procese
  - ISA predstavlja interfejs između OS i mašine

Procesna virtuelna mašina

- Virtuelna platforma za izvršavanje jednog procesa
- Primeri
  - Java VM
  - Microsoft .NET framework
- Stvara se kada i proces i ukida kada se on završi
- Kompajliranje aplikacije stvara virtuelni binarni kod
- Kod se izvršava u VM, tako da može da se izvrši na svakoj platformi za koju postoji VM

Procesna virtuelna mašina



Sistemska virtuelna mašina

- Jedna hardverska platforma može da podrži više izolovanih gostujućih OS
- Primeri
  - VMware
  - VirtualBox
- Virtuelizujući softver je *host* određenom broju gostujućih OS

Sistemska virtuelna mašina