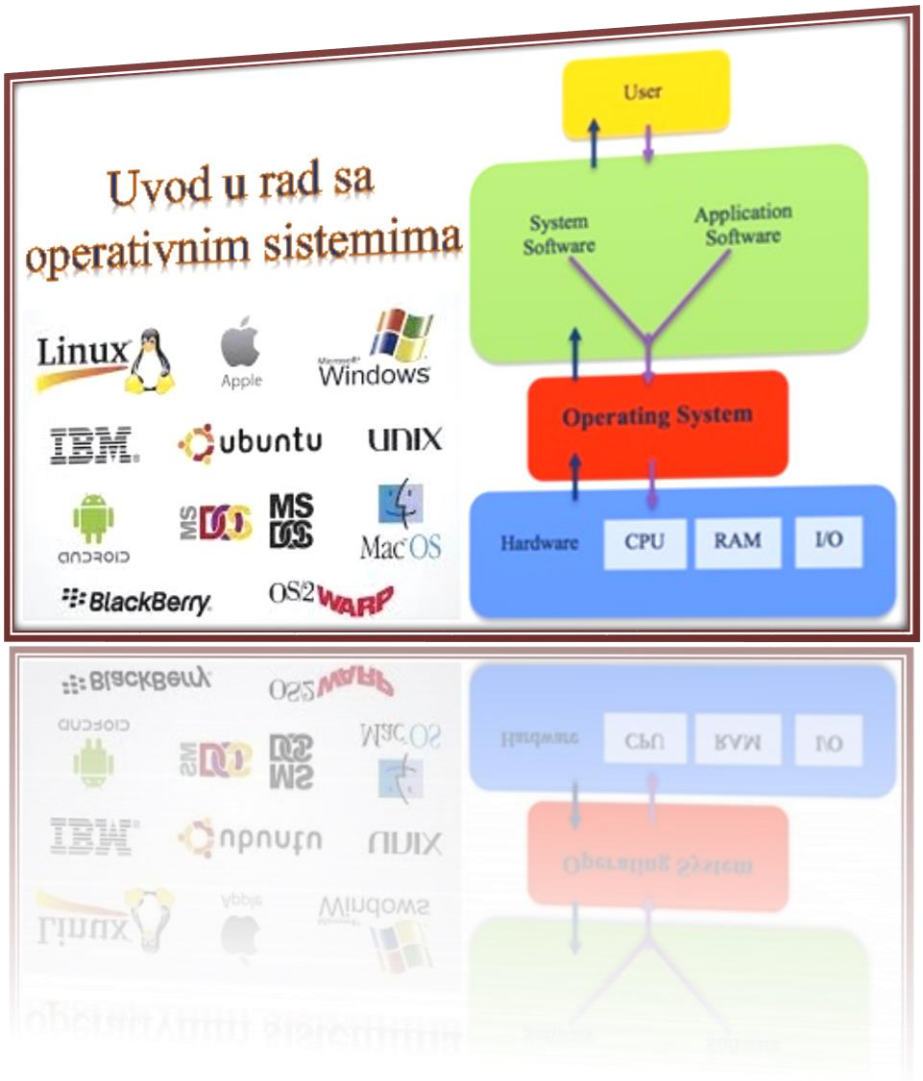
OPERATIVNI SISTEMI

resavska škola XXI/CopyPaste metod



Treće *onLine* izdanje
Banjaluka, *mojTim*, decembar 2022.

Ovaj priručnik ima isključivo edukativnu namjenu.

Ovdje prezentovani materijali su preuzeti sa *mreže*. Ako želite da preuzmete neke od tekstova potrebno bi bilo da pronađete autora i sa njim se konsultujete oko daljeg prezentovanja materijala.
Autore možete pronaći koristeći reference, koje su date na kraju priručnika (klikom na sadržaj koji vam se čini interesantnim.)

Uvod u rad sa operativnim sistemima

Sadržaj

Šta je računar i šta je OS

Računar je uređaj (hardware) koji obrađuje, pamti ili razmjenjuje informacije. Način na koji to radi je određen u programski. Program je niz komandi koje određuju šta treba uraditi sa podacima.
Dva neodvojiva dijela računara su:

- hardver – uređaj (hardware) i
- softver – program (software).

Pošto hardver ne može da radi bez softvera, a softver nema smisla bez hardvera jasno je da oni čine računar. Na primjer, ako zamislimo da je ljudsko biće računar, onda bi hardver predstavljale ćelije (tkiva, organi, ...), a misli i ideje bi predstavljali softver.

Računar je mašina koja može pamtiti i obrađivati informacije. U širem smislu računar (*computer*) se definiše kao matematička mašina.
Češće se koristi nešto uža definicija koja **računar definiše kao elektronsku, digitalnu, reprogramibilnu mašinu koja može da obavlja logičko matematičke operacije, unos, obradu i pamćenje podataka**.

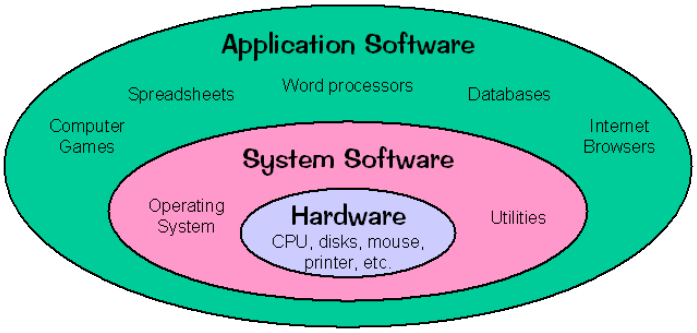
- elektronska znači da osnov građe računara čine elektronske komponente
- digitalna da obavlja operacije sa brojevima (digit - broj)
- reprogramibilna znači da se redoslijed operacija može programirati i mijenjati.

Najrašireniji tip računara poznat kao PC (*Personal Computer* - lični računar), je univerzalan računar namijenjen za jednog korisnika koji na njemu obavlja različite, ali relativno proste poslove unosa obrade i pamćenja (čuvanja) podataka.
Računar čini jedinstvo fizičkih i programskih resursa.

Teoretski, svaki korisnik, može sam da bira i odlučuje o načinu i konceptu rada računara. No, to bi od korisnika zahtijevalo multidisciplinarno znanje: od matematike, elektronike, komunikacija, mehanike do ergonomije i ko zna čega još. Praktično od takvog računara bi rijetko ko imao stvarne koristi.

Hardverom treba upravljati, tj. na neki način treba „natjerati“ procesor da sabere dva broja, disk da zapamti određenu sliku, monitor da prikaže podatke kako treba, pointer da prati kretanje miša, štampač da odštampa željene podatek itd. To je posao softvera ili programa. Dakle, ispravan hardver ne radi ako nema softvera.

Tip softvera koji upravlja hardverom zove se sistemski softver (jer upravlja sistemom).



Softver ne služi samo da upravlja hardverom nego i za pružanje usluga korisniku (čovjeku). Tako su za razne poslove kreirani različiti programi. Ako želite da otkucate neki tekst određeni program vam nudi tu mogućnost. Ako želite da kreirate neku sliku koristićete neki drugi program. Čak i korisnik može napraviti neki svoj program za neku posebnu svrhu. Ova vrsta softvera koja pruža razne usluge korisnicima tj. ima konkretnu primjenu (*application*) u svakodnevnom poslu se zove aplikativni softver (primijenjeni ili upotrebni program).

Softver dijelimo na:

- sistemski (operativni sistem za upravljanje hardverom) i
- aplikativni (primjenjivi).

Softver se kreira pomoću nekog programskog jezika.
Najpoznatiji programski jezici su: Basic, Pascal, Fortran, C, C++, Java, Delphi, VisualBasic itd.
Većina današnjih operativnih sistema je kreirana („napisana“) u C-u.

Generalno, ne postoji kompletna i adekvatna definicija operativnog sistema.
Operativni sistem se može uporediti sa vladom. Komponente računarskog sistema su hardver, softver, i podaci, a operativni sistem osigurava sredstva za pravilno korištenje navedenih komponenti. *Poput vlade operativni sistemi ne izvršavaju operacije radi sebe samih, nego jednostavno osiguravaju okruženje u kojem drugi programi mogu obavljati koristan posao.*

Operativni sistem se može vidjeti i kao raspoređivač resursa (**resource allocator**), koji se u računarskom sistemu ponaša kao upravitelj (**manager**) kompjuterskih resursa kao što su CPU time ili ciklusi na procesoru, memorija, ulazno-izlazni uređaji itd., koje operativni sistem dodjeljuje specifičnim programima i korisnicima kako bi oni obavili "koristan" posao. Pod pojmom operativnog sistema u klasičnom smislu podrazumjeva se "softver potreban za izvršavanje (aplikativnih) programa i za koordinaciju aktivnosti računarskog sistema".

Najčešća definicija operativnih sistema kaže da je operativni sistem jedan program koji se cijelo vrijeme izvršava na kompjuteru, poznatiji pod nazivom kernel, s tim da se svi ostali dijelovi softvera posmatraju kao aplikativni programi.

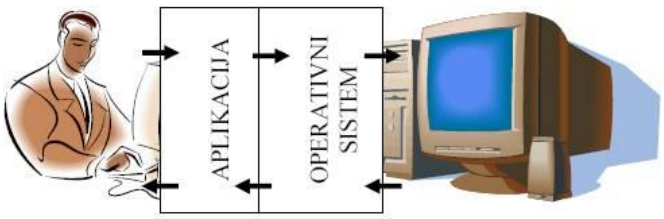
Prema međunarodnom standardu ISO/IEC*, **Operativni sistem je softver koji kontroliše izvršavanje programa i koji može pružati servise (usluge), kao što su dodjeljivanje resursa, raspoređivanje, U/I kontrola i menadžment podacima.**
** ISO/IEC predstavlja zajednički tehnički komitet koji čine Internacionalna organizacija za standardizaciju (ISO) i Internacionalna komisija za elektrotehniku (IEC).*

Međutim po modernom shvatanju operativni sistem je "Sve što proizvođač isporuči pod tim nazivom"što najbolje ilustruje primjer Microsoft Windows-a, gdje pod operativnim sistemom dobijamo: od "stvarnog" OS-a do standarnih korisničkih alata, kalkulatora, programa za crtanje i sviranje, Internet mejlera i čitača...

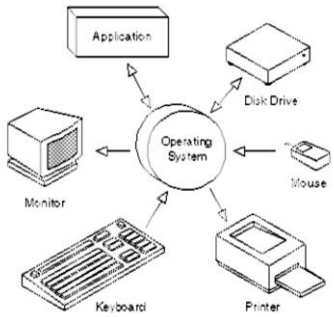
Funkcije i osobine OS

Pod pojmom **računarski resursi** podrazumjevamo sve što je programu potrebno za rad.
Hardverski resursi su procesori, operativna memorija i ulazno/izlazni uređaji, a **softverski resursi** su programi i podaci, to jest fajlovi.

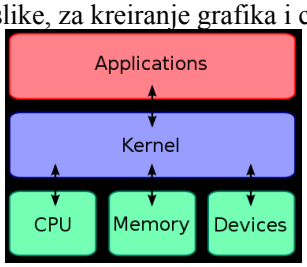
OS kontroliše i upravlja računarom uz pomoć instrukcija korisnika. Ako bi grafički prikazali odnos korisnika i računara onda bi to izgledalo kao na slici 1. Naime, korisnik radi na nekoj aplikaciji (unosi tekst, sluša muziku, projektuje, računa, ...). Aplikacija koristi operativni sistem da bi izvršila obradu podataka na hardveru.



Slika 1. Odnos korisnika i djelova računara



Aplikativni softver ili prosto aplikacije se koriste za izvršavanje određenih poslova na računaru: obrada teksta, tabelarni proračuni, crtanje, obrada slike, obrada zvuka, komponovanje, kreiranje animacija, učenje, zabava itd. Aplikacije su onaj dio računara koji krajnji korisnik najviše koristi.
Aplikacije koriste operativni sistem da bi pristupile hardveru. Zbog toga se aplikacije kreiraju i prilagođavaju za razne operativne sisteme. Drugm riječima, aplikacija izrađena za Windows obično neće raditi na PC koji koristi Linux OS.
Poznate aplikacije koje se svakodnevno koriste su: za unos i obradu teksta, za tabelarnu obradu podataka, za kreiranje prezentacija, za kreiuranje baza podataka, za kreiranje WEB stranica za čitanje WEB stranica, za kreiranje i obradu slike, za kreiranje grafika i crteža, za obradu zvuka, itd.



Slične aplikacije postoje za sve OS. Ovo su aplikacije opšte namjene.
Pored njih, postoje aplikacije za posebne namjene (bankarsko poslovanje, evidencija građana, statistika, finansijko poslovanje, itd).
Korisnici kroz aplikacije koriste funkcije računara: obrada, pamćenje i prenos informacija.
Da bi aplikacije pravilno funkcionisale treba da su prilagođene operativnom sistemu i da koriste njegove prednosti.

Operativni sistem ima dvostruku ulogu.
S jedne strane, on upravlja dijelovima od kojih se sastoji računar (procesor, I/O kontroleri, radna memorija), sa ciljem da oni budu što cjelishodnije upotrebljeni.
S druge strane, operativni sistem stvara za krajnjeg korisnika računara pristupačno radno okruženje, tako što pretvara računar od mašine koja rukuje bitima, bajtima i blokovima u mašinu koja rukuje datotekama i procesima.

Kako je arhitektura računara i hardvera krajnje komplikovana za programiranje za većinu korsinika, **operativni sistem takođe obezbeđuje takozvanu virtuelnu mašinu**, odnosno jedinstven pogled na računarski sistem sa tačke gledišta korisnika, nezavistan o konfiguraciji računara, konkretnom hardveru i samoj arhitekturi.

Osobine OS

Iz niza osobina koje mogu posjedovati OS naglašavamo:

- Istovremenost - paralelizam (**Concurrency**)
- Zajedničko korišćenje, odnosno dijeljenje resursa (**Sharing**)
- Pouzdanost (**Reliability**)
- Sigurnost (**Security**)
- Upotrebljivost (**Usability**) i
- Djeljivost - modularnost (**Modularity**)

Funkcije OS

Za razliku od karakteristika, koje **su poželjna** svojstva, **funkcije** sistema su zadaci koje sistem **mora** da realizuje. Navešćemo samo globalne funkcije koje moraju biti rješene u svakom OS:

- upravljanje računarskim resursima:
  1. upravljanje procesorom (CPU),
  2. upravljanje memorijom (RAM),
  3. upravljanje I/O uređajima,
  4. upravljanje podacima i
  5. upravljanje aplikacijama
- interpretiranje i izvođenje kontrolno upravljačkih komandi i programa,
- upravljanje poslovima i zaštitu,
- podršku daljinske obrade i rada u mreži.

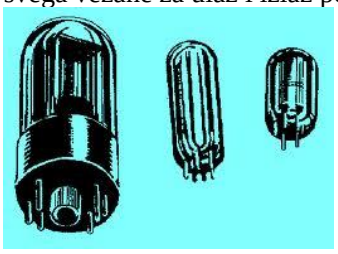
Kako bi se omogućila efikasnost svih računarskih resursa, u svakom operativnom sistemu moraju biti na određen način riješene **opšte funkcije**, a to su:
• upravljanje zadacima obrade (Job Managment)
• upravljanje podacima (Data Managment)
• upravljanje ulazom i izlazom (Device Menagment)
• upravljanje memorijom (Memory Menagment)
• obrada prekida (Interrupt Handling)
• dodjeljivanje procesora (Processor Scheduling)
• zaštita (Protection)
• podržavanje daljinske obrade (TP Monitoring)
• interpretiranje i izvođenje kontrolno-upravljačkih naredbi (J/CL)...

Moduli koji upravljaju svakim od navedenih resursa računarskog sistema je da treba da obezbjede:

- vođenje evidencije o resursu
- donošenje odluke o dodjeli resursa
- alokaciju resursa i
- dealokaciju resursa.

Razvoj OS

U prvoj generaciji računarskih sistema operativni sistemi nisu postojali, tako da je operater koji je opsluživao računarski sistem, bio u obavezi da sve potrebne radnje, prije svega vezane za ulaz i izlaz podataka nekog programa, obavi sam.



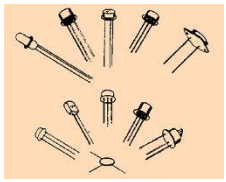
Kompjuteri prve generacije (1945–1955), čiju su osnovu činile vakuumske cevi (do 20.000 cevi po računaru), bili su ogromnih dimenzija i veoma skupi. Koristila ih je uglavnom vojska.

Ovi računari su bili jako spori, programiralo se na mašinskom jeziku, dok su simbolički jezici (uključujući i asembler), kao i operativni sistemi, u to vrijeme bili nepoznati. Ljudi koji su radili na tim računarima obavljali su sve poslove – od programiranja do održavanja računara.

U prvoj generaciji računara, opsluživanje računarskog sistema bilo je potpuno prepušteno operateru, koji je morao da pripremi sve što je potrebno da se zadatak obrade može obaviti. Čovjek je, dakle, imao punu kontrolu nad računarskim sistemom. Operater je bio u mogućnosti da sve potrebne radnje obavi na vrijeme, jer je sistem bio spor i izvršavao se samo jedan program, tj. obavljao se samo jedan zadatak.
Može se reći da je iskorišćenjee računarskog sistema, tj. njegovih najvažnijih resursa – centralnog procesora i centralne memorije – bilo slabo. Najveći dio vrijemena trošio se na poslove operatera i ulazno-izlazne operacije, a mnogo manji dio na rad centralnog procesora. Iako je ovakav sistem bio krajnje neefikasan, odnos tih vrijemenskih perioda bio je u prihvatljivim granicama zbog relativno male brzine samog centralnog procesora.

U drugoj generaciji računarskih sistema dolazi do nastanka prvih kontrolnih programa koji su pomagali operateru da opslužuje računarski sistem. Tada su, dakle, prvi put funkcije opsluživanja i upravljanja sistemom podijeljene između operatera i kontrolnih programa. Ali ti kontrolni programi nisu bili toliko sofisticirani da bi ih mogli nazvati operativni sistem.

Osnovu računara druge generacije (1955–1965) činili su tranzistori, pa su računari postali manji, pouzdaniji i jeftiniji. Računare druge generacije su, osim vojske, kupovale i velike korporacije i univerziteti. Računari su bili smješteni odvojeno, u posebnim sobama, koje su se dijelile u tri funkcionalne celine: ulazna soba, centralni računar i izlazna soba.

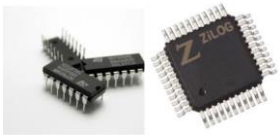


Programeri su pisali programe na papiru, većina na programskom jeziku FORTRAN, zatim su se ti programi prenosili na **bušene kartice**, koje su se ostavljale u sobi sa ulaznim računarom (**input room**).
Operator sistema je, zatim, uzimao bušene kartice i ubacivao ih u računar, i to prvo kartice sa prevodiocem FORTRAN-a, a potom bušene kartice s programom koji treba izvršiti. Glavni računar je obavljao posao, a rezultat se dobijao takodje na bušenim karticama, koje su se prenosile u prostoriju sa rezultatima (**output room**). Ovde se mnogo vrijemena trošilo na šetače izmedju raznih prostorija sa bušenim karticama. Operativni sistem kao zaseban pojam još uvek nije postojao.
Dalje se kao poboljšanje uvodi paketna, tj. grupna obrada (**batch processing**), zasnovana na upotrebi **magnetne trake** – uređaja mnogo bržeg od bušenih kartica. Pri paketnoj obradi, u ulaznoj sobi sa poslovima sakuplja se jedna količina sličnih programa (na Primjer, svi programi koji zahtevaju prevodilac FORTRAN-a), koji se pomoću jeftinijeg računara (npr. IBM 1401) s bušenih kartica prenose na magnetnu traku. Posle toga se magnetna traka prenosi u sobu s glavnim računarom, tj. sa moćnijim i skupljim računarom, predvidjenim za izvršavanje samog programa (npr. IBM 7094). U glavni računar se učitava poseban program koji je zadužen da sa trake sa poslovima programe redom učitava i izvršava. Taj program se može **smatrati pretkom operativnih sistema.**

Nakon izvršavanja programa, rezultati se snimaju na drugu magnetnu traku koju operater prenosi do trećeg računara, zaduženog za prebacivanje rezultata sa magnetne trake na bušene kartice. Manji računari (ulazno-izlazni) nisu direktno vezani za glavni računar, što znači da rade u off-line režimu (u režimu gdje nisu bili u medjusobnoj komunikaciji).

U drugoj generaciji računarskih sistema povećava se brzina rada centralnog procesora, kapaciteti centralne memorije i eksternih (masovnih) memorija, pojavljuju se nove i brže ulazno-izlazne jedinice. Programi se pišu na simboličkom mašinskom jeziku, asembleru ili na višem programskom jeziku (FORTRAN). Operater više nije u stanju da efikasno opslužuje računarski sistem, jer su njegove reakcije suviše spore. Jedino rješenje se moglo naći u prebacivanju niza kontrolnih funkcija sa operatera na sam računarski sistem, to jest na posebne kontrolne programe. Otuda su funkcije opsluživača i upravljača sistemom bile podjelljene izmedju operatera i kontrolnih programa. Ti programi se uključuju u odredjenim situacijama, kao što su, na Primjer, priprema programa za izvodjenje, kontrola ulaznih i izlaznih uredjaja i razni poslovi oko učitavanja programa i pripreme za njegovo izvodjenje. **Dakle, u računarima druge generacije razlikuju se dvije osnovne vrste programa: kontrolni i korisnički.**

U **trećoj generaciji računarskih sistema**, kontrolno upravljačke funkcije se još više prebacuju na sam računar i zbog toga dolazi do nastanka većeg broja kontolnih i upravljačkih programa velike kompleksnosti. Skup svih tih programa nazvan je operativni sistem.



Računari treće generacije (1965–1980) prave se od integrisanih kola (IC). Početkom šezdesetih godina većina proizvođačaa pravi dvije vrste računara:

- jednu bržu verziju (kao IBM 7094) i
- jednu slabiju (kao IBM 1401),

što je skup poduhvat.

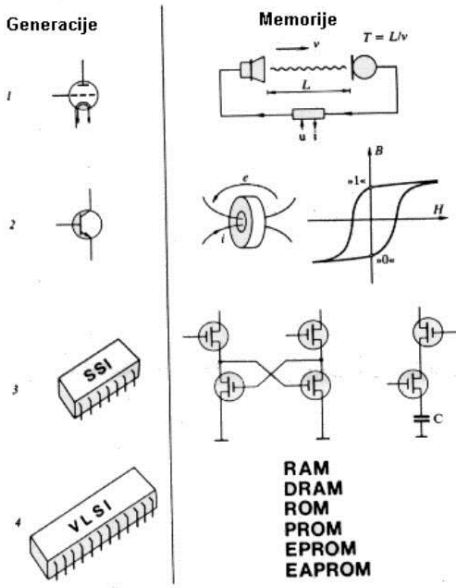
Novi korisnici računara najpre žele slabije modele koji su jeftiniji, dok će im jači, brži i skuplji modeli biti potrebni tek nakon izvesnog vrijemena. IBM taj problem pokušava da razreši uvodjenjem klase računara IBM System/360. To je serija kompatibilnih računara različitih snaga. Svaki od ovih računara je pogodan i za naučnu i za poslovnu primenu, pa je time podjela računara na ove dvije vrste nestala. Ovaj koncept su preuzeli i ostali proizvoođači računara.
Računari iz serije System/360 radili su pod operativnim sistemom OS/360, koji je bio veoma glomazan i prepun grešaka. S razvojem discipline poznate pod imenom softversko inženjerstvo (software engineering), uvode se nove funkcije:

- multiprogramiranje, (multiprogramming)
- višestruke ulazno-izlazne (U/I) operacije (spool)
- podjela računarskog vrijemena (time-sharing).

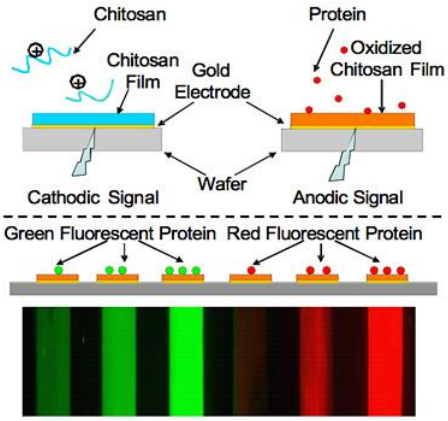
Kod treće generacije računara posebno treba istaći pojavu dva operativna sistema : MULTICS i UNIX.
Projekat **MULTICS** (MULTIplexed Information and Computing Service) neuspela je ideja kompanija MIT, Bell Labs i General Electric da se napravi moćan računar i operativni sistem koji će biti u stanju da radi sa velikim brojem terminala. Osnovna ideja je preuzeta iz modela distribucije električne energije – u tom modelu, dovoljno je da svaki korisnik koji poželi da upotrebi neki električni aparat, taj isti aparat samo priljući na električnu mrežu. Sličan model su pokušali da naprave i sa računarima: ideja je da u jednom radu postoji moćan centralni računar, a da gradjani kod kuće imaju terminale kojima preko modema pristupaju glavnom računaru. Ovaj model se može smatrati pretečom računarskih mreža i Interneta.
Drugi operativni sistem, **UNIX**, uprošćena je varijanta MULTICS sistema, koja je dživela praktičnu realizaciju i ekspanziju do današnjih dana. **Ken Thompson**, jedan od naučnika i programera kompanije Bell Labs, koji je radio na razvoju projekta MULTICS, napisao je za računar PDP-7 mini verziju MULTICS sistema. Posle toga je nastao UNIX (UNI= jedan, X = CS = Computing Service).
Računari treće generacije zvali su se **mini računari**: prvi računar je DEC-ov (Digital Equipment Corporation) PDP-1, do tada najmanji i najjeftiniji računar. Koštao je tada "samo" 120.000 dolara. U trećoj generaciji računarskih sistema, zbog pojave multiprogramiranja i porasta brzina, veličine memorije i broja ulaznih-izlaznih jedinica, još više kontrolno-upravljačkih funkcija prebacuje se sa čovjeka na računar. Dakle, čovjek definitivno gubi mogućnost kontrole interne situacije u računarskom sistemu i upravljanja njome i sve što je moguće prebacuje se na računarski sistem, tj. na pojedine sistemske programe. Skup svih tih programa naziva se jednim imenom: operativni sistem. Programer se oslobadja niza složenih rutinskih poslova i pruža mu se mogućnost većeg angažovanja na kreativnom dijelu posla. No, pored kontrolno-upravljačkih programa u, računarima treće generacije razvijen je i čitav niz uslužnih programa, čiji je zadatak da dalje

olakšaju i pojednostave upotrebu računarskih sistema. Zbog toga, prema nameni, softver možemo podijeliti na sistemski i korisnički (aplikativni).

U četvrtoj generaciji računara (1980–pa nadalje), prvi put se pojavljuju personalni računari.
Razvoj personalnih računara započinje pojavom LSI čipova (large scale integration), to jest čipova visokog stepena integracije. Računari su bili dovoljno jeftini, tako da su ih mogli priuštiti i više odseka iste firme ili univerziteta, dok su personalni računari postali dovoljno jeftini da ih mogu imati i pojedinci. Poznatiji personalni računari su Spectrum, Commodore, Atari, zatim IBM PC, Apple Macintosh itd. **U prve operativne sisteme za personalne računare spadaju MS-DOS i UNIX.**
Paralelno s razvijanjem korisničkog softvera, razvija se i korisnički interfejs programa, to jest korisnčko okruženje. Na taj način se osobama koje računarski sistem i same programe ne poznaju detaljno, omogućava da te programe uspešno koriste.

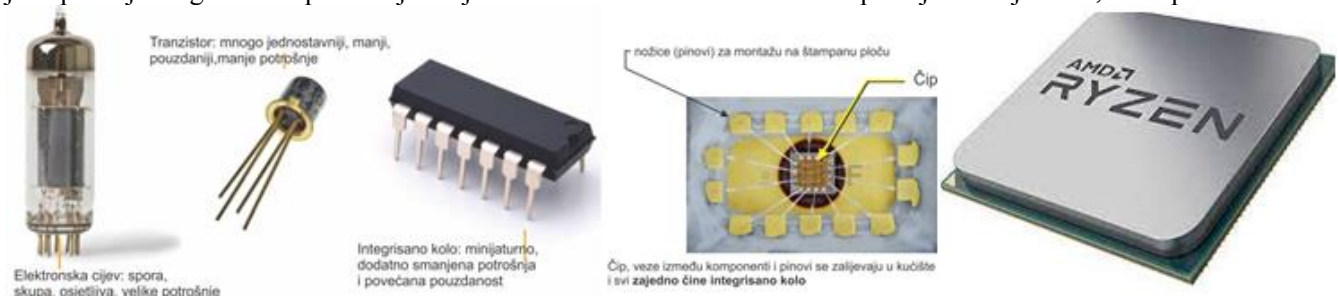


Smatra se da će računari u petoj generaciji biti izgrađeni pomoću organskih sastavnih dijelova.



Karakteristike savremenih OS

Raniji OS (MS DOS, UNIX, VMS, ...) su radili u tzv. *tekstualnom modu*. To znači da je korisnik kucao tekstualne naredbe koje su uglavnom bile komplikovane. Korišćenje računara je bilo dosta nepraktično i teško za učenje. To je bio jedan od razloga zašto je malo ljudi tada koristilo računare.
Poznato je da obrada jedne slike traži mnogostruko više računarskih resursa od obrade bilo kog teksta. Razvojem hardvera računari su imali sve bolje performanse i rad sa grafikom je postajao sve lakši. To je bio preduslov za pojavu tzv. *grafički orjentisanih* operativnih sistema (Windows, Linux, Mac OS).
Prije dvadesetak godina (krajem osamdesetih) se isprofilisalo, a i do danas se zadržalo da rad u tekstualnom modu označava *računare stare generacije.* Novi OS su uveli pojam pokazivača (*pointer*) koji se koristi kao kažiprst.
Pomjeranje ovog pointera se obavlja pomoću miša. Ovim pointerom se biraju komande ili sličice koje se zatim startuju nekim od tastera miša. Ova grafička veza između korisnika i računara se naziva GUI (*Graphical User Interface*).
Prva (ali svakako ne najvažnija) osobina modernih računara je **korisnički grafički interfejs.**
Prednosti pri korišćenju GUI-a su jednostavnost pri radu, intuitivno i brzo učenje. Korišćenje GUI-a liči na sporazumijevanje „rukama“. Dakle, umjesto da se kucaju komplikovane naredbe prosto se prstom pokazuje na njih. To je doprinijelo ogromnom povećanju broja korisnika PC računara. Danas ne postoji OS koji nema, ili ne podržava GUI.



Poređenje cijevi, tranzistora, integrisanih kola i mikroprocesora, kao osnov za hardversku podjelu računara i OS

Popularnost PC računara dovela je do masovnosti što opet omogućilo nova tehnioloških poboljšanja. Dizajn operativnih sistema zasnovan je na konstrukciji paralelne arhitekture koji omogućavaju istovremeni rad više kompjutera (procesora) na rešavanju određenog zadatka. Poseban impuls bila je masovna upotreba Interneta.
Savremeni računari treba da imaju mogućnost višekorisničkog i višeprogramskog rada u mrežnom okruženju, da su pouzdani, zaštićeni, podržavaju GUI, jednostavni za rad (user frendly).

Peta generacija: nevidljivi ili ugrađeni računari

XXI vijek

Mada se računari današnjice ubrajaju u računare četvrte generacije ipak je početkom XXI vijeka došlo do kvalitativnih promjena, prije svega umrežavanjem personalnih računara (pojavom mrežnih operativnih sistema i dijeljenjem resursa) a pogotovo **masovnom upotrebom interneta** (globalnom svjetskom mrežom), kao i pojavom novih tipova računara: **tableta i pametnih telefona**. **Multimedija**, mobilna telefonija i internet su postali dijelom računarskih sistema.

Generacija	Hardver	Softver	Primjeri računara
I (1939.-1958.)	Elektronske cijevi, bušene kartice, feritne memorija	Mašinski kod, memorisani programi	ENIAC, UNIVAC I, IBM 700
II (1959.-1964.)	Tranzistori, memorija	Programski jezici visokog nivoa, paketna obrada	IBM 7094
III (1965.-1971.)	integrisana kola LSI IC, poluprovodničke memorije, mikroprocesori	time-sharing, grafika, GUI, multiprogramiranje, strukturno programiranje	IBM 360 370 PDP 11
IV (1972.-1992.)	VLSI, mreže, optički diskovi, personalni računari	Objektno orijentisani jezici, ekspertni sistemi, mrežni i distribuirani OS	IBM 3090, Cray XMP, Altair 8800 IBM PC, Mac Apple
V (1993.-?)	ULSI, GaAs, paraleleni sistemi	veštačka inteligencija paralelno procesiranje	smartPhone, Sun Sparc

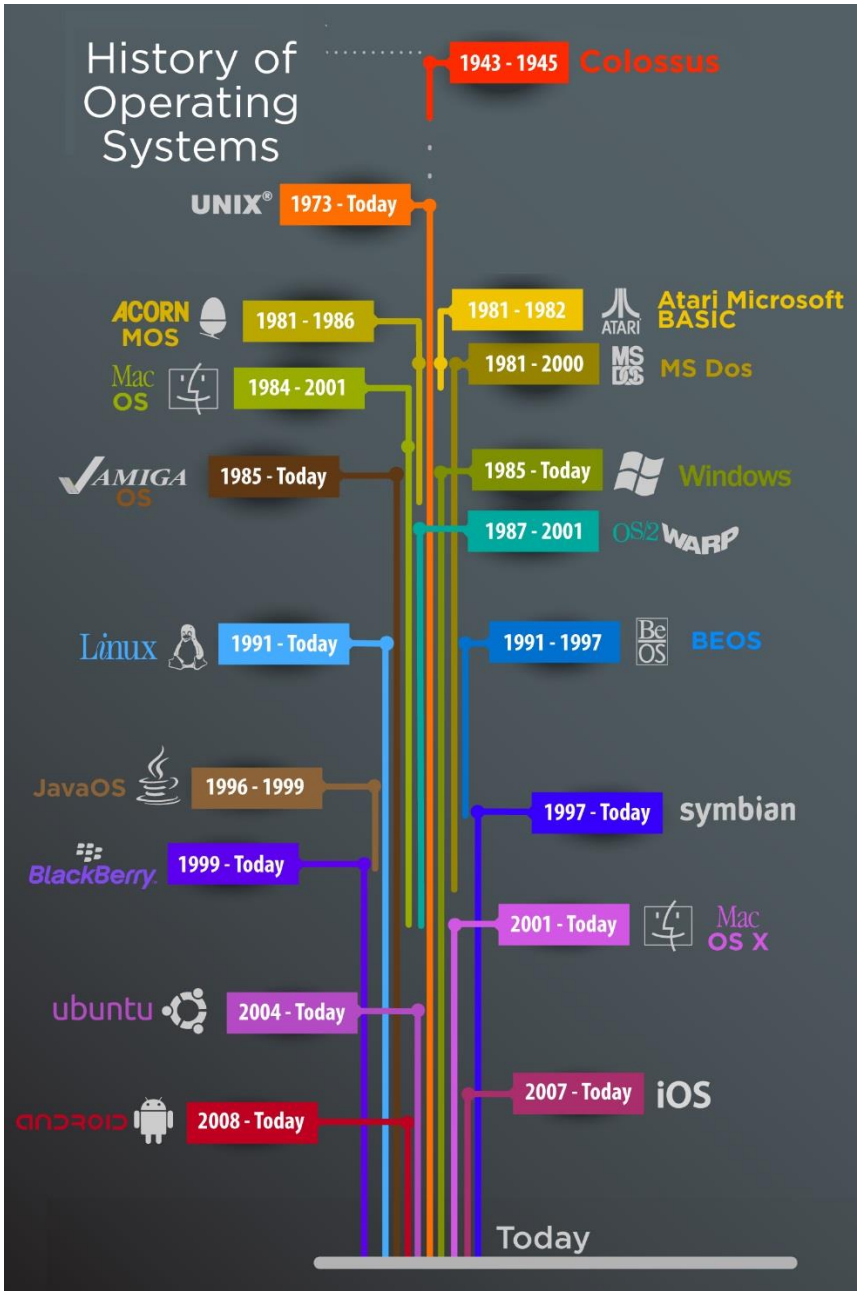
Pregled razvoja računara

Peta generacija računara se i dalje razvija i **nije precizno definisana**.
Napredak i izmjene u petoj generaciji nisu prevashodno tehnološke, već je to prije svega **algoritamska i softverska evolucija programiranja i sistemi bazirani masivnom paralelnom procesiranju i vještačkoj inteligenciji**. Računari bi trebali biti sposobni odlučiti šta je najbolje u tom momentu, sam raditi neke stvari.
Računari ove generacije poznati i pod nazivom **nevidljivi računari** (Invisible Computers. Računari se ugrađuju u razne aplikacije kao što su digitalni satovi, bankarske karte i u razne druge proizvode, tako da se koriste, a da se ne primjećuje postojanje računara koji obavlja poslove nevidljiv, u pozadini.

U praksi su realizovane različite verzije **ugrađenih računarskih sistema** (engl. embeded), koji su integrisani u uređaje (automobile, kućne aparate, mobilne telefone, konzole za video igre...), što je prikazano na Slici 1.14.

Embedded sistemi predstavljaju računarske sisteme kod kojih je izvršena **integracija hardvera i softvera koja omogućava izvršavanje specifičnih funkcija.** Računar je ugrađen (enkapsuliran) u uređaj koga kontroliše.
Termini *nevidljiv* i *ugrađen* pokazuju da su ovakvi sistemi dio nekog većeg sistema. Ugradbeni sistemi se razlikuju od računara opšte namjene kakvi su u osnovi računari predhodnih generacija. Namjenjeni su za manji broj unaprijed definisanih zadataka (ponekad i samo jedan), što im daje mogućnost bolje optimizacije.

Zadatak programera i naučnika je da konstruišu računar koji "razmišlja". U ovakve računare možemo ubrojati robote. Već sad postoje roboti koji imaju mogućnost ograničenog samostalnog odlučivanja.
Cilj razvoja pete generacije računara je da računare osposobi za razumiju prirodni govor i da budu sposobni za samoorganizaciju.
Smatra se da će računari u budućnosti umjesto na silikonskoj (poluprovodničkoj) osnovi biti izgrađeni pomoću organskih sastavnih dijelova, ili korištenjem novih nanotehnologija, ali njihov razvoj je još uvijek u eksperimentalnoj fazi.



Vremenska osa koja pokazuje istoriju razvoja najpopularnijih operativnih sistema današnjice
izvor: https://in.pinterest.com/pin/396809417140897454/

Naglasimo još jednom **šta OS treba da radi - omogući**:
Izvršavanje programa
OS puni glavnu memoriju programom i počnje njegovo izvršavanje. Korisnički programi ne mogu da sami sebi dodijele procesor.
U\I operacije
Sistem mora da komunicira sa diskovima, USB uredjajima, magnetnim trakama i drugim uredjajima niskog nivoa. Korisnik zadaje uredjaj i operaciju koju treba izvrsiti a sistem konvertuje taj zahtjev u specificne komande uredjaja i ili kontrolera. Korisnicki programi ne mogu da pristupe nedjeljivim uredjajima svaki put kada im je to potrebno vec samo kada se ti uredjaji ne koriste od strane nekog drugog uredjaja.
Komunikacija
Slanje poruka izmedju sistema zahtijeva da se poruke dijele u pakete podataka, salju do mreznog kontrolera, prenose preko komunikacionih medijuma i na mjestu odredista isporuce kao cjelina.Korisnicki programi nisu u mogucnosti da koordiniraju pristupom mrezi vec je to zadatak OS.
Upravljanje sistemom datoteka
Postoji puno detalja o kojima korisnici ne moraju da vode računa pri kreiranju datoteke, brisanju, dodjeli memorije... Npr. potrebno je voditi evidenciju o blokovima na disku koji se koriste za datoteke. Brisanje datoteke zahtijeva brisanje informacija o imenu datoteke i oslobadjanje dodijeljenih blokova. Da bi se obezbijedio autorizovan pristup datoteci potrebno je provjeriti zastitu. Korisnicki programi ne mogu da obezbijede zastitu niti dodijele ili oslobode memoriju.

Klasifikacija operativnih sistema

Postoje brojne podjele operativnih sistema na osnovu različitih kriterijuma: prema broju korisnika i/ili procesa, prema načinu obrade poslova, prema distribuciji procesorske snage i ostalih resursa, prema nameni i funkcionalnim osobinama. Za početak daćemo **tri** osnovne klasifikacije, a proširenu klasifikaciju možete pogledati u narednom.

1. **Klasifikacija prema broju korisnika i procesa**

Prema broju korisnika, operativni sistemi se dijele na:
- jednokorisničke (single user) i
- višekorisničke (multiuser).

2. **Klasifikacija prema broju simultanih aktivnosti**

Prema broju simultanih aktivnosti, tj. prema broju procesa koji se mogu izvršavati paralelno ili kvaziparalelno, operativni sistemi se dijele na:
- jednoprocesne (single tasking, singleprocess) i
- višeprocesne (multitasking, multiprocess).

3. **Klasifikacija po strukturi**

Podjela OS prema strukturi:
- Slojeviti OS
- OS sa monolitnim jezgrom (Kernel OS)
- OS sa mikro jezgrom (Mikrokernel OS)

Ovdje date klasifikacije predstavljaju uobičajeni presjek koji je nemoguće formalizovati zbog nepostojanja opšteprihvaćenih standarda. ali i isprepletenosti pojedinih tehniologija, koje nisu čiste, već najčešće plod kompromisa I kao takve svojevrsni **hibridi,** pa svi moderni operativni sistemi imaju ponešto od svih klasa koje su ovdje predstavljene.

Pregled dodatnih klasifikacija operativnih sistema

Predlažem da se sa narednim klasifikacija (*datim na sivoj podlozi*), za početak, upoznate tek informativno, jer ono sadrži mnoge pojmove koji će naknadno biti objašnjeni. Uostalom, nema univerzalno prihvaćene čak ni definicije operativnog sistema, a kamoli klasifikacije.

Klasifikacija prema načinu obrade poslova

Prema načinu obrade poslova, operativni sistemi se klasifikuju kao:

1] **Sistemi sa grupnom obradom** (batch)
Grupna (serijska, paketna) obrada je takav način rada računara u kome korisnici predaju svoje poslove na izvršenje posredstvom ulaznih jedinica, i koji se zatim odvijaju jedan za drugim u nizu, pri čemu korisnik nema mogućnost komuniciranja sa svojim poslom

2] **Interaktivni sistemi** (interactive systems)
Interaktivne sisteme (nazivaju se još i time-sharing sistemi) karakteriše postojanje terminala za svakog korisnika, preko kojih korisnici zadaju poslove i komuniciraju sa svojim poslovima. Paralelnost u radu se postiže tako što se svakom korisničkom programu dodeljuje jedan kvantum vrijemena centralnog procesora, pa se na svaku poruku korisnika odaziva u roku od nekoliko sekundi. Po isteku vrijemenskog
kvantuma dodeljenog jednom procesu, on se prekida, bilo da je završio s radom ili nije, a procesor se dodeljuje sljedećem procesu u redu čekanja.

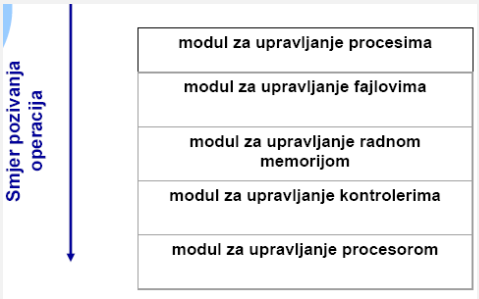
3] **Kombinovani sistemi**
Kombinovane sisteme karakteriše mogućnost istovrijemenog obavljanja interaktivnih poslova i paketne obrade. Na Primjer, korisnik može u pozadini pokrenuti nekoliko vrijemenski zahtjevnih poslova koji ne zahtjevaju interakciju sa njim; dok čeka na njihovo izvršenje, može čitati elektronsku poštu ili Web stranice.

Klasifikacija po strukturi

- Podjela OS prema strukturi:
- Slojeviti
- Monolitni
- Virtualne mašine
- Exokernel-i
- Klijent – server

Slojeviti

Kod slojevite realizacije operativni sistem se dijeli na različite slojeve na hijerarhijski način: svaki sloj se gradi na slojeve ispod njega.



Monolitni

Monolitni operativni sistemi nemaju pravilnu strukturu kao slojeviti operativni sistemi, jer se sastoje od modula čija saradnja nije ograničena pravilima kao kod slojevitih operativnih sistema.
To znači da se iz svakog od modula monolitnih operativnih sistema mogu slobodno pozivati operacije svih ostalih modula.

Virtuelne mašine

Srce sistema, poznato kao monitor virtuelne mašine, radi na čistom hardveru i obavlja multipleksiranje, obezbijeđujući ne jednu, već nekoliko virtuelnih mašina na sljedećem, višem nivou.
Stvarni operativni sistem izvodi prevođenje naredbi korisnika iz njemu predstavljene virtuelne slike računara na konkretni hardver računara. Struktura virtuelne mašine korištene su za mainframe računre sa velikim brojem korisnika kako bi im se omogućila autonomnost u radu sa operativnim sistemom.
Danas se ideja virtuelne mašine puno koristi i to u različitim kontekstima. Možda najčešća upotreba je izvršavanje MS-DOS programa na Pentium-ima. Pošto su uvidjeli potrebu korisnika za upotrebom starog sofvera na novom hardveru, dizajneri softvera za Pentium su obezbijedili virtuelni 8086 mod na Pentium-u. U tom modu se mašina ponaša kao 8086, uključujući 16-bitno adresiranje i ograničenje na 1 MB.
Od drugih upotreba naglasićemo JAVA programe koji se izvršavaju pomoću virtuelnih mašina.

Exokernel-i

Idući korak dalje, napravljen je sistem koji svakom korisniku daje kopiju kompjutera, ali ne sa svim resursima, već samo sa jednim dijelom.
Struktura exokernela korisnicima prezentira virtuelnu sliku mašine (računara) koja se razlikuje od hardvera računara koje korisnici koriste. To je omogućeno povezivanjem virtuelnih resursa koje koriste korisnici sa raspoloživim hardverskim resursima računara.
U najnižem sloju softvera je program zvani exokernel, koji se izvršava u kernel modu. Njegov zadatak je da alocira resurse za virtuelne mašine i onda provjerava njihove pokušaje za korišćenje tih resursa, da bi se uvjerio da nijedna mašina ne pokušava da koristi resurs tuđe mašine. Svaka virtuelna mašina može imati svoj operativni sistem, s tim što je ograničena na korišćenje samo onog dijela resursa kojeg zahtjeva i koji je alociran za nju.

Klijent–server model - mikrokernel

Trend u modernim operativnim sistemima je premještanje koda u više nivoe, na taj način ga otklanjajući iz kernel moda, ostavljajući minimalni mikrokernel. To se obično radi implementiranjem
većeg dijela sistema u korisničke procese.
Da bi dobio uslugu, korisnički proces (sad proces klijent) šalje zahtjev procesu serveru, koji odrađuje posao i vraća odgovor.
U ovom modelu sve što kernel radi je komunikacija između klijenta i servera.

Klasifikacija prema vrsti računara kojom OS upravlja

Jedan od mogućih kriterijuma za klasifikaciju operativnih sistema je vrsta računara ili uređaja kojim operativni sistem upravlja. Po tom kriteriju mogu se izdvojiti:

1. operativni sistemi realnog vrijemena RTOS

Real-time sistemi se koriste kada postoje stroga vrijemenska ograničenja za izvršavanje definisanih poslova. Često se koriste i kao "kontrolni uređaji" u namjenski razvijenim sistemima, kao naprimjer u automobilskoj industriji kada kontrolišu robote koji rade na sklapanju automobila, prilikom medicinskih eksperimenata, u vojnoj industriji (kod testiranja projektila) ili u istraživanju svemira. Senzori služe kao primarni ulazni uređaji, pomoću kojih podaci dolaze do računarskog sistema. Ovi sistemi su podržani sa sistemima opšte namjene i naprednim operativnim sistemima kao što su UNIX .
Sistemi za rad u realnom vrijemenu (Real Time System, RTS) predstavljaju mikroračunarski sistem koji upravlja i nadgleda fizičke procese.

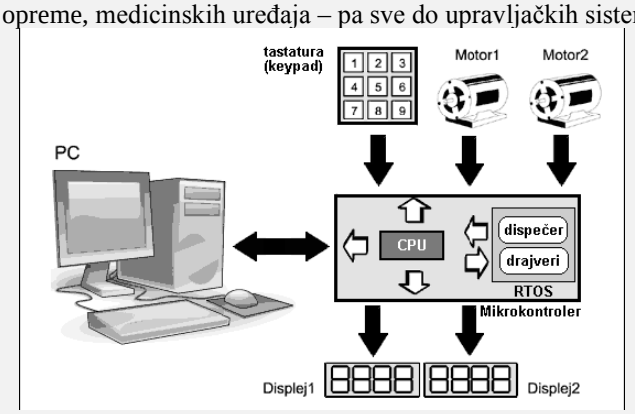
2. operativni sistemi za ugrađene (embedded) sisteme

Embedded sistem je kompjuterski sistem specijalne namjene, koji je potpuno zatvoren od strane kontrola uređaja. Za razliku od ličnih računara opšte namjene, embedded sistem ispunjava specifične zahtjeve i izvršava prethodno definisane zadatke. Embedded sistem je programirani hardverski uređaj. Programibilni hardverski čip je "sirov materijal" i on je programiran određenom aplikacijom.
Neki embedded sistemi nemaju operativni sistem, ili imaju specijalizovani embedded operativni sistem (često real-time operativni sistem), ili je programer dodijelio portu jedan od njih novom sistemu.

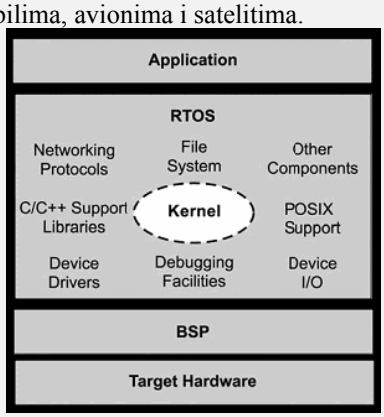
3. operativni sistemi za inteligentne kartice (smart card)

Najmanji računarski sistemi u pogledu gabarita, funkcionalnosti i zahtjeva su veličine kreditne kartice. Obično se cjelokupan sistem (procesor, memorija, I/O uređaj) nalazi u jednom integrisanom kolu i programiran je na izvršavanje malog broja operacija. Najčešće je omogućen rad samo jedne aplikacije sastavljene u jeziku nižeg nivoa. U slučaju mogućnosti obavljanja više poslova, na uređaju mora postojati i najjednostavniji alokator resursa.

Sistemi klasirani kao 2 i 3 mogu se podvesti pod klasu real-time operativne sisteme (RTOS).
(*Svi embedded sistemi nisu ujedno i sistemi za rad u realnom vrijemenu (RTS), niti obrnuto. Međutim, veoma često RTS čine dio nekog većeg sistema ili uređaja i iz tog razloga se nazivaju ugrađeni (embedded) mikroračunarski sistemi za rad u realnom vrijemenu ili, kraće, embedded sistemi za rad u realnom vrijemenu (Real Time Embedded System).)*
Ovi sistemi su u protekle tri decenije, od jednostavnih kontrolera sa specifičnim softverom za određen upravljački zadatak prerasli u složene sisteme na kojima se izvršava mnoštvo aplikacija – od kritičnih upravljačkih aplikacija do grafičkog korisničkog interfejsa.
Ovi sistemi imaju danas široku primenu: od mobilnih telefona, digitalnih fotoaparata, fiskalnih registar-kasa, računarske opreme, medicinskih uređaja – pa sve do upravljačkih sistema u automobilima, avionima i satelitima.



Prikaz jednostavnog sistema na bazi mikrokontrolera sa RTOS

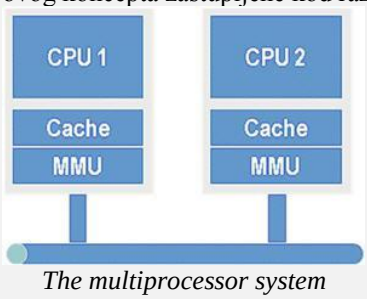


Uopštena arhitektura RTOS za embedded sisteme

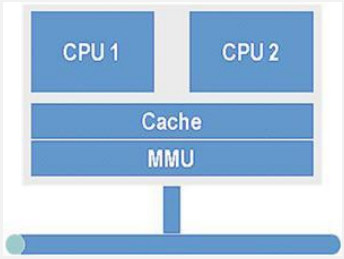
4.multiprocesorski (višeprocesorski) operativni sistemi

Multiprocesorski operativni sistemi predstavljaju koncepciju OS-a sličnu multiprogrammingu. Krajnji korisnik ovakve sisteme vidi kao dase radi o jednom vrlo moćnom procesoru.
Ovi sistemi imaju više procesora koji tijesno komuniciraju dijeleći sabirnicu, sat i ponekad memoriju i periferne uređaje. Oni se Takođe često nazivaju i čvrsto povezani sistemi (tightly coupled sistemi).
Osnovni razlog razvoja multiprocesorskih sistema je povećanje brzine rješavanja korisničkih problema, drugim riječima više posla se može obaviti u kraćem vrijemenskom periodu.
U odnosu na više jednoprocesorskih sistema, jedan višeprocesorski sistem ima i značajnu prednost i u uštedi novca s obzirom da procesori dijele iste resurse. Naprimjer više se isplati da se na jednom disku nalaze određeni podaci sa kojima radi više procesora, nego da postoji više računara svaki sa svojim diskom i sa mnogo kopija podataka. Veća je i sigurnost rada, jer ukoliko bi se desilo da jedan procesor prestane sa radom, to ne bi stopiralo rad sistema, nego bi ga samo usporilo, jer bi se posao nastavio obavljati na preostalim procesorima.

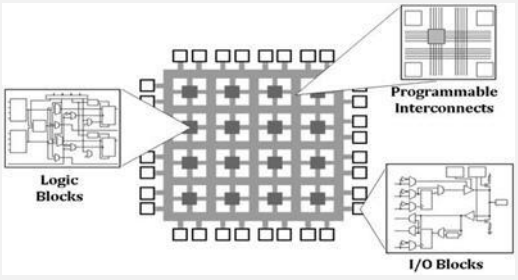
Multiprogramski i multiprocesorski sistemi se javljaju u drugoj generaciji OS, a na slikama su prikazane neke od varijanti ovog koncepta zastupljene kod različitih tehnoloških nivoa.



The multiprocessor system has a divided cache and MMU with long-interconnects



The multicore processors share the cache and MMU with short interconnects



FPGA allows the user to program gates into parallel hardware paths

Danas višeprocesorski sistemi najčešće koriste simetrično multiprocesiranje, u kojem svaki procesor pokreće identičnu kopiju operativnog sistema, koje opet komuniciraju jedna sa drugom kada je to potrebno.

5. mrežni operativni sistemi

Mrežne operativne sisteme *karakterišu računari povezani u mrežu. Ovi računari zadržavaju relativno visok stepen autonomije – svaki računar ima svoj operativni sistem – a u mogućnosti su da međusobno razmenjuju podatke pomoću odgovarajućih protokola.*
Operativni sistemi mogu biti različiti, potreban je samo zajednički protokol*, tj. zajednički jezik za komunikaciju.*
Korisnik jednog računara može se prijaviti na drugi, preuzeti neke datoteke itd. Korisnik zna da nije sam u mreži, tj. svestan je različitih računara s kojima komunicira preko mreže.

Mrežnih operativnih sistema ima dva osnovna tipa:

- Mrežni operativni sistem za mreže ravnopravnih korisnika (Peer-to-Peer (P2P))

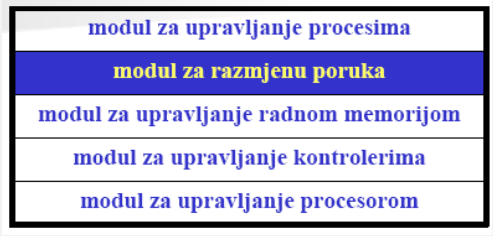
U ravnopravnim mrežama bilo koja stanica može da radi kao server datoteka ili kao klijent (potrošač) mrežnih usluga. Ravnopravni mrežni operativni sistemi obično su jednostavniji od operativnih sistema serverskih mreža. Često se ravnopravni mrežni operativni sistemi izvršavaju kao i svaki drugi.
Mnogi sadašnji operativni sistemi **obezbjeđuju uslove** za mreže ravnopravnih korisnika (Windows 7, Vista, XP, Linux).

- Mrežni operativni sistem za serverske mreže (Klijent-server)

Klijent-server je arhitektura gdje su korisnik (klijent) i server odvojeni ili neravnopravni. Najočitiji je primjer pregledanja Internet stranica. Korisnikov računar i Internet pregledač su klijent – oni zahtijevaju, dok su računar i baza podataka koji čine web stranicu server – on poslužuje. Klijent je obično aktivan korisnik, koji šalje zahtjeve i čeka dok se isti ne ispune, dok je server pasivan, čeka na zahtjeve te ih ispunjava i šalje korisniku.
Na serverima se najviše koriste Linux, Solaris i FreeBSD operativni sistemi koji su razvijeni po uzoru na operativni sistem Unix. U posljednje vrijeme se koristi i server iz Microsoft Windows Server 2008.

6. distribuirani operativni sistemi - mikrokernel

Distribuirani operativni sistemi *su mnogo ozbiljnija varijanta u mrežnom okruženju, zato što osim dijeljnja i migracije datoteka i štampača omogućavaju i dijeljenje procesa, tj. programa. Distribuirani operativni sistemi upravljaju međusobno povezanim računarima, koji su prostorno udaljeni.*
Korisnici ovaj sistem vide kao jednoprocesorski sistem, ali se, u stvari, radi o operativnom sistemu namjenjenom za rad sa više procesora koji su fleksibilno povezani preko mreže. ***To znači da postoji više računara povezanih u mrežu, ali samo jedan operativni sistem, upravlja svim resursima u mreži.***
Ovakav operativni sistem se naziva i mikrokernel (microkernel), jer ima smanjenu funkcionalnost u odnosu na "običan" operativni sistem. Mikrokernel ***ne*** *sadrži sloj za upravljanje fajlovima, jer on nije potreban za svaki od računara iz distribuiranog računarskog sistema i jer se on prebacuje u korisnički sloj (iznad mikrokernela), predviđen za korisničke procese (NFS).*



Hijerarhijska struktura mikrokernela

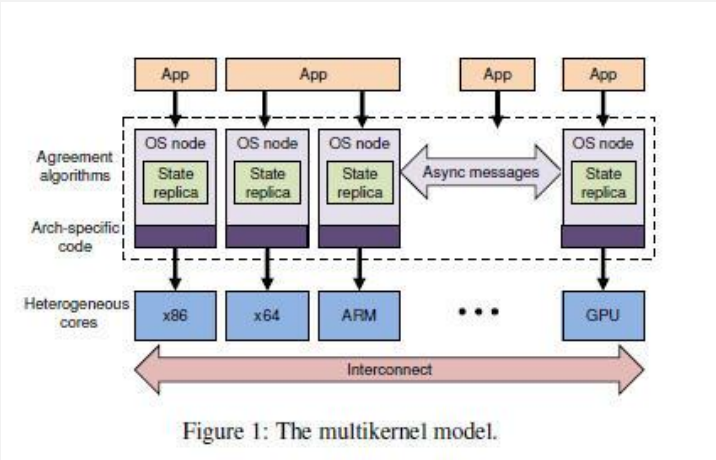


Figure 1: The multikernel model.

U pravom distribuiranom sistemu, korisnik ne treba da vodi računa o tome gdje su smještene njegove datoteke ili gdje se izvršava njegov program – to je posao distribuiranog operativnog sistema. Distribuirani operativni sistem se, dakle, ponaša kao jedinstvena cjelina. Korisnik ne mora znati da je umrežen s drugim računarima – on cio sistem vidi kao jedan računar.
Svaki procesor tj računar ima sopstvenu lokalnu memoriju, a međusobna komunikacija se ostvaruje putem mreže realizovane kao LAN ili WAN. Osim podatka, datoteka i štampača distribuiraju se i procesi. Četiri glavne prednosti distribuiranih sistema su dijeljenje resursa, ubrzavanje izračunavanja, pouzdanost i komunikacije.
Distribuirani sistemi zahtjevaju mrežnu infastrukturu i mogu biti realizovani kao klijent/server ili kao ravnopravni računarski sistemi koji dijele resurse na mreži (peer-to-peer systems).

Najveći, danas prisutan distribuirani sistem predstavlja World Wide Web sistem koji je zaslužan za veliku popularizaciju umrežavanja i distribuiranih sistema uopšte.
Sama ideja Web-a je u predstavljanju svega kao dokument, te ovaj sistem pripada grupi sistema sa distribuiranim dokumentima. Web se danas sastoji od miliona klijenata i servera na kojima je dijeljeno više milijardi dokumenata.

Klasifikacija prema namjeni i načinu obrade

Klasifikacija prema namjeni i načinu obrade poslova
Prema namjeni, operativni sistemi se dijele na operativne sisteme opšte namjene (general purpose systems), koji mogu da obavljaju razne poslove, kao što su obrada teksta i slike, i operativne sisteme specijalne namjene, koji, po pravilu, služe za upravljače procesima.

Klasifikacija prema vrsti računarskog okruženja

Standardno govorimo o tri vrste računarskog okruženja (computing environments):

1] Tradicionalno (traditional computing)
Pod tradicionalnim okruženjem podrazumjevamo jednokorisničke računare ili sisteme s dijeljenjem vrijemena, gdje se korisnici preko svojih terminala povezuju na servere.

[2] Zasnovano na Webu (Web-based computing)
Ovo okruženje je zasnovano na umrežavanju po principima globalne mreže (kao što je Internet) na kojoj postoje Web serveri i klijenti u vidu PC računara i malih mobilnih sistema.

[3] Ugrađeno okruženje (embedded computing)
Ugrađeno okruženje je tipično za tvrde real-time sisteme koje odlikuje nepostojanje tastature, monitora i diskova. Aplikacija koja se izvršava u realnom vrijemenu dobija ulazne informacije preko ureoeaja kao što su senzori, dok se statusne i izlazne informacije prikazuju na LED diodama ili na nekom malom displeju.

Jednokorisnički OS
MS DOS

MS DOS (Microsoft Disk Operating System) je nastao 1981 godine i označio je početak ere PC računara. U slobodnom prevodu DOS znači operativni sistem za diskove (flopi disk i hard disk). *Novi operativni sistemi više ne koriste D u svom nazivu, jer rade mnogo više poslova od samog upravljanja diskovima.*

Kao uvod za moderne OS upoznaćemo se sa najslavnijim predstavnikom OS XX vijeka, već zaboravljenim DOS-om. Neki tvrde da nije baš zaboravljen, već ukopan u temelje Windows-a, i to toliko duboko da je njegov kernel.
DOS je ovdje, relarivno, detaljno obrađen sa brojnim korisničkim primjerima. Za to postoje dva razloga:

- Jednostavnost DOS to čini mogućim, za razliku od kompleksnijih operativnih sistema
- Principe koje sadrže drugi operativni sistem možete shvatiti na primjeru DOS-a

U ostatku priručnika daćemo principske postavke koje se univerzalno implementiraju u operativne sistema i koristiti konkretne operativne sisteme i primjere da ih ilustrujemo.

Operativni sistem DOS se koristi, kao i ostali OS, za pokretanje sistema, upravljanje hardverskim resursima kojima mašina raspolaže, rad sa njegovim vlastitim programima i podacima, manipulaciju datotekama i podršku u izvršavanju aplikacijskim programima koje korisnik instalira na svoju mašinu.

Po svojoj strukturi DOS spada u monolitne slojevite operativne sisteme mada je pseudovišeslojni (pseudolayer structure), što se može vidjeti na slikama ispod. Pošto slojevi nisu dobro odvojeni, programi mogu da direktno pristupaju I/O uređajima.

DOS se može podijeliti na četiri dijela:

- Modul eksternih korisničkih (user) naredbi
- Procesor naredbi (command processor)
- DOS jezgra (kernel)
- DOS BIOS.

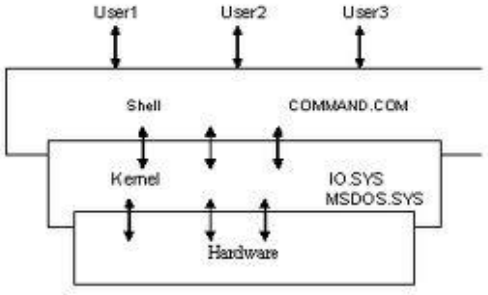
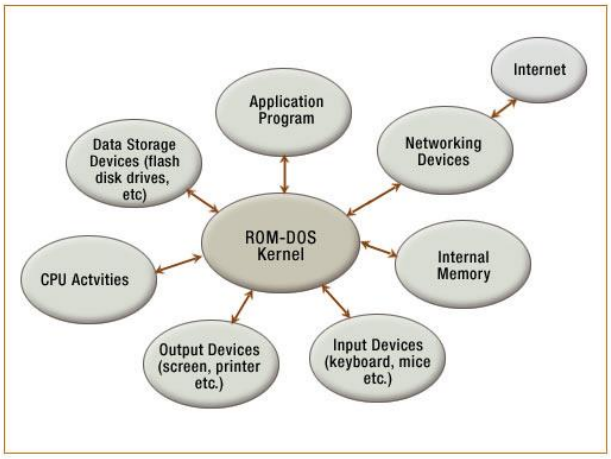
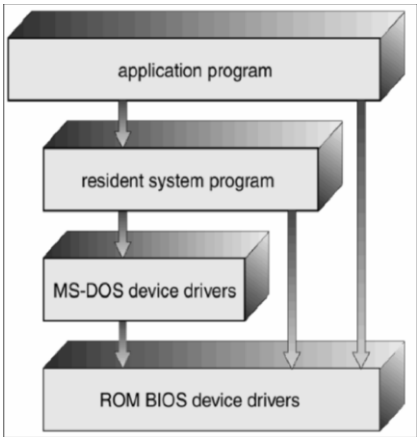


Fig - Hierarchy of interaction with MS-DOS

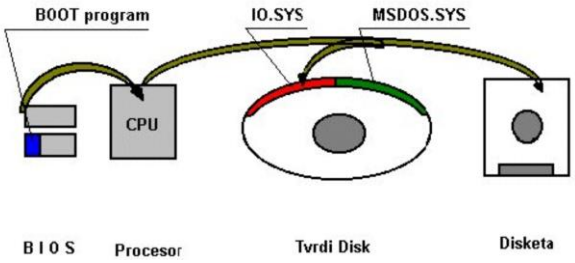




Na ilustracijama je prikazan način kako korisnik pristupa OS i kako pojedini dijelovi međusobno komuniciraju, što je dodatno objašnjeno u poglavljima *Proces učitavanja DOS operativnog sistema Korišćenje i izvršenje naredbi DOS operativnog sistema* i Dodatku.

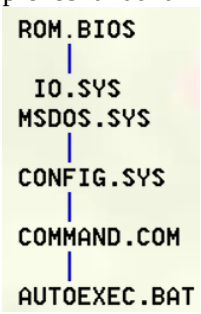
Proces učitavanja DOS operativnog sistema

Nakon uključenja računara, dolazi do pokretanja BIOS-a i testa hardvera sa kontrolom svih komponenti sistema i utvrdio da su sve komponente sistema priključene.
BOOT program, po zadatim instrukcijama, nastavlja proces podizanja sistema.
Prvo BOOT program pretražuje disketnu jedinicu u nadi da će tamo pronaći jednu formatiranu disketu na kojoj bi našao dalje instrukcije.
Ukoliko je nađe BOOT program je očitava i nastavlja dalje.
Ukoliko je ne nađe BOOT program nastavlja sa pretraživanjem te pretražuje tvrdi disk kao sljedeću jedinicu.
Ako BOOT program ni tamo ne nađe sledeće instrukcije, odnosno sistemske fajlove potrebne za nastavak procesa podizanja sistema, proces podizanja sistema se prekida i korisnik obaviještenje o grešci na sistemu.
Instrukcije koje BOOT program traži nalaze se na dva sistemska fila koja su inače smještena na sektoru 0 i tragu 0. Radi se o filovima io.sys i msdos.sys. Oni su ustvari sakriveni tako da korisnik nikad ne može doći u kontakt sa njima kako ih ne bi greškom izrisao. Oni se ne vide ni u DOS-fajl katalogu. pod ovim imenima ovi fajlovi su samo ako je riječ o Microsoftovom DOS operativnom sistemu dok se kod drugih proizvođača drugačije zovu.



Proces učitavanja sistemskih fajlova

Kad je BOOT program pronašao ta dva fajla na hard disku pročitao je program podatke koji se nalaze na prvom sektoru tvrdog diska kopirao ih na određenu adresu u radnu memoriju.
Na svim formatiranim diskovima je prvi sektor rezervisan za DOS. To se zove BOOT record koji ima informacije o tome kako ce BOOT program postupiti sa ta dva sakrivena fila. Kad je BOOT program prenio informacije iz BOOT recorda u radnu memoriju na hexadecimalnu adresu 7C00 prepušta BOOT program kontrolu BOOT recordu i ukazuje procesoru da je sada BOOT record taj sto vodi igru te da se njemu obrati za instrukcije.
Procesor se sada okreće adresi 7C00 u radnoj memoriji i radi dalje po instrukcijama.
Tako BOOT record preuzima kontrolu nad računarom i očitava ona dva sakrivena fila. Prije se očitava fil IO.SYS i prenosi u radnu memoriju.



IO.SYS sadrzava podatke koji kompletiraju BIOS. Jedan dio fila se naziva SYSINIT koji preuzima na sebe brigu o nastavku BOOT procesa.
Sada kontrolu na racunaru preuzima SYSINIT i prvo sto radi je da ocita fil msdos.sys te isti pohrani u radnu memoriju. Fil msdos.sys u nastavku zajedno sa BIOS preuzima rukovanje fajlovima te rukovanje signalima između pojedinih jedinica sistema.

SYSINIT traži u korijenskom direktoriju na tvrdom disku fajl config.sys. To je fajl koji pravi korisnik.
On sadrži informacije o podešavanjima u računaru kao što smo opisali ranije. On ima takođe instrukcije o tome koje zadatke SYSINIT dozvoljava msdos.sys da izvrši. To mogu biti instrukcije o tome kako povecati moc BIOS-a da rukuje memorijom racunara ili nekim od prikljucaka racunara npr. misom.
SYSINIT daje nalog msdos.sys da očita fajl command.com te istog pohrani u radnu memoriju.

Fajl command.com se sastoji iz tri dijela. Prvi dio je ustvari uvecanje mogućnosti BIOS za unošenje i iznošenje podataka. Ovaj dio se polaže u memoriju kao permanentni dio operativnog sistema.
Drugi dio fajla command.com sadrzi interne DOS komande kao DIR, COPI, i TYPE. One se memorisu u gornjem dijelu radne memorije i mogu biti upisani preko drugih programa.
Treći dio fajla command.com upotrebljava se samo jedanput i kasnije se izbacuje iz memorije. Taj dio traži u korijenskom direktoriju fajl autoexec.bat. Očitanjem fajla autoexec.bat i config.sys sistem je konfigurirsan i spreman je za podizanje operativnog sistema /Windows naprimjer/ a nakon toga i za upotrebu.
Program COMMAND.COM se mora nalaziti na svakoj sistemskoj disketi.
I pored toga, *on ne spada u operativni sistemi* (vidi POZICIJU U PREDHODNOM POGLAVLJU). Radi se o izvršnom (.COM) programu koji je sličan svakom drugom izvršnom programu, ali je on zadužen za vrlo bitne operacije.

Kod DOS-a je COMMAND.COM je procesor naredbi ili tzv. korisnička ljuska (shell).

Predstavlja vezu između korisnika i operativnog sistema. On interpretira naredbe i omogućava izvršavanje sistemskih datoteka
Nakon što je sistem izvršio zadane naredbe i nakon što je COMMAND.COM izvršio odgovarajuće pripreme i postavio odzivni znak (prompt) računar i operativni sistem su spremni za rad.

Korišćenje i izvršenje naredbi DOS operativnog sistema

DOS se koristi na dva načina:

- kroz linijske naredbe, ili
- kroz druge programe, u kojima preko menija koristimo mogućnosti DOS-a, a da pritom ne poznajemo način rada DOS naredbi niti sintaksu pisanja istih.

Komandna linija je linija u kojoj kucate komande (naredbe).
Komandni prompt (odzivnik) pokazuje da ste u komandnoj liniji.
Prompt može biti slovo za logičku oznaku disk jedinice praćeno obrnutom kosom crtom (backslash; "beksleš") (c:\ ili a:\, na Primjer) i nazivom direktorijuma (na Primjer, c:\dos).
Slovo pokazuje koja je disk jedinica aktivna jedinica.

MS-DOS pretražuje aktivnu disk jedinicu kako bi pronašao informaciju koja je potrebna da bi procesirao komande koje ste otkucali. Da biste naveli MS-DOS da izvrši zadatak, vi kucate komandu (na monitoru desno od komandnog prompta pojavljuju se znaci koje kucate), a zatim pritiskate taster sa natpisom "Enter".

Sve naredbe DOS-a prolaze kroz proces navođenja:

- Svaki put kad se zada naredba (preko tastature ili programski), DOS najprije provjerava da li naredba postoji u listi internih naredbi.
- Ako je naredba prisutna u memoriji, odmah se izvršava.
- Ako naredbe nema u popisu internih naredbi (znači da je eksterna), DOS pretražuje tekući direktoriji traži datoteke koje imaju ekstenziju .EXE, .COM ili .BAT.
- Ako se pronađe tražena datoteka, ona se učitava u memoriju, te se izvršava.
- U slučaju da DOS ne nađe traženu eksternu naredbu, šalje i ispisuje na monitoru: *Bad command or file name.*
- Kada je završen proces koji je pokrenula eksterna naredba, oslobađa se dio u memoriji u kojem je učitana eksterna naredba.

Poziv batch datoteke (tip BAT) uzrokuje čitanje svake linije u datoteci i izvršenje navedenih naredbi, redoslijedom navođenja.

Implementacije DOS-a

Pod implementacijom se podrazumjeva način primjene OS-a, a kod imlementacije DOS-a se najčešće govori o različitim pristupima ranih verzija PC računarima koje su najviše zavisile od proizvođača (tačnije distributera) OS-a. Postoji nekoliko implementacija a najjpoznatije su:

- PC-DOS – je IBM implementacija;
- MS-DOS- je Microsoft implementacija
- DR-DOS – je Digital Research-ova (kasnije Novell) implementacija

Sve ove impementacije funkcionišu na gotovo isti način.
DOS je također i apstrakcijski sloj u operativnim sistemima, ili samo dio u ranim verzijama Windows operativnog sistema.

Procesor naredbi

Procesor naredbi je odgovoran za izvršenje naredbi koje pozivamo.
Inicijalno (i standardno) DOS koristi COMMAND.COM kao procesor naredbi.

Moguće je definisati drugi procesor naredbi pomoću parametra
COMSPEC= u datoteci CONFIG.SYS: COMSPEC=DATOTEKA.COM
Većina korisnika DOS-a nema potrebe za korištenjem drugog procesora naredbi.
Procesor naredbi sastoji se od tri dijela:

- Inicijalni dio;

- Rezidentni dio;
- Tranzijentni dio.

Inicijalni –startni dio procesora naredbi
Inicijalni dio pokreće proceduru AUTOEXEC.BAT, ako ona postoji u glavnom (root) imeniku. Kada je AUTOEXEC.BAT procedura završena, inicijalni dio se briše i oslobađa svoje mjesto u memoriji.

Rezidentni dio procesora naredbi
Rezidentni dio je prisutan u memoriji cijelo vrijeme.
Ovaj dio izvršava funkcije kojima DOS mora pružiti trenutni odgovor, npr.:

- Učitavanje tranzijentnog dijela procesora naredbi,
- Izvršavanje i povratak iz aplikacijskih programa,
- Prekid programa kad korisnik pritisne tastere <Ctrl + Break>,
- Obrada standardnih grešaka.
- Takođe, ovaj dio ispisuje poruke tipa: Abort, Retry, Ignore

Tranzijentni dio procesora naredbi
Ovaj dio se nalazi na najvišim memorijskim adresama prisutnog RAM-a. Možemo reći da je to procesor naredbi u užem smislu, jer u sebi sadrži interpreter internih naredbi i procesor batch datoteka. Ovaj dio ispisuje odzivni znak OS-a, učitava i izvršava internu naredbu sa tastature ili batch datoteke. Kod eksternih naredbi, naredbe se moraju učitati s diska u memoriju. Kada se završi proces, koji je pokrenula eksterna naredba, oslobađa se dio memorije u koji je naredba učitana.

DOS jezgra (kernel)

Ovaj dio DOS-a je odgovoran za

- Rukovanje datotekama (kreiranje, brisanje ili promjena DOS datoteka),
- Rukovanje imenicima-direktorijima (kreiranje, brisanje ili modificiranje),
- Vezu između aplikacija i DOS servisnih funkcija.

Najznačajnije funkcije MS-DOS jezgre su: upravljanje memorijskim resursima, upravljanje sadržajem datoteka, iniciranje počeka i završetka programa, kontrola ulazno-izlaznih priključaka – portova računara.

DOS BIOS

Svaki računar ima set programskih funkcija, razvijenih da omoguće korišćenje ulaznih i izlaznih rutina niskog nivoa, za servisiranje priključnih komponenti računara. BIOS (Basic Input/Output System).
BIOS rutine su hardverski zavisne, te ih kod promjene hardverskih komponenti treba mijenjati (ako se priključuje komponenta različita od postojeće.
Ove rutine su smještene u ROM.
Svaki put, kad uključimo računar, DOS koristi rutine iz ROM-a, zajedno s datotekom IBMBIO.COM (PC-DOS) ili IO.SYS (MS-DOS) da kreira područje u memoriji koje je zaduženo za ulazno/izlazne operacije.
Programi koje nazivamo device driver prevodi paket naredbi iz formata kojeg DOS razumije u format naredbi koji razumije hardverski uređaj. Ovi programi olakšavaju priključivanje novog uređaja u konfiguraciju računara.

Zapis za pokretanje (dizanje) sistema (BOOT loader)

Zapis za podizanje sistema BOOT (Boot Record) je mali program veličine 512 bajta, pisan u mašinskom jeziku i zadatak mu je učitavanje operativnog sitema sa diska

Zapis za podizanje sistema zauzima nulti (i samo jedan) sektor na disku.
Pri resetu (poništenju, dovođenju u početno i poznato stanje) ili pri uključenju, računar uvijek očita sadržaj tog prvog sektora u svojoj memoriji i izvršava prvu naredbu tog sektora. Prva naredba je skok na dio programa koji resetuje disk, očitava osnovne parametre diska i očitava direktorij tog diska, kako bi provjerio da li se na disku nalazi operacijski sistem. Ako je operacijski sistem prisutan na disku, program za podizanje ga učitava i izvršava. Parametri diska koji su Takođe zapisni u Boot zapisu, govore o karakterističnim svojstvima diska.

DOS SHELL - alternativni komandni procesor

Komunikacija korisnika sa DOS-om zasniva se na programu COMMAND.COM koji nazivamo procesorom komandi ili komandnim procesorom (interpretatorom). Deklaracija SHELL omogućava izbor alternativnog komandnog interpretatora i, samim tim, kompletnu promjenu korisničkog interfejsa koji DOS nudi.
SHELL = datoteka [parametri]
datoteka - sadrži novi komandni procesor.
parametri - po startovanju se prosljeđuju komandnom procesoru. Za standardni COMMAND.COM parametri su /E:xxxxx (definiše veličinu radnog prostora) i /P (izvršava AUTOEXEC.BAT ako postoji).
SHELL = \YUCOM.COM - startovanje (imaginarnog) domaćeg komandnog interpretera; sve komande DOS-a su sada na našem jeziku!

Ako datoteka sadrži neispravan komandni procesor, sistem će se zakočiti. Moraćete da ga "podignete" sa diskete i da izbacite red SHELL iz CONFIG.SYS.

Uslužni programi

Uslužni su programi su sastavni dio MS-DOS operativnog sistema u širem smislu. Uslužne programe čine:

- DOSKEY (rezidentni program koji pamti zadnjih 20 instrukcija unesenih sa tastauture).
- EDIT (ASCII editor-uređivaĉ teksta) i
- HELP (korisniku omogućava prikaz kratkih uputa o instrukcijama MS-DOS-a).

Rad sa DOS-om

Operativni sistem DOS se koristi za pokretanje sistema, upravljanje hardverskim resursima kojima mašina raspolaže, rad sa njegovim vlastitim programima i podacima, manipulaciju datotekama i podršku u izvršavanju aplikacijskim programima koje korisnik instalira na svoju mašinu.
Obično korisnik ne poznaje šta operativni sistem radi u pozadini i na koji način pokreće sistem, raspoređuje memoriju, kako vodi organizaciju spoljne memorije, kako upravlja perifernim jedinicama, niti ijednu drugu radnju koju OS izvršava. Korisnik poziva komande koje nudi operativni sistem i druge aplikacije instalirane na mašini.
Druga je situacija, ako korisnik piše programe koji trebaju izvršiti određene zadatke, on u izvesnoj meri mora poznavati strukturu i način rada operativnog sistema kako bi na najoptimalniji način iskoristio sve resurse računara.

OPREZ! Neke komande mogu dovesti do gubitka svih prethodno snimljenih podataka na diskovima.

Na neke operacije operativnog sistema korisnik ne može uticati.
Nakon uključivanja računara doći će do podizanja sistema, a da korisnik ništa ne doprinese toj radnji.

Sa menjanjem sadržaja dva fajla, CONFIG.SYS i AUTOEXEC.BAT možemo uticati na neke operacije.
DOS provjerava sadržaj ta dva fajla i iz njih učitava niz naredbi koje definišu okruženje u kojem će mašina raditi. Većina ovih parametara definisanja sistema može se i kasnije, u toku rada predefinisati na nove vrijednosti.
Resursi na kojima upravlja operativni sistem (hardver i softver) korisnik pristupa preko komandi (naredbi).

Organizacija diska kod DOS-a

Da bi operativni sistem znao gdje se nalaze pojedine vrste informacija, on disk organizuje na način koji mu omogućava lak pristup svakoj lokaciji na disku i praćenje gdje se koji podatak nalazi i koliko slobodnog prostora na disku preostaje za buduće podatke koje treba smestiti.
Površina hard-diska (diskete) se dijeli na sektore i tragove. Od prvog traga se smješta dio operativnog sistema (programa) neophodnih za podizanje sistema. Tako je kod formatiziranja diska neophodno navesti opciju kojom će program za formatiranje izvršiti rezervaciju prostora za smještaj sistemskih datoteka IO.SYS i MSDOS.SYS.

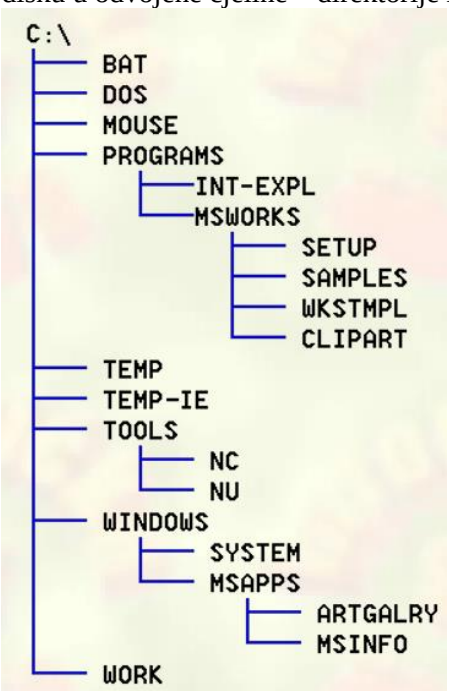
Hard diskovi se mogu dijeliti na više particija različitog kapaciteta, čiji je ukupni zbir kapaciteta jednak kapacitetu samog diska. Ovo se čini pomoću specijalnog programa FDISK.EXE koji se dobija u paketu sa operativnim sistemom DOS. Svakoj

particiji dodeljuje se jedno logičko ime. Ako se disk dijeli na particije, prva particija je obavezno butabilna (na njoj se nalazi operativni sistem za podizanje i upravljanje računarom) i ona ima logičko ime C:.

Operativni sistem koristi dvije tabele (također snimljene na disku), registratore u kojima se vode podaci o rasporedu dijelova datoteka po površini diska. Ove tabele se nazivaju FAT-tabelama (File Allocation Table), od kojih je jedna rezervna tabla i koristi se u slučaju uništavanja originala. Ove tabele se obavezno ažuriraju prilikom svake promjene (snimanja ili brisanja, odnosno promjene veličine neke datoteke). U FAT tabeli se vodi spisak kataloga, datoteka, tragova i sektora gdje su smješteni pojedini dijelovi datoteka.

Logička organizacija podataka

Kako se na diskove može smjestiti velika količina podataka, OS mora brzo prepoznati na kojem se dijelu diska nalazi korisniku trenutno potrebni podatak (datoteka). Zbog toga se koristi organizovana podjela svih podataka na disku u odvojene cjeline – direktorije i poddirektorije.



Konfiguracija podsjeća na stablo kojemu je korijen i osnova (ROOT) s koje se razvijaju grane (imenici) i grančice (podimenici).
Na slici uz samostalne direktorije BAT, DOS ili TEMP, prikazani su direktoriji programske grupe MSWORKS i dio programske grupe WINDOWS.

DOS komande

Komande –naredbe- su skup karaktera veličine od jednog do osam karaktera (istih onih koji se koriste za imenovanje datoteka), nakon čega obično ne slijedi ništa ili jedan prazan karakter i skup parametara koji definišu način na koji se naredba izvršiti, odnosno nad kojim medijem.

DOS komande mogu biti:
- interne i
- eksterne.

Interne komande su permanentno prisutne u operativnoj memoriji računara od momenta podizanja sistema do njegovog gašenja.
Eksterne komande se učitavaju sa diska i u memoriji su prisutne samo dok su aktivni programi koje one izvršavaju.
Druga podjela komandi operativnog sistema je:
- komande za rad sa katalozima,
- komande za rad sa datotekama,
- komande za predstavljanje parametara sistema i
- pomoćne komande.

Naziv direktorija, komandi i naredbi, odnosno datoteka ograničen je na najviše 8 znakova bilo slova abecede ili brojki (ne preporučuje se upotereba naših slova).

MS-DOS komanda može imati do tri (sastavna) dijela.

Svaka komanda ima ime.

Neke komande zahtjevaju jedan ili više parametara koji identifikuju objekt sa kojim vi želite da MS-DOS nešto uradi.

Neke komande takođe uključuju (obuhvataju) jedan ili više tzv. prekidača (switches; "svičevi"), koji modifikuju aktivnost koja treba da bude izvršena. Naziv (ime) komande, koji prvo kucate, određuje akciju koju želite da MS-DOS izvede.

Neke komande (takve kao **CLS** - CLear Screen - komanda koja briše ekran) sastoji se samo od naziva. Međutim, većina MS-DOS komandi zahtjeva još nešto osim prostog navođenja naziva. MS-DOS ponekad zahtjeva dodatnu informaciju koja se specifikuje parametrima posle naziva komande. Parametar definiše objekat na kome vi želite da MS-DOS izvrši neku aktivnost.

Na primjer, komanda DEL zahtjeva da imenujete datoteku koju želite da obrišete.

Pretpostavimo da želite da obrišete datoteku SPISAK.TXT. To ćete postići ako otkucate: **del** spisak.txt

Neke komande zahtjevaju više od jednog parametra. Na primjer, želite da preimenujete ("prekrstite") datoteku koristeći komandu **RENAME** (skraćeni oblik REN). U tom slučaju morate uključiti i originalno ime (naziv) datoteke i novo ime datoteke.

Kod nekih komandi parametri su opcioni (neobavezni).

Na primjer, ako koristite DIR komandu bez parametara, na ekranu će se pojaviti lista datoteka iz direktorijuma koji trenutno koristite. Ako uključite parametar (na primjer, oznaku druge disk jedinice), izlistaće se datoteke u drugom direktorijumu.

Prekidači

Prekidač je kosa crta (/. forward slash), obično praćena jednim slovom ili brojem. Prekidači se koriste da bi se modifikovao način na koji komanda izvršava zadatak. Na primjer, pretpostavimo da želite da upotrebite komandu DIR da biste vidijeli listing direktorijuma koji sadrži veliki broj datoteka. Kada otkucate samo komandu DIR, nazivi datoteka se izlistavaju na ekranu tako brzo da ne možete stići da ih pročitate. Ako dodate prekidač /p, možete pregledati listu datoteka dio po deo, tj. ekran po ekran. Neke MS-DOS komande nemaju uopšte prekidače, dok neke imaju po nekoliko. Ako komanda ima više od jednog prekidača, oni se navode jedan posle drugog. Prekidači se razdvajaju "blankom" (jednim praznim mestom), ali to je opciono (neobavezno).

Kucanje komande

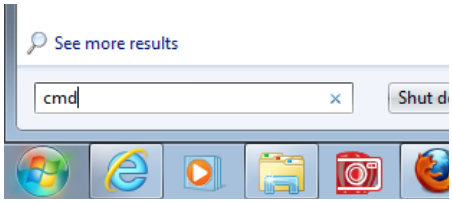
Trepćuća crtica (flashing underscore) u komandnoj liniji predstavlja kursor. Kursor vam pokazuje gdje da kucate komandu. Kada otkucate znak, kursor se pomera za jedno mesto udesno. Ako ukucate pogrešan znak, pritisnite tzv. "bekspejs" (backspace) taster ("<-" - taster u gornjem desnom uglu alfanumeričke tastature) da obrišete znak lijevo od kursora.

Komande možete kucati velikim ili malim slovima.
Ukoliko nije drugačije specifikovano, komandu morate razdvojiti od parametara praznim mestom (pritiskom na razmaknicu - najduži taster na alfanumeričkoj tastaturi). Ako želite da ponovo otkucate (retype) komandu, pritisnite "Esc" taster. Kursor se premješta na početak nove linije pa možete ponoviti komandu. Sve što ste otkucali pre pritiska na taster "Esc" biće ignorisano.

U DOS-u imamo dva džoker-znaka:

- ”?” znak zamjenjuje bilo koji dozvoljeni karakter, a znak
- “*” zamjenjuje grupu karaktera u imenu i ekstenziji fajla.

Umjesto starog DOS-a pokrenite njegovu simulaciju



Ako DOS pokrećemo iz Windowsa (kao virtuelnu mašinu) pozivamo ga komandom **cmd.**
Umjesto GUI okruženja dobićemo komandolinijski mod, koji je emulacija DOS operativnog sistema..

Saznajte sve CMD komande (tačnije sve DOS komande podržane u cmd modu).
Umjesto da pamtite **ukucajte help** (nakon što pokrenete cmd) i dobićete informacije o svim komandama

Osnovne komande su korisne za obavljanje jednostavnih zadataka, ali realizacija komplikovanijih problema zahtjeva šire komande.
Sa svakom novom verzijom operativnog sistema Windows komande za DOS komandu liniju su promjenjene. U svakoj verziji komande su dodate, izbrisane i izmjenjene, ali zadržavaju se osnovni principi.

DOS komandna linija je korisna kad je Windows sistem datoteka oštećena ili ste slučajno izbrisali fajl, ili greška nastala zbog virusa. Popravka iz DOS-a je često jedina opcija.

Netvork komande (izvorno: DOS network commands) pružaju mnoge mrežne alatke za komandu liniju. DOS komande za mrežu pružaju vitalne informacije za pomoć inženjerima i administratorima za dijagnozu problema.

Primjer upotrebe i objašnjanje nekih komandi

Razgledanje tekstualnih datoteka
Da biste na monitoru vidjeli sadržaj tekstualne datoteke koristite komandu TYPE.
Na primjer, da biste vidjeli sadržaj datoteke LISTA.TXT na disketi u drajvu B, treba da koristite sljedeću komandu: **type b:lista.txt**

Ako je datoteka koju želite da razgledate velika, treba da koristite znak. "pajp" (pipe; tj. znak |) iza koga slijedi komanda MORE: type b:lista.txt | more
Uključujući komandu MORE dobili ste mogućnost da razgledate datoteku ekran po ekran.

PREPORUKA: Ukoliko ne koristite komandu MORE kada koristite TYPE, možete privremeno zaustaviti prikaz datoteke na monitoru istovremenim pritiskanjem tastera "Ctrl" i "S", odnosno pritiskom na taster "Pause". Da biste nastavili "skrolovanje", pritisnite bilo koji taster (osim "Pause"). Da biste trajno prekinuli prikazivanje datoteke na monitoru, pritisnite istovremeno tastere "Ctrl" i "C" ili "Ctrl" i "Break"; ove kombinacije trastera poništavaju komandu TYPE.

Kopiranje datoteke na printer
Da biste kopirali datoteku na svoj printer kao odredišnu datoteku treba da specifikujete naziv porta (priključka) na koji je printer priključen.
Na primjer, sljedeća komanda kopira datoteku NAPOMENA.TXT sa diskete u drajvu A na printer priključen na port LPT1: **copy a:napomena.txt lpt1**
Koristeći CON i naziv porta na koji je priključen printer možete kopirati direktno unos sa tastature na printer. Na primjer, neka je printer priključen na LPT1 port : **copy con lpt1**
Kada završite sa slanjem informacija (sa kucanjem) printeru, pritisnite istovremeno tastere "Ctrl" i "Z", a zatim pritisnite taster "Enter" da bi se odštampalo to što ste otkucali.

Komande za redirekciju (preusmjeravanje) ulaza i izlaza
Ukoliko ne specifikujete drugačije, MS-DOS prihvata unos (ulaz) sa tastature i šalje izlaz na ekran (monitor).
Ponekad je korisno preusmjeriti ulaz ili izlaz u datoteku ili na printer.
Na primjer, listing nekog direktorijuma možete preusmjeriti sa ekrana u datoteku.
Za redirekciju ulaza i izlaza komandi koriste se sljedeći znakovi:
• znak "veće od" (>) šalje izlaz komande u datoteku ili na uređaj (npr. printer);
• znak "manje od" (<) prihvata ulaz potreban za komandu iz datoteke, a ne sa tastature;
• dvostruki znak "veće od" (>>) dodaje izlaz komande na kraj datoteke bez brisanja informacija koje su već u datoteci.

Redirekcija izlaza komande
Skoro sve MS-DOS komande šalju izlaz na ekran.
Čak i komande koje šalju izlaz na drajv ili printer prikazuju poruke i promptove na ekranu. Da biste preusmjerili izlaz sa ekrana u datoteku ili na printer, koristite znak "veće od" (>).
Na primjer, u sljedećoj komandi listing direktorijuma koji formira komanda DIR preusmjerava se u datoteku DIRLIST.TXT: **dir > dirlist.txt**

Upravljanje memorijom
MS-DOS sadrži sljedeće programe za upravljanje memorijom:
• HIMEM, koji omogućuje pristup proširenoj memoriji
• EMM386, koji koristi proširenu memoriju da simulira (podražava) produženu memoriju.

MS-DOS obično radi u konvencionalnoj memoriji, a tako manje konvencionalne memorije ostaje na raspolaganju za programe. Međutim, ako vaš sistem ima proširenu memoriju, MS-DOS može da radi u proširenoj memoriji. Kada je to ispunjeno, on koristi prva 64 kB proširene memorije, koji se zovu visoki memorijski prostor (High Memory Area - HMA). Kako samo nekoliko programa može da koristi HMA, to ima smisla da se MS-DOS izvršava baš tu. Da bi se koristile ove pogodnosti, u konfiguracionu datoteku (CONFIG.SYS) treba dodati komande kojima se instaliraju programi za upravljanje memorijom.
Na primjer: **device = c:\dos\himem.sys dos = high,umb**
Ove komande prvo učitaju program za upravljanje memorijom HIMEM, a zatim učitaju MS-DOS u proširenu memoriju.

Komanda UMB specifikuje da MS-DOS treba da održava vezu između konvencionalne memorije i gornjeg memorijskog prostora.
Da biste instalirali EMM386 kao expanded-memory emulator, u konfiguracionu datoteku treba da dodate "device" komandu:
device = c:\dos\emm386.exe 640
Ova komanda specifikuje da EMM386 treba da koristi proširenu (extended) memoriju kako bi simulirao produženu (expanded) memoriju i rezerviše 640 kB proširene memorije u tu svrhu.

SMARTDrive ("smartdrajv") predstavlja program za ubrzavanje ("keširanje") hard diska kod računara koji imaju hard disk i proširenu ili produženu memoriju.
device = c:\dos\smartdrv.sys 1024
Ova komanda specifikuje da je SMARTDRV.SYS u direktorijumu c:\dos i da je veličina keša 1024 kB. Podrazumeva se da SMARTDRV radi u proširenoj memoriji.

Izbor boja DOS prozora

Izbor boja se vrši comandom color i sa dva hheksadecimalna broja: **color XX;** npr**. color 2F**

Prvi broj definiše pozadinu (background), a drugi boju teksta.
U tabeli je dat set koji definiše boje.
Ako se broj ne navede vrati se predefinisane.

0 = Black	8 = Gray
1 = Blue	9 = Light Blue
2 = Green	A = Light Green
3 = Aqua	B = Light Aqua
4 = Red	C = Light Red
5 = Purple	D = Light Purple
6 = Yellow	E = Light Yellow
7 = White	F = Bright White

Izrada komandnih skripti: batch fajlova

DOS se može „natjerati“ da u izvjesnom smislu **koristi sam sebe** da bi obavio neki posao. Reč je o tzv. BAT datotekama; one u stvari sadrže nizove DOS komandi koje se, kada ih startujete, izvršavaju potpuno automatski.
Šta je BATCH datoteka-fajl / skripta ?
To je datoteka sa nastavkom (extenzijom) BAT koja sadrži niz naredbi sistema ili posebnih "BATCH" naredbi.
Naredbena skripta ili Batch datoteka je program ili neformatirana tekstualna datoteka koja sadrži jednu ili više naredbi sistema ili "batch" naredbi.
Batch fajlovi ili batch programi kako se najčešće nazivaju, omogućavaju kreiranje programa koji upravljaju radom operativnog sistema DOS.
Cilj kreiranje batch fajla je da se automatizuje DOS-ov proces, odnosno da se operater oslobodi od dugotrajnog unošenja komandi u DOS linijama.
Batch fajlovi omogućavaju izvršavanje određenih operacija zadatih od strane korisnika i to na sljedeći način. Batch skripte pišemo u txt editoru (notepad-u) a da bi smo dobili batch fajl, jednostavno ćemo dodati .bat ili .cmd extenziju.

primjer 1

```
@echo off
title Test batch fajla

echo OBJASNJENJE:

echo.
echo Ispod se nalaze batch komanda
echo koje ce izvrsiti pokretanje aplikacije kalkulator.
echo Na kraju ce vam pisati: "press any key to continue..."
echo i nakon pritiska na enter izbrisace se predhodni sadrzaj cmd prozora
echo pa cete nakon ponovne poruke "press any key to continue..."
echo izacicete iz cmd-a
```

```
echo u okviru koga je i pokrenut ovaj batch.

echo.
echo Pokrecem kalkulator
start calc

echo.
pause
cls
pause
exit
```

primjer 2
Evo još jednog primjera - batch fajl koji otvara dva programa istovremeno.
Ukucajte u Notepad (sa navodnicima):

```
@echo off
title Start dva programa

"C:|Program Files|Microsoft Office|Office|OUTLOOK.EXE"
"C:|Program Files|Internet Explorer|IEXPLORE.EXE"

pause
exit
```

Nakon što snimite ovo sa nastavkom .bat imate batch fajl kojim možete jednim duplim klikom (ili kako već to volite da radite) istovremeno da pokrenete 'Internet Explorer' i 'Outlook' (ako se kod Vas IE i Outlook nalaze na drugoj lokaciji, onda prilagodite putanju).

primjer 3
batch fajl ispisuje konfiguraciju vašeg računara.

```
@echo off
ipconfig /all
pause
exit
```

primjer 4
Skripta koja kopira sadržaj direktorija u datoteku v1.0

```
@echo off
%ime   rem ime = Ime datoteke u koju ce se ispisati sadržaj direktorija:
dir > %ime
pause
exit
```

primjer 6 Backup:
Imate, npr. pod C:\My Documents fajlove koje želite da iskopirate na drajv D:\ u folder 'Backup'. Pri tome želite da iskopirate samo one fajlove koji su promjenjeni u odnosu na prethodni backup, a želite da iskopirate i foldere i podfoldere kao i sistemske i skrivene foldere. Da bi to uradili možete upotrebiti batch fajl poput ovog:

```
@echo off
md d:|Backup
xcopy "C:|My Documents|*.*" d:|Backup| /d /y /i /h /s
cls
```

Ovaj batch fajl pravi folder 'Backup'; a ako već imate folder 'Backup', onda izostavite drugi red.
Možete napraviti i log fajl koji će prikazati rezultate: dodajte na kraj trećeg reda > *d:\Backup\xcopy.log*.
Da bi se kopirali samo fajlovi nastali posle određenog datuma, dodajte uz */d* parametar željeni datum.
Ostale parametre za xcopy možete da vidite ako otvorite cmd i ukucate help xcopy (i pritisnete Enter).

primjer 6
Kreiranje menija korišćenje i poziv korišćenje programskih skokova

```
ECHO OFF
title DOS nije mrtav, a ja pozdravljam Jovu i Peru
REM pogledaj ime programa-prozora

:MENU
CLS
color 0F
REM pogledaj boje
ECHO .....
ECHO IZABERI 1 ili 2 da popricamo ili 3 ZA KRAJ.
ECHO .....
ECHO 1 - Ako si Jovan izaberi 1
```

```
ECHO 2 - Ako si Pero izaberi 2
ECHO 3 - EXIT-izlaz
ECHO.

SET /P M=Type 1, 2, ili 3 then press ENTER:
cls
IF %M%==1 GOTO jovo
IF %M%==2 GOTO pero
IF %M%==3 GOTO kraj

ECHO POGRESAN IZBOR
pause>nul
GOTO MENU

:jovo
color 47
ECHO Zdravo Jovo. Biraj ponovo
pause>nul
GOTO MENU

:PERO
color 1e
ECHO Zdravo Pero. Kako si?
pause>nul
GOTO MENU

:KRAJ
ECHO Sa ostalim ne komuniciram!
pause>nul
EOF
```

primjer SAMO za napredne i poznavaoce: Advanced Windows batch example - conditional shutdown

```
@echo off
color 0A
title Conditional Shutdown.

:start
echo Welcome, %USERNAME%
echo What would you like to do?
echo.
echo 1. Shutdown in specified time
echo 2. Shutdown at a specified time
echo 3. Shutdown now
echo 4. Restart now
echo 5. Log off now
echo 6. Hibernate now
echo.
echo 0. Quit
echo.

set /p choice="Enter your choice: "
if "%choice%"=="1" goto shutdown
if "%choice%"=="2" goto shutdown-clock
if "%choice%"=="3" shutdown.exe -s -f
if "%choice%"=="4" shutdown.exe -r -f
if "%choice%"=="5" shutdown.exe -l -f
if "%choice%"=="6" shutdown.exe -h -f
if "%choice%"=="0" exit
echo Invalid choice: %choice%
echo.
pause
cls
goto start

:shutdown
cls
set /p sec="Minutes until shutdown: "
set /a min=60*%sec%
shutdown.exe -s -f -t %min%
echo Shutdown initiated at %time%
echo.
```

```
goto cancel

:shutdown-clock
echo.
echo the time format is HH:MM:SS (24 hour time)
echo example: 14:30:00 for 2:30 PM
echo.
set /p tmg=enter the time that you wish the computer to shutdown on:
schtasks.exe /create /sc ONCE /tn shutdown /st %tmg% /tr "shutdown.exe -s -t 00"
echo shutdown initiated at %tmg%
echo.

:cancel
set /p cancel="Type cancel to stop shutdown: "
if not "%cancel%"=="cancel" exit
shutdown.exe -a
cls
schtasks.exe /end /tn shutdown
cls
schtasks.exe /delete /tn shutdown
cls
echo Shutdown is cancelled.
echo.
pause
exit
```

Višestruke konfiguracije sistema

CONFIG.SYS sadrži informacije o hardverskoj konfiguraciji sistema. Niz naredbi u ovoj datoteci se izvršava samo jednom pri podizanju sistema, tako da svaka promjena u datoteci dolazi do izražaja tek prilikom slijedećeg podizanja sistema.
Config.sys je obična tekstualna datoteka koju možemo editirati bilo kojim editorom. Unutar datoteke config.sys može se upotrijebiti 18 različitih naredbi kako bi se izvršilo podešavanje sistema, od kojih se tri (BREAK, SET i REM) mogu koristiti i u komandnoj liniji.

Čest je slučaj da za razne poslove PC treba konfigurisati na različite načine - kada se radi sa Windows-om pogodno je da u memoriji bude samo HIMEM.SYS, ako se radi u Venturi potreban je EMS drajver, za "normalan" rad treba imati disk keš i neke rezidentne programe i tako dalje.
Prepravke CONFIG.SYS iz komandne linije je naporno i nepotrebno
Kreiranje raznih CONFIG datoteka i njihovo aktiviranje preko .BAT programa je znatno prihvatljivije, ali i dalje zahtjeva dosta pažnje i pravi priličan haos na disku.
Jednostavan CONFIG.SYS koji ilustruje ove tehnike mogao bi da glasi:

```
[Menu]
MenuItem=Standard
MenuItem=DosOnly
MenuDefault=Standard,4

Common]
BREAK ON
BUFFERS = 25
FILES = 99

Standard]
DEVICE=C:\DOS\HIMEM.SYS
DEVICE=C:\DOS\EMM386.EXE RAM
DOS=HIGH,UMB
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /E:1024 /P

[DosOnly]
DEVICE=C:\DOS\HIMEM.SYS
```

Po startovanju računara ispisuje se meni u kome figurišu pobrojane opcije Standard i DosOnly.
Ukoliko tokom sljedeća 4 sekunda (parametar MenuDefault direktive) ništa ne učinite, biće izvršena [Common] i [Standard] sekcija dok će izbor opcije 2 izazvati izvršenje [Common] i [DosOnly] deklaracija.

CONFIG.SYS mora da se uskladi sa AUTOEXEC-om, na primjer na sljedeći način:

```
@ECHO OFF
PROMPT $P$G
PATH C:\DOS;C:\BAT
GOTO %CONFIG%

:Standard
LOADHIGH C:\UTIL\CED -FC:\UTIL\CED.DEF
LOADHIGH C:\BAT\XK
LOADHIGH C:\DOS\MOUSE
C:\KWIK\SUPERPCK /E /&U+ /S:3000 /I- /P- /-A /-B
GOTO End

:DosOnly
C:\DOS\DOSKEY
:End

CLS
D:
```

Organizacija, naravno, može da bude i drukčija ali poželjno je da se podjeli na

- obaveznu: gdje se navedu stvari koje u svakom slučaju treba izvršiti (npr. prompt, path itd)
- opcionu: izbornu sa GOTO %CONFIG%, "skoči" na dio procedure koji odgovara izabranom stanju

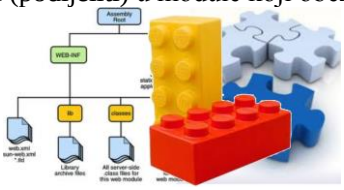
Promjenljiva CONFIG se automatski definiše tokom interpretiranja CONFIG.SYS-a. Svaka od "grana" AUTOEXEC-a završava se sa **GOTO End** da program preskoči narednu.

Greške u datoteci CONFIG.SYS bivaju prijavljene u toku "podizanja" sistema i da računar pogrešne redove jednostavno ignoriše.
Može se, međutim, dogoditi da neki neispravno instaliran drajver ili linija SHELL izazovu blokadu sistema u kom slučaju treba startovati računar bez CONFIG.SYS-a kako bi se greška mogla otkloniti.
Kada se pojavi poruka*: Starting MS-DOS,*
pritisnete F5 pa će kompletan CONFIG.SYS i AUTOEXEC.BAT će biti "preskočen"
dobićete MS-DOS is bypassing your CONFIG.SYS and AUTOEXEC.BAT files
i uobičajeni prompt C>>.
Ukoliko niste sigurni koja linija vam pravi probleme, umjesto F5 pritisnite F8 i CONFIG.SYS će biti izvršavan liniju po liniju, pri čemu sa Y ili N odlučujete da li će svaka konkretna linija biti izvršena ili preskočena.

Moduli kod OS

Kao što je naglašeno savremeni OS se može dekomponovati (podijeliti) u module koji obezbjeđuju sljedeće funkcije:

- upravljanje procesima
- upravljanje memorijom
- upravljanje U/I uređajima
- upravljanje datotekama
- upravljanje mrežom.



Zadatak OPERATIVNOG SISTEMA je da upravlja fizičkim (procesor, kontroleri i radna memorija) i logičkim (fajlovi i procesi) djelovima kompjutera, pa se operativni sistem može podijeliti na module, koji izvršavaju te funkcije.

Postoji nekoliko prednosti modularnih OS:

- lakše je modifikovati sistem i ispravljati greške jer promjene utiču samo na neke dijelove sistema,
- informacije se čuvaju samo gdje je to potrebno,
- informacijamma se pristupa samo unutar definisane i ograničene oblasti.

Modul za upravljanje procesorom

Priključenje procesora sa jedne na drugu nit je zadatak ovog modula.Uvodi **operaciju preključivanja** čiji poziv dovodi do preključivanja procesora **sa jedne niti na drugu**, koje mogu pripadati istom ili raznim procesima.
U toku preključivanja procesora između niti istog procesa ne dolazi do izmene adresnog prostora procesa, pa je ovakvo preključivanje brže (kraće) nego preključivanje procesora između niti raznih procesa.

Modul za upravljanje I/O kontrolerima

Ovaj modul poziva **operacija preključivanja**.
Pošto upravljanje I/O uređajima zavisi od vrste uređaja, modul za upravljanje kontrolerima se sastoji od niza komponenti, nazvanih drajveri. Modul za upravljanje kontrolerima ostvaruje svoj zadatak tako što uvodi (drajverske) operacije ulaza i izlaza. Upravlja raznim ulazno/izlaznim uređajima koji su priključeni na kontrolere (tastatura, miš, ekran, štampač, odnosno uređaji masovne memorije kao što su diskovi, diskete, CD/DVD ROM-ovi, itd.).

Modul za upravljanje radnom memorijom

Modul za upravljanje radnom memorijom ostvaruje svoj zadatak tako što uvodi **operacije zauzimanja i oslobađanja**.
Vodi evidenciju o slobodnoj radnoj memoriji radi zauzimanja zona slobodne radne memorije, odnosno radi oslobađanja prethodno zauzetih zona radne memorije.
Iz modula za upravljanje radnom memorijom pozivaju se operacije ulaza i izlaza.

Modul za upravljanje fajlovima

Modul za upravljanje fajlovima ima **operacije čitanja i pisanja**.
Omogućava otvaranje i zatvaranje fajlova, odnosno čitanje i pisanje njihovog sadržaja i vodi evidenciju o blokovima (masovne memorije) u kojima se nalaze sadržaji fajlova.
Brine se i o prebacivanju djelova sadržaja fajlova između radne i masovne memorije (druge dvije funkcije), a za ovo prebacivanje su potrebni baferi, pa se poziva i operacija zauzimanja dovoljno velikog baferskog prostora.
Iz modula za upravljanje fajlovima pozivaju operacije oslobađanja, ulaza i izlaza.

Modul za upravljanje procesima

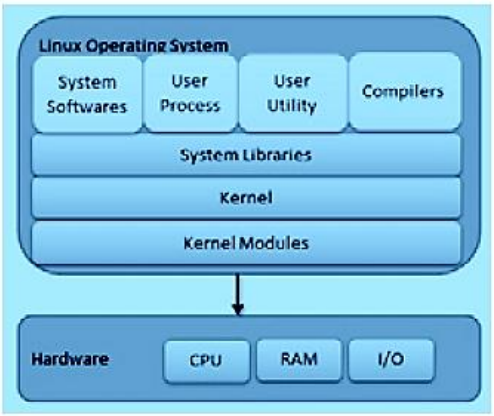
Modul za upravljanje procesima ima **operacije stvaranja i uništavanja**.

Omogućava stvaranje i uništavanje procesa, kao i stvaranje i uništavanje njihovih niti, odnosno omogućava istovremeno postojanje više procesa (višeprocesni režim rada), tj. više niti.
Poziva operaciju čitanja, radi preuzimanja sadržaja izvršnih fajlova, koji su potrebni za stvaranje slike procesa, a pošto je za stvaranje slike procesa potrebna radna memorija, pozivaju se i operacije zauzimanja, odnosno oslobađanja.
Iz modula za upravljanje procesima pozivaju se operacije čitanja, pisanja, zauzimanja i oslobađanja.

Modularni operativni sistemi

Modularni operativni sistem je izgrađen sa svojim različitim funkcijama podeljenim u različite procese, svaki sa svojim sopstvenim interfejsom. Nasuprot tome, tradicionalni monolitni operativni sistem koristi jednu statički kompajliranu sliku i radi u režimu „sve ili ništa“. Ako bilo koji element ili aplikacija unutar monolitnog operativnog sistema otkaže ili treba ažuriranje, cijeli sistem mora biti ugašen i ponovo pokrenut, prekidajući sve tokove paketa.
Primarna prednost modularnog pristupa je da svaki proces radi nezavisno, a ako jedan od njih ne uspije ili je potrebno ažuriranje, to neće utjecati na druge funkcije.
Modularni operativni sistem značajno poboljšava vrijeme trajanja infrastrukture kompanije do nivoa koji se približava željenim 99,999% (pet devetki) i maksimizira dostupnost svih njenih poslovnih kritičnih aplikacija.

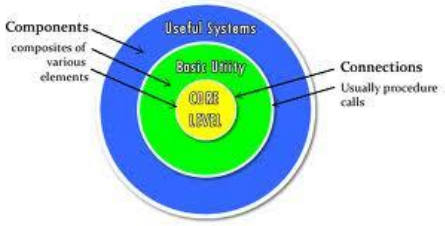
Glavni elementi modularnog operativnog sistema su kernel i skup dinamički učitivih aplikacija sa sopstvenim diskretnim memorijskim prostorom. Kernel je zaštićen od kvarova servisa i aplikacija.
Svaki proces se može pratiti kako bi se utvrdilo da li radi ispravno, a ako je potrebno, neispravan proces može se dinamički ponovo pokrenuti. Cjelokupni sistem može nastaviti s radom tokom popravka. Ovo održava vrijeme neprekidnog rada osnovne infrastrukture, sistemskih aplikacija i cjelokupnog operativnog sistema. Ova mogućnost elegantnog pokretanja i zaustavljanja aplikacije, bez uticaja na cio sistem, omogućava brz odgovor na novonastale pretnje i bezbjednosne propuste i olakšava upravljanje aplikacijama uz česte cikluse osvežavanja.



Linux kernel je modularan, što znači da može proširiti svoje mogućnosti kroz dinamički učitane module kernela.
Vodeće distribucije Linuxa za preduzeća kao što je SUSE Linux Enterprise Server grade na ovom modularnom Linux kernelu i nude mnoge dodatne module koji sadrže softverske pakete koji dodaju funkcionalnost operativnom sistemu sa svim prednostima jednostavnog upravljanja i visoke dostupnosti.

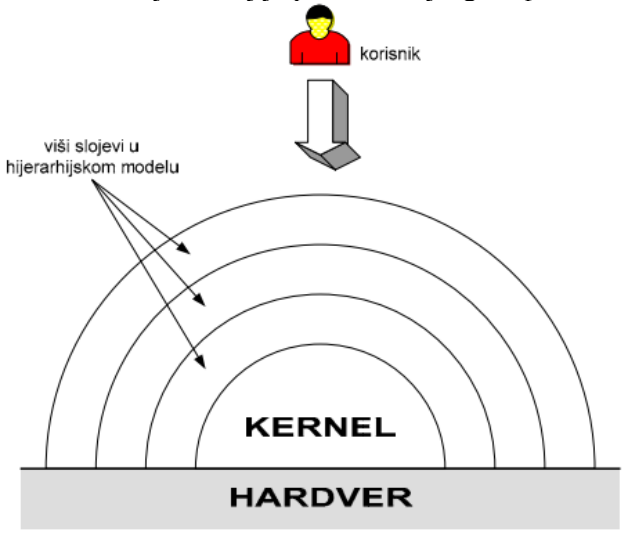
Moduli i hijerajhija

Korišćenje modularnog pristupa kod OS čiji kodovi imaju više miliona instrukcija nije bilo dovoljno, pa su moduli grupisani u hijerarhijske nivoe.



Generalno, možemo reći da jednostavni operativni sistemi mogu imati monolitnu strukturu, a složeniji (oni koji upravljaju računarima koji obavljaju kompleksnije zadatke) slojevitu hijerarhijsku realizaciju.

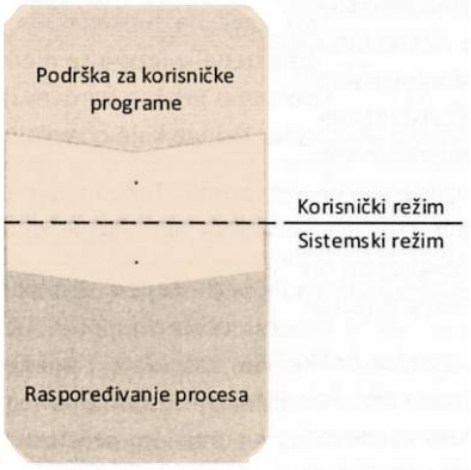
Kod hijerarhijskog modela na posmatranom nivou operativnog sistema mogu se zahtjevati usluge samo od njegovih nižih nivoa, a nikako od viših. Najniži sloj je poznat kao jezgro operativnog sistema (nucleus, kernel).



Hijerarhijski model operativnog sistema

Slojeviti operativni sistemi (layered systems)

Slojevita arhitektura podrazumjeva da je operativni sistem izgrađen od zasebnih slojeva (cjelina) koji se nadograđuju jedan na drugi.

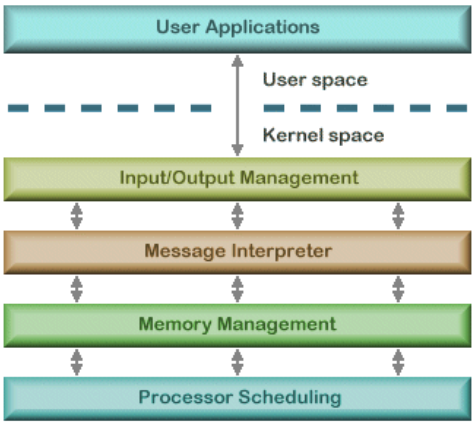


Pri tome, svaki sloj ima određene funkcije koje su opisane kroz njegov interfejs ka višem sloju. Slojevi se implementiraju tako da mogu da koriste isključivo usluge prvog sloja ispod sebe. Projektovanje slojeva je zahtjevan posao jer se mora voditi računa o raspodjeli funkcija.

Problem kod ovakvih sistema je neefikasnost. Naime, sistemski poziv prolazi kroz više slojeva a pri svakom prolazu se prosljeđuju podaci, mijenjaju parametri itd. što dovodi do usporenja.

Prvi operativni sistem koji je napravljen na ovaj način je operativni sistem sa imenom THE (Technische Hogeschool Eindhoven) od strane E.W.Dijkstre.
Sistem je imao 6 slojeva kao što je prikazano na sledećim slikama:

Layer	FUNCTION
5.	The Operator
4.	User Programs
3.	INPUT/OUTPUT Managment
2.	Operator-Process Comunication
1.	Memory And Drum Managment
0.	Processor Allocation And Multiprogramming



Nulti sloj (*processor scheduling–raspored rada procesora-*) namijenjen je poslovima alokacije procesora, promjeni procesa kada se desi prekid ili istek tajmera. Uz ovo sloj 0 objezbjeđuje osnovne funkcije za multiprogramiranje (mogućnost da se pokrene više programa od jednom).
Sloj 1 obavlja upravljanje memorijom. On alocira (dodjeljuje) prostor procesima u glavnoj memoriji.
Sloj 2 upravlja komunikacijom izmedju svakog procesa i operatorske konzole.
Sloj 3 upravlja ulaznim izlaznim(I/O) uredjajima i privremeno memoriše (BUFFERING) informacioni tok ka i od I/O uređaja.
Sloj 4 je sloj na kome se nalaze korisnički programi a na sloju 5 obavlja se proces sistemskog operatora.

Moduli ovakvih operativnih sistema formiraju hijerarhiju.

Svaki od slojeva je predodređen da sadrži jedan od modula operativnog sistema. Raspodjelu modula po slojevima diktira pravilo koje nalaže da se **iz svakog sloja pozivaju samo operacije uvedene u nižim slojevima hijerarhije**. Primjena pomenutog pravila dovodi do smještanja modula za upravljanje procesima u sloj na vrhu hijerarhije. U prvi niži sloj dospeva modul za upravljanje fajlovima, dok je sledeći niži sloj namjenjen modulu za upravljanje radnom memorijom. Predposlednji sloj hijerarhije sadrži modul za upravljanje I/O kontrolerima, a u poslednjem sloju nalazi se modul za upravljanje procesorom.

Svrha predstavljanja operativnog sistema kao hijerarhije slojeva je motivisana željom da se zadatak operativnog sistema raščlani na više jednostavnijih međusobno nezavisnih zadataka i zatim svaki od njih objasni zasebno.

Primjeri OS sa slojevitom organizacijom : Multics , THE (6 layers), **MS-DOS** (4 layers)**.**

Razlika između slojevite i monolitne realizacije OS

Osnovna razlika je u tome, što se OS kod monolitne strukture sastoji od skupa procedura bez ikakvog grupisanja ili hijerarhije, a kod slojevite realizacije OS se dijeli na više slojeva od kojih se svaki oslanja na slojeve ispod, i gdje svaki sloj ima tačno određenu funkciju (upravlja tačno određenim resursima).

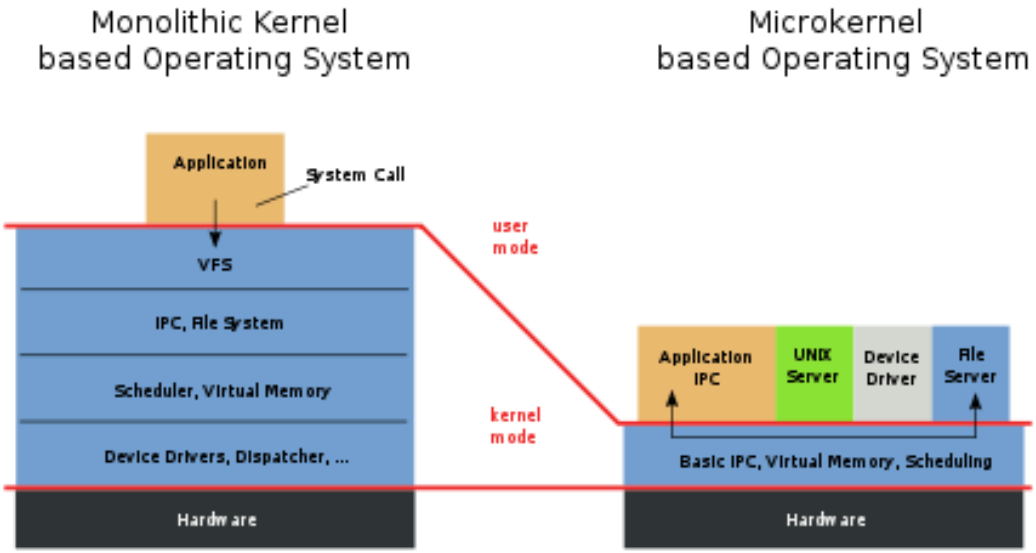
Monolitni operativni sistemi se sastoje od modula čija saradnja nije ograničena pravilima kao kod slojevitih operativnih sistema.
To znači da se iz svakog od modula monolitnih operativnih sistema mogu slobodno pozivati operacije svih ostalih modula.

Kod **slojevitih modularnih** sistema raspodjelu modula po slojevima diktira pravilo koje nalaže da se iz svakog sloja pozivaju samo operacije uvedene u nižim slojevima hijerarhije. Primena pomenutog pravila dovodi do smještanja modula za rukovanje procesima u sloj na vrhu hijerarhije. U prvi niži sloj dospeva modul za rukovanje datotekama, dok je sledeći niži sloj namenjen modulu za rukovanje radnom memorijom. Predposlednji sloj hijerarhije sadrži modul za rukovanje kontrolerima, a u poslednjem sloju nalazi se modul za rukovanje procesorom.

Većina modernih sistema je neka vrsta **hibrida** i najčešće koristi neke elemenata slojevite i monolitne strukture.

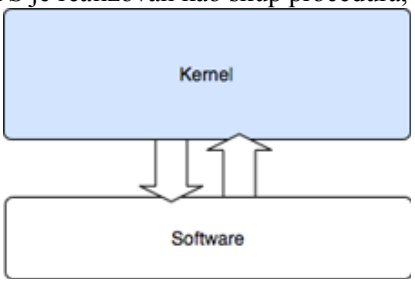
Tipovi jezgre

Što se samog dizajna i naravno same implementacije jezgre tiče, postoji nekoliko tipova.
Dva najinteresantnija i ujedno najviše suprotstavljena tipa su monolitna jezgra (monolithic kernel) i mikrojezgra (microkernel).



Monolitni OS –Monolitno jezgro- -Kernel-

Operativni sistemi koji koriste monolitnu kernel strukturu (npr. UNIX) realizovani su kao skup procedura (tj. modula), od kojih svaka može pozvati svaku ako je to potrebno. vidi *Jezgro operativnog sistema: KERNEL*, u Dodatku.
OS je realizovan kao skup procedura, od kojih svaka može pozvati svaku ako je to potrebno.



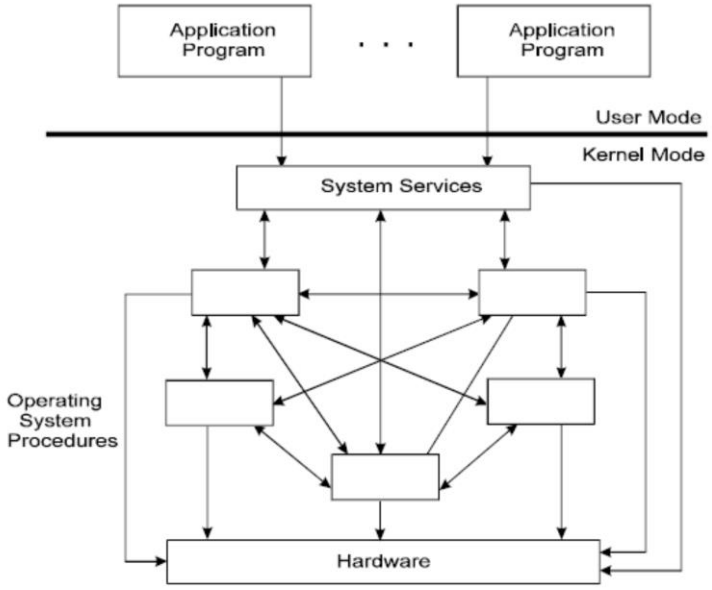
Kod monolitne jezgre se cijela jezgra izvršava u jezgrinom prostoru i to upravo u sistemskom (zaštićenom) načinu rada.
Za takvu jezgru je karakteristična vrlo snažna apstrakcija nad sklopovima (i hardverskim i softverskim) i interfejs prema aplikacijama sa vrlo bogatim skupom sistemskih poziva.
Uprkos činjenici da su razne funkcionalnosti najčešće implementirane u različitim dijelovima jezgre, stvarno odvajanje takvih dijelova je u praksi nemoguće zbog isprepletenosti izvornog koda i struktura podataka.

Visok nivo međusobne integracije svih dijelova utiče na dobre performanse zbog minimalnih nivoa apstrakcije između samih dijelova kao i visoke efikasnosti koja je posljedica takvog dizajna.
Glavna mana ovakvog pristupa je prvenstveno problem da greška u bilo kojem dijelu ovakve jezgre najčešće utiče na pad cijelog sistema, budući da sve komponente nužno vjeruju jedna drugoj.
Poznatiji primjeri monolitnih jezgri su: Linux (postojanje modula ne mijenja činjenicu da je riječ o monolitnom dizajnu jezgre), BSD jezgre, Solaris, DOS, većina Unix jezgri kao i Windows jezgra (hibridnog dizajna budući da je dio jezgrinih modula implementiran u vidu korisničkih aplikacija, no smatra se u osnovi monolitnom jezgrom).

Korisnički programi servise OS-a koriste na sledeći način: parametri sistemskog poziva se smještaju u određena mesta, kao što su registri procesora ili stek, pa sledi specijalna operacija – **poziv jezgra** OS-a (**kernel call**). Ova operacija

prebacuje procesor iz **korisničkog režima** rada u **sistemski režim** rada i kontrolu predaje OS-u. U korisničkom režimu rada nisu dozvoljene neke komande procesora, a u sistemskom režimu mogu se koristiti sve operacije poznate od strane procesora.
Poslije poziva jezgru, OS preuzima kontrolu, na osnovu parametara poziva određuje koju sistemsku proceduru treba pozvati i poziva tu proceduru i na kraju se kontrola vraća korisničkom programu.

Iz svakog od modula monolitnih operativnih sistema mogu slobodno pozivati operacije svih ostalih modula.

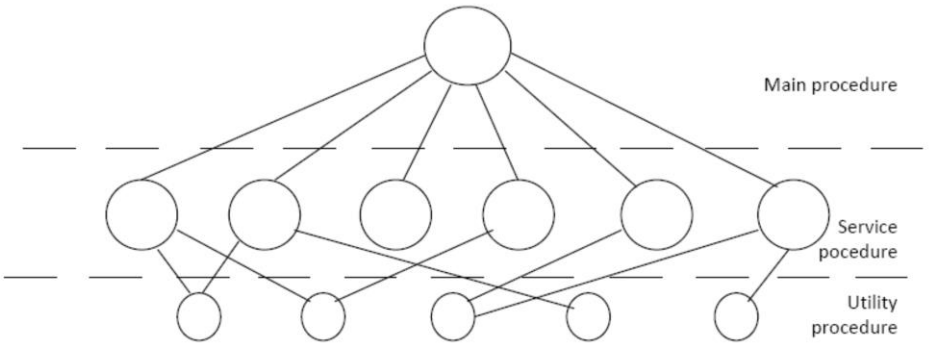


Ovo je dosad najviše korišćena organizacija i može se sa punim pravo nazvati "**Velika zbrka**" *("The Big Mess" or spaghetti code).*
Struktura je takva da uopšte nema strukture.

OS je napisan kao kolekcija procedure takvih da svaka može pozvati bilo koju drugu. Svaka procedura ima definisan interfejs u smislu parametara i rezultata. Međutim i u ovim sistemima ipak se može uočiti neka mala struktura.
Servisi (sistemski pozivi) objezbjeđeni od OS-a zahtijevaju parametre o utvrdjenim pravilima (npr. Da se prenose putem steka). Ovakve instrukcije prebacuju *mašinu* iz korisničkog načina rada i prebacuju upravljanje na OS. OS tada pribavlja parametre i utvrđuje koji sistemski poziv da se realizuje.

Ovakva organizacija odgovara označenoj strukturi za OS:
1. Postoji glavni program koji poziva servisnu proceduru
2. Postoji skup servisnih procedura koji izvršavaju sistemske pozive
3. Postoji skup pomoćnih procedura koje pomažu sevisne procedure

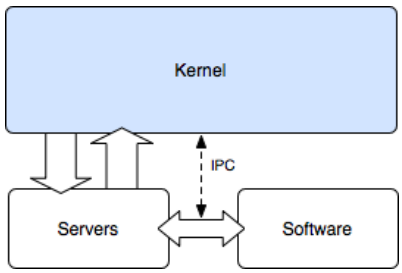
U ovom modelu za svaki sistemski poziv postoji jedna servisna procedura koja realizuje taj poziv.
Pomodne (utility) procedure se bave problemima koji su važni za nekoliko servisnih procedura (npr. Prikupljeni (FETCHING) podatak iz korisničkog programa). Podjela procedura se može izvršiti u tri sloja (layers) kao što je prikazana na slededoj slici:
Što se tiče **sakrivanja** informacija, to ovdje uopšte **ne postoji**, tj. svaka procedura je vidljiva u odnosu na bilo koju drugu proceduru, za razliku od strukture koja sadrži module ili pakete, u kojoj je većina informacija sakrivena unutar modula, tako da se samo u određenim tačkama mogu pozvati van modula.



Da bi se konstruisao stvaran objektni program operativnog sistema pomoću ovog pristupa, prvo se prevode sve pojedinačne procedure, ili datoteke koje sadrže te procedure, a zatim se sve zajedno povezuju u jednu objektnu datoteku koristeći sistemski povezivač. Pri tome, mogućnost skrivanja informacija ne postoji jer je svaka procedura vidljiva za svaku proceduru.

Mikro jezgro (mikrokernel)

Alternativa strukturi monolitnog operativnog sistema je arhitektura sa mikro jezgrom (microkernel)
Osnovna zamisao je napraviti minimalno i pouzdano jezgro visokih performansi, a sve ostale funkcije jezgra potisnuti u takozvani korisnički prostor (user space).



Mikrojezgra je primjer kernela minimalističkog dizajna: aplikacijama se pružaju tek najosnovnije nužne usluge (odnosno sistemski pozivi), a to su već spomenuto upravljanje **adresnim prostorom**, **vremenskim prekidima**, **nitima** (threads) pojedinog procesa te **komunikacijom među procesima**.

Sve ostale dodatne usluge poput grafičkih interfejsa, mrežnog steka i sl. su ugrađene (implementirane) u korisničkom prostoru u vidu posebnih servisa (servers, **pa se ovaj tip jezgra ponekad naziva i klijent server kernel**).

Zbog pojave takvih servisa van prostora kernela, kod mikrojezgri dolazi do izražaja potreba za efikasnom komunikacijom među procesima (inter-process communication) kako je prenošenje poruka jedan od glavnih aspekata rada.
Zbog toga što se daleko više poruka prenosi iz kernela u korisnički prostor i obrnuto, mikrojezgre obično imaju nešto lošije performanse od monolitnih jezgri.
Dodatni servisi su u praksi obične korisničke aplikacije koje se mogu pokretati i gasiti bez loših posljedica za ostatak sistema: jedina njihova privilegija je da mogu pisati i čitati neke dijelove memorije koji su nedostupni ostalim procesima.

Glavni problem takvog pristupa gdje se neki sistemski servisi mogu gasiti i paliti po potrebi je nestanak strogo definisanog stanja kao kod monolitnih jezgri. Zato je moguće da će u nekom trenutku neki servisi jednostavno doći u nedefinisano stanje. Za sprečavanje pada sistema uvodi se mogućnost pauze i čekanja do povratka usluge.

Primjeri primjene arhitekture mikrojezgri su prisutni u modernim operativnim sistemima: Minix, AmigaOS, Mach (GNU Hurd, XNU odnosno Mac OS X), QNX, L4 familija OS, Symbian OS, Singularity.

Korisnički moduli međusobno komuniciraju slanjem poruka (message passing).

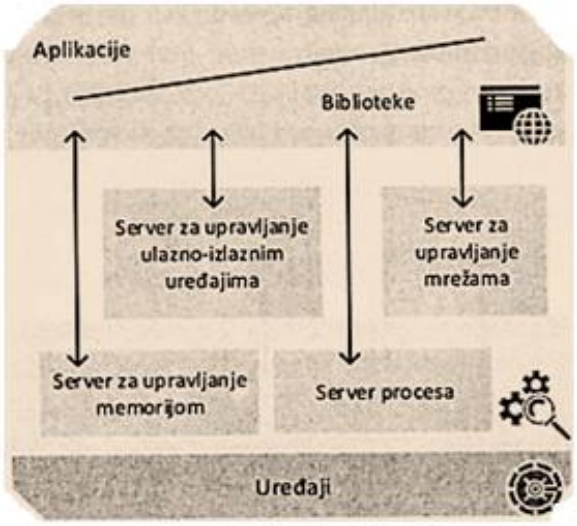
Kod arhitekture sa mikro jezgrom **samo najvažnije** funkcije OS-a se nalaze u jezgru.
Manje važni servisi i aplikacije su van jezgra i izvršavaju se u korisničkom režimu rada.
Komponente OS izvan mikro jezgra se implementiraju kao server procesi. Ove komponente komuniciraju međusobno tako što šalju poruke preko mikro jezgra.

Mikro jezgro vrši validaciju poruka, prenosi poruke između komponenata i daje komponentama pristup hardveru.
Komponente mogu biti drajveri uređaja, server datoteka, server procesa, programi za upravljanje virtuelnom memorijom...
Npr. ako aplikacija treba da otvori datoteku tada ona šalje poruku serveru datoteka preko mikro jezgra. Svaki od servera može da pošalje poruku drugim serverima i može da poziva primitivne funkcije unutar mikro jezgra.
Tako je ostvarena klijent-server arhitektura unutar jednog računara.

Osnovne prednosti OS sa mikro jezgrom su sljedeće:
- Dodavanje novog servisa ne zahtijeva modifikovanje jezgra OS,
- Sistem je bezbjedniji jer se više operacija izvršava u korisničkom režimu rada,
- Predstavlja podršku distribuiranim sistemima,
- Predstavlja podršku objektno-orijentisanim OS,
- Omogućava jednostavnije projektovanje jezgra i pouzdaniji OS.

Hibridni sistemi

Hibridna arhitektura sistema, odnosno hibridna jezgra (hybrid kernel) predstavljaju kompromis između monolitne i arhitekture koja se zasniva na mikrojezgru.



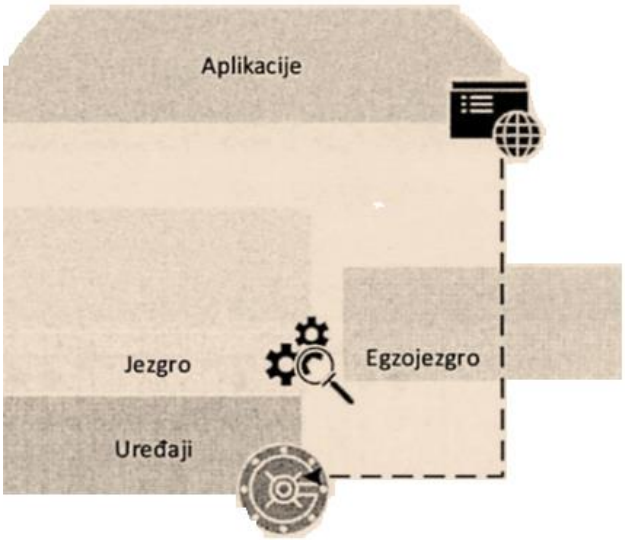
Neke veoma bitne, kao i funkcije koje se često izvršavaju, spuštaju se u jezgro kako bi se povećala brzina i efikasnost, ali se dobar dio funkcija zadržava u nivoima iznad jezgra.

Kod ovakvog pristupa najvažnije je precizno odrediti koje funkcije treba spustiti u jezgro, a koje ostaviti van, jer će u suprotnom hibridni pristup ličiti na sistem sa monolitnim (u slučaju da se previše funkcija implementira u jezgro) ili mikrojezgrom (ako dosta važnih funkcija ostane van jezgra).

Hibridna jezgra se koriste u većini komercijalnih operativnih sistema: Apple Mac OS X i kod novijih Microsoft Windows OS (od NT3.1 i XP-a).

Sistemi zasnovani na egzojezgru

Tradicionalni pristup pri dizajniranju operativnih sistema podrazumeva da hardverske komponente ne budu vidljive korisničkim aplikacijama, odnosno da programi komuniciraju sa hardverom isključivo preko jezgra operativnog sistema.
Na ovaj način olakšava se programiranje, ali se, sa druge strane, ograničavaju performanse. Suprotni pristup bi podrazumevao da se dozvoli da aplikacije direktno pristupaju hardveru, tj. da se jezgro potpuno zaobiđe. U tom slučaju bi funkcionisanje sistema zavisilo od aplikacija koje pišu programeri i aplikacije bi zavisile jedne od drugih.



Koncept egzo jezgra (exokernel) predstavija kompromis između ova dva suprotna pristupa (slika 1.14.). Ideja je da jezgro obezbedi osnovne resurse i da aplikacijama prepusti rad sa njima. Ovo se postiže premeštanjem apstrakcije iznad hardvera u posebne biblioteke koje obezbeđuju minimalne apstrakcije uređaja. Na taj način se programeri mogu osloniti na odgovarajuću biblioteku ali im se ostavlja i mogućnost da implementiraju svoje biblioteke. Drugim rečima, programeri imaju slobodu u izboru nivoa apstrakcije kada je pristup hardveru u pitanju.
Ovakav način rada može da doprinese znatnom ubrzanju i poboljšanju performansi. Sa druge strane, dodatna fleksibilnost za korisničke aplikacije može da dovede do smanjenja konzistentnosti i neurednosti koda.

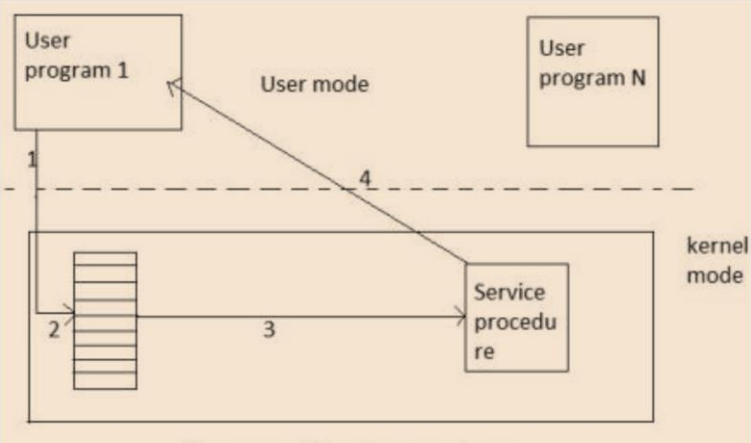
Pošto je veći deo funkcija izmešten iz egzojezgra, ono je obično manje od prethodno pomenutih tipova jezgara. Primeri sistema koji imaju arhitekturu zasnovanu na egzojezgru su: XOK, ExOSitd

Privilegovani režim rada i sistemski pozivi

Nema jasno i nedvosmisleno propisanih i definisanih pravila koja regulišu raspodjelu funkcija operativnog sistema po nivoima.
Odnos veličine operativnog sistema i operativne memorije, kod većine operativnih sistema je takav da ne može da stane u operativnu memoriju. Zato se u memoriji **uvijek** nalaze **samo najvažniji** delovi operativnog sistema, takozvani rezidentni delovi, koji aktiviraju i završavaju korisničke programe, vrše dodjelu memorije i datoteka i obavljaju operacije ulaza-izlaza.
Rezidentni dio operativnog sistema mora obavezno podržavati mehanizam prekida, jer je on osnova višeprogramskog rada i komuniciranja računara sa spoljnim svijetom. (*Rečeno je da je taj dio operativnog sistema koji mora uvek biti prisutan u operativnoj memoriji obično se naziva jezgro*). Funkcije koje koriste svi nivoi moraju se smestiti u jezgro operativnog sistema. Ostali dijelovi se ubacuju u memoriju kada su potrebni i izbacuju kada više nisu potrebni.

OS radi u sistemskom - privilegovanom režimu rada (**sve naredbe izvršive**), a korisnički procesi se izvršavaju u korisničkom (neprivilegovanom) režimu rada. Tu se neke naredbe ne izvršavaju (npr. za promjenu režima rada, za rad sa privilegovanim registrima, za I/O...) jer bi inače OS mogao ostati bez "kontrole".

Cilj uvođenja dva režima rada računara je obezbeđivanje zastite racunarskog sistema od namjernih ili nenamjernih upada u sistem. Ovaj režim rada mora biti hardverski podržan, mora spostojati indikatorski registar, koji će sadržati flag u kome se nalazi informacija o režimu rada (trenutnom, u kome se nalazi računar).
P=0, korisnički režim rada; P=1, privilegovani režim rada.



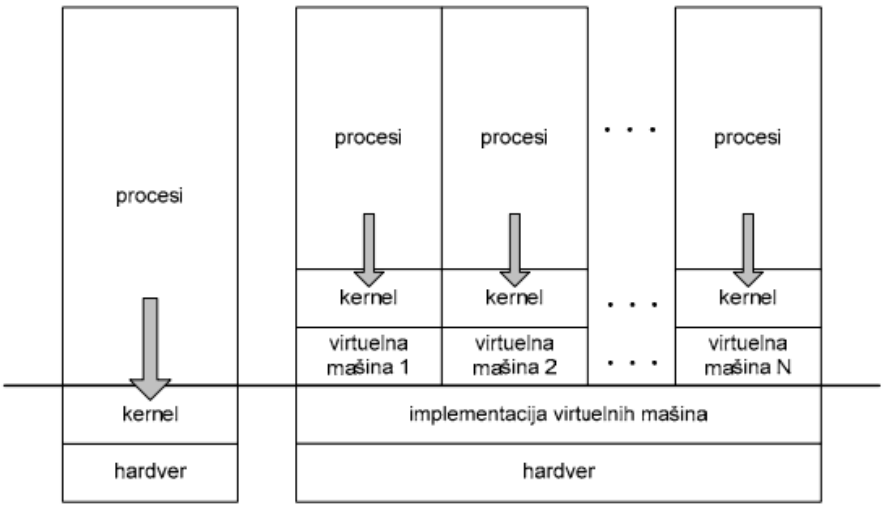
Način realizacije prelaska iz korisničkog u privilegovani mod rada kod monolitnih OS

Ako korisničkom programu treba usluga programa koji radi u provilegovanom režimu rada onda korisnički program mora da koristi takozvane sistemske pozive.
Sistemski poziv je specijalna instrukcija kojom se pokreće neka rutina iz privilegovanog režima rada, to je u stvari softverski prekid.
Samo prekidom se može preći iz korisničkog u privilegovani režim rada. Kad se desi prekid, procesor automatski prelazi u privilegovani režim rada i onda operativni sistem izvršava operacije koje je zahtjevao korisnički program.

Virtuelne mašine

Operativni sistem može da se definiše kao virtuelna mašina (Što smo i mi uradili kod opisa funkcija operativnog sistema)

IBM je razvio posebnu softversku strukturu koja ima velike prednosti za realizaciju novih operativnih sistema i koja se takođe naziva virtulena mašina.



Koncept virtulene mašine

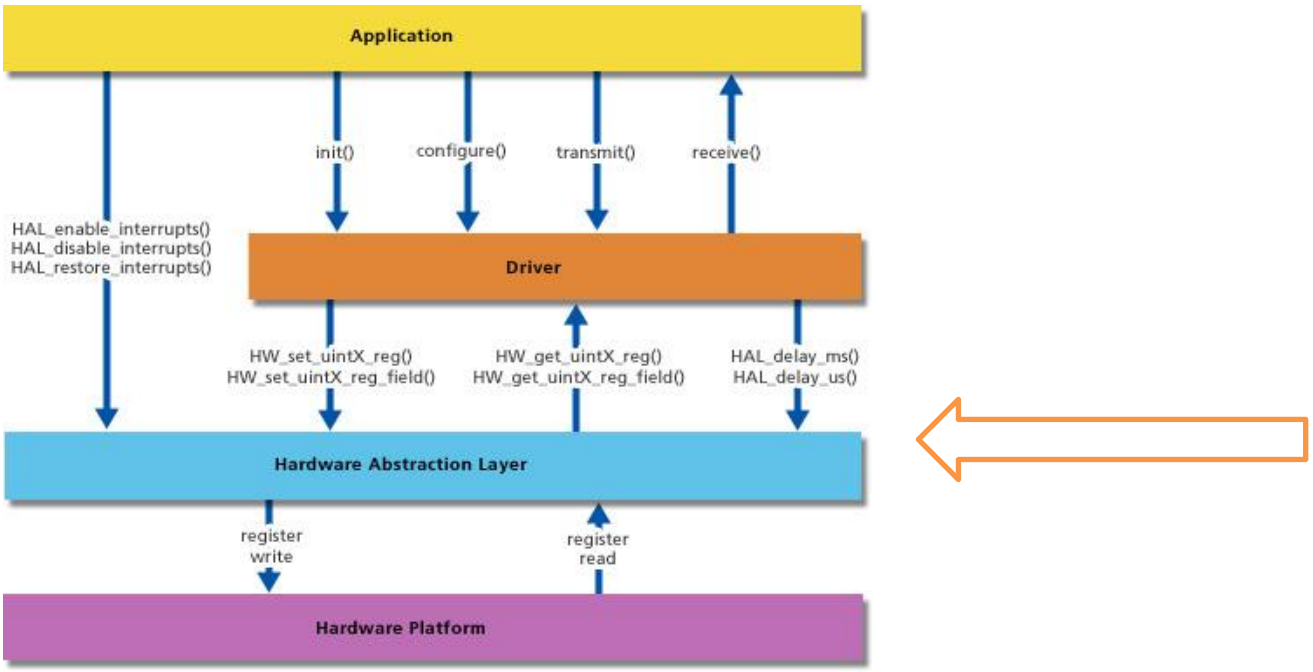
Strukturu virtuelnih mašina IBM definiše na sljedeći način: na najnižem nivou se nalazi hardver, a iznad hardvera monitor virtuelnih mašina (virtual machine monitor), odnosno poseban sistem koji obezbjeđuje niz virtuelnih mašina (tačnih kopija hardvera). Na te virtuelne mašine mogu instalirati različiti operativni sistemi.

Odgovarajući operativni sistemi primaju sistemske pozive korisničkih programa, a hardverske operacije koje šalju ti operativni sistemi prema svojim virtuelnim mašinama prihvata monitor virtuelnih mašina i realizuje ih u skladu sa hardverom ispod sebe. Virtuelna mašina je bazirana na slojevitoj organizaciji i tretira realni hardver i realni kernel kao da su hardver za operativni sistem koji predstavlja. **Virtulena mašina obezbjeđuje identičan interfejs kao da je realni hardver ispod virtulene mašine, a ne čitav niz slojeva softvera.**

Operativni sistem kreira **iluziju** o višestrukim procesima koji se izvršavaju na svom virtuelnom procesoru i svojoj virtuelnoj memoriji. Virtulene mašine obezbjeđuju kompletnu zaštitu sistemskih resursa, pošto su sve virtulene mašine međusobno izolovane, i ne mogu direktno dijeliti resurse.

Sloj apstrakcije hardvera HAL –virtuelni uređaji i particionisanje

Razvojem hardvera i softvera, a u velikoj mjeri i analizom na koji način se oni koriste, došlo je do izdvajanja posebne komponente operativnog sistema, koja je dobila naziv **sloj apstrakcije hardvera** (Hardware Abstraction Layer – **HAL**).
Uloga HAL-a jeste da obezbjedi interfejs između kernela i samog hardvera. HAL se bavi eventualnim konverzijama formata podataka (mada je potreba za tim zanemarljiva), a najbitniji zadatak jeste obezbjeđivanje tzv. virtuelnog uređaja.



Virtuelni uređaj predstavlja standardizaciju načina na koji se pristupa hardverskim uređajima.
Zahvaljujući ovoj komponenti operativnom sistemu se omogućava da koristi iste mehanizme za posmatranje stanja hardvera i razmjenu podataka, bez obzira na hardver koji se ispod njega nalazi.
Na ovaj način se adaptacija operativnog sistema za drugu arhitekturu računara svodi na pisanje novog HAL-a.

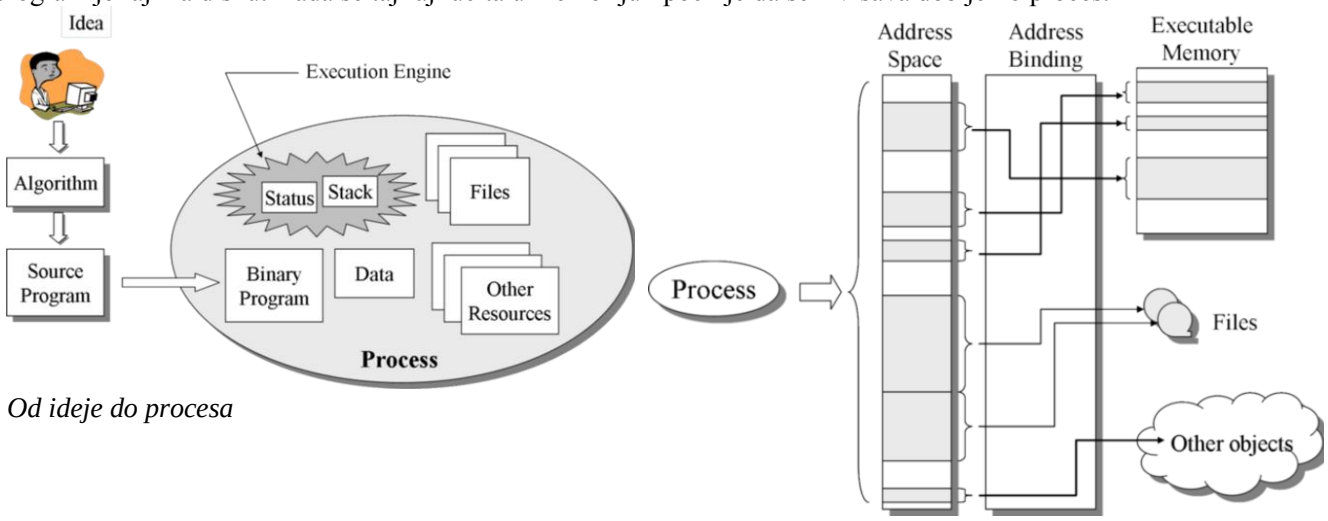
Virteulne mašine su prefektan razvojni sistem za realizaciju novih operativnih sistema, jer se razvoj realizuje na virtulenoj mašini, a ne na realnoj, može lako da se kontroliše, mogu se ispravljati greške i nikako ne može doći do narušavanja normalnih sistemskih operacija, jer će sistem koji nudi virtulene mašine da radi bez obzira na stanje pojedinih virtulenih mašina.

Podjela resursa sistema na virtuelna okruženja je poznat pod nazivom particionisanje.
Osnovni tipovi particionisanja su:
a) hardversko (fizičko) particionisanje – predstavlja fizičko razdvajanje resursa sistema korištenjem odvojenih hardverskih uređaja. Aplikacije koje su projektovane tako da rade samostalno u okviru operativnog sistema (ili za koje se ne preporučuje paralelno korištenje fizičkih resursasa drugim aplikacijama) imaju najviše koristi od hardverskog particionisanja.
b) logičko particionisanje – predstavlja razdvajanje radnih okruženja unutar računarskog sistema, pri čemu se radi o softverskoj implementaciji particija. Na sličan način kao kod hardverskog particionisanja, kreirane particije predstavljaju samostalna okruženja u okviru kojih je moguće izvršavanje odvojenih operativnih sistema, ali se upravljanje particijama vrši putem posebnog softverskog međusloja koji se naziva monitor virtuelne mašine (hipervizor).

Upravljanje procesima

Program i proces

Procesi predstavljaju jedan od najvažnijih koncepata operativnih sistema. Program je niz instrukcija koji ostvaruje neki algoritam. Proces je program u statusu izvršavanja, zajedno sa svim resursima koji su potrebni za rad programa. Znači: program je fajl na disku. Kada se taj fajl učita u memoriju i počinje da se izvršava dobijemo proces.



Od ideje do procesa

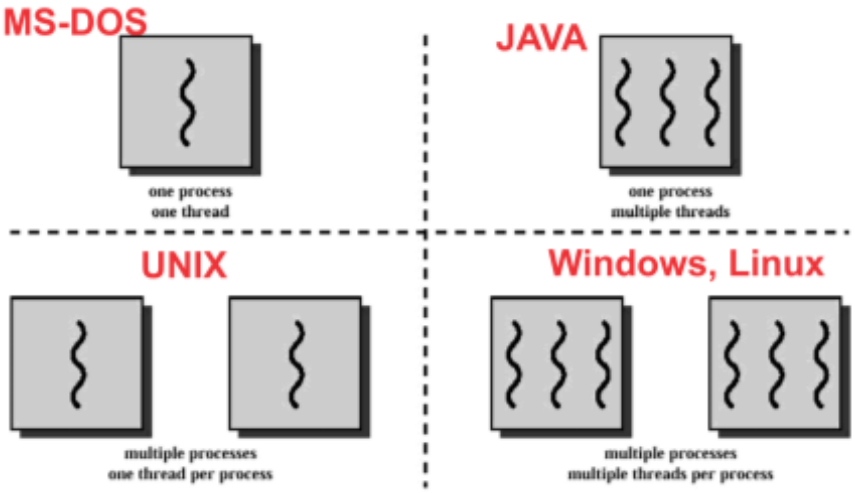
Svaki proces gleda svoju memoriju kao granični skup logičkih memorijskih adresa. Operativni sistem upravlja prevođenjem (određivanjem) logičkih adresa u odgovarajuće fizičke adrese koje su rezervisane za proces.

Operativni sistemi izvršavaju različite programe: u sistemima sa paketnom obradom to su poslovi (jobs), a u sistemima sa razdijeljenim vremenom – korisnički programi ili zadaci (tasks).
Program je fajl koji sadrži nizove naredbi mašinskog jezika (koji se može izvršavati na posmatranom procesoru) i eventualno podataka. Kao takav on je opisan samo svojim sadržajem i ima statičku strukturu.
Proces (ili task) je dinamičke prirode. To je objekat multiprogramiranja i predstavlja program kome je pridodata posebna struktura podataka koje se naziva kontrolni blok procesa PCB (Process Control Block) ili PD - deskriptor procesa.
Posao (job) je objekat paketne obrade i obuhvata niz naredbi kontrolno-upravljačkog jezika, programa i podataka za te programe. Njegova obrada se odvija sekvencijalno, tako što se izvršava niz naredbi koji pokreće odgovarajuće programe.

Ovo su samo neke od mogućih definicija za program, proces i posao.
Jednostavno rečeno, **program je pasivan objekat, to jest datoteka na disku.**
Kada se program učita u memoriju, postaje proces, tj. aktivan objekat koji ima svoje resurse poput procesorskih registara i memorije.

Procesi i niti

Najjednostavnija definicija procesa (**task**) je da je to program koji se izvršava.
Nit (**thread**)se može shvatiti kao proces unutar procesa.
Nit i proces predstavljaju moguće metode realizacije paralelnog izvršavanja programa.
Grubo govoreći nit možemo shvatiti kao podprogram, a task kao program. Program koji ima mogućnost rada sa podprogramima ima veću modularnost, brži je i manji, ali i nestabilniji –teži za kontrolu, jer pojedini podprogrami dijele zajedničke resurse, što često dovodi do kolizije, pada programa i sistema.



Ilustracija za pokazuje razne modele multiprocesiranja kod različitih OS korišćenjem niti i procesa (threads and processes)

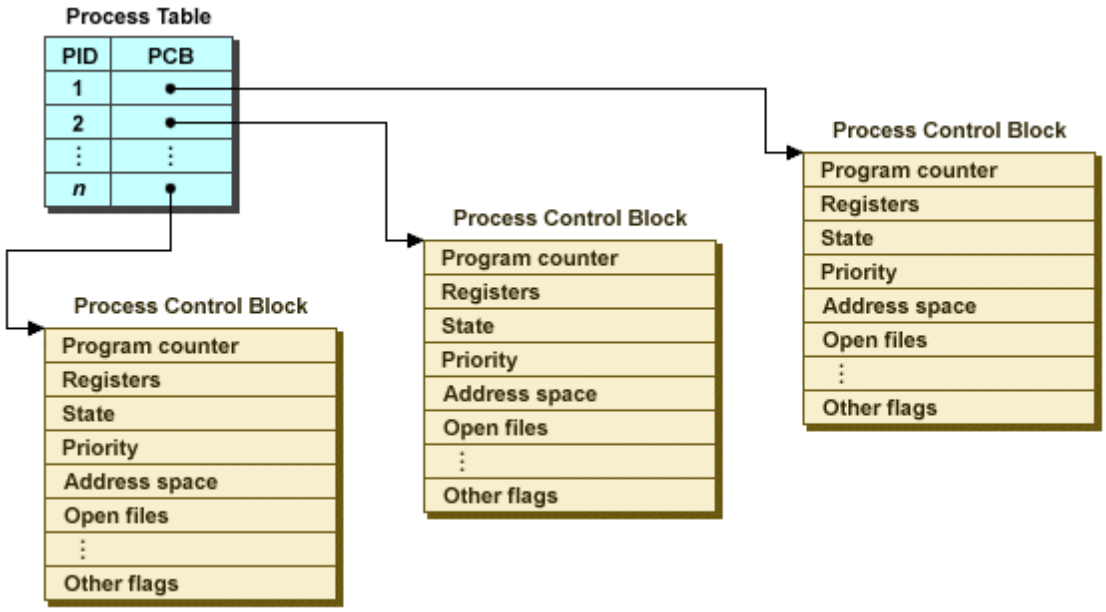
Kako jedan proces može da se izvršava i u korisničkom režimu i u režimu jezgra (*kernel*), potrebno je obezbediti istu podršku i na korisničkom nivou za korisničke niti (*user threads*), i na nivou jezgra za niti jezgra (*kernel threads*).
Podrška za **korisničke niti** realizuje se preko biblioteke za rad sa korisničkim nitima (***user thread library***). Ova biblioteka obezbeđuje podršku za stvaranje niti, raspored izvršavanja niti i upravljanje nitima, ali bez uticaja jezgra(*kernel*).
Najznačajnije vrste niti su : POSIX Pthreads, Mach C-threads i Solaris UI-threads.
Niti jezgra (*kernel*) direktno podržava operativni sistem. Konkretno, jezgro (***kernel*) izvršava operacije** stvaranja **niti**, raspoređivanja niti i upravljanja nitima u prostoru jezgra.
Primjeri : Windows 95/98/NT/2000, Solaris, Tru64 UNIX , BeOS, Linux.

Predstavljanje procesa: Kontrolni blok procesa –PCB-

Proces je niz aktivnosti koje nastaju kao posljedica izvršenja programa. Dio procesa su i podaci koji tu aktivnost opisuju, odnosno podaci koji su neophodni za upravljanje procesom. Ove podatke generiše i koristi operativni sistem, odnosno dispečer, a opisuje se raznim imenima: kontrolni blok procesa (PCB - Process Control Block), vektor stanja ili process descriptor.
Svaki proces ima tri fundamentalna memorijska dijela (sekcije):
oblast koda (programska sekcija –read only)u kojoj se čuvaju instrukcije programa,
oblast podataka (sekcija podataka data section) u kojoj se čuvaju globalne promjenljive i
oblast steka (stek section) u kojoj se čuvaju privremeni podaci poput parametara potprograma, adrese povratka i lokalne promjenljive.
Osim memorije proces koristi i procesorske registre poput brojača naredbi, ulazno izlazne resurse poput datoteka i raznih ulazno izlaznih uređaja. dio procesa su i podaci koji ga opisuju, to jest podaci koji su neophodni za upravljanje procesom. Ove podatke generiše i koristi operativni sistem.

Kontrolni blok procesa je dio radne memorije, to jest memorijska struktura sa osnovnim informacijama o procesu, koje operativni sistem koristi za upravljanje tim procesom.
U informacije iz kontrolnog bloka spadaju:

- ime ili jedinstveni identifikator procesa (PIO);
- kontekst (okruženje) procesa;
- prioritet procesa;
- informacije o memoriji procesa;
- lista otvorenih datoteka;
- status zauzetih ulazno-izlaznih resursa;
- trenutno stanje procesa.



Kontrolni blok procesa PCB: struktura i elementi

"Laki" i "teški" procesi

Postoje dva glavna tipa procesa u svijetu multitaskinga - threads ili "laki" procesi i "teški" procesi. Ova dva tipa procesa razlikuju se uglavnom po načinu kako upravljaju memorijom.
Postoje operativni sistemi, koji podržavaju:
1. samo teške procese (UNIX)
2. samo lake procese (Oberon)
3. podržavaju i teške i lake procese (Windows)

Podjela procesa na teške i lake procese vrši se na osnovu toga, kako koriste memoriju (koje dijelove memorije):
- svaki teški proces (**task)** ima sopstveni **memorijski prostor za kod, globalne promjenljive, stek i heap koju ne dijeli ni sa kim, pristup tim djelovima ima samo dati proces**.
- laki procesi (niti, **threads**) **mogu dijeliti memorijski prostor za kod, globalne promjenljive i heap**. **Stek se ne može dijeliti** jer ne možemo unapred znati koji proces koristi stek: proces A stavi nešto na stek i nastavlja rad, dolazi proces B i on isto stavi nešto na stek, A je završio svoj rad i sad su mu potrebni podaci sa steka, a tamo su podaci procesa B… Znači laki procesi imaju sopstveni stek a mogu dijeliti kod, globalne promjenljive i heap.

Ako neka aplikacija treba da izvrši više funkcija simultano, dijeli se u više procesa (npr. korišćenjem UNIX *fork* sistemskog poziva). Svaki proces obezbeđuje resurse potrebne za izvršavanje programa - ima virtualni adresni prostor, izvršni kôd, podatke, module za rukovanje objektima - hendlere, environment promjenljive, bazni prioritet, minimalnu i maksimalnu veličinu radnog prostora.

Nasuprot tome, u operativnim sistemima Windows osnovna jedinica izvršavanja je **thread - nit** ili tok izvršavanja. Task ili proces sastoji se od jednog ili više thread-ova (jedan je main). Svaki od njih ima različit stek i status mašine, ali svi thread-ovi istog procesa dijele kôd i podatke u memoriji, pa prelazak sa jednog na drugi thread oduzima manje sistemskog vremena nego prelazak sa jednog "teškog" procesa na drugi.

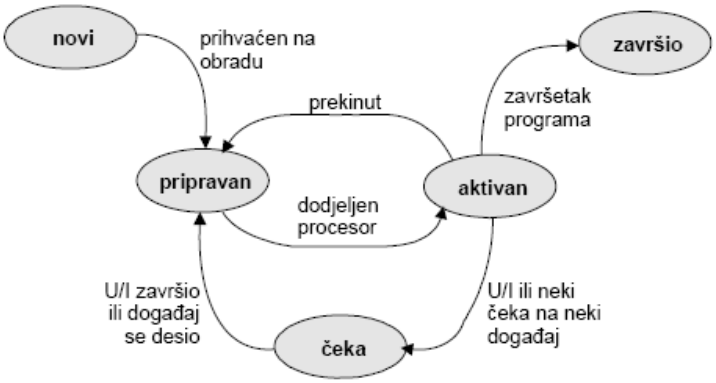
Kontekst "lakog" procesa uključuje skup procesorskih registara, kernel stek, thread environment blok i user stek u adresnom prostoru matičnog procesa.

Programski jezici, koji omogućavaju kreiranje lakih procesa: Java, Modula-2, Concurrent Pascal, Ada, itd.

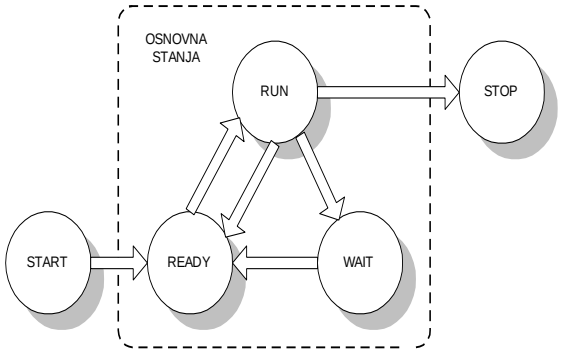
Osnovna stanja procesa

Proces čini niz koraka koji slijede jedan za drugim. Između dva koraka proces može biti prekinut, a njegovo izvršavanje se može nastaviti u nekom drugom trenutku vremena, na istom ili nekom drugom procesoru. Svi procesi koji uđu u računarski sistem prolaze kroz niz stanja tokom svog boravka u računaru. Jednostavno rečeno, stanje procesa (process state) opisuje ono što se u datom trenutku događa sa procesom. Provođenje procesa iz jednog stanja u drugo (state transition) obavlja operativni sistem.
U zavisnosti od koncepta mašine (ili virtuelne mašine) na kojoj je realizovan opratívni sistem definiše se različit broj stanja (od 3 do 7)

Srandardno (i minimalno) procesi se mogu nalaziti u jednom od tri stanja:



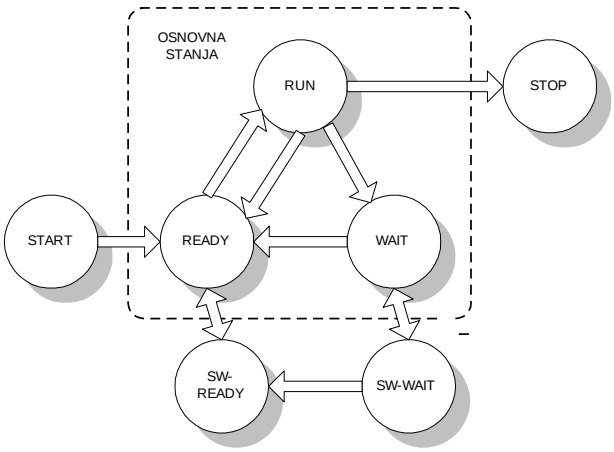
• **Novi** (new) – stanje u kojem se proces nalazi nakon što je stvoren, kreira se PCB i proces ide u sljedeće stanje pripravan.
• **Spreman** -pripravan (ready) – označava proces koji se nalazi u redu za izvršavanje. Nakon nekog vremena, kada dobije pravo korištenja procesora, proces prelazi u stanje aktivan.
• **Aktivan** – (run) označava proces koji se trenutno izvršava. Ukoliko OS prekine aktivan proces tada se taj proces prebacuje u stanje pripravan, a ukoliko proces mora obaviti neku I/O operaciju, tada ide u stanje čeka. (U oba slučaja se zapamti trenutno stanje popunjavanjem PCB-a).



Osnovna stanja procesa

U stanju RUN se nalazi uvijek samo jedan proces (onaj koji se trenutno izvršava - kome je dispečer dodijelio CPU).
U stanjima READY i WAIT se može nalaziti više procesa.
Iz stanja RUN u READY proces prelazi po isteku kvanta vremena (tj. po prekidu od časovnika).
Iz stanja RUN u WAIT proces prelazi ako zatraži neki resurs koji je u tom trenutku zauzet (koji ne može da dobije) ili ako mora da sačeka da mu neki drugi proces završi posao.

U složenijim OS se pojavljuju još dva stanja. Suspendovanje procesa podrazumeva prebacivanje iz operativne memorije na eksternu. U OM ostaje samo PCB procesa.



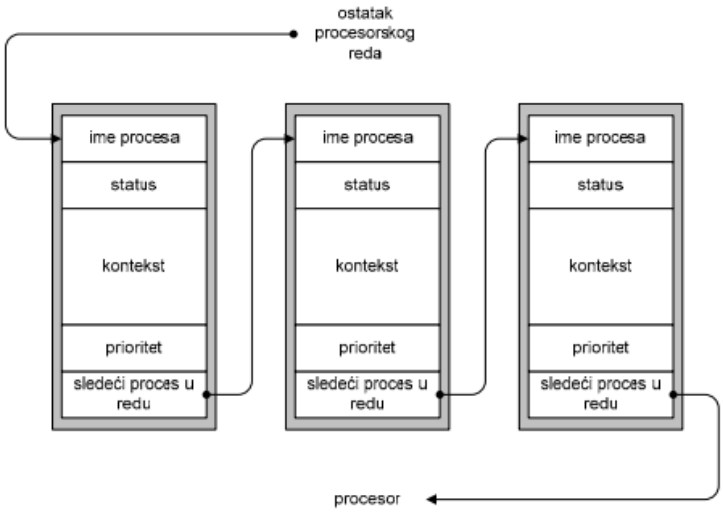
Slika 2 - Stanja procesa

Na taj način dobijamo još dva stanja:

- proces je **suspendovan i spreman** (ako je došlo do suspendovanja u stanju spreman)
- proces je **suspendovan i blokiran** (ako je došlo do suspendovanja u stanju blokiran).

Proces koji je suspendovan, prestaje da se takmiči za resurse, oslobađaju se resursi koje je zaouzeo, ali ostaje i dalje proces.

Proces koji je u stanju suspendovan i blokiran prelazi u stanje suspendovan i spreman, ako postaje spreman, tj. ako može da nastavi sa radom (npr. proces pošalje zahtev skeneru da skenira sliku, čeka da skener završi sa radom, pa se blokira, u međuvremenu se suspendira, pa postaje suspendovan i blokiran, kada skener završi skeniranje, proces prelazi iz stanja suspendovan i blokiran u stanje suspendovan i spreman.)
Iz stanja suspendovan i blokiran u stanje blokiran i iz stanja suspendovan i spreman u stanje spreman procesi mogu preći samo explicitno, tj. zahtjevom korisnika.



U stanju RUN se u jednom trenutku može naći najviše onoliko procesa koliko procesora ima na sistemu. Procesi koji se nalaze u stanju RUN su tekući procesi. U stanju RUN se proces ne može zadržati do svog završetka, zato što se u opštem slučaju posmatra višeprocesno okruženje u kome se procesor dodeljuje na određeno vrijeme, po nekom algoritmu. Na taj način se sprečava da jedan proces crpi resurse računara, dok drugi procesi čekaju u redu. Dodatno, ukoliko se operativni sistem karakteriše osobina pretpražnjenja, procesor se može oduzeti i u slučaju da naiđe proces višeg prioriteta, čime se obezbeđuje blagovremeno izvršavanje kritičnih procesa.

Svi procesi koji se nalaze u stanju READY mogu se povezati u red koristeći jednostavnu strukturu liste. Ova linearna lista, prikazana. čini procesorski red u koji je uključen i tekući proces. *Dispečer* (tj. OS) koristi procesorski red prilikom upravljanja procesima.

Operacije nad procesima

Operativni sistem može da izvrši sledeće operacije nad procesima:
- kreiranje procesa
- uništavanje procesa
- izrada veza proces – proces roditelj
- promjena stanja procesa
- promjena prioriteta procesa.

Kreiranje procesa

Za svaki program odnosno posao OS kreira bar jedan proces.
U okviru tog procesa može se realizovati niz radnji, npr. editovanje, prevođenje, povezivanje i izvršenje (za vrijem e kartica se radilo slično tome).
Povoljnije rješenje je da se navedene faze kreiraju kao posebni procesi, odnosno editovanje, prevođenje, eventualno povezivanje i obavezno izvršenje. Svi prethodno pomenuti procesi su kreirani od strane OS. Korisnik ne mora uopšte da zna da je za njegov program formiran bilo koji proces.
Međutim, za složenije programe se pokazuje da je izuzetno korisno da se razbiju na dijelove i da se za te dijelove formiraju procesi.

Programski jezici imaju sredstva za kreiranje procesa koja stoje na raspolaganju programeru (npr. funkcije *CreateProcess*, *CreateThread* u C++)

Proces u toku svog izvršavanja može, odgovaraqjućim sistemskim pozivom (npr. *fork* u UNIX-u), da kreira nove procese. Proces koji proizvodi nove procese naziva se **roditeljski proces** (parent process), dok se tako dobijeni proces naziva **dijete proces** (child process) i može i sam da kreira svoje naslednike. Tako nastaje stablo procesa. Odnosi proces roditelj – proces dijete mogu se opisati na osnovu načina dijeljenja resursa i načina izvršavanja.
Postoje sledeći načini dijeljenja resursa:
- Roditelji i djeca procesi dijele sve resurse
- Djeca procesi dijele podskup roditeljskih resursa
- Roditelji i djeca ne dijele resurse.

Kao što smo ranije rekli, svaki proces za svoj rad koristi različite resurse – procesorske registre, memorijski prostor za kod, podatke i stek, datoteke i ulazno/izlazne uređaje. Specijalno je osejtljiv memorijski adresni prostor koji se između procesa roditelja i djece može raspodjeljivati na sljedeće načine:
- Dijete duplikat roditelja
- Dijete ima program učitan u sebe

Prema načinu izvršavanja, odnos između procesa roditelja i djece mogu biti sledeći:
- Roditelji i djeca nezavisno i konkurentno rade
- Roditelji se blokiraju i čekaju da djeca završe sa radom.

Završetak procesa

Proces završava svoju aktivnost kada izvrši poslednju instrukciju svog koda i traži od OS da ga završi tako što mu upućuje sistemski poziv ***exit.*** Pri tome, proces dete može da vrati izlazne podatke roditeljskom procesu preko sistmskog poziva ***wait*** Operativni sistem zatim dealocira resurse pridružene procesu i tako proces na tzv. normalan način završava sa radom.
Proces se može završiti i nasilno. Procesi roditelji mogu da prekinu izvršavanje procesa djece koju su kreirali korišćenjem sistemskog poziva ***abort*** iz jednog od sledećih razloga:
- Djeca prevazilaze dodijeljene resurse
- Task pridružen dijetetu više nije potreban
- Roditelji završavaju rad

Operativni sistemi ne dozvoljava dijetetu da nastavi sa radom, ako je roditelj završio i nasilno uništavaju sve procese potomke što se naziva kaskadno završavanje (**cascade termination**).

Odnosi među procesima

Procesi mogu biti međusobno nezavisni ili zavisni.
Ako se procesi izvršavaju uporedo i nezavisno onda nema interakcije niti razmjene informacija između njih, nasuprot kooperativnim procesima koji moraju da razmenjuju informacije ili da se međusobno sinhronizuju - dijele stanje (podatke).
U multiprocesnom OS izolovanih procesa praktično nema – svi oni dijele resurse sistema (memoriju, fajlove, I/O uređaje)
Kod nezavisnih procesa rezultat izvršavanja procesa ne zavisi od redosljeda izvršavanja i preplitanja sa drugim nezavisnim procesima – ne zavisi od raspoređivanja.
Kod procesa koji dijele podatke rezultat izvršavanja zavisi od redosleda izvršavanja i preplitanja – dakle, zavisi od raspoređivanja.

Odnose među procesima koje ćemo analizirati možemo svrstati u tri osnovne grupe:

- međusobno isključenje procesa (mutual exclusion)
- sinhronizacija procesa (synchronization)
- uzajamno blokiranje - zastoj (deadlock)

Međusobno isključenje procesa

Međusobno isključenje procesa je takav odnos među procesima koji ne dopušta istovremeno izvođenje dva procesa u pojedinim njihovim djelovima (kritičnim sekcijama), dok se u ostalim djelovima procesi mogu bez problema uporedo izvršavati.
U ovakav odnos dolaze nezavisni procesi koji treba da koriste neki nedjeljiv resurs, npr. datoteku ili područje u memoriji u koje oba mogu da upisuju podatke.

Na primjer, pretpostavimo da procesi p1 i p2 rade u nekom sistemu za rezervaciju avionskih karata. Pri tome bi rezervaciji karate odgovarao upis u neko memorijsko područje R. Onaj proces koji prvi dobije resurs R mora izvršiti rezervaciju, odnosno upisati rezervaciju u području R, ili ako je neki proces već ranije izvršio rezervaciju i upisao svoje podatke u područje R, proces mora odustati od rezervacije. Zbog toga, svaki proces kada pristupi u područje R mora pre rezvisanja provjeriti da li je područje slobodno i ako jeste, upisati svoje podatke. Problem može nastati ako proveru da li je područje R slobodno proces izvrši nakon što je istu proveru izvršio, a pre no što je izvršio rezervaciju neki drugi proces. Da bi se ovo sprečilo, od trenutka kada neki proces P krene u proveru, mora se zabraniti pristup području R drugim procesima sve dok proces P ne završi sa rezervacijom.
Ovaj dio koda u kome neki proces koristi nedjeljivi resurs naziva se **kritična sekcija.** OS mora da posjeduje mehanizam koji će vremenski uskladiti izvođenje procesa i pri svakom korišćenju nedjeljivog resursa provoditi međusobno isključivanje procesa. Ukoliko procesi istovremeno ulaze u svje kritične sekcije treba takođe provesti međusobno isključenje, dakle propustiti samo jedan proces.

Sinhronizacija procesa

Uopšteno gledano, procesi unutar računarskog sistema teku asinhrono. Međutim, u pojedinim slučajevima mogu postojati djelovi procesa, pa i čitavi procesi čiji rad neophodno uskladiti. To usklađivanje tokova dvaju ili više procesa međusobno nazivamo sinhronizacijom procesa. Sinhronizacija procesa sastoji se u tome da se djelovi procesa koji treba da budu sinhronizovani odvijaju prema određenim pravilima, odnosno prema određenom redosledu. Taj redosled može biti od slučaja do slučaja različit, pa imamo različite vidove sinhronizacije procesa.
Sinhronizacija nije jed noznačan odnos kao međusobno isključenje.

Uzajmno blokiranje procesa – zastoj

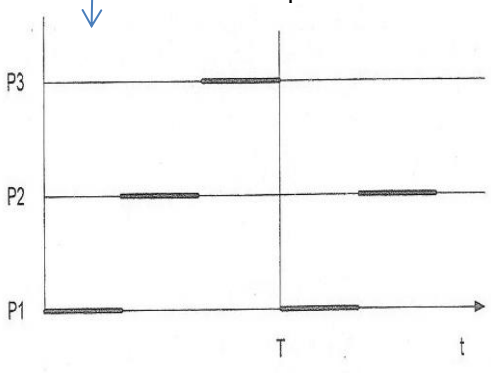
Treća vrsta odnosa je tzv. deadlock - zastoj. Ovakav odnos je nepoželjan u sistemu i treba ga, ako je moguće sprečiti ili, ako se dogodi, intervencijom operativnog sistema razrešiti. Do međusobnog blokiranja dolazi zbog konkurisanja za resure računarskog sistema ili zbog čekanja na događaj koji ne može nastupiti.

Raspodjela poslova

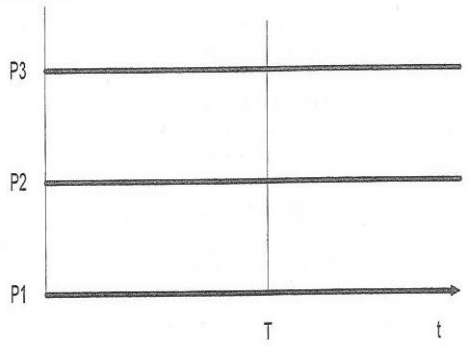
Mehanizam raspodjele poslova OS omogućava predaju upravljanja između taskova.
Multitasking se izvršava između dva ili više procesa.
Multiprogramiranjem se se postiže bolje iskorišćenje procesora - ukoliko tekući proces čeka na resurs, procesor može izvršavati neki drugi proces koji se nalazi u stanju čekanja na procesor. Podjela vremena (timesharing) je poseban oblik multiprogramiranja koji omogućava da svaki korsinik radi interaktivno na računaru preko posebnog terminala kome pripada dodjeljeno procesorsko vrijeme. Poslije isteka vremenskog kvantuma, odnosno dodjeljene količine procesorskog vremena, porcesor se dodeljuje drugom terminalu.
Na jednoprocesorskim računarskim sistemima se u jednom vremenskom trenutku može izvršavati samo jedan proces. Na višeprocesorskim sistemima se u jednom trenutku može izvršavati onoliko procesa koliko na sistemu ima procesora.
Bez obzira na vrstu računarskog sistema (jednoprocesorski ili višeprocesorski sistem), ukoliko se na sistemu izvršava višeprocesni operativni sistem (kao što su Windows 2003 Server ili UNIX), veći broj procesa istovremeno čeka na procesor.
Operativni sistem treba da obezbjedi odgovarajući mehanizam za dodjelu procesora različitim procesima. Dijelovi tog mehanizma su redovi čekanja na procesor i planeri poslova.

Konkurentni procesi

Nit (*thread*) je entitet koji se izvršava koristeći program i druge resurse od pridruženog procesa. Svaka nit je pridružena nekom procesu. Jednom procesu se može pridružiti više niti. Rad sa više niti (*multithreading*) se odnosi na mogućnost operativnog sistema da podrži izvršavanje više niti u okviru jednog procesa. Tradicionalan pristup je izvršavanje jedne niti po procesu.
Primjeri operativnih sistema sa takvim pristupom su MS-DOS i više verzija UNIX-a. Većina savremenih operativnih sistema podržava niti.
Na primjer, sve novije verzije operativnog sistema Windows i Solaris podržavaju rad sa nitima. Takođe, savremeni programski jezici kao što su Ada i Java podržavaju niti.
Paralelno (uporedno, istovremeno, konkurentno) izvršavanje više procesa je moguće **samo ako se računarski sistem sastoji od više centralnih procesora**. Za takav računarski sistem se kaže da je višeprocesorski. Na sistemu sa jednim procesorom moguće je samo kvaziparalelno izvršavanje procesa. U bilo kom trenutku izvršava se samo jedan proces.
Primjer kvaziparalelnih procesa R1, R2 i R3 je prikazan na slici dole.
Primjer paralelnih procesa R1, R2 i R3 je dat na slici dole desno. Pretpostavljeno je da sistem ima 3 procesora i da se na svakom od njih izvršava samo po jedan proces.



Slika 1.



Slika 2.

Jedna nit može da zahtjeva **servis neke druge niti**. Tada ta nit mora da sačeka da se pozvani servis završi. Potreban je mehanizam sinhronizacije jedne niti sa drugom niti ili sa hardverom. To se može uraditi pomoću signala koji ima značenje da je pozvani dio posla završen.
Konkurentne niti su niti koje se izvršavaju u isto vrijeme. **Konkurentne** niti mogu da se takmiče za ekskluzivno korišćenje resursa. Niti mogu da postavljaju istovremene zahtjeve za istim resursom ili istim servisom. Operativni sistem mora da obezbijedi interakciju između konkurentnih niti. Kritična sekcija je segment koda čije instrukcije mogu da utiču na druge niti. Kada jedna nit izvršava kritičnu sekciju ni jedna druga nit ne smije da izvršava tu istu kritičnu sekciju.

Slika 3.

Teškoće koje nastaju u softverskoj realizaciji algoritama za upravljanje kritičnim sekcijama su:

- stalno testiranje promjenljivih ili stanja čekanja, što troši procesorsko vrijeme,
- svi detalji implementacije direktno zavise od programera i mogućnost greške je uvijek prisutna,
- ne postoji način da se nametne protokol koji zavisi od kooperacije, programer može da izostavi neki dio,
- ovi protokoli su suviše komplikovani.

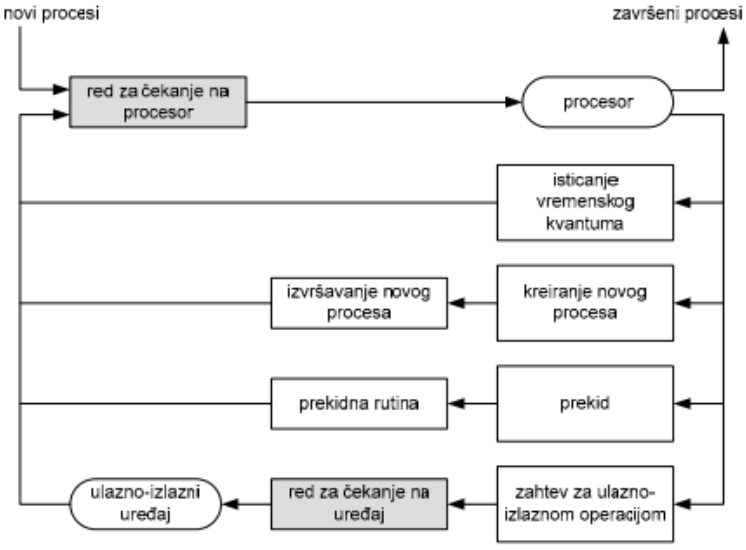
Problem upravljanja konkurentnim procesima

Za upravljanje konkurentnim procesima potrebno je da:

- Metod za dijeljenje vremena mora da bude implementiran tako da omogući svakom od kreiranih procesa da dobije pristup sistemu. Ovaj metod uključuje mogućnost prekidanja onih procesa koji ne ustupaju procesor dovoljno.
- Proces i sistemski resursi moraju da imaju zaštitu i moraju da budu zaštićeni međusobno. Veličina memorije koju dati proces može da koristi mora da bude ograničena za bilo koji proces, kao i operacije koje može da izvršava na uređajima kao što su diskovi.
- Sistem ima ugrađene mehanizme unutar jezgra za prevenciju potpunog zastoja između procesa.

Redovi čekanja na procesor

Proces u svom životnom ciklusu prolazi kroz nekoliko redova čekanja (scheduling queues).
Nakon kreiranja, proces se ubacuje u takozvani red čekanja za poslove (job queue), koji obuhvata sve postojeće procese na sistemu. Procesi se mogu nalaziti u raznim stanjima i raznim lokacijama (u memoriji, na disku, djelimično u memoriji, delimično na disku).



Svi procesi koju su spremni za rad i nalaze se u operativnoj memoriji čuvaju se u redu čekanja na procesor, odnosno u redu čekanja spremnih procesa (ready queue).
Redovi čekanja na procesor se po pravilu realizuju kao povezane liste, formirane od kontrolnih blokova procesa sa definisanim redosledom izvršavanja procesa. Redosled se specificira preko zaglavlja liste (queue header), koje sadrži informacije o početnom i poslednjem kontrolnom bloku u listi, i pokazivača na sledeći kontrolni blok. Opertivni sistem uvodi i poseban red čekanja za svaki ulazno-izlazni uređaj (I/O queue).
Svaki red čekanja na uređaj sadrži povezanu listu kontrolnih blokova procesa koji taj uređaj zahtevaju. Na slici lijevo prikazana je tehnika za raspoređivanje procesa (**scheduling**). Prisutne su dvije vrste redova čekanja: red čekanja na procesor i redovi čekanja na ulazno-izlazne uređaje.

Novi proces se inicijalno postavlja u red čeknja za spremne procese u kome čeka dodelu procesora, nakon čega napušta red i počinje da se izvršava.
Proces koji se nalazi u stanju izvršavanja može:
• ostati bez procesora kada mu istekne dodjeljeno vrijeme,
• kreirati novi proces i čekati u blokiranom stanju da se novi proces izvrši,

• ostati bez procesora kada se dogodi prekid,
• postaviti I/O zahtev, nakon čega se prebacuje u red čekanja na ulazno-izlazni uređaj, odnosno postaje blokiran.
Proces se vraća u red čekanja na procesor sve dok se ne završi, posle čega oslobađa sve zauzete resurse.

Planer poslova i dispečer sistema

Mogli ste da vidite da proces u svom životnom ciklusu prolazi kroz razna stanja i redove čekanja. Programi za raspoređivanje (schedulers) odlučuju o tome kada će proces ući u neki red čekanja ili napustiti taj red.
Ove komponente operativnog sistema se pominju pod imenima:
• planer poslova (job scheduler, long-term scheduler, high-level scheduler) i
• dispečer (dispatcher, short-term scheduler, low-level scheduler).

Planer poslova se u hijerarhijskom modelu nalazi iznad kernela, obavlja sljedeće funkcije:
• dijeli poslove na procese,
• na osnovu određenih algoritama dodeljuje prioritete procesima i
• dovodi procese u red čekanja na procesor.
Planer poslova se poziva kada se pojave novi procesi ili kada jedan ili više procesa završi aktivnosti. Planer poslova reguliše stepen multiprogramiranja, odnosno broj simulatnih procesa u memoriji.

Od njega se očekuje da pravi dobru selekciju procesa koji će dobiti memoriju kako bi sistem što efikasnije funkcionisao. On vrši selekciju procesa koji će dobiti memoriju kako bi sistem efikasno funkcionisao. *Planer poslova potiče sa main-frame sistema, koje je obavljao grupnu obradu poslova (batch).*

Dispečer sistema (odnosno planer poslova niskog nivoa; *dispatcher, low level scheduler*) je dio kernela koji dodeljuje procesor procesima (vidi PCB). Procesor se uvijek dodeljuje na osnovu nekog algoritma, kao što je na primer Shortest Job First (prvo poslovi koji zahtjevaju najmanje procesorskog vremena).
Zadatak dispečera sistema je da dodeljuje procesor procesima koji se nalaze u procesorskom redu. Dispečer dodeljuje procesor svaki put kad tekući proces pređe iz stanja RUN i stanje WAIT ili READY. Dispečer utvrđuje koji je proces najpovoljniji za dodjelu procesora, odnosno koji je proces u stanju READY najvišeg prioriteta.
Ukoliko se procesorski red formira sa inkrementalnim prioritetima (dinamički izmjenljiva lista), najpovoljniji proces je prvi proces u redu. Ukoliko na sistemu postoji više procesorskih redova, odnosno po jedan red za svaki prioritet, uzima se prvi proces iz odgovarajućeg reda.

Pojam, struktura, upravljanje i zaštita fajlova

Bilo koji operativni sistem treba da omogući jednostavno organizovanje i pretraživanje podataka na računaru.

Pojam fajla - datoteka (file)

Datoteka, fajl (file) predstavlja osnovnu organizacionu cjelinu, u koju smještamo podatke.

Fajl (file) znači papir, dokument unutar fascikle.

Pojam fajl je uveden zbog potrebe povezivanja programa sa ulazno-izlaznim uređajima koji omogućavaju memorisanje podataka.
Pod fajlom nazivamo imenovani prostor na memorijskom medijumu (disk, cd, usb) koji sadrži određene informacije.

Datoteke omogućuju korisnicima da organizuju podatke u skladu sa svojim potrebama. Za korisnika, datoteka (file) predstavlja kolekciju povezanih informacija, tj logičku cjelinu sa značenjem.
Za operativni sistem, to je **objekat koji se čuva u sekundarnoj memoriji**.

Naglasimo datoteke se mogu posmatrati na dva načina:
- **kao logička sliku** koja prikazuje datoteku kao redove informacijskih cjelina
- **koo fizička slika** u kojoj se datoteka sastoji iz reda informacionih blokova iste veličine

Za datoteku se može reći i da je to apstraktni tip podataka koji OS definiše i implementira kao niz logičkih slogova.

Tip podataka podrazumijeva skup vrijednosti koje podatak može da ima, memorijski prostor potreban za smiještanje podatka, kao i operacije koje mogu da se izvrše nad podatkom.
Za razliku od podataka koji se nalaze u operativnoj memoriji, **podaci u datoteci su postojani**, tj. ostaju sačuvani i nakon isključenja napajanja ili ponovnog uključenja sistema. Podaci koji se čuvaju u datotekama se mogu izgubiti jedino otkazom uređaja na kojima se datoteke nalaze.

Bitne karaktersitike sistema datoteka su: struktura, imenovanje, zaštita, fizička organizacija datoteka i načini korišćenja.
Na jednom disku se može nalaziti više različitih sistema datoteka.

Upisivanje logičke datoteke na momorijski uređaj je jedna od osnovnih funkcija operativnih sistema, tj. glavna funkcija sistema za upravljanje datotekama.
U opštem slučaju veličina logičkog sloga je različita od veličine fizičkog sloga na uređaju na kome se datoteka skladišti. Datoteka nije uvijek memorisana kao cjelina na jednom mjestu na disku, već kod realnih sistema sadržaj datoteke može da bude razbacan u više blokova koji nisu susjedni, odnosno može doći do fragmentacije podataka.

Za pojam fajla su vezani sadržaj i atributi.
Sadržaj fajla obrazuju korisnički podaci.

Atributi fajla se čuvaju u deskriptoru fajla poznatim pod imenom kontrolni blok datoteke (**file control block, FCB**). Kontrolni blok datoteke najčešće je direktorijumska struktura, ali se može implementirati i kao zasebna tabela (na Primjer, indeksni čvor u sistemu datoteka ext2).
Najvažniji atributi datoteke su: ime, veličina, datum i vrijeme posljednjeg ažuriranja, vlasnik, dozvole, lokacija podataka, itd.

Zaštita datoteka je važna osobina OS koja omogućava da različiti korisnici skladište svoje informacije na djeljenom računaru i da tim informacijama mogu da pristupe samo vlasnici tih podataka i autorizovani korisnici.
Važno je i spriječiti da ne dođe do gubljenja podataka. Datoteke ili dijelovi datoteka mogu da budu uništeni na više načina: hardverske greške, otkazi napajanja, otkazi glava diska, prašina, velike temperaturne promjene, velike promjene vlažnosti, softverske greške, vandalizam drugih korisnika, prisustvo jakih magnetnih polja, itd.

Format ili tipovi datoteka

Format ili tip datoteke određuje na koji način se vrši memorisanje računarskih podataka.
Formati su neophodni zbog činjenice da su datoteke sa stajališta operativnog sistema jedno-dimenzionalni redovi bajta Takve ograničene, linearne strukture ne mogu jasno opisati različite vrste podataka što znači da se kompleksni podaci moraju intepretirati putem konvencija. Sve konvencije za jednu vrstu datoteka označavaju se formatom.
U početnom stadiju računara, kompleksni i komprimirani, takozvani binarni formati su bili standardni. Razlog je jednostanan - skupocjenost memorije na čvrstom disku. Dodatni razlog predstavljala je i činjenica da su komercijalne firme strogo čuvale formate kao poslovne tajne. Razmjena podataka sličnih programa iz različitih firmi je stajala u pozadini. Prvi koraci u pravcu slobodne razmjene podataka između različitih programa sačinjavali su RIFF formati iz 80-tih godina npr. WAV format za audio datoteke.

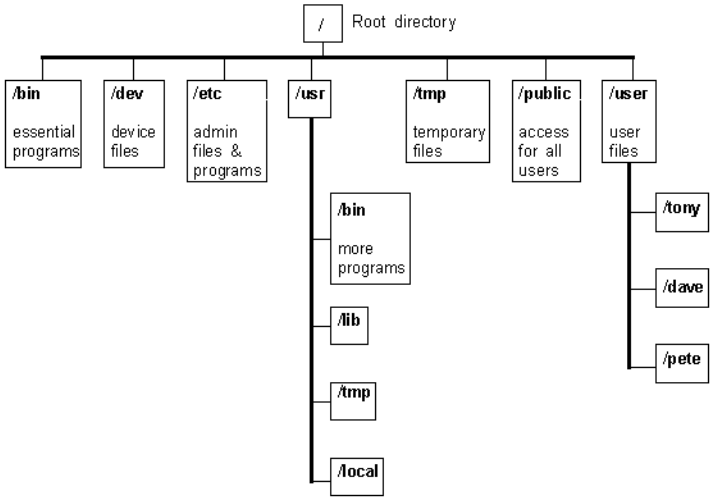
U posljednje vrijeme sve se više koristi metoda koja omogućava memorisanje podataka koje je nezavisno od programa i proizvodjača - XML. Preduslov za korištenje ove tehnologije je upotreba internacionalno priznatih standarda. XML dokument se sastoji od deklaracija, elemenata, atributa, procesnih instrukcija i komentara. To je obična tekstualna datoteka čitljiva na svakoj platformi koja može čitati tekstualne podatke. XML podržava Unicode i omogućuje prikaz teksta na svim danas poznatim jezicima.
XML je standardizovani jezik i za njegovu standardizaciju brine se World Wide Web Consortium. Koristi se za različite namjene: odvajanje podataka od prezentacije, razmjenu podataka, pohranu podataka, povećavanje dostupnosti podataka i izradu novih specijaliziranih jezika za označavanje.

Datoteke se najjednostavnije mogu podijeliti na izvršne datoteke (tj prevedene i povezane programe) i datoteke s podacima, koje mogu biti tekstualne i binarne.
U većini operativnih sistema, posle izrade, datoteka osim imena dobija i tip. Pomoću tipa, operativni sistem može preliminarno da odredi vrstu datoteke i da je poveže s nekom aplikacijom.

Tip datoteke se može realizovati korišćenjem nastavka imena datoteke **tj ekstenzijom (extension)**, što je slučaj u operativnim sistemima **DOS/Windows**.
U sledećoj tabeli navedene su neke od značajnijih ekstenzija koje se koriste u ovim operativnim sistemima.

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

Kod **Unixolikih** OS tip datoteke se alternativno može zadati **nekim karakterističnim zapisom**.



Ova metoda zadavanja tipa primenjuje se na **UNIX/Linux sistemima** – **magični brojevi** koji se nalaze na samom početku datoteke određuju njen tip.

Iz komandnog intepretera **UNIX ne** prepoznaje ekstenziju (datoteka sa ekstenzijom txt bez problema može biti arhiva ili izvršna datoteka). Međutim, Linux u grafičkom okruženju preliminarno određuje vrstu datoteke na osnovu ekstenzije i otvara datoteku pridruženom aplikacijom.

Prepoznavanje i otvaranje fajla kod Unixa

Na UNIX sistemima postoji nekoliko osnovnih tipova datoteka:

- tekstualne datoteke - ASCII (neformatiran tekst), English text (tekst sa interpunkcijskim karakterima)
- izvršni shell programi. Ovaj tip datoteka se može pro čitati korišćenjem programa za pregled sadržaja tekstualnih datoteka (cat, less) ilimodifikovati pomo ću editora teksta (kao što su ***vi*** i ***joe***);
- izvršne (binarne) datoteke;
- datoteke u koje su smješteni podaci (na primjer, Open Office Writer dokument). Ove datoteke mogu se pročitati korišćenjem programa iz koga su kreirane i programakoji je kompatibilan sa tim formatom datoteka (na primjer, Open Office može daotvori dokumente kreirane Microsoft Office paketom).

Kako bi bez otvaranja datoteke u nekom editoru (koji inače služe samo za pregled tekstualnih datoteka) saznali o kakavoj se datoteci radi, možemo se poslužiti naredbom file: **$ file /bin/ps**

Prilikom određivanja tipa datoteke komanda file izvršava slede će testove:

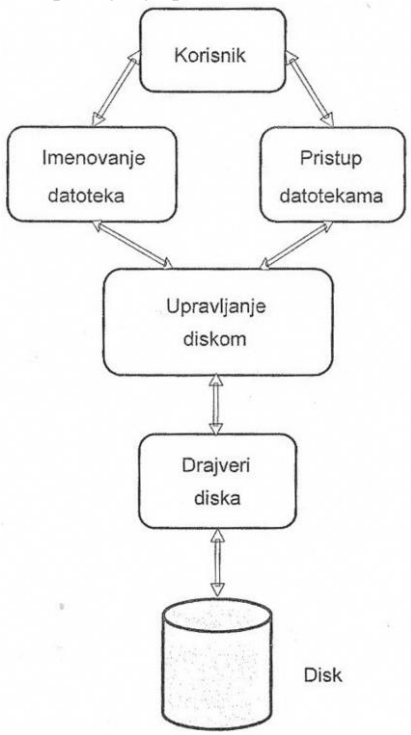
- **Test specijalnih datoteka** se na osnovu sistemskog poziva **stat()** određuje da li jedatoteka regularna ili specijalna (odnosno node, simbolički link, imenovani pipe). Ukolikodatoteka prođe ovaj test na ekranu se prikazuje tip datoteke, a dalje izvršenje komande seobustavlja.
- **Test magičnih brojeva** određuje se programski paket kojim je datoteka kreirana. Svaki program prilikom kreiranja datoteka upisuje neke kontrolne informacije (**overhead**) koje ovaj test posmatra kao identifikator tipa datoteke, odnosno magični broj. Bilo kojadatoteka sa nepromenjenim identifikatorom, koji se nalazi na fiksnom ofsetu od početka datoteke, može biti opisana na ovaj na čin.
  Magični brojevi se smještaju u **datoteke/usr/share/magic.mgc** i **/usr/share/magic** i prilikom izvršavanja ovog testa redom traže nafiksnom ofsetu u datoteci. Ukoliko se određeni magi čni broj poklopi sa magi čnim brojemdatoteke, program file prekida dalje izvršenje i na ekranu ispisuje informaciju o programukojim je datoteka kreirana.Ukoliko datoteka ne prođe ni test magi čnih brojeva, program file će pokušati da odredi dali je datoteka tekstualna i u tom smislu ćenajpre izvršiti test seta karaktera.
- **Test seta karaktera** (**character settest**). U datoteci postoje tri tipa karaktera:
  - normalni karakteri (karakteri koji mogu biti prikazani na ekranu),
  - kontrolni karakteri, kao što su space i tab (karakteri koji ne mogu biti prikazani naekranu, ali se pojavljuju u običnim tekstualnim datotekama),
  - binarni karakteri (karakteri koji se ne mogu prikazati direktno na ekranu, i ne pojavljuju se u običnim tekstualnim datotekama
- **Karakter set (character set)** je jednostavna deklaracija normalnih, kontrolnih i binarnihkaraktera. Na osnovu karaktera koji se u datoteci pojavljuju komanda file određuje karakter set kome datoteka pripada. Ukoliko datoteka pro đe ovaj test komanda file naekranu prikazuje ime karakter seta datoteke (ASCII, ISO-8859-x, UTF-8, UTF-16,EBCDIC).
- Nakon toga se izvršava **jezički test (language test)** kojim će komanda file pokušati da odredi programski jezik u kome je datoteka napisana - u datoteci se traženizovi karaktera koji su karakteristi čni za određene programske jezike. Na primjer, ukolikose u datoteci pojavi riječ **keyword.br**, to znači da se ispitivanje vrši nad troff(1) ulaznom datotekom, dok riječ struct nagoveštava da se radi o C programu.
  U slučaju da program **file** jezičkim testom ne može da odredi programski jezik, datoteka se smatra tekstualnom

Logički i fizički sistem za upravljanje datotekama

Najvažnije aktivnosti operativnog sistema u dijelu za upravljanje podacima na sekundarnoj memoriji su:
1) planiranje dodjele sekundarne memorije,
2) dodjela slobodne sekundarne memorije,
3) upravljanje slobodnim memorijskim prostorom na sekundarnoj memoriji.

Sistem za upravljanje podacima ima sljedeće funkcije:
- identifikovanje i lociranje izabranih datoteka,
- korišćenje direktorijuma za opisivanje lokacije svih datoteka i njihovih atributa,
- opis kontrole pristupa korisniku u djeljenom sistemu, rad sa blokovima radi pristupa datotekama.

Sistem za upravljanje podacima upravlja radom sekundarne memorije i može se dekomponovati na logički i fizički sistem za upravljanje podacima.



Komponente sistema za upravljanje podacima

Postoje dva odvojena aspekta sistema za upravljanje podacima: **interfejs** ka krajnjem korisniku i i**mplementacija**. Interfejs obuhvata datoteke i direktorijume, a implementacija fizičku organizaciju podataka.

Na slici su prikazane komponente sistema za upravljanje podacima.
Zadatak OS je da preslika podatke sa kojima radi krajnji korisnik preko datoteka na fizički uređaj ili više fizičkih uređaja.
Za svaki uređaj koji se povezuje sa datim računarskim sistemom potreban je poseban program koji se zove drajver uređaja (**device driver**).
Drajver uređaja je ili dio OS ili je raspoloživ operativnom sistemu. Namjenjen je za direktnu komunikaciju sa datim uređajem, kontrolerom ili kanalom. Drajver uređaja je odgovoran pokretanje U/I operacije na uređaju i procesiuranje završetka zahtjeva za U/I operacijom.

Drajver diska pristupa sektorima, stazama, cilindrima, upisno-čitajućim glavama diska, nosačima glava i svim drugim mehaničkim dijelovima koji obezbjeđuju da disk ispravno radi. Kada drajver diska zna koju komandu treba proslijediti disku, on je upisuje u registre kontrolera diska.

Logička struktura i organizacija datoteka (FILE SYSTEM)

U najjednostavnijem slučaju, datoteka nema svoju logičku strukturu i predstavljena je kao kolekcija reči, tj bajtova.

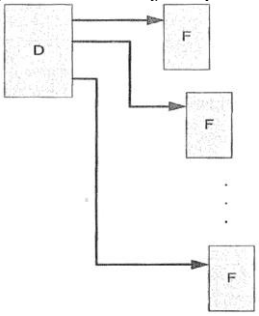
U jednostavnije logičke strukture spadaju strukture zapisa, pri čemu jedan zapis u datoteci može biti fiksne ili promjenljive dužine. Primjer zapisa je red u tekstualnoj datoteci. Složenije strukture predstavljaju formatirani dokumenti.
Datoteka sa logičkom strukturom može se simulirati umetanjem kontrolnih znakova u datoteke bez logičke strukture, npr umetanjem kontrolnih znakova Line Feed (LF) i Carriage Return (CR) u tekst, dobijamo jednostavan tekstualni dokument.

Datoteke se grupišu u skupove datoteka. Na primjer, prirodno je da datoteke sa podacima o studentima pojedinih godina studija istog smjera pripadaju jednom skupu datoteka.
Skupu datoteka pristaje naziv direktorijum (directory, folder), ako se prihvati gledište da skup datoteka sadrži imena svih datoteka koje su obuhvaćene pomenutim skupom.

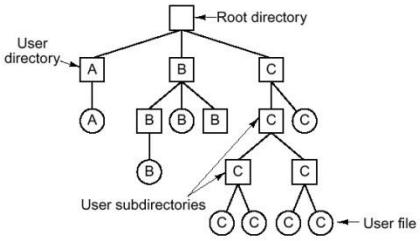
Radi razlikovanja direktorijuma, svaki od njih posjeduje ime koje bira korisnik. Za direktorijume su dovoljna jednodjelna imena, jer nema potrebe za klasifikacijom direktorijuma (pogotovo ne po njihovom sadržaju).

Direktorijumi

Direktorijum ili katalog je struktura podataka koja sadrži listu datoteka i poddirektorijuma.
Direktorijum omogućava automatsko vođenje evidencije o datotekama i preslikavanje između imena datoteka i samih datoteka.
I sam direktorijum je datoteka. Direktorijum je posebna datoteka koja sadrži jednu ili više datoteka. Osnovna razlika u odnosu na korisničke datoteke je da podaci unutar direktorijuma nisu korisnički podaci, već sistemski podaci o sistemu datoteka. Direktorijum sadrži informacije o atributima, lokaciji i vlasniku datoteke.
Direktorijumi uspostavljaju logičku organizaciju sistema datoteka koja je nezavisna od organizacije uređaja. Za pristup datotekama koje se nalaze u direktorijumima koriste se sistemski pozivi koji odgovaraju osnovnim operacijama za rad sa datotekama. Datotekama se može pristupati pomoću apsolutne staze i pomoću relativne staze. Datoteke na različitim putanjama mogu imati isto ime.
Diskovi mogu da sadrže na stotine hiljada ili više miliona datoteka. Direktorijum obezbjeđuje sistematičan način za imenovanje i lociranje tih datoteka. Dio sistema za upravljanje datotekama omogućava administraciju organizacije datoteka koje se mogu nalaziti na više uređaja uključujući uređaj koji se samo povrijemeno povezuju na dati sistem.
Postoji nekoliko načina za izbor strukture podataka kojom se predstavlja sadržaj direktorijuma.
Prva mogućnost je da direktorijum podržava **linearni prostor imena**, kao što je prikazano na slici 2.

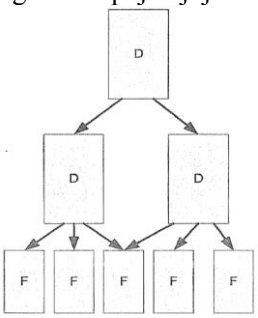


Slika 2. Linearni prostor imena



Slika 3. Hijerarhijski prostor imena sistemi direktorijuma DOS/Windows

Druga mogućnost je **hijerarhijski prostor imena**, kao što je prikazano na **slici 3.**
Najjednostavniji oblik hijerarhijskog direktorijuma je struktura podataka u obliku stabla. Na svaki direktorijum sem korijenog direktorijuma i na svaku datoteku pokazuje tačno jedan direktorijum. Direktorijumi koji imaju hijerarhijsku strukturu u obliku stabla se često primjenjuju u različitim OS. Ovakav način omogućava da datoteke sa istim imenom mogu da se pojavljuju u različitim direktorijumima.



Slika 4. Aciklični graf

Treći mogući način za strukturu podataka koja se koristi kao sadržaj direktorijuma je **aciklični graf**, kao što je pokazano na slici 4. Grafovi u opštem slučaju mogu biti ciklični i aciklični.
Kod cikličnih grafova postoji zatvorena putanja. Kod acikličnih grafova to nije slučaj.

Označavanje datoteka

Svaka datoteka posjeduje ime koje bira korisnik. Poželjno je da ime datoteke ukazuje na njen konkretan sadržaj, ali i na vrstu njenog sadržaja (radi klasifikacije datoteka po njihovom sadržaju).
Zato su imena datoteka dvodijelna, tako da prvi dio imena datoteke označava njen sadržaj, a drugi dio označava vrstu njenog sadržaja, odnosno njen tip. Ova dva dijela imena datoteke obično razdvaja tačka.

Realizacija sistema datoteka

Prilikom realizacije sistema datoteka, definiše se:

- **Logička struktura sistema datoteka** – logička struktura je način predstavljanja sistema datoteka korisniku; pod ovim se podrazumjeva definicija datoteka, direktorijuma, njihovih atributa i operacija dozvoljenih nad njima tj. sve što predstavlja logičku sliku sistema datoteka.
- **Fizička struktura sistema datoteka** – čine je strukture podataka na disku koje služe za skladištenje podatakač u ove strukture spadaju, npr blokovi (UNIX, Linux) i klasteri (clusters (sistem datoteka FAT))
- **Preslikavanje logičke strukture sistema datoteka u fizičku** – pod ovim preslikavanjem podrazumjeva se uspostavljanje veze između sadržaja konkretne datoteke ili direktorijuma i struktura na disku; npr, preslikavanje određuje da se sadržaj datoteke mydoc.txt iz direktorijuma c:\1, nalazi u blokovima 15,16 i 25...

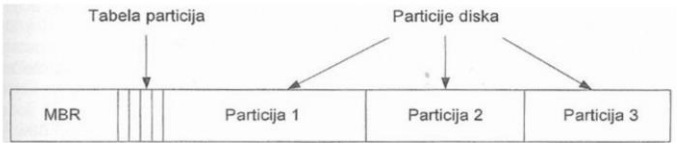
Po pravilu, sistem datoteka se realizuje u više nivoa, tj slojeva (layers). Pri radu sa objektima sistema datoteka –na Primjer, sa datotekama – potrebno je preći put od imena datoteke do konkretnog fizičkog bloka na disku.

Organizacija i fizička struktura sistema datoteka kod sekundarne memorije (diskova)

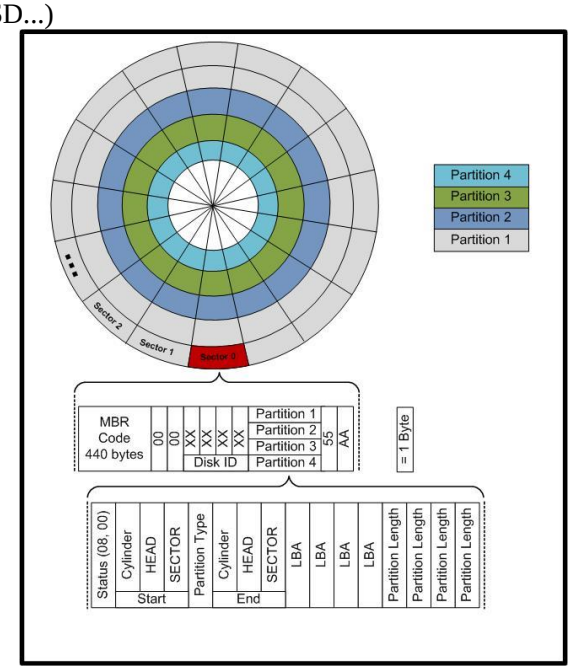
Sistemi datoteka se skladište na diskovima ili nekom drugom medijumu sekundarne memorije.
Svaki disk se može podijeliti na jednu ili više particija, kao na slici 1, pri čemu na particijama mogu da budu nezavisni sistemi datoteka.

Nulti sektor diska se naziva glavni startni slog (Master Boot Record, MBR) i koristi se za startovanje računara. Tabela particija koja se nalazi na kraju glavnog startnog sloga sadrži početnu i krajnju adresu svake particije i jedna od tih particija je označena kao aktivna.

MBR je nezavisan od operativnog sistema, dok je BOOT SECTOR aktivne particije zavisan o operativnom sistemu (DOS, Windows, Unix...) i vrsti fajl sistema (FAT, NTFS, FreeBSD...)



Slika 1. Primjer strukture sistema datoteka

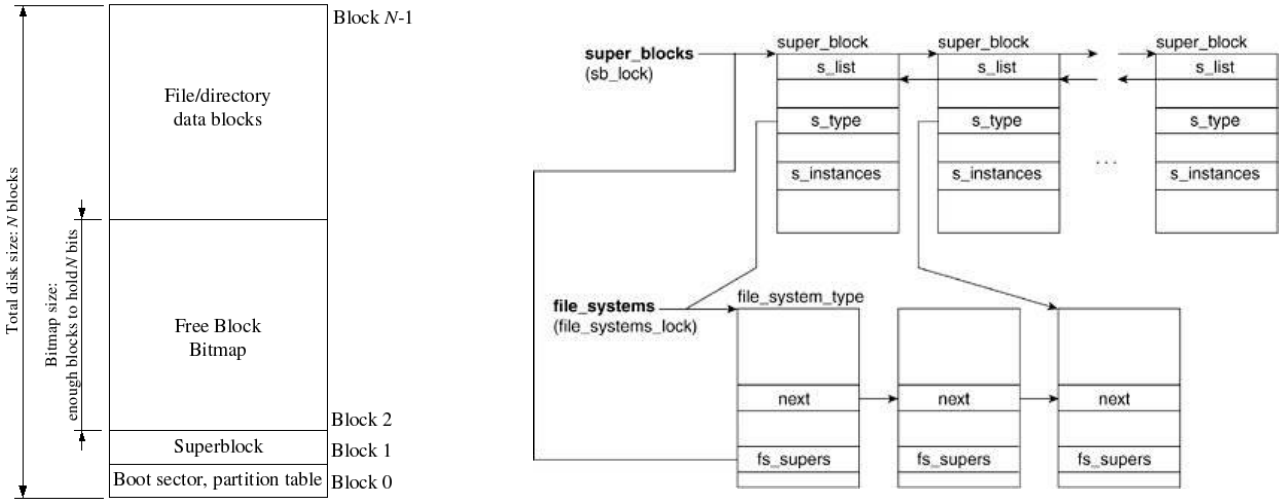


Nakon startovanja računara BIOS (Basic Input-Output System) čita i izvršava glavni startni slog. Na osnovu informacije u glavnom startnom slogu locira se aktivna particija na disku i učitava se u njen prvi blok koji se naziva **boot blok** u operativnu memoriju.

Program iz boot bloka puni operativni sistem koji se nalazi na aktivnoj particiji.

Svaka particija ima boot blok bez obzira da li se na njoj nalazi operativni sistem koji se može boot-irati ili ne.
Osim boot bloka struktura particije diska zavisi od sistema do sistema.

Bitan dio particije je i **superblok** koji sadrži ključne parametre o sistemu datoteka kao što su: tip sistema datoteka, veličina sistema datoteka, broj slobodnih blokova u sistemu datoteka i drugi administrativni podaci.
Jezgro operativnog sistema održava superblok u operativnoj memoriji i periodično ga upisuje na disk.
S obzirom da superblok sadrži kritične podatke neophodne za rad sistema, operativni sistem replikuje sadržaj superbloka na disk za slučaj da otkaže dio diska na kome je upisan superblok.



Pozicija superbloka

Pored superbloka u particiji se nalaze podaci o slobodnim i zauzetim blokovima datog sistema datoteka u obliku **bitmape ili povezane liste pokazivača**.
Takođe, u svakoj particiji se nalaze direktorijumi i datoteke koji pripadaju datom sistemu datoteka.
Particije mogu da budu primarne, logičke i dodatne.
Primarne particije su one sa kojih je moguće podizanje operativnog sistema. Svaki disk mora da ima bar jednu primarnu particiju. Upotrebom više primarnih particija moguće je instalirati i koristiti više operativnih sistema na istom disku.
Logičke particije su particije čija je namjena skladištenje podataka. Sa logičkih particija se ne može podizati operativni sistem.
Dodatne particije omogućavaju prevazilaženje ograničenja koje postoji po pitanju maksimalnog broja mogućih particija na jednom disku. Dodatna particija može da sadrži više logičkih particija.

Logička struktura i organizacija datoteka

Organizacija datoteka označava logičku strukturu slogova datoteke na osnovu načina na koji im se pristupa.

Kriterijumi koji se koriste kod izbora organizacije datoteke su:
- brz pristup podacima,
- jednostavnost ažuriranja,
- jednostavnost održavanja,
- pouzdanost.

Operativni sistemi podržavaju datoteke koje na implementacionom nivou mogu biti nizovi bajtova, nizovi slogova ili nizovi složenijih organizacija podataka.Postoji veliki broj organizacija datoteka koje su implementirane ili predložene za implementaciju. Od svih do sada poznatih organizacija datoteka, najčešće se koriste sljedeće:

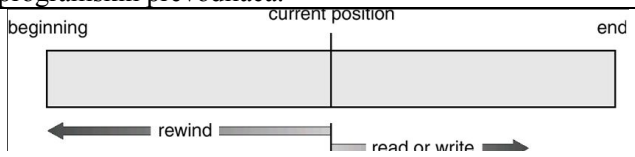
- **Serijska organizacija** datoteke omogućava da se podaci upisuju redosljedom kojim nastaju. Mogu se upisivati slogovi promjenljive dužine i promjenljiv skup polja unutar sloga. Unutar serijske organizacije ne postoji struktura.
- **Sekvencijalna organizacija**. Razlika između serijske i sekvencijalne organizacije datoteka je u tome što kod sekvencijalne organizacije postoji informacija o vezama između slogova logičke strukture datoteke. I kod jedne i kod druge organizacije slogovi se upisuju u susjedne lokacije unutar memorisjkog prostora dodjeljenog datoteci. Svi slogovi su iste dužine.

- Kod spregnute organizacije se koriste pokazivači. Postoje jednostruko, dvostruko i višestruko spregnute datoteke.
- **Rasuta organizacija**. Za rasutu organizaciju je karakteristično da se adresa lokacije dobija transformacijom vrijednosti identifikatora sloga. Neophodno je da svaki slog ima svoj ključ.
- **Indeks-sekvencijalna organizacija** datoteke uvodi tri zone: primarnu zonu (ili zonu podataka), zonu indeksa i zonu prekoračenja. Zbog poboljšanja performansi indeks-sekvencijalne datoteke se periodično reorganizuju.
- **Indeksna sa B stablima** otklanja potrebu periodičnog obnavljanja. Ovaj nedostatak je otklonjen kod indeksne organizacije sa B- stablima koja ima primarnu zonu i zonu indeksa. Osnovna karakteristika indeksne organizacije sa B-stablima je da postoji automatska reorganizacija po potrebi.
- **Organizacija sa više ključeva**. Datoteke sa više ključeva su najčešće indeksne datoteke sa B-stablom ili nekom varijantom B-stabla. Za svaki ključ postoji po jedna zona indeksa.

Metode pristupa datotekama i podacima

Da bi se podaci iz datoteke koristili potrebno je da se podacima pristupi i da se nakon toga učitaju u operativnu memoriju računara. Informacijama u datoteci se može pristupiti na više načina. Načini pristupa uskladištenim podacima se zovu metode pristupa.

Najjednostavniji metod pristupa je **sekvencijalni pristup**. Sekvencijalni pristup je zasnovan na modelu datoteke u obliku trake. Kod sekvencijalnog pristupa postoji pokazivač na trenutnu lokaciju unutar datoteke i podaci se procesuiraju slog po slog. Ovakav način pristupa se koristi kod editora i programskih prevodilaca.

Kod sekvencijalnog pristupa informacije se prosljeđuju tačnim redosljedom, jedna iza druge, u odnosu na vrijednost tekućeg pokazivača (**current pointer, CP**). Posle svakog pristupa datoteci, vrednost tekućeg pokazivača se ažurira.	

Sekvencijalni pristup zahteva da postoji mogućnost premotavanja datoteke na početak, tako da vrednost tekućeg pokazivača bude nula (CP=0).
Znači, pri sekvencijalnom pristupu datoteci, mogu se izvesti sljedeće operacije:

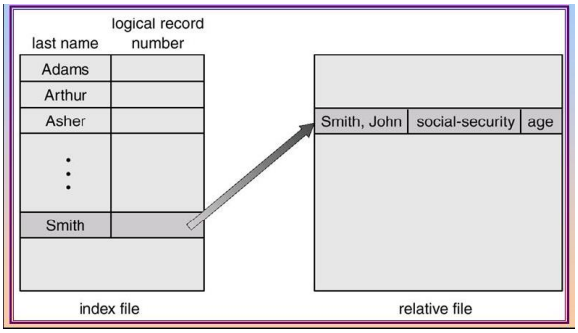
- čitanje sledećeg bloka
- upis u sledeći blok
- pozicioniranje na početak

Drugi metod **je direktan pristup** ili relativni pristup. Pretpostavlja se da se u datoteci nalaze logički slogovi fiksne dužine. Direktan pristup je zasnovan na modelu datoteke u obliku diska. Datoteka se može posmatrati kao numerisan niz blokova ili slogova i zbog toga se slogovima u datoteci može pristupati brzo i to po proizvoljnom redosljedu. Direktan pristup omogućava korisniku da prustupi krajnjem bloku datoteke bez čitanja prethodnog sadržaja. Pristup datotekama na disku je direktan. U operacije koje se mogu izvesti ukoliko je pristup datoteci direktan spadaju:

- čitanje n-tog bloka
- upis u n-ti blok
- pozicioniranje na n-ti blok
- čitanje sledećeg bloka
- čitanje prethodnog bloka

Postoje **i druge metode pristupa** koje se zasnivaju na direktnom pristupu, kao što je **indeksirani pristup** koji se zasniva na primjeni indeksa. Kod indeksiranog pristupa za datu datoteku se kreira indeks. Indeks sadrži pokazivače na blokove datoteke. Kod takvih metoda pristupa prvo se pretražuje indeks, a zatim se na osnovu pokazivača direktno pristupa željenom slogu.

Metoda pristupa pomoću indeksnih datoteka koristi se za pristup odgovarajućem zapisu u bazi podataka. Svakoj datoteci pridružena je indeksna datoteka, uređena po nekom kriterijumu, pomoću koje se prilikom čitanja brzo može naći odgovarajući zapis.
Prilikom upisa novog zapisa u datoteku, ažurira se i indeksna datoteka.



Performanse pristupa podacima direktno zavise od toga gdje se podaci nalaze. Keš memorija omogućava komponentama računara, koje imaju različite brzine rada, da efikasnije kumuniciraju privrijemenim premještanjem

podataka sa sporijeg uređaja na brži uređaj. Keš memorija je skuplja od uređaja sa kojih se podaci privrijemeno prebacuju, tako da povećanjem veličine keš memorije cijena sistema raste.

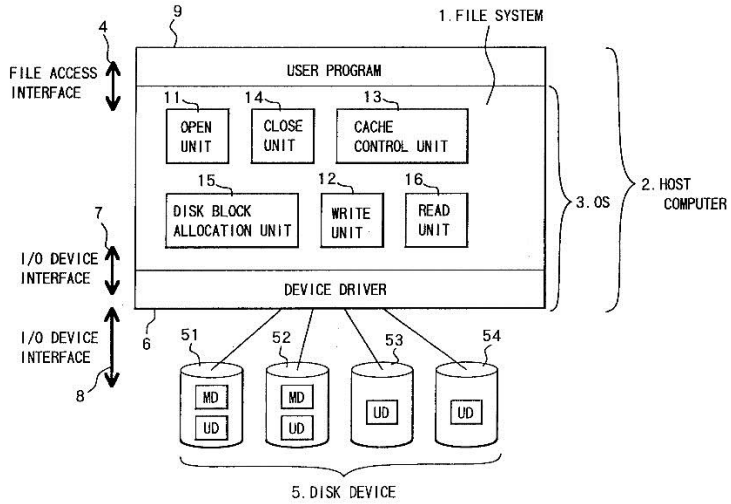
Metode dodjele prostora na disku

Jedan od problema koji se riješava u okviru sistama za upravljanje datotekama je kako dodijeliti prostor datotekama na disku tako da im se može brzo pristupiti i da korišćenje prostora na disku bude što je moguće bolje. Postoji više mogućih metoda za dodjelu prostora.
Tri glavne metode za dodjelu prostora na disku su:

1. **Metoda dodjele susjednih memorijskih lokacija** za skladištenje date datoteke koristi susjedne blokove diska. Dodjela susjednih blokova je jednostavna za realizaciju. Susjednost blokova poboljšava performanse. Na taj način ova implementaciona metoda omogućava brzo i jednostavno izračunavanje adrese bloka u kome se nalaze podaci. Za pristup podacima potreban je samo pomjeraj od početka datoteke. Moguć je sekvencijalan i direktan pristup podacima. Nedostatak strategije dodjele susjednih memorijskih lokacija jeste eksterna fragmentacija.
2. **Metoda dodjele povezanih blokova fiksne veličine** omogućava skladištenje svih datoteka tako što se koriste blokovi fiksne veličine. Susjedni blokovi se povezuju u povezanu listu. Osnovna prednost ove implementacione metode je da nema eksterne fragmentacije. Upravljanje memorijom je pojednostavljeno jer su svi blokovi iste veličine. Ne postoji potreba za premještanjem ili kompakcijom neke datoteke. Blokovi mogu da budu razbacani bilo gdje na disku. Nedostatak može da bude degradacija performansi kod direktnog pristupa podacima, jer je potrebno slijediti pokazivače od jednog bloka diska do sljedećeg.
3. **Metode koje koriste šemu sa indeksima** svakoj datoteci pridružuju tabelu indeksa. Svaki indeks u tabeli indeksa pokazuje na blokove diska koji sadrže stvarne podatke date datoteke. Slog za svaku datoteku unutar direktorijuma sadrži broj bloka indeksa i ime datoteke. Za male datoteke neiskorišćenost bloka indeksa može da bude velika, jer i u slučajevima kada se mali broj pokazivača stvarno koristi cio blok indeksa mora da bude dodijeljen.Ovakav način implementacije sistema datoteka obezbjeđuje brz direktan pristup podacima.

Upravljanje podacima (datotekama)

Upravljanje podacima je jedna od osnovnih funkcija operativnih sistema.



Korisnici savremenih računarskih sistema ne moraju da vode računa o upravljanju podacima, koje je neophodno kod aplikacija koje pristupaju datote-kama memorisanim na sekundarnoj memoriji.

Dio operativnog sistema koji upravlja podacima naziva se **sistem za upravljanje podacima** ili sistem za upravljanje datotekama (**File Manager**). Njegov zadatak je da omogući organizaciju podataka na takav način da krajnji korisnik može da im pristupi brzo i lako.

Upravljanje datotekom obuhvata ne samo upravljanje njenim sadržajem, nego i upravljanje njenim imenom. Tako, na primjer, stvaranje datoteke podrazumijeva i zadavanje njenog sadržaja, ali i zadavanje njenog imena (što se dešava u toku editiranja, kompilacije, kopiranja i slično). Takođe, izmjena datoteke može da obuhvati ne samo izmjenu njenog sadržaja, nego i izmjenu njenog imena (što se dešava, na primjer, u editiranju).

Operacije nad datotekama

Operacije nad datotekama su dio operativnog sistema.
Najčešće komande sistema za upravljanje podacima su:

- kreiranje datoteke,
- čitanje i pisanje unutar datoteke,

- pozicioniranje unutar datoteke radi operacije čitanja i pisanja,
- postavljanje i korišćenje mehanizma zaštite,
- promjena vlasništva nad datotekom,
- listanje datoteka u datom direktorijumu,
- brisanje datoteke.

Mogu se podijeliti na osnovne i izvedene. Zbog lakšeg snalaženja daćemo pregled nekih komandi za upravljanje podacima sa orginalnimm (engleskim) nazivima

osnovne	**izvedene**
open	create
close	delete
read	append
write	get attributes
seek	set attributes
	rename

Zaštita fajlova

Fajlovi su namjenjeni za trajno čuvanje podataka.
Za uspešnu upotrebu podataka neophodna je zaštita fajlova, koja obezbjeđuje da podaci, sadržani u fajlu, neće biti izmjenjeni bez znanja i saglasnosti njihovog vlasnika, odnosno, koja obezbijeđuje da podatke, sadržane u fajlu jednog korisnika, bez njegove dozvole drugi korisnici ne mogu da koriste.

Podaci, sadržani u fajlu, ostaju neizmjenjeni, ako se onemogući pristup fajlu i radi pisanja (radi izmjene njegovog sadržaja). Takođe, podaci, sadržani u datoteci, ne mogu biti korišćeni, ako se onemogući pristup datoteci, radi čitanja (radi preuzimanja njenog sadržaja).
Na ovaj način uvedeno **pravo pisanja** i **pravo čitanja** datoteke omogućuju da se za svakog korisnika jednostavno ustanovi koja vrsta upravljanja datotekom mu je dozvoljena, a koja ne. Tako, korisniku, koji ne posjeduje pravo pisanja datoteke, nisu dozvoljena upravljanja datotekom, koja izazivaju izmjenu njenog sadržaja. Ili, korisniku, koji ne posjeduje pravo čitanja datoteke, nisu dozvoljena upravljanja datotekom, koja zahtijevaju preuzimanje njenog sadržaja.
Za izvršne datoteke uskraćivanje prava čitanja je prestrogo, jer sprečava ne samo neovlašteno uzimanje tuđeg izvršnog programa, nego i njegovo izvršavanje. Zato je uputno, radi izvršnih datoteka, uvesti posebno **pravo izvršavanja programa**, sadržanih u izvršnim datotekama. Zahvaljujući posjedovanju ovog prava, korisnik može da pokrene izvršavanje programa, sadržanog u izvršnoj datoteci, i onda kada nema pravo njenog čitanja.
Pravo čitanja, pravo pisanja i pravo izvršavanja datoteke predstavljaju **tri prava pristupa** datotekama, na osnovu kojih se za svakog korisnika utvrđuje koje vrste upravljanja datotekom su mu dopuštene. Da se za svaku datoteku ne bi evidentirala prava pristupa za svakog korisika pojedinačno, uputno je sve korisnike razvrstati u klase i za svaku od njih vezati pomenua prava pristupa.
Iskustvo pokazuje da su **dovoljne tri klase korisnika. Jednoj pripada vlasnik datoteke, drugoj njegovi saradnici, a trećoj ostali korisnici.** Nakon razvrstavanja korisnika u tri klase, evidentiranje prava pristupa datotekama omogućuje matrica zaštite (protection matrix) koja ima tri kolone (po jedna za svaku klasu korisnika) i onoliko redova koliko ima datoteka. U presjeku svakog reda i svake kolone matrice zaštite navode se prava pristupa datoteci iz datog reda za korisnike koji pripadaju klasi iz date kolone. Na slici 2 je prikazan primjer matrice zaštite.

	owner	**group**	**other**
file1	**r w x**	**r - x**	**- - x**
file2	**r w x**	**- - x**	**- - x**
file3	**r w -**	**- w -**	**----**
file4	**r w x**	**---**	**r - -**

Slika 2. Matrica zaštite

U primjeru matrice zaštite sa slike 2 vlasnik (owner) datoteke file1 ima sva prava pristupa, njegovi saradnici (group) nemaju pravo pisanja, a ostali korisnici (other) imaju samo pravo izvršavanja (pretpostavka je da je reč o izvršnoj datoteci).
Ima smisla uskratiti i vlasniku neka prava, na primjer, da ne bi nehotice izmjenio sadržaj datoteke file2, ili da ne bi pokušao da izvrši datoteku koja nije izvršna (file3).
Primijetite i malu nelogičnost za file4.

Za uspjeh izloženog koncepta zaštite datoteka neophodno je onemogućiti neovlašteno mijenjanje matrice zaštite. Jedino vlasnik datoteke smije da zadaje i mijenja prava pristupa (sebi, svojim saradnicima i ostalim korisnicima). Zato je potrebno znati za svaku datoteku ko je njen vlasnik.

Takođe, potrebno je i razlikovanje korisnika, da bi se među njima mogao prepoznati vlasnik datoteke.
To se postiže tako što svoju aktivnost svaki korisnik započinje svojim predstavljanjem. U toku predstavljanja korisnik predočava svoje ime (username) i navodi dokaz da je on osoba za koju se predstavlja, za šta je, najčešće, dovoljna lozinka (pasword). Predočeno ime i navedena lozinka se porede sa spiskom imena i (za njih vezanih) lozinki registrovanih korisnika.
Predstavljanje je uspešno, ako se u spisku imena i lozinki registrovanih korisnika pronađu predočeno ime i navedena lozinka.
Predstavljanje korisnika se zasniva na pretpostavci da su njihova imena javna, ali da su im lozinke tajne.
Zato je i spisak imena i lozinki registrovanih korisnika tajan, znači, direktno nepristupačan korisnicima.
Jedina dva slučaja, u kojima ima smisla dozvoliti korisnicima posredan pristup ovom spisku, su:

- radi njihovog predstavljanja i
- radi izmjene njihove lozinke.

Za predstavljanje korisnika uvodi se posebna operacija, koja omogućuje samo provjeru da li zadani par ime i lozinka postoji u spisku imena i lozinki registrovanih korisnika. Slično, za izmjenu lozinki uvodi se posebna operacija, koja omogućuje samo promenu lozinke onome ko zna postojeću lozinku.
Sva druga upravljanja spiskom imena i lozinki registrovanih korisnika (kao što su ubacivanje u ovaj spisak parova imena i lozinki, ili njihovo izbacivanje iz ovog spiska) nalaze se u nadležnosti povjerljive osobe, koja se naziva administrator (superuser, root, admin, ...).
Zaštita datoteka potpuno zavisi od odgovornosti i poverljivosti administratora, odnosno od tačnosti pretpostavke da on neće odavati lozinke, niti ih koristiti, radi pristupa korisničkim datotekama. Zbog prirode njegovog posla, administratora ima smisla potpuno izuzeti iz zaštite datoteka, s tim da on tada svoje nadležnosti mora vrlo oprezno da koristi.
Nakon prepoznavanja korisnika (odnosno, nakon njegovog uspešnog predstavljanja), uz pomoć matrice zaštite moguće je ustanoviti koja prava pristupa korisnik posjeduje za svaku datoteku.
Da bi se pojednostavila provjera korisničkih prava pristupa, uputno je, umjesto imena korisnika, uvesti njegovu numeričku oznaku. Radi klasifikacije korisnika zgodno je da ovu numeričku oznaku obrazuju dva redna broja. Prvi od njih označava grupu kojoj korisnik pripada, a drugi od njih označava člana grupe. Podrazumijeva se da su svi korisnici iz iste grupe međusobno saradnici.
Prema tome, redni broj grupe i redni broj člana grupe zajedno jednoznačno određuju vlasnika. Saradnici vlasnika su svi korisnici koji imaju isti redni broj grupe kao I vlasnik.
U ostale korisnke spadaju svi korisnici čiji redni broj grupe je različit od rednog broja grupe vlasnika. Posebna grupa se rezerviše za administratore.
Numerička oznaka korisnika pojednostavljuje provjeru njegovog prava pristupa datoteci. Ipak, da se takva provjera ne bi obavljala prilikom svakog pristupa datoteci, umesno je takvu provjeru obaviti samo pre prvog pristupa.
Zato se uvodi posebna operacija otvaranja datoteke, koja prethodi svim drugim operacijama (kao što su pisanje ili čitanje datoteke). Pomoću operacije otvaranja se saopštava i na koji način korisnik namjerava da koristi datoteku. Ako je njegova namjera u skladu sa njegovim pravima, otvaranje datoteke je uspešno, a pristup datoteci je dozvoljen, ali samo u granicama iskazanih namjera.
Pored operacije otvaranja, potrebna je i operacija zatvaranja datoteke, pomoću koje korisnik saopštava da završava korišćenje datoteke.
Nakon zatvaranja datoteke, pristup datoteci nije dozvoljen do njenog narednog otvaranja.
Numerička oznaka vlasnika datoteke i prava pristupa korisnika iz pojedinih klasa predstavljaju atribute datoteke.

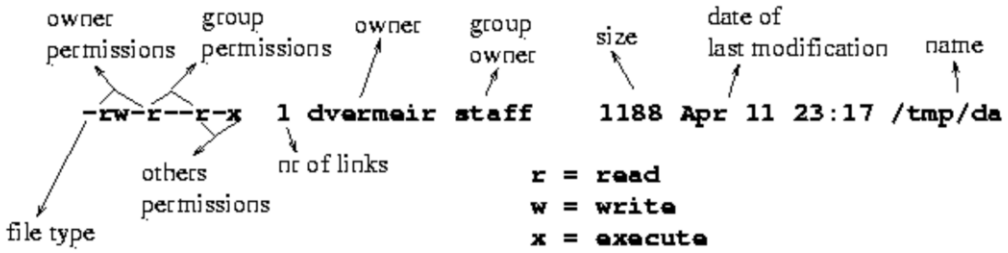


Figure 2: File access permissions

U Unix/Linux sistemima zaštita se uglavnom ostvaruje pomoću dva mehanizma:

- Definisanja vlasničke kategorije
  - Vlasnik (korisnik koji je napravio datoteku)

- Grupa (korisnička grupa kojoj je datoteka formalno priključena)
- Svi (svi korisnici sistema)

- Definisanje pristupnih prava se formira kao proizvoljan kombinacija osnovnih prava:
  - Čitaj (Read)
  - Piši (Write)
  - Izvršavaj (Execute)

U Windows operativnim sistemima, administracija kontrole pristupa i prava nad datotekama je ostvarena kroz sistem Korisnika i Grupa. Korisnik pripada Grupi koja može pripadati nadgrupi itd. Pri određivanju prava se poštuju principi:

- Nasleđivanja
- Sabiranja dozvola
- Jače zabrane

Dozvole za rad sa određenim datotekama se formiraju kao standardne dozvole u koje spadaju:

- Čitanje
- Pisanje
- Čitanje i izvršavanje
- Modifikacija
- Puna kontrola

Upravljanje memorijom

Upravljanje memorijom je jedna od osnovnih funkcija operativnih sistema.
Memorija se sastoji od niza memorijskih reči, a svaka ima jedinstvenu adresu. Prilikom izvršavanja procesa, procesor na osnovu programskog brojača (Program counter, PC) čita instrukciju iz memorije (fetch). Pročitane instrukcije u toku izvršenja dodatno mogu zahtjevati čitanje operanada ili upisivanje podataka na druge memorijske lokacije.
Razni operativni sistemi koriste različite metode upravljanja radnom memorijom. Ove metode mogu biti krajnje jednostavne, ali i veoma složene, poput straničenja (paging) i segmentacije, a svaka ima svoje prednosti i mane. Izbor metode za upravljanje memorijom u velikoj mjeri zavisi i od hardverske podrške, tj od procesorske arhitekture.

Savremeni operativni sistemi omogućavaju svakom procesu da dobije više virtuelne memorije, nego što je ukupna veličina stvarne (fizičke) memorije na datom računarskom sistemu.

Glavni cilj kod upravljanja memorijom je da se **kombinovanjem velike spore memorije sa malom brzom memorijom ostvari efekat velike brze memorije.**
Za upravljanje memorijom bitni su programski prevodilac, operativni sistem i hardver.

1. Programski prevodilac struktuira adresni prostor date aplikacije.
2. Operativni sistem preslikava strukture programskog prevodioca u hardver.
3. Na kraju, hardver izvršava stvarne pristupe memorijskim lokacijama.

Dodjeljivanje memorije

U radnu memoriju se, pored korisničkih procesa, smješta i rezidentni dio operativnog sistema (npr jezgro).
Da bi u višeprocesnom okruženju obezbjedio stabilan i pouzdan rad sistema, neophodno je što efikasnije dodjeliti različite dijelove memorije.
Mamorija se dijeli na najmanje dvije particije od kojih je jedna (najčešće niži deo) namenjena rezidentnom dijelu operativnog sistema (**kernel space**) a druga particija (viši dijelovi) – korisničkim procesima (**user space**).
Obično se u najnižem dielu memorije nalazi tabela prekidnih rutina.

U korisničkom adresnom prostoru nalazi se više procesa koji formiraju red čekanja na procesor. Takođe, postoje i procesi koji nastoje da uđu red čekanja, obično sa diska. Da bi proces ušao u red čekanja na procesor, neophodno je da najpre dobije potrebnu memoriju. Glavni problem pri upravljanju memorijom jeste dodjela slobodne memorije procesima koji se u ulaznom redu (alokacija memorije).

Tehnike za dodjelu memorije procesima grubo se mogu podijeliti na dvije vrste:

- Kontinualna alokacija (contiguous allocation) – i logički i fizički adresni prostor procesa sastoje se od kontinualnih niza memorijskih reči, pri čemu memorijske particije koje se dodjeljuju procesima po veličini mogu biti jednake ili razčličite
- Diskontinualna alokacija (discontiguous allocation) – fizički adresni prostor procesa nije realizovan kao kontinualan niz memorijskih adresa; diskontinualna alokacija obuhvata metode straničenja, segmentacije i straničenja sa segmentacijom.

Dodjela memorije može da bude kontinualna i diskontinualna. *Ako se za dati proces koriste susjedne memorijske lokacije u okviru datog dijela memorije, tada je to kontinualna dodjela memorije. Ukoliko se za dati proces koriste dijelovi memorije kod kojih postoji diskontinuitet u pogledu susjednosti lokacija, tada je to diskontinualna dodjela memorije.*

Vremenski i prostorni problem upravljanja memorijom

Najvažnije aktivnosti operativnog sistema u dijelu za upravljanje memorijom su:

1) vođenje evidencije o tome koji se dijelovi memorije trenutno koriste i ko ih koristi,
2) donošenje odluke o učitavanju procesa u memoriju, odnosno koje procese prebaciti u memoriju kada memorijski prostor postane raspoloživ,
3) dodjela i oslobađanje memorijskog prostora po potrebi.

Da bi se jedan program izvršio neophodno je da se u memoriju unesu i njegove instrukcije i podaci, kako bi bili dostupni centralnom procesoru. To ne znači da sve instrukcije i svi podaci moraju da budu u memoriji sve vrijeme tokom izvršavanja programa. Moguće je unijeti u memoriju samo jedan dio instrukcija programa sa podacima neophodnim za njihovo izvršavanje. Po izvršavanju tog dijela programa u memoriju se može unijeti, u sve lokacije, sljedeći niz naredbi sa podacima potrebnim za njihovo izvršavanje.
U uslovima višeprogramskog rada ovakva mogućnost je posebno zanimljiva. Držanjem u memoriji dijelova, a ne cijelih programa, moguće je aktivirati više programa u jednom vremenskom intervalu čime se povećava stepen višeprogramskog rada, a time i stepen iskorišćenja ostalih resursa računara. Naravno, ovakav način rada zahtijeva dodatne hardverske komponente i povećava složenost operativnog sistema.

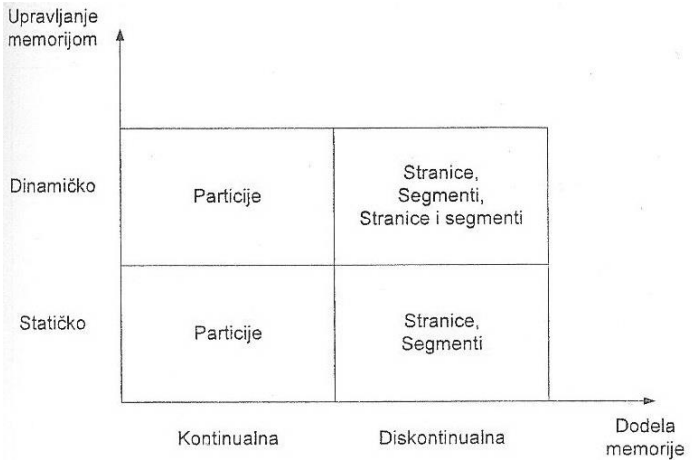
Sa stanovišta operativnog sistema memorija je podijeljena u dva nivoa. Prvi nivo čini primarna (glavna) memorija u kojoj se nalaze trenutno aktivni dijelovi različitih programa, dok drugi nivo čini sekundarna (pomoćna) memorija sa relativno brzim pristupom, na kojoj se čuvaju kompletne kopije svih aktivnih programa.

Operativni sistem problem upravljanja memorijom svodi **na problem vremenske i prostorne raspodjele** programa ili dijelova programa **između dva nivoa memorije**.
Drugim riječima, **upravljanje memorijom se sastoji od sljedeće tri komponente**:

1) upravljanje unošenjem ("*fetch policy*") – u smislu donošenja odluke o tome kada će se program ili njegovi dijelovi unijeti u memoriju,
2) upravljanje smještanjem ("*placement policy*") – u smislu donošenja odluke o tome gdje će se program ili njegovi dijelovi smjestiti u memoriji,
3) upravljanje zamjenom ("*replacement policy*") – u smislu donošenja odluke o tome koji će se program ili dijelovi programa izbaciti iz memorije da bi se oslobodio prostor za unošenje drugog programa ili dijelova drugog ili istog programa.

Različite metode i tehnike upravljanja memorijom koje se primjenjuju u operativnim sistemima razlikuju se upravo po tome kako i na osnovu čega donose neku od navedenih odluka.

Samo upravljanje memorijom može da bude statičko i dinamičko. Statičko upravljanje memorijom je kada se cio program unosi u memoriju prije izvršavanja programa. Dinamičko upravljanje memorijom je kada se veličina memorije određuje na osnovu veličine programa u trenutku unošenja programa u memoriju ili kada se dijelovi programa mogu unostiti u memoriju u toku izvršavanja programa.



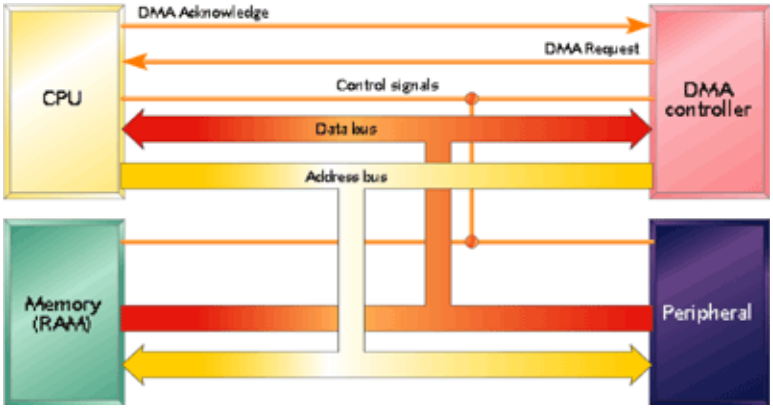
Načini upravljanja memorijom

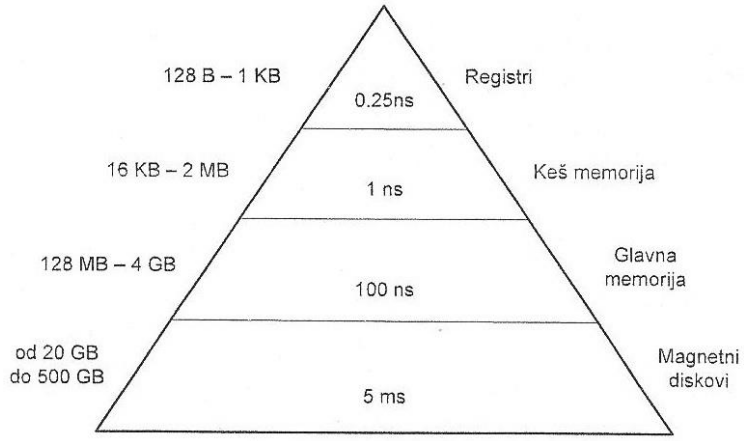
Prema tome, u opštem slučaju postoje sljedeći načini upravljanja memorijom:

- pomoću statičkih particija,
- pomoću dinamičkih particija,
- pomoću statičkih stranica,
- pomoću statičkih segmenata,
- pomoću dinamičkih stranica,
- pomoću dinamičkih segmenata,
- pomoću dinamičkih stranica i segmenata.

Memorijski sistem sa više nivoa

Da bi procesor mogao da čita instrukcije ili izvršava operacije nad podacima potrebno je da te instrukcije, odnosno podaci budu smješteni unutar fizičke (RAM) memorije.
Procesor i RAM memorija su povezani pomoću magistrale podataka veoma velike brzine.
Magistrala podataka najčešće može imati širinu 32, 64 ili 128 bita. Širina magistrale podataka definiše količinu podataka koja se može prenijeti u toku jednog ciklusa magistrale. Širina magistrale podataka ne definiše maksimalnu veličinu programa ili maksimalnu veličinu podataka. To je određeno širinom adresne magistrale.
Adresna magistrala može da ima bilo koju širinu u zavisnosti od procesora.





Memorijska piramida

Memorijski sistem savremenih računarskih sistema je više nivoski sistem, najčešće predstavljen kao piramida .

Memorijski sistem je podijeljen na četiri nivoa:

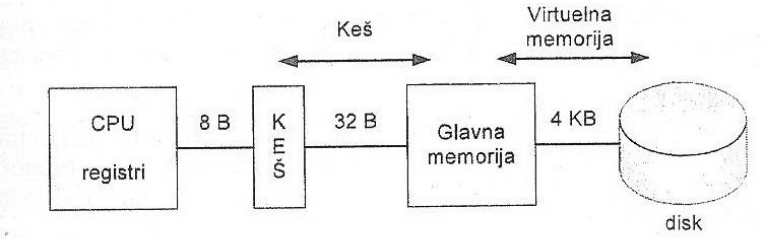
- registri,
- keš memorija,
- glavna memorija (RAM) i
- magnetni diskovi.

Za svaki nivo prikazan je kapacitet, prosječno vrijeme pristupa i naziv memorije.
Kapacitet i prosječno vrijeme se stalno poboljšavaju.
Takođe, na svakom memorijskom nivou postoji kompromis brzine i cijene.

Iako brzina prenosa podataka između procesora i RAM memorije može da bude veoma velika, u mnogim slučajevima između procesora i RAM memorije se koristi keš memorija. Keš memorija je mnogo brža, ali i skuplja. Ona ubrzava rad aplikacija. Iz ugla krajnjeg korisnika keš memorija je nevidljiva. U prosjeku oko 10% ukupne veličine programa se nalazi u glavnoj memoriji, a oko 1% programa se nalazi u keš memoriji. U registrima se nalazi jedna instrukcija i nekoliko podataka.
Na datom računaru najčešće postoji više programa ili procesa koji su aktivni u isto vrijeme i svaki od njih pristupa RAM memoriji. Da bi omogućio različitim procesima da koegzistiraju na datom računaru operativni sistem svakoj aplikaciji dodjeljuje virtuelni adresni prostor.

Takođe, operativni sistem preslikava virtuelni adresni prostor u fizički memorijski prostor.
U opštem slučaju svaki put kada se program učita u glavnu memoriju on će da bude na različitim memorijskim lokacijama. Odnos između brzine pristupa i cijene kod različitih nivoa memorije prikazan je na slici 3.

Program za upravljanje memorijom (*memory manager*) izvršava dvije osnovne operacije: dodjelu kontinualnog memorijskog prostora i oslobađanje zauzete memorije. Ponekad se koristi i treća operacija pomoću koje se može promijeniti veličina već dodjeljene memorije, tako što se veličina dodjeljene memorije smanji ili poveća.



	Registri	Keš	Glavna memorija	Memorija diska
Veličina	32 B	32 KB – 4 MB	1024 MB	100 GB
Brzina	1 ns	2 ns	30 ns	5 ms
Cena		$125/MB	$0.20/MB	$0.001/MB
Veličina linije	8 B	32 B	4 KB	

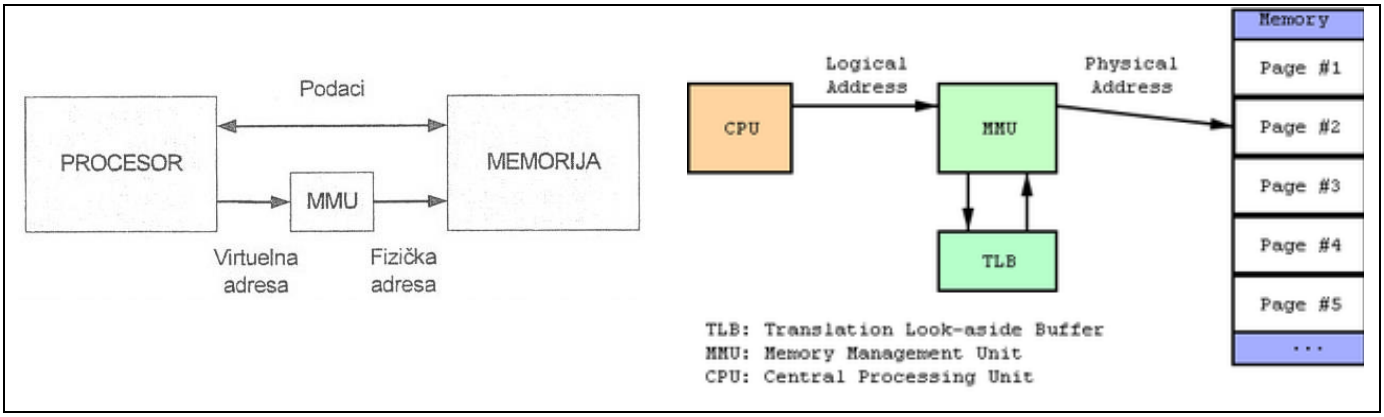
Veće, sporije, jeftinije

Slika 3. Memorijski sistem kod savremenih računarskih sistema

Logičko i fizičko adresiranje memorija

Razlikujemo logički i fizički memorijski adresni prostor.
Adresa koju generiše procesor naziva se logička adresa, dok se adresa kojom se puni memorijski adresni registar naziva fizička adresa.
Fizička adresa je adresa operativne memorije.
Preslikavanje logičkih adresa u fizičke je obavezno.
Fizičke i logičke adrese su identične ako se primjene **metode vezivanja adresa (*address binding methods*)** u vrijeme prevođenja i punjenja programa.

Upravljanje memorijom obuhvata preslikavanje adresa kao što je to prikazano na slici:



Preslikavanje virtuelne adrese u fizičku, pomoću posebnog hardvera MMU jedinice

Virtuelna adresa je adresa u programu i nju generiše procesor.
Fizička adresa je adresa na računarskom hardveru.

Kod savremenih računara preslikavanje iz virtuelne u fizičku adresu vrši poseban hardver koji se naziva **jedinica za upravljanje memorijom (*Memory Management Unit, MMU*)**. Za ovo preslikavanje adresa se kaže i da je to preslikavanje ili translacija iz logičke u stvarnu (fizičku) adresu. Hardver šalje jedinici za upravljanje memorijom fizičke adrese i na taj način vrši adresiranje glavne memorije. Kod većine savremenih mikroračunarskih sistema jedinica za upravljanje memorijom je ugrađena u čip procesora.

U okviru jedinice za upravljanje memorijom nalazi se poseban **registar za relociranje adresa (TLB).** Vrijednost upisana u registar za relociranje se dodaje svakoj virtuelnoj adresi.

Vezivanje adresa

Program se nalazi na disku kao binarna izvršna datoteka ***(binary executable).***

Program sa diska se mora mora učitati u memoriju, unutar adresnog prostora novostvorenog procesa. Zavisno od metode upravljanja memorijom koja se koristi, proces se u toku izvršavanja može više puta pomjerati na relaciji disk- memorija.

Kolekcija procesa na disku, koja čeka povratak u memoriju i nastavak izvršenja, naziva se ulazni red (input queue).

U sistemima sa omogućenim multiprogramiranjem, veći broj procesa dijeli radnu memoriju računara.

Programer ne može unapred znati koji će procesi biti zastupljeni u memoriji prili-kom izvršenja programa u takvom okruženju, niti koje će memorijske lokacije biti slobodne.

On ne može unapred odrediti fiksne memorijske lokacije za smještaj programa i zato koristi relativne ili simboličke (zavisno od programskog jezika i konkretne upotrebe).

Dužnost operativnog sistema je da prevede relativne u fiksne prilikom učitavanja programa u memoriju.

Vezivanje adresa se može shvatiti kao transformacija, tj prevođenje adrese sa „jezika“ jednog adresnog prostora u drugi.

Povezivanje instrukacija i podataka sa memorijskim adresama obavlja se u sledećim fazama:

- ***Vrijeme prevođenja (compile time)***

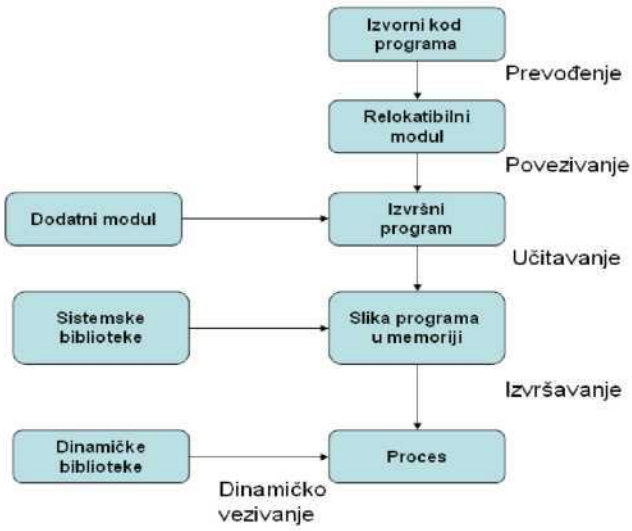
Kako nije poznato gdje će proces koji će izvršavati generisani kod biti smješten u memoriji, prevodilac generiše relativne, a ne apslolutne adrese. Program se kasnije može smestiti bilo gdje u memoriji.

- ***Vrijeme učitavanja u memoriju (load time)***

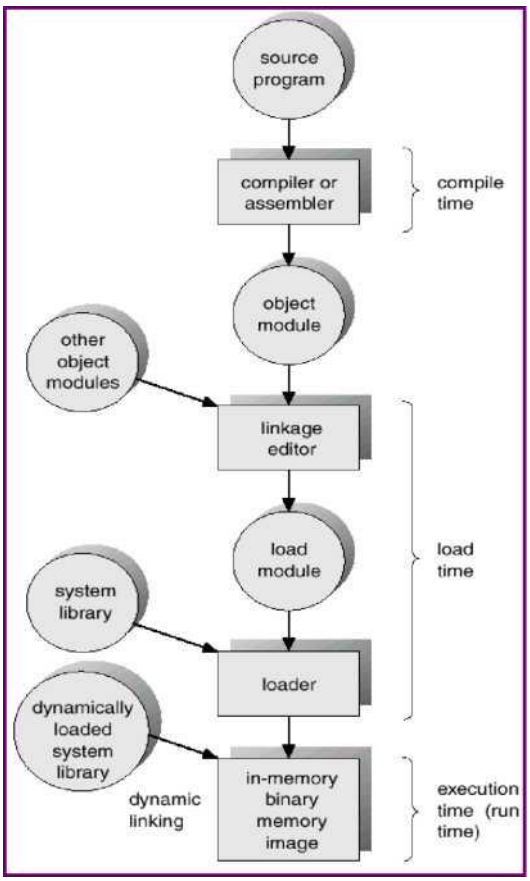
U fazi učitavanja povezivač (linker) i punilac (loader) na bazi relokatibilnog koda generišu apsloutne adrese i pune memoriju programom. Povezivač može povezati korisnički program sa drugim relokatibilnim modulima (object modules).

- ***Vrijeme izvršavanja (execution time)***

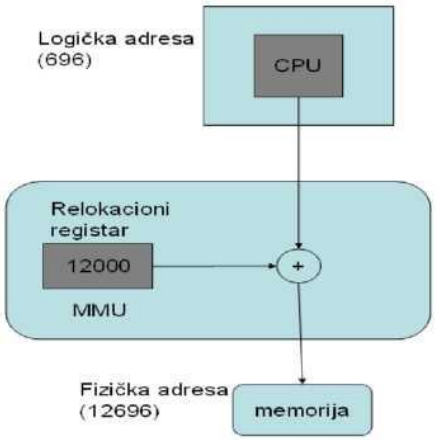
Za vrijeme izvršavanja, proces se može pomjerati s jednog segmenta na drugi (uključujući i disk, ukoliko se koristi i virtuelna memorija)

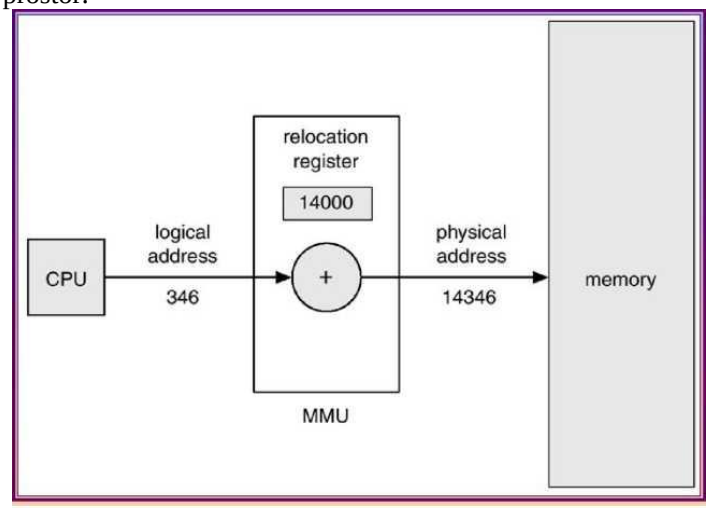


Faze koje prethode izvršavanju programa



Adresa koju generiše procesorska instrukcija je logička, a adresa same memorijske jedinice je fizička.
Logičke i fizičke adrese su potpuno iste u fazi prevođenja i u fazi učitavanja programa, ali se razlikuju u fazi izvršavanja (u fazi izvršavanja nazivaju se i virtuelnim adresama).
Skup svih logičkih adresa koje generiše program naziva se logički ili virtuelni adresni prostor, a skup svih fizičkih adresa koje njima adgovaraju naziva se fizički adresni prostor.





Mapiranje pomoću relokacionog registra

Relokacioni registar definiše adresu fizičkog početka programa. Svaka logička adresa koju generiše progrma se sabira s vrednošću relokacionog registra i tako se dobija fizička adresa.
Korisnički program uvijek počinje od nulte adrese i ne treba voditi računa o svom fizičkom prostoru, osim o gornjoj granici programa (max).
Logički adresni prostor koji se nalazi u opsegu [0,max] mapira se u opseg [R+0, R+max], gdje je R vrijednost relokacionog registra, tj fizička adresa početka programa.

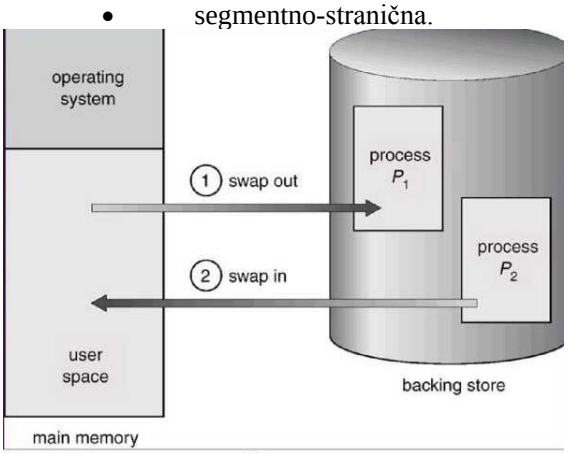
Privremena razmjena - SWAP

Prilikom izvršavanja, proces se mora nalaziti u radnoj memoriji. Postoje situacije kada se proces može privremeno prebaciti iz memorije na disk, kako bi se oslobodila memorija. Oslobođenja memorija puni se drugim procesom.
Razmena (swap) koristi se u prioritetnim šemama za raspoređivanje procesa gdje se procesi visokog prioriteta čuvaju u memoriji, dok se svi procesi niskog prioriteta upisuju na disk i čekaju da se oslobodi memorija. Ova varijanta razmjenjivanja naziva se ***roll out, roll in.*** Proces koji se razmenjuje mora biti potpuno oslobođen aktivnosti - ne sme da radi, niti da čeka kraj neke ulazno-izlazne operacije (stanja WAIT i READY).

Virtuelna memorija

Koncept virtuelne memorije je jedna od najboljih ideja primjenjenih na računarske sisteme. Glavni razlog za uspjeh ove ideje je da virtuelna memorija radi automatski, tj. bez intervencija programera aplikacija.
Osnovna prednost korišćenja virtuelne memorije je mogućnost izvršavanja programa koji zahtijeva memorijski prostor veći od fizičke (operativne) memorije, raspoložive na datom računarskom sistemu. Koncept virtuelne memorije stvara utisak korisniku da je njegov program u potpunosti učitan u memoriju i izvršen. Prije nastanka koncepta virtuelne memorije programer je morao da vodi računa da njegov program može da stane u fizičku memoriju. Osim toga, primjenom virtuelne memorije moguće je dijeljenje računara između procesa čija je veličina ukupnog adresnog prostora veća od veličine fizičke memorije.
Ako se primjeni vezivanje adresa u vrijeme izvršavanja programa fizička i logička adresa su različite i **tada se logička adresa naziva virtuelna adresa.**
Program za upravljanje memorijom **proširuje RAM memoriju sa rezervisanim dijelom memorijskog prostora na disku**. Prošireni dio RAM memorije se naziva **zamjenski (*swap*) prostor**.
Proširenje RAM memorije zamjenskim prostorom ima isti efekat kao instaliranje dodatne RAM memorije. U mnogim slučajevima dovoljno je povećati zamjenski prostor kako bi se izvršavali veći programi. Operativni sistem jedino mora da obezbjedi da program i podaci budu raspoloživi u RAM memoriji u trenutku kada su potrebni. Dijelovi programa kojim se ne pristupa često i dijelovi programa koji se koriste za upravljanje greškama se prenose na zamjenski prostor. Kada bilo koji od tih dijelova zatreba, program za upravljanje memorijom taj dio prenosi u RAM memoriju.
Koncept virtuelne memorije posebno dolazi do izražaja kod višekorisničkih operativnih sistema, jer kod prenošenja dijelova programa ili podataka u/iz operativne memorije procesor ne mora da čeka, vec odmah prelazi na izvršavanje drugog posla. Svaka aplikacija ima svoj virtuelni adresni prostor koji operativni sistem preslikava u fizičku memoriju. Veličina virtuelne memorije je ograničena samo veličinom zamjenskog prostora na disku. Jednostavno rješenje korišćeno kod prvih Unix sistema je posebna particija diska koja se koristi samo kao zamjenski prostor. Kod takvog rješenja prostor diska se dijeli na dva glavna dijela: jedan koji se koristi za straničenje i drugi koji se koristi za *file* sistem. Problem sa ovakvim pristupom je nefleksibilnost.
Postoje sljedeće osnovne vrste organizacije virtuelne memorije:

- segmentna,
- stranična,
- segmentno-stranična.



Tehnika razmjene zahtjeva postojanje 3 komponente:

- Prostor na disku ***(swap space***) na koji će se smještati uspavani procesi
- Mehanizam ***swap-out*** koji prebacuje proces iz memorije na disk
- Mehanizam ***swap-in*** koji vraća uspavani proces sa diska u memoriju

Najveći dio vremena u ciklusima razmene otpada na prenos podataka između memorije i diska. Trajanje jedne razmene zavisi od količine podataka za prenos, karakteristike diskova i pratećeg hardvera. Kako je to vrijem e ogromno u odnosu na vrijem e izvršavanja memorijskih ciklusa, ne preporučuje se često korišćenje tehnike razmjenjivanja.
Swap postoji na svim modernim operativnim sistemima i to u različitim modifikovanim varijantama. Rijetko se razmenjuju cjeli procesi - uglavnom se razmenjuju manji dijelovi memorije (npr stranice).

Raspodjela memorije

Ukoliko se koristi kontinualno dodjeljivanje memorije, i logički i fizički adresni prostor procesa se sastoje od kontinualnog niza memorijskih adresa. Prosto rečeno, svaki proces dobija jedan kontinualni dio memorije. Metode kontinualnog dodjeljivanja memorije su:

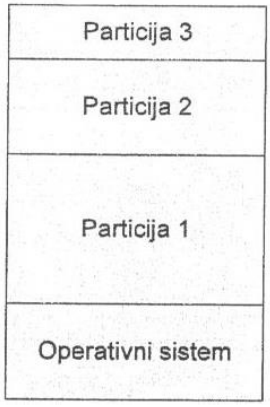
- Multiprogramiranje sa fiksnim praticijama
- Multiprogramiranje sa particijama promjenljive veličine

Uvođenje particija je jednostavan metod koji omogućava višeprogramski rad, tj. da više programa koji se izvršavaju bude u isto vrijeme u operativnoj memoriji.

Multiprogramiranje i statičke particije

statičke particije-unaprijed nepromjenjvo izdjeljene

Upravljanje memorijom pomoću statičkih particija je jedan od najosnovnijih načina upravljanja memorijom. Za višeprogramski rad neophodna je dodjela više particija. Memorija se dijeli u particije fiksne veličine, kao što je to pokazano na slici 1.



Slika 1. Statičke particije

Particije su fiksirane u vrijeme inicijalizacije sistema i ne mogu se mijenjati u toku izvršavanja programa.
Svakoj particiji se dodjeljuje po jedan proces.
Programi se raspoređuju tako da budu smješteni u najmanjoj particiji koja je dovoljno velika da prihvati cio program. Izvršni program koji je pripremljen za izvršavanje u datoj particiji ne može da se izvršava u nekoj drugoj particiji bez ponovnog povezivanja (*relinking*).
Kod ovog načina upravljanja memorijom postoji potreba zaštite koda operativnog sistema od mogućih promjena od strane korisničkih procesa. Zaštita se može uraditi pomoću baznog i graničnog registra.

Jedan od najstarijih i najprostijih metoda dodjeljivanja memorije je ovakva podjela cjele fizičke memorije na više dijelova fiksne veličine, pri čemu se u jednom dijelu može naći samo jedan proces. **U ovakvoj organizaciji, stepen multiprogramiranja je jednak broju memorijskih particija.** Ova metoda je korišćena u sistemu IBM OS/360.

Cijela memorija se izdijeli na više dijelova. Svi procesi se stavljaju u red čekanja (input queue) koji može biti jedinstven za cjeli operativni sistem ili poseban za svaku particiju.
Višestruki redovi čekanja obično se formiraju za opsege veličina Q=1 KB, Q=2 KB, Q=4 KB.
Ukoliko za proces koji je došao na red nema dovoljno memorije, uzima se sledeći manji proces iz liste.
Kada postoji više redova čekanja, veći broj malih procesa može čekati u redu za male particije, dok su velike particije neiskorišćene. U tom slučaju ima dovoljno memorije, ali se ne koristi. Bolji je jedinstveni red čekanja, jer ako nema mjesta u memoriji u redu čekanja za particiju koja odgovara veličini procesa, procesu se dodjeljuje veća particija.
Dva procesa ne mogu biti smještena u jednoj particiji.

Multiprogramiranje sa fiksnim particijama korišćeno je u sistemima s grupnom obradom (batch).
Ova metoda je nepogodna za interaktivne sisteme zbog postojanja većeg broja procesa promjenljive veličine (obično malih) koji se pojavljuju po slučajnom rasporedu.

Ova metoda se više ne koristi.

Diskontinualno dodjeljivanje memorije- *dinamička alokacija straničenjem*

Umesto fiksnih particija, memorija se dijeli dinamički, a svaka šupljina ***(hole),*** tj slobodan kontinualni dio memorije, može se iskoristiti za smještanje procesa - pod uslovom da je dovoljno velika. Šupljine dinamički nastaju i nestaju, zajedno sa procesima, mogu biti bilo gdje u memoriji i imati bilo koju veličinu, što odgovara procesima na interaktivnim sistemima.
Kada proces naiđe u sistem, traži se šupljina dovoljno velika za proces. Sav prostor koji proces ne zauzme od cijele šupljine, predstavlja novu šupljinu u koju se može smestiti novi proces.

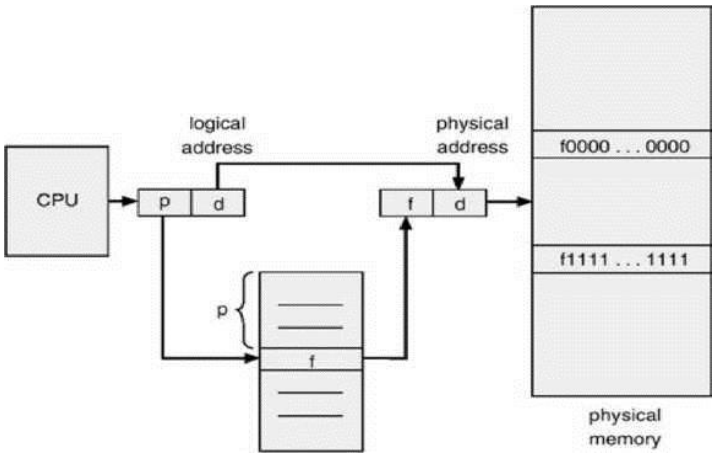
Alokacija memorije je dinamička - **memorija se sastoji od procesa i šupljina**, a **OS dinamički vodi evidenciju** o zauzetosti memorije na jedan od sledećih načina:

- Bit mape ***(bit maps)***
- Povezane liste ***(linked lists)***
- Sistem udruženih parova - drugova ***(buddy system)***

Koriste se dvije metode:
straničenje *(paging)*
segmentacija (segmentation), kao i njihove kombinacije ***straničenje sa segmentacijom*** i ***segmentacija sa straničenjem.***

Straničenje

Straničenje je metoda sa hardverskom podrškom na nivou procesora koja se koristi u svim operativnim sistemima i na svim računarskim arhitekturama. Gotovo da ne postoji nijedan savremeni procesor koji hardverski ne podržava straničenje.
Fizička memorija, tj fizički adresni prostor, podijeli se na blokove fiksne veličine, koji se **nazivaju fizičke stranice ili okviri *(page frames).***
Logički adresni prostor takođe se podijeli na blokove istih veličina koji se nazivaju **logičke stranice *(pages*).**
U daljem tekstu, pod terminom stranice podrazumevaćemo logički stranicu, a pod terminom okvir - fizičku. Veličine stranica su po pravilu stepen broja 2, najčešće u opsegu od 512 B do 8 KB, mada mogu biti i veće, 16 MB.
Swap prostor na disku takođe se dijeli na stranice koje po veličini odgovaraju memorijskim stranicama.



Straničenje

Metoda straničenja funkcioniše na sledeći način: svakoj logičkoj stranici odgovara jedna fizička, a korespodencija između njih se čuva u tabeli stranica ***(page table).*** To omogućava da se kontinualni logički prostor procesa razbaca svuda po memoriji.
Svaka logička adresa koju generiše procesor dijeli se na dva dijela:

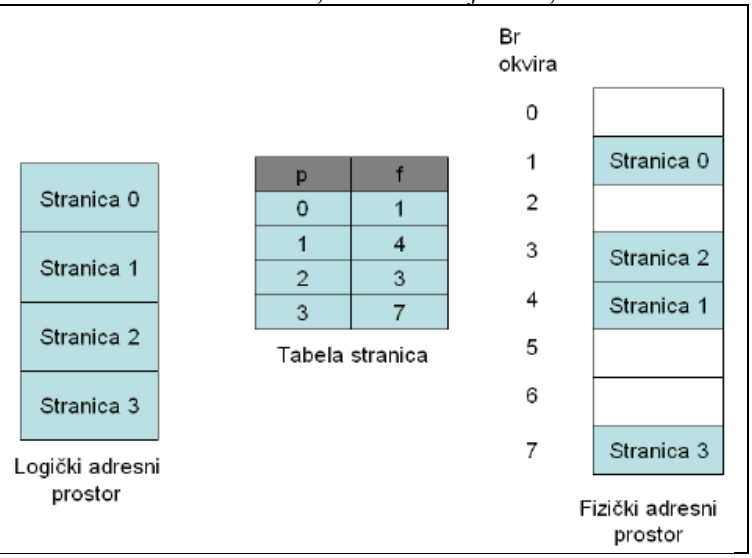
- broj stranice (p, ***page number)*** - koristi se kao indeks u tabeli stranica koja sadrži baznu adresu okvira. Bazna adresa predstvlja viši dio adrese
- pomjeraj unutar stranice (d, ***page offset)*** - definiše položaj u odnosu na samu stranicu i - u kombinaciji sa baznom adresom (čisto sabiranje) - definiše punu fizičku adresu koja se šalje memorijskoj jedinici.

Naglasimo da je osnovna adresabilna jedinica bajt i da je pomjeraj isti i za logičku i za fizičku adresu
Ako je veličina logičkog adresnog prostora 2^m, a veličina stranice 2^n, tada viši dio adrese dužine m-n definiše broj stranice, dok najnižih n bitova adrese predstavljaju pomjeraj unutar stranice.

Primjer straničenja:
Zamislimo da imamo memoriju veličine 32 B. Definišemo 8 okvira veličine 4 B, što znači da je m=6, a n=2.

Uzmimo korisnički proces koji zauzima 4 logičke stranice, sa logičkim adresama:
- stranica 0 (0-3)
- stranica 1 (4-7)
- stranica 2 (8-11)
- stranica 3 (12-15)

Logička adresa 0 ima broj logičke stranice 0. Pomoću broja logičke stranice ulazimo u tabelu stranica i nalazimo da je ona smještena u okvir broj 1, i tako redom:
logička stranica 1 smještena je u okvir 4,
logička stranica 2 u okvir 3 i
logička stranica 3 u okvir 7.



Proces koji ulazi u stanje izvršavanja mora da dobije potrebnu memoriju, tako da se za dati proces preračunava koliko mu stranica memorije treba. Svaka stranica mora da se mapira u okvir. Ako proces zahteva n stranica, tada se alocira n okvira, koji se pune procesom, pri čemu se mapiranje stranica - okvir upisuje u tabelu stranica.

Svako straničenje predstavlja dinamičku relokaciju, a tabela stranica predstavlja relokacioni registar za svaki okvir fizičke memorije. Smanjenje stranica dovodi do povećanja njihove tabele, jer je svaka stranica opisana jednim zapisom u tabeli. Kao posljedica, povećava se kašnjenje peilikom mapiranja ili pretraživanja. U slučaju korišćenja ***swap*** tehnike sa stranicama, poželjnije su veće stranice zato što je rad sa diskovima efikasniji kada su transferi veći.

Značajan aspekt straničenja je jasno razdvajanje korisničkog pogleda na memoriju i aktuelne fizičke memorije. Korisnik svoj dio memorije doživljava kao kontinualni prostor iako su stranice raznih procesa razbacane po memoriji.

Mapiranje logičkog i fizičkog prostora zadatak je OS i korisnik ga ne vidi.
Operativni sistem mora da prati koji su okviri slobodni, a koji su dodjeljeni i to kom procesu. Sve se to čuva u jezgru, u tabeli okvira ***(frame table)*** u kojoj je svaki okvir opisan jednim zapisom.
Operativni sistem mora i da za svaki proces generiše tabelu stranica koja se odnosi samo na njegove stranice i koja definiše mapiranje za taj proces.

Segmentacija memorije

Prethodno razmatrani slučajevi odnose se na korisničke procese, tj programe, koji zahtevaju kontinualan memorijski prostor.

Sama struktura programa nije kontinualna, jer programi se logički dijele na više nezavisnih cjelina.
Npr, sam kod se sastoji od više cjelina kao što su tabele, polja, stek, promjenljive, itd. Svaki od ovih logičkih segmenata dobija ime pomoću koga se referencira i može nezavisno da se učita u memoriji. Po pravilu, programer definiše logičke segmente u izvornom programu, a prevodilac na osnovu toga pravi memorijske segmente.

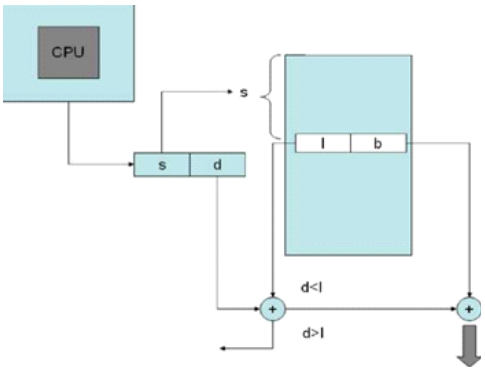
Segmentacija je metoda upravljanja memorijom koja podržava logički korisnički pogled na memoriju. Logički adresni prostor sastoji se od kolekcije segmenata, a svaki segment ima jedinstveno ime i dužinu.

Logička adresa se sastoji od dva dijela:
- imena segmenta (umesto imena segmenta obično se zadaje broj koji predstavlja identifikator segmenta)
- pomjeraja unutar segmenta

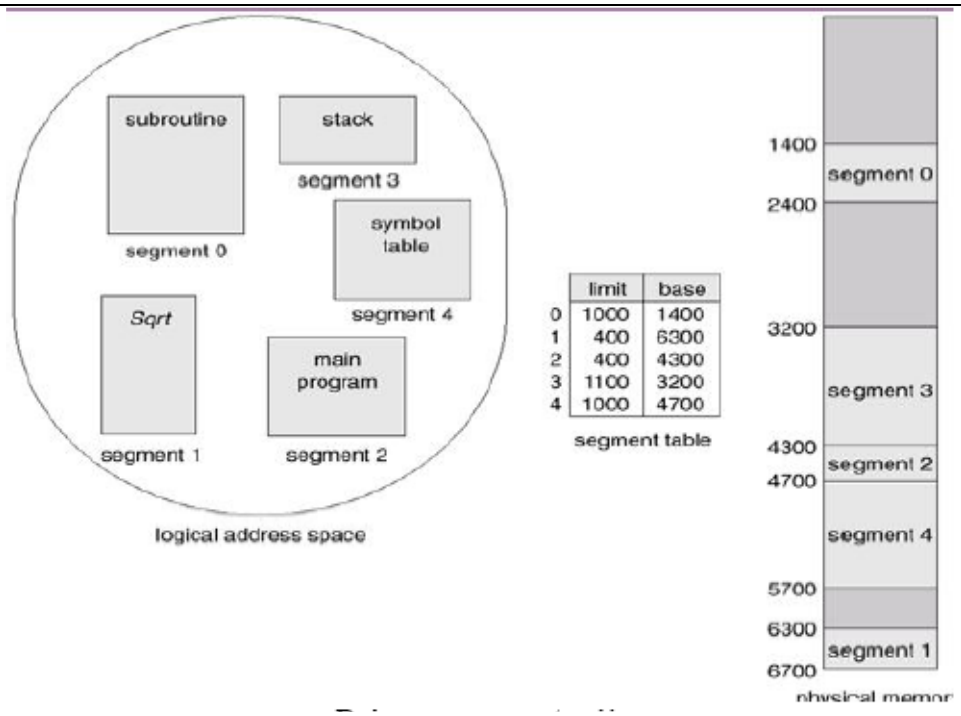
Kod straničenju logička adresa jednodimenzionalna, pri čemu **razdvajanje dijelova adrese i transliranje obavlja hardver.**

Pri segmentaciji, kao i pri multiprogramiranju s particijama promjenljive dužine, javlja se eksterna fragmentacija. Slobodna memorija se ne može iskoristiti za smještaj segmenata ukoliko ne postoji dovoljno velika šupljina, bez obzira na količinu slobodne memorije. Problem eksterne fragmentacije, koja zavisi od veličine segmenata i raspodjele nailaska procesa, može se smanjiti sažimanjem memorije.

Mapiranje se obavlja preko tabele segmenata a ima podršku i u hardveru mikroprocesora.



Na slici je prikazan slučaj segmentacije za 5 segmenata, čije su definicije date u tabeli segmenata.



Segmentacija sa straničenjem

Najpopularnije serije procesora Intel (80x86 i Pentium) i Motorola (68000) imaju ugrađenu podršku i za segmentaciju i za straničenje, tako da omogućavaju primenu kombinovanih metoda diskontinualne alokacije pa se procesi mogu dijeliti na fizički diskontinualne logičke celine.
Pri tome, straničenje poništava eksternu fragmentaciju segmenata.

Procesor Intel 80386 koristi segmentacije sa straničenjem. Logička adresa se sastoji od identifikatora segmenta ***(selector)*** i pomjeraja u okviru segmenta ***(offest).*** Iz tabele segmenata čita se adresa logičke stranice u katalogu stranica. Straničenje je realizovano u dva nivoa:

- spoljna tabela se zove katalog stranica ***(page directory)***
- unutrašnja - tabela stranica

Fragmentacija memorije

Fragmentacija se odnosi na neiskorišćenu memoriju koju sistem za upravljanje memorijom ne može da dodijeli procesima.
Uz nju je povezan i sljedeći problem. Neka postoji više (m) slobodnih segmenata u koji se može upisati sljedeći proces. Postavlja se pitanje koji od njih odabrati. Moguća su sljedeća rješenja:

- Prvi koji zadovoljava (**first-fit**): Procesu se dodjeljuje prvi segment koji zadovoljava postavljene memorijske zahtjeve. Obično pretraživanje počinje od početka liste slobodnih segmenata ili se nastavlja od mjesta gdje je prethodno ispitivanje zaustavljeno.
- Najbolje poklapanje (**best-fit**): Procesu se dodjeljuje onaj segment koji nabolje odgovara njegovim memorijskim zahtjevima. Iako na prvi pogled ovim pristupom se najbolje iskorištava slobodna memorija, rezultat je stvaranje malih segmenata, šupljina, koji su posljedica razlike veličine segmenta i programa.
- Najlošije poklapanje (**worst-fit**): Procesu se dodjeljuje najveći slobodni segment. Ovaj algoritam ima za cilj stvaranje što većih šupljina, suprotno prethodnom algoritmu.

Provedene simulacije pokazale su da su prva dva algoritma bolja od posljednjeg u smislu bolje iskoristivosti memorije kao i prosječnog vremena izvođenja procesa.
Postoje dva tipa fragmentacije: interna i eksterna.

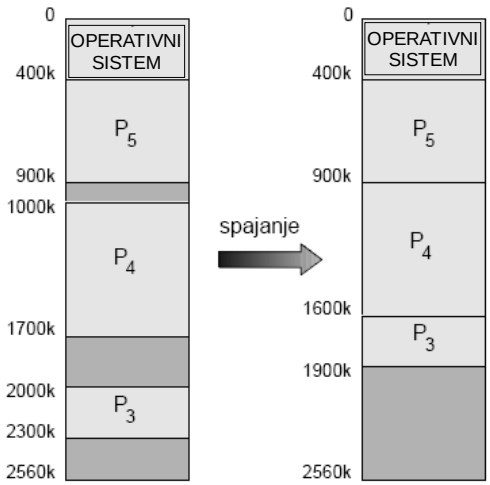
Interna fragmentacija je dio memorije unutar regiona ili stranice koja je dodjeljena datom procesu i ne koristi se od strane tog procesa. Interna fragmentacija je prouzrokovana različitom veličinom dodjeljene memorije i programa koji je učitan u taj dio memorije. Taj dio memorije nije raspoloživ za korišćenje drugim procesima sistema sve dok dati proces ne završi sa radom ili ne oslobodi dodjeljenu memoriju. Interna fragmentacija ne postoji kod upravljanja memorijom pomoću dinamičkih particija, kod statičkih segmenata i kod dinamičkih segmenata.

Eksterna fragmentacija je neiskorišćena memorija između particija ili segmenata. Ova memorija nije kontinualna, već se sastoji iz više manjih dijelova. Eksterna fragmentacija ne postoji kod upravljanja memorijom pomoću statičkih i pomoću dinamičkih stranica.

Za prevazilaženje problema eksterne fragmentacije koristi se tehnika sažimanja ili kompakcije (*compaction*).

Kompakcija se izvršava u tri faze.

1. Prvo se određuje nova lokacija za svaki blok koji se premješta.
2. Zatim se ažuriraju svi pokazivači na taj blok u skladu sa novom lokacijom.
3. U trećoj fazi se podaci premještaju na novu lokaciju



Primjer *spajanja memorije, premještanjem procesa P3 i P4*

Zaštita memorije

Kada program šalje podatke na pogrešne adrese ili ih smješta na nedozvoljene lokacije dolazi do zastoja ili čak pada kompletnog sistema. Jedan od osnovnih zadataka sistema za upravljanje memorijom je zaštita procesa koji dijele memoriju.
Ako nekoliko procesa dijeli memoriju, procesima se ne smije dozvoliti da mjenjaju lokacije koje nisu njima dodjeljene. Ovim se ostvaruje određeni oblik **privatnosti procesa**.

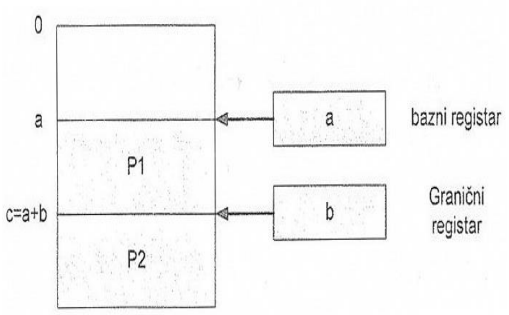
Važna karakteristika sistema sa straničenjem je da svaki proces može da pristupi samo onim okvirima koji se pojavljuju u njegovoj tabeli stranica.

Zaštita memorije se može implementirati hardverski i softverski.

Hardversko rješenje je preslikavanje adresa (*address translation*) i ovo rješenje je opisano u nastavku teksta.
Hardverska zaštita memorije se može ostvariti korišćenjem dva registra koji se zovu: bazni (*base*) i granični (*limit*) registar.

U baznom registru se nalazi najmanja adresa fizičke memorije gdje je dati program smješten, dok se u graničnom registru nalazi veličina opsega memorije unutar koga program može pristupati, kao što je to prikazano na slici.

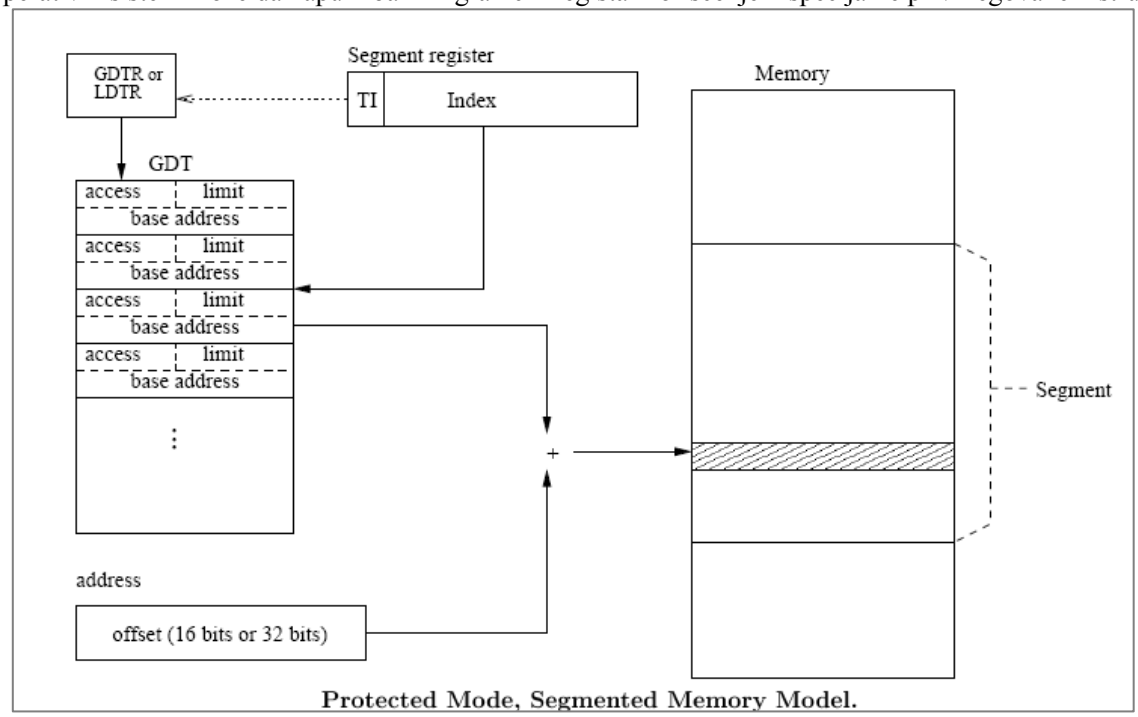
- Prvo se provjerava da li je generisana adresa veća ili jednaka vrijednosti koja se nalazi u baznom registru.
- Ako je ovaj uslov ispunjen tada se provjerava da li je generisana adresa manja od zbira vrijednosti u baznom i graničnom registru.
- Ako je i ovaj uslov ispunjen dozvoljava se pristup fizičkoj memoriji.



Logički adresni prostor

Ovakvom hardverskom zaštitom eleminisana je mogućnost da neki korisnički program nakon adresiranja pristupi dijelu memorije koji je izvan njegovog dozvoljenog adresnog prostora. Ako je generisana adresa datog programa van dozvoljenog adresnog prostora tada će doći do sistemske greške, tj. **do takozvane fatalne greške**.

Jedino operativni sistem može da napuni bazni i granični registar korišćenjem specijalne priviliegovane instrukcije.

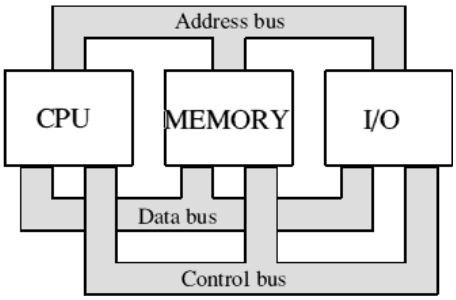


Protected Mode, Segmented Memory Model.

Operativni sistem takođe, sprečava programe da mijenjaju sadržaj ovih registara.

Upravljanje uređajima *U/I processing&control*

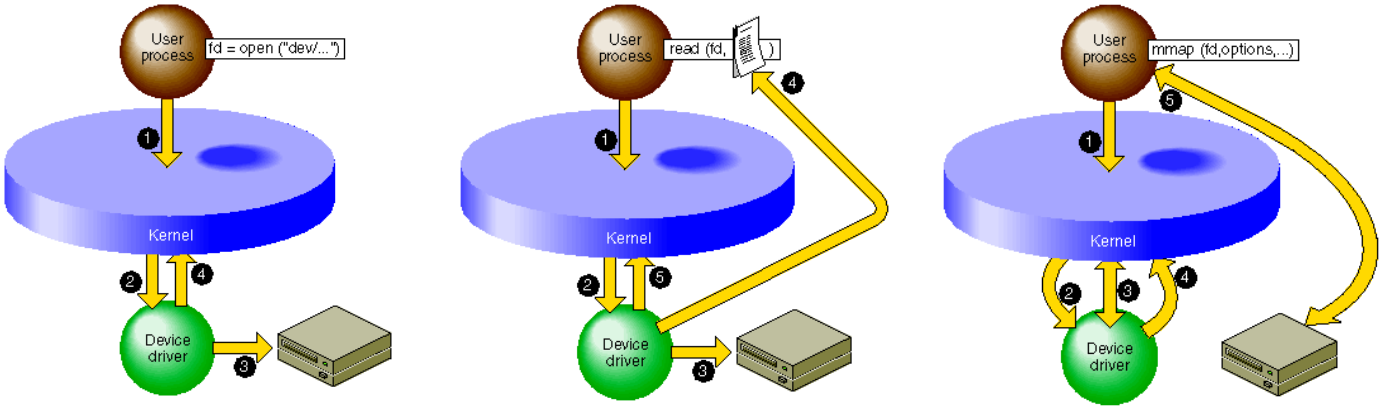
Ulazni i izlazni uređaji su veoma važni za efikasno korišćenje računarskog sistema. Oni predstavljaju vezu ljudi sa računarom. Ulazni uređaji prikupljaju podatke iz okoline i prevode ih u oblik pogodan za obradu u računarskom sistemu. Izlazni uređaji preuzimaju podatke dobijene obradom i prosljeđuju ih na dalju obradu ili prikazuju u obliku upotrebljivom za ljude.



Računari rade sa velikim brojem različitih uređaja.
To su: uređaji za memorisanje (npr. diskovi, trake), uređaji za prenos (npr. modemi), uređaji koji omogućuju interfejs sa ljudima (npr. tastatura, monitor, štampač) i drugi specijelizovani uređaji.

Svaki od ovih uređaja treba da ima mogućnost da pošalje ili primi podatke iz računarskog sistema.

Spoljašnji uređaji se priključuju na računar preko U/I modula.

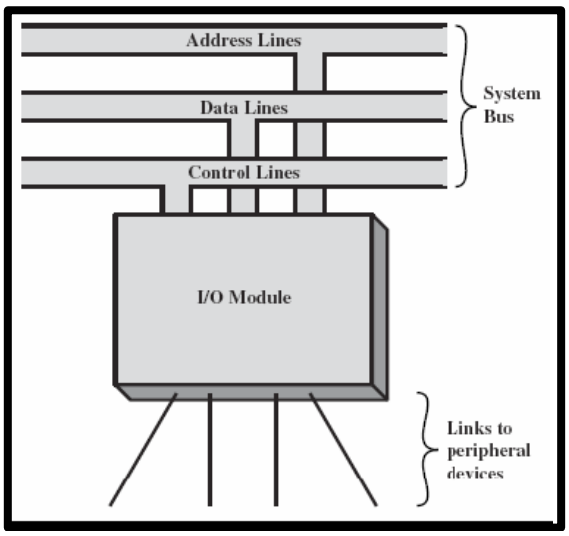


Različiti modeli realizacije pristupa U/I uređajima i njiov odnos prema OS (kernelu)

U/I moduli

Spoljašnji uređaji računarskog sistema se još nazivaju i periferni uređaji ili periferali.
U većini računarskih sistema procesor ne kontroliše direktno periferne uređaje, nego su oni priključeni na uređaj koji se naziva U/I modul. U/I modul komunicira direktno sa procesorom i procesor preko njega obavlja svu potrebnu komunikaciju sa U/I uređajima.
Periferni uređaji su znatno sporiji od procesora tako da bi njihova direktna komunikacija sa procesorom i memorijom znatno usporila cio sistem. Osim toga, svaki periferni uređaj radi u skladu sa određenim pravilima. Nije isplativo uključivati u procesor logiku rada velikog broja različitih perifernih uređaja. Sa jedne strane bi to znatno usporilo operacije procesora, dok bi sa druge strane onemogućilo dodavanje novih tipova U/I uređaja.

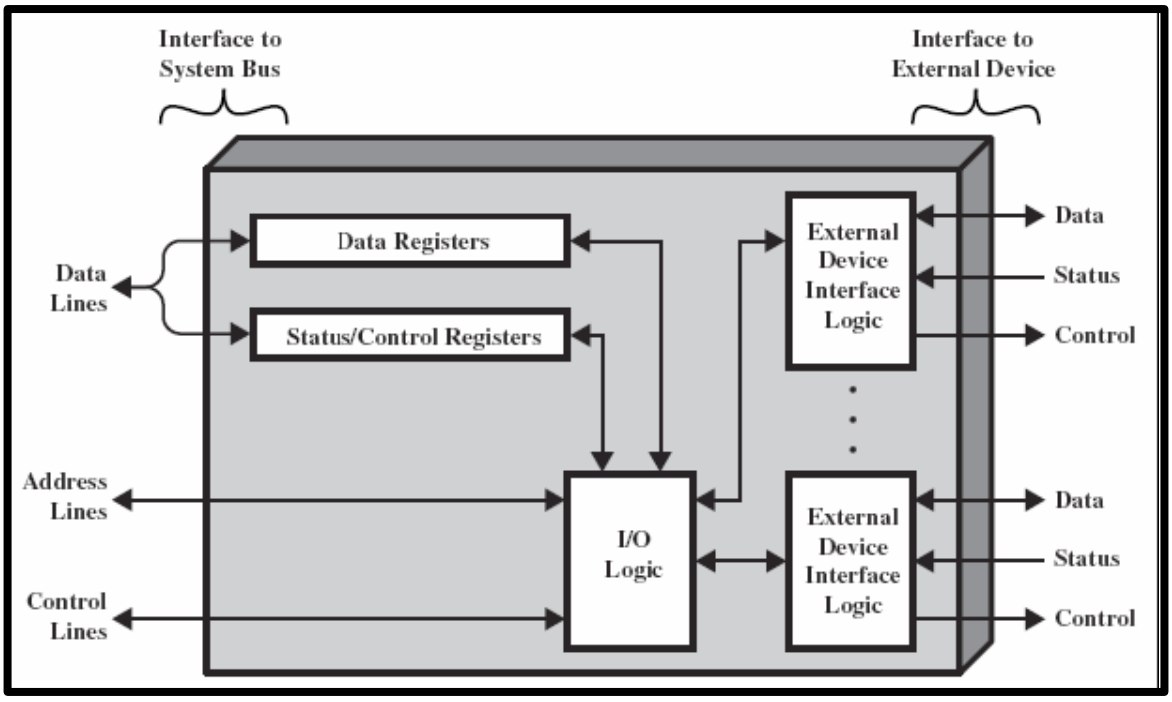
U/I modul je direktno priključen na sistemsku magistralu i obezbjeđuje razmjenu informacija sa U/I uređajima i kontrolu njihovog rada na način koji najmanje utiče na performanse računarskog sistema.



Glavne funkcije U/I modula su:

- Kontrola i usklađivanje saobraćaja
- Komunikacija sa procesorom
- Komunikacija sa uređajima
- Prihvatanje podataka
- Otkrivanje grešaka

Blok dijagram U/I modula



Blok diagram U/I modula

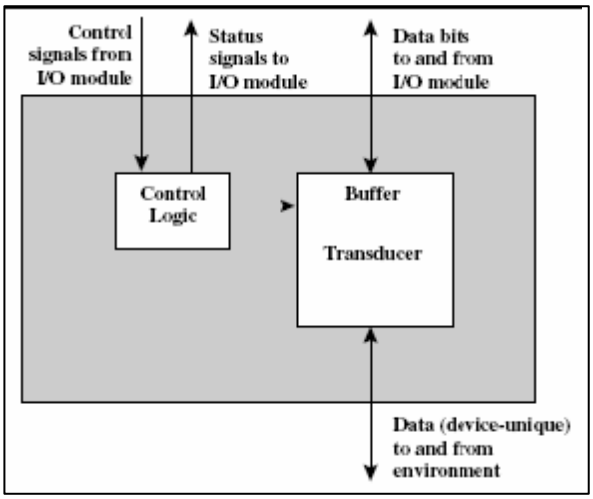
Tehnike izvršavanja U/I operacija

Programirani U/I model

Najjednostavniji način izvršavanja U/I operacija je programirani U/I (Programmed I/O: **PIO**), poznata i kao metoda čekanja ((busy and wait).
Kada procesor izvršava program i naiđe na zahtjev za U/I operacijom, on određuje potrebne adrese i šalje komandu odgovarajućem U/I modulu.

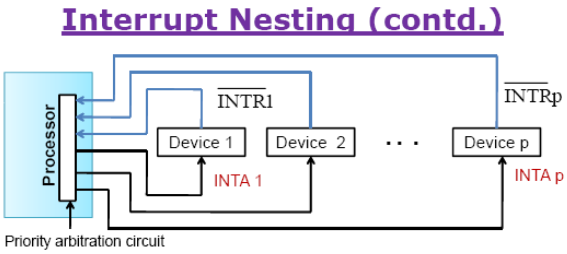
Postoje četiri tipa komandi koje procesor može da pošalje U/I modulu:

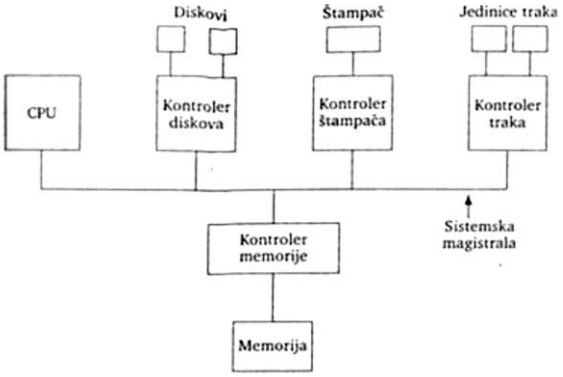
- **Kontrolna komanda** koja se koristi za aktiviranje perifernog uređaja i kojom se naznačava akcija koju treba preduzeti.
- **Test komanda** kojom se ispituje stanje U/I modula i odgovarajućih perifernih uređaja.
- **Komanda za čitanje** izdaje direktivu U/I modulu da pročita podatak iz perifernog uređaja i smjesti ga u interni bafer.
- **Komanda za pisanje** izdaje direktivu U/I modulu da pročita podatak sa magistrale i prenese ga u periferni uređaj.



Prekidina tehnika U/I – Interapt U/I model

Prekidima upravljan U/I se primjenjuje na skoro svim računarskim sistemima, bez obzira na njihovu veličinu. Izuzetak mogu da budu jedino računarski sistemi sa vrlo ograničenom funkcijom.
Primjer obavljanja U/I operacije sa perifernim uređajema (diskovima, štampačem...) uz korišćenje prekida je prikazan na slikama dole.





Razlike između programiranog i prekidnog modela pristupa U/I uređajima

Suštinski razlka između ova dva pristupa nema.
Razlika je u načinu pokretanju procedure pristupa uređaju. Kod programskog modela zahtjev je implementiran unutar samog programa (procesa), a kod interapt modela zahtjev za komunikacijom sa U/I uređajem dolazi izvana.

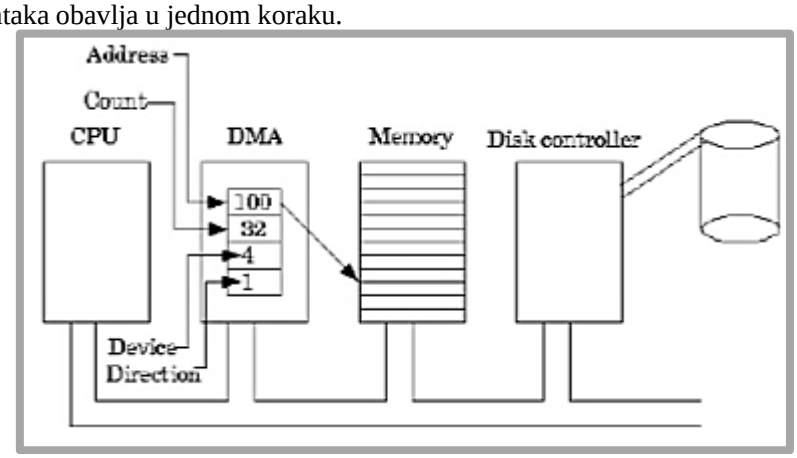
Kod programiranog U/I prenosa vrši se razmjena podataka između CPU-a i U/I interfejsa. CPU izvršava program koji upravlja U/I operacijom, kao što je čitanje statusa spoljnjeg uređaja, izdaje komande READ ili WRITE i vrši prenos podataka. Kada procesor izda komandu U/I uređaju on zatim čeka sve dok se U/I operacija ne završi. Kako je CPU brži od U/I interfejsa, evidentno je da procesor suviše mnogo vremena gubi na testiranje spremnosti za prenos.

Osnovni mehanizam koji se koristi kod prekidne U/I je isti kao i kod programirane U/I. Kod prekidnog U/I-a prenos podataka se inicira od strane U/I uređaja, koji koristi prekidni mehanizam da ukaže CPU-u na svoju spremnost. Na ovaj način ne postoji više potreba za permanentnim testiranjem statusa U/I uređaja. U/I uređaj može takođe koristiti mehanizam prekida kada želi da CPU obrati njemu pažnju iz drugih razloga, kao što su: očekuje se (javila se) određena greška, ukazuje na/ završetak lokalne operacije i dr.

Moglo bi se reći da je razlika istorijska, i da se odnosi na vrijeme kad se proces izvršenja programa stvarno prekidao. Kod modernih OS-a i programa za obradu U/I zahtjeva procesi se obavljaju paralelno pa se programski i interapt pristup ne razlikuju bitno.

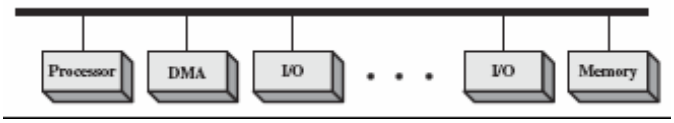
Direktan pristup –DMA model pristupa U/I uređajima

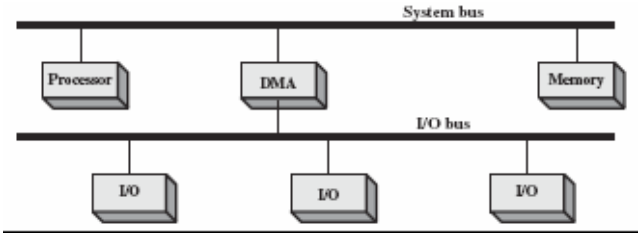
Nedostaci oba prethodno opisana načina izvršavanja U/I operacija su što zahijevaju intervenciju procesora pri prenosu podataka između memorije i U/I modula. Prenos podatak se odvija preko procesora i brzina prenosa je ograničena brzinom kojom procesor može da testira i servisira U/I modul. Pri izvršavanju svake U/I operacije procesor mora da izvrši veliki broj instrukcija zbog čega ostali programi moraju da čekaju. Ovi nedostaci posebno dolaze do izražaja pri prenosu velikih količina podataka.
U tom slučaju je efiksnije primjeniti tehniku prenosa nazvanu direktan pristup memoriji (*Direct memory access*, *DMA*), kod koje se prenospodataka obavlja u jednom koraku.

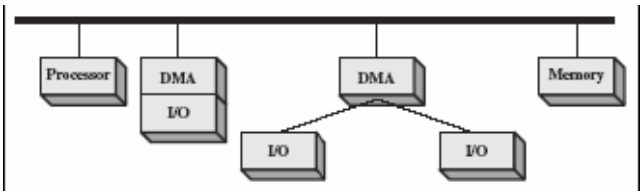


Direktan pristup memoriji zahtijeva dodatni modul priključen na sistemsku magistralu koji se *DMA* kontroler. ***DMA* kontroler predstavlja specijalizovani procesor koji može da izvršava programirani U/I.**
Kod PC-a DMA prenos obavlja se posredstvom posebnog mikrokontrolera specijalizovanog za prenos blokova podataka. Svaki od DMA kontrolera ima po četiri kanala – četiri nezavisne cjeline koje omogućavaju zasebne direktne prenose podataka. Svaki od kanala može se programirati da u datom trenutku obavi prenos bloka podataka.
Moguća su različiti tipovi DMA prenosa:

- memorija -> I/O port,
- I/O port -> memorija i
- memorija -> memorija







Da bi se izvršio prenos potrebno je defnisati tip prenosa, početnu (ili krajnju) adresu u memoriji gdje se nalazepodaci i broj bajtova koje treba prenijeti.
Kada procesor treba da izvrši U/I operaciju, on upiše *DMA* kontrolni blok u memoriju. Kontrolni blok sadrži adresu uređaja sa koga se vrši prenos ili na koji treba upisati podatke.
Zatim procesor predaje adresu *DMA* kontrolnog bloka *DMA* kontroleru i prelazi na druge poslove.
DMA kontroler nastavlja izvršavanje operacije i prenosi jedan po jedan bajt radeći direktno sa magistralom bez pomoći centralnog procesora.
DMA dobija kontrolu nad magistralom samo kada ona nije zauzeta od strane centralnog procesora. Magistrala može da bude slobodna kada je procesor ne koristi ili kada *DMA* kontroler zahtijeva od procesora da privremeno suspenduje

svoje operacije sa magistralom. Ovakav način rada *DMA* kontrolera se naziva krađa ciklusa, jer u suštini *DMA* krade cikluse na magistrali od centralnog procesora.
Po završetku operacije, *DMA* šalje prekid procesoru kojim ga obavještava da je operacija izvršena. Na taj način, **procesoru se šalje samo jedan prekid, bez obzira na količinu prenesenih podataka.**

Zaštita operativnih sistema

Problem zaštite postoji kod svih računarskih sistema, a posebno kod sistema koji su povezani na Internet. Pitanje zaštite je postalo jedno od najvažnijih pitanja u svakom poslovnom sistemu. Operativni sistem ima značajnu ulogu u riješavanju problema zaštite.
Apsolutna zaštita računarskih sistema se ne može ostvariti. Osnovni cilj je obezbjediti visok nivo zaštite i zato pristup riješavanju problema zaštite mora biti sveobuhvatan sa stalnim razvijanjem novih mehanizama zaštite u skladu sa bezbjednosnim problemima koji nastaju. Kod savremenih računarskih sistema primjenjuje se sistem zaštite na više nivoa, tj. zaštita se primjenjuje:

1. na nivou mreže,
2. na nivou operativnog sistema,
3. na nivou aplikacije,
4. na nivou baze podataka,
5. kao proceduralna zaštita.

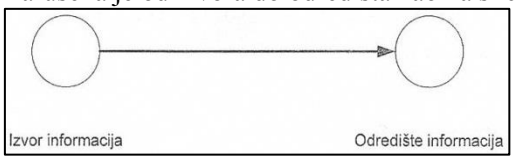
Osnovna potreba za zaštitom u okviru operativnog sistema nastaje zbog dijeljenja resursa kao što su memorija, U/I uređaji, programi i podaci. Operativni sistem može da obezbjedi sljedeće načine zaštite:
Bez zaštite – kada se dijelovi koda sa kritičnim sekcijama izvršavaju u različito vrijeme;
Izolaciju – kada se svaki proces izvršava nezavisno od drugih procesa bez dijeljenja resursa i bez međusobne komunikacije. Svaki proces ima svoj adresni prostor, datoteke i druge objekte;
Sve je djeljivo ili nema dijeljenja resursa – vlasnik objekta deklariše objekat kao javni ili privatni. Ako je objekat javni svako mu može pristupiti, ako je privatni može mu pristupiti samo vlasnik;
Dijeljenje preko ograničenja pristupa – kada operativni sistem obezbjeđuje da samo autorizovani korisnik može da pristupi datom objektu. U ovom slučaju operativni sistem kod svakog pristupa datog korisnika nekom objektu provjerava dozvolu pristupa;
Dijeljenje preko dinamičkih sposobnosti – kada se koncept kontrole pristupa proširuje tako da omogući dinamičko kreiranje prava za dijeljenje objekata;
Ograničeno korišćenje objekata – kada se ograničava ne samo pristup datom objektu, već operacije koje se mogu vršiti nad objektom.
Svaki višekorisnički operativni sistem mora da obezbjedi zaštitu od neautorizovanog pristupa jednog korisnika resursima drugog korisnika. Sistem lozinki koji se uglavnom primjenjuje nije potpuno siguran. Korisnici obično biraju lozinke koje se lako pogađaju ili biraju složenije, ali ih zapisuju i ostavljaju na vidljivim mjestima. Narušavanjem sistema lozinki dolazi se do neautorizovanog pristupa resursima datog sistema.
Zahtjevi sistema zaštite savremenih sistema
Kod današnjih sistema koji se zasnivaju na komunikaciji korisničkog procesa i servisa koji obezbjeđuju neku vrstu usluge ili obrade podataka postoje sljedeći zahtjevi sistema zaštite:

1) Međusobna autentikacija,
2) Kontrola pristupa ili autorizacija,
3) Zaštićena komunikacija,
4) Neporicanje slanja, odnosno prijema podataka,
5) Ne ponavljanje slanja,
6) Nema odbijanja servisa.

Međusobnom autentikacijom se obezbjeđuje verifikacija identiteta obe strane koje učestvuju u komunikaciji. Tek nakon završene međusobne autentikacije se može nastaviti dalja komunikacija. Kontrolom pristupa se obezbjeđuje da samo autorizovani korisnici mogu pristupiti traženim podacima. U suprotnom, neautorizovani korisnik bi mogao da naruši integritet podataka, tako što bi mogao da ih mjenja. Zaštićena komunikacija garantuje tajnost podataka koji se prenose preko komunikacionog kanala. Neporicanje ima značenje da ni jedna strana koja učestvuje u komunikaciji ne može da poriče slanje, odnosno prijem podataka prenijetih u toku procesa komunikacije. Ne ponavljanje slanja obezbjeđuje sistem od mogućnosti da treća strana kopira cijelu ili neki dio poslate poruke i nakon toga vrši ponovo slanje tih istih podataka. Zahtjev da nema odbijanja servisa obezbjeđuje da nema degradacije performansi datog sistema i garantovanje legitimnim korisnicima sistema da mogu da koriste potreban servis.

Vrste napada

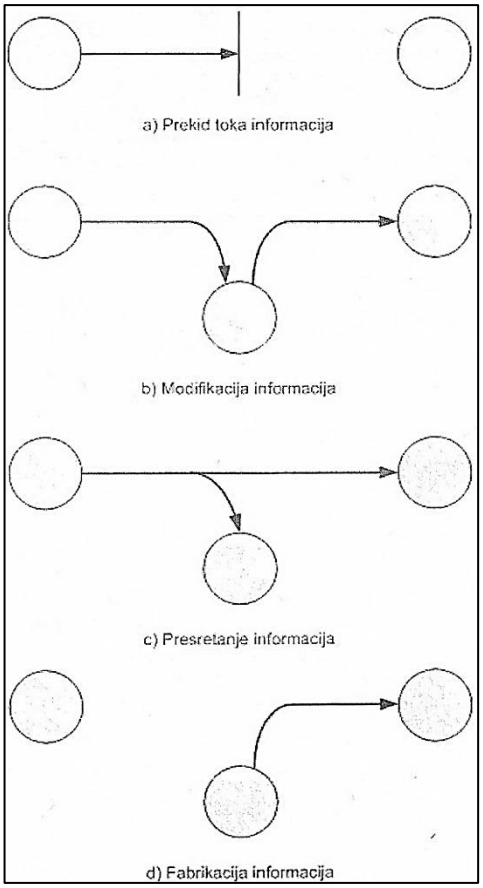
Dati računarski sistem ili dio mreže možemo posmatrati sa aspekta obezbjeđivanja informacije. Uobičajen tok informacija kada zaštita sistema nije narušena je od izvora do odredišta kao na slici 1.



Slika 1. Uobičajen tok informacija kada zaštita sistema nije narušena

Kada je zaštita sistema narušena, tada postoje sljedeće mogućnosti:

- prekid toka informacija,
- modifikacija informacija,
- presretanje informacija,
- fabrikacija informacija (slika 2).



Slika 2 Tok informacija kada je zaštita sistema narušena.

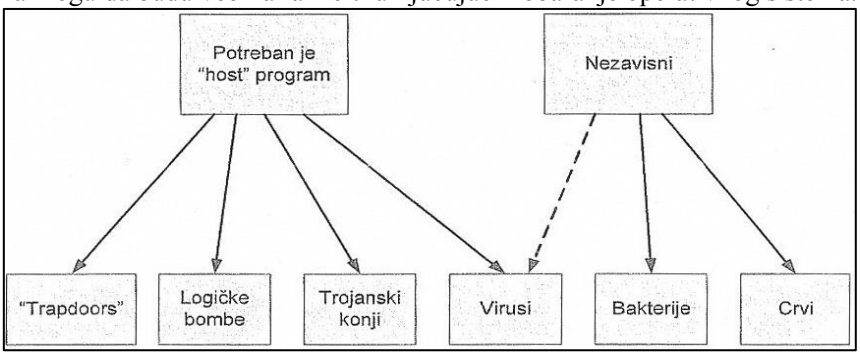
Napadi koji prouzrokuju prekid toka informacija se nazivaju napadi na raspoloživost sistema. Primjeri takvih napada su: prekid komunikacione linije, uništenje neke hardverske komponenete ili nedostupnost datog sistema za upravljanje datotekama.

Napadi koji imaju za cilj modifikaciju informacija pripadaju klasi napada na integritet podataka. Primjeri takvih napada su: modifikacija sadržaja elektronske pošte, modifikacija sadržaja date datoteke, modifikacija elektronskih dokumenata koji se prenose preko mreže.

Treća vrsta napada su napadi na tajnost ili povjerljivost podataka. Oni su prouzrokovani presretanjem informacija koje se prenose, tako što neautorizovani subjekt dobija pristup tim informacijama. Primjeri takvih napada su: korišćenje sniffer programa, nedozvoljeno kopiranje datoteka ili programa koji se prenose.

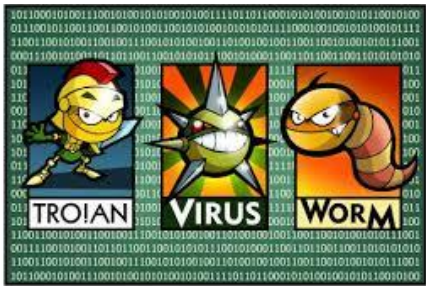
Četvrta vrsta napada su napadi na autentičnost, koji su prouzrokovani fabrikacijom informacija. Nastaju tako što neautorizovani subjekt generiše falsifikovane ili lažne informacije unutar datog sistema. Primjeri napada na autentičnost su: dodavanje sadržaja unutar nekog elektronskog dokumenta, generisanje nepostojeće elektronske pošte ili ponovno slanje pošte poslatih u nekom prethodnom periodu

Napadi na računarske sisteme se mogu takođe klasifikovati u dvije grupe: pasivni i aktivni. Kod pasivnih napada nema promjene podataka, dok kod aktivnih može biti promjene podataka, generisanje novih podataka, može se vršiti ponovno slanje i može se primjeniti masovno slanje podataka čime se ugrožava raspoloživost napadnutog sistema. Krajnji efekat zlonamjernih programa mogu da budu veoma različiti uključujući i obaranje operativnog sistema.



Slika 3. Klasifikacija zlonamjernih programa

Napade na operativne sisteme od strane zlonamjernih programa možemo podijeliti u nekoliko grupa: **virusi, crvi i trojanske konje.**



U opštem slučaju postoje zlonamjerni programi koji mogu postojati nezavisno od drugih programa, kao i programi koji se instaliraju i izvršavaju kao dodatni dijelovi nekih drugih programa.
Logička bomba je zlonamjeran program koji se aktivira na neki događaj (npr. na neki datum). *Trapdoor* je nedokumentovani dio koda koji omogućava pristup neželjenim korisnicima.
Bakterije su zlonamjerni programi koji se replikuju sve dok ne napune cio disk ili dok ne potroše sve procesorske resurse.

Virusi

Računarski virus je program koji se može sam reprodukovati tako što dodaje sopstveni kod nekom drugom programu. Razultat izvršavanja virusa može biti ispisivanje neke poruke na ekranu, prikaz slike na ekranu, modifikovanje ili brisanje neke datoteke, poziv nekog telefonskog broja, itd. Drugim riječima, virus može uraditi sve što računarski program može uraditi.
Ciljevi autora virusa su:
brzo širenje virusa,
da se virus teško detektuje,
da odbrana od virusa bude što komplikovanija.
Tipičan virus u toku svog životnog ciklusa prolazi kroz sljedeće faze:

- faza spavanja,
- faza propagacije,
- faza trigerovanja,
- faza izvršavanja.

U fazi spavanja virus je besposlen. Iz ove faze virus se može aktivirati na neki događaj, kao što je datum, prisustvom nekog drugog programa ili datoteke, itd. Ova faza nije obavezna za sve viruse. U fazi propagacije se vrši kloniranje virusa. Identična kopija virusa se smješta unutar nekog programa ili datoteke i na taj način se vrši dalje inficiranje. U fazi trigerovanja dolazi do aktiviranja virusa i tada on započinje funkciju za koju je i namjenjen. Posljednja faza je faza izvršavanja u kojoj virus može samo ispisati neku poruku na ekranu ili izvršiti neku destruktivnu operaciju.
Najpoznatiji tipovi virusa su: paraziti, stalno prisutni u operativnoj memoriji, boot sektor, stealth i polimorfni. Paraziti se uvijek nalaze kao dio nekog izvršnog programa. Stalno prisutni virusi u memoriji su obično dio nekog sistemskog programa i inficiraju sve programe koji se izvršavaju. *Boot* sektor virusi inficiraju glavni *boot* slog (*master boot record*) i šire se nakon podizanja sistema sa diska. *Stealth* virusi su posebno projektovani da budu nevidljivi prilikom detekcije od strane antivirusnih softvera. Polimorfni virusi su virusi koji mutiraju prilikom svakog inficiranja, čime ostvaruju da detekcija na osnovu karakterističnog uzorka virusa nije moguća

Crvi

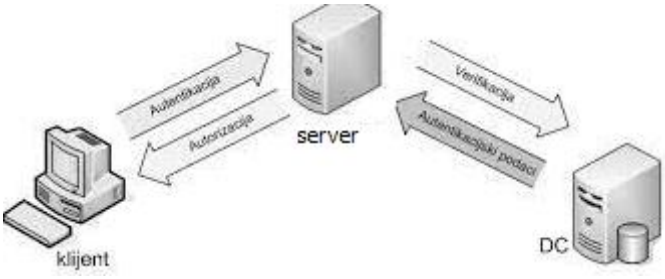
Crv je računarski program koji kopira samog sebe sa jednog računara na drugi. Crvi se mogu koristiti za prenošenje virusa ili za zamjenu postojećih datoteka verzijama datoteka koje predstavljaju Trojanske konje. Obično se crvi prenose preko računarske mreže koristeći nedostatke operativnih sistema u pogledu zaštite. Crvi se brzo replikuju i troše puno memorije na host računarima. Najčešće se prenose uz elektronsku poštu i uz dodatke elektronskoj pošti.
Između crva i virusa ne postoji baš uvijek jasna granica. Crvi su dosta slični virusima, ali između njih postoji razlika. Za razliku od virusa, za aktiviranje crva nije potreban korisnik i oni skrivaju svoje širenje na druge računare. Propagacija crva može biti mnogo brža od propagacije virusa. Brzina propagacije je proporcionalna broju ranjivih računara.

Trojanski konj

Trojanski konj je program koji se najčešće prenosi na ciljni računar kao nevidljivi dodatak uz neki drugi program, a zatim dolazi do njegovog aktiviranja. Mogu se prenijeti kopiranjem programa, skidanjem sa Interneta, kao i otvaranjem dodatka u elektronskoj pošti. Aktivnosti Trojanskih konja mogu biti veoma različite. Mogu da obrišu podatke, pošalju svoju kopiju na sve računare sa liste elektronske pošte i da omogući dodatne napade na dati računar. Obično ostvaruju povezivanje sa nekim udaljenim računarom i prenose informacije sa računara na kome su instalirani.
Trojanski konji koji se izvršavaju u okviru jezgra operativnog sistema su najopasniji. Tada ovi programi imaju potpunu kontrolu nad datim sistemom. To znači da programi kao što su, na primjer, drajver uređaja, screen saver i bilo koji drugi programi koje operativni sistem izvršava predstavljaju potencijalni izvor napada.
Trojanski konji se najčešće prikazuju kao korisni programi, ali u suštiti oni uvijek imaju neku neželjenu aktivnost po onoga ko ih koristi. Oni se koriste za indirektno izvršavanje funkcija koje neautorizvani korisnici ne bi mogli da ostvare direktno.

Mehanizmi zaštite i autentikacija

Zaštita sistema može biti implementirana na više različitih načina. Pod fizičkom zaštitom podrazumijeva se obezbjeđenje zgrada, obezbjeđenje prostorija i neki način kontrole identiteta korisnika sistema.



Glavni problem zaštite kod svih operativnih sistema je **autentikacija** (***proces određivanja identiteta nekog subjekta***).

Tradicionalan mehanizam zaštite je sistem kod koga se koristi korisničko ime i lozinka kojim se verifikuje identitet korisnika i na taj način isključuje mogućnost rada neidentifikovanih korisnika. Tajnost lozinke je glavni dio sistema kojim se naovaj način obezbjeđuje zaštita.

Od načina implemenatacije sistema lozinki najviše zavisi koliko će sistem biti zaštićen.

Zato većina današnjih sistema ne omogućava korisnicima unos onih lozinki koje nisu dovoljno sigurne i koje se nazivaju slabe lozinke. Primjeri slabih lozinki su ime i prezime, riječ iz riječnika, ime člana porodice, itd.
U cilju bolje zaštite, potrebno je da operativni sistem podrži **jake lozinke**, odnosno:

1. kontroliše da lozinke budu riječi koje se ne mogu naći u riječniku,
2. kontroliše da lozinke budu riječi najmanje 6 karaktera dužine,
3. kontroliše da lozinke budu sastavljene i od slova i brojeva,
4. obezbjedi da lozinke imaju period važenja,
5. ograniči broj pokušaja prijavljivanja na sistem sa pogrešnom lozinkom, tako što će nakon maksimalnom broja pokušaja automatski zabraniti korisnički nalog.

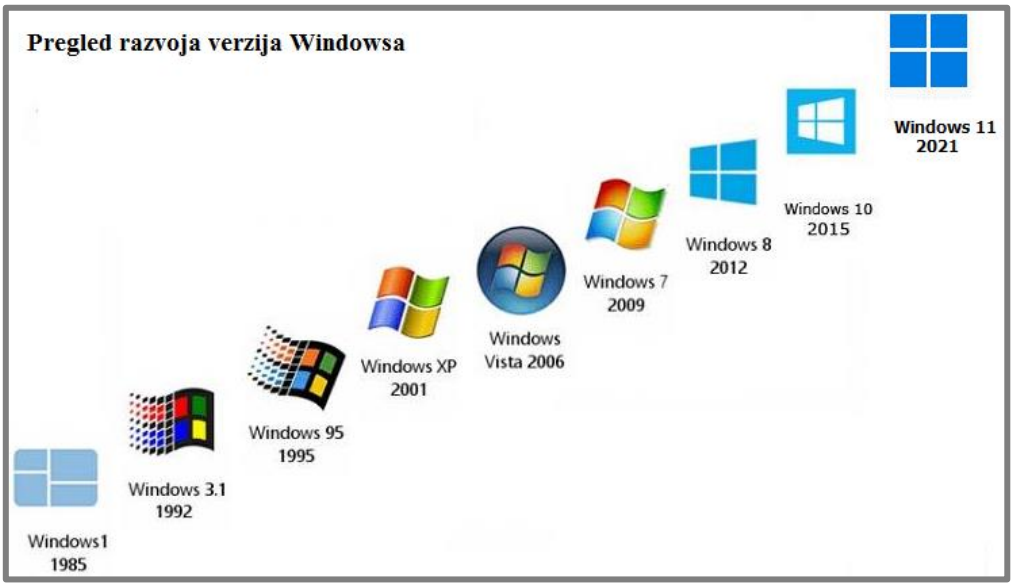
Najvažnija lozinka u datom sistemu je lozinka sistem administratora, jer ona ima kompletnu kontrolu nad sistemom. Upravo zbog toga najveći broj napada na sistem ima za cilj pronalaženje lozinke sistem administratora.

Sistem zaštite koji se zasniva na lozinkama se može narušiti **pogađanjem** lozinki.
Drugi način je metodom **grube sile** gdje se korišćenjem današnjih računara veoma brzo može pretražiti kompletan skup mogućih lozinki čija je maksimalna dužina unaprijed poznata.

Narušavanje sistema zaštite koji se zasniva na lozinkama se može narušiti kao rezultat vizuelnog ili elektronskog monitoringa.
Vizuelni monitoring nastaje gledanjem u tastaturu prilikom unosa korisničkog imena i lozinke.
Elektronski monitoring se može uraditi pomoću sniffing alata kojim se može snimiti identitet korisnika i njegova lozinka. Kriptovanje podataka koji sadrže lozinku rješava ovaj problem.

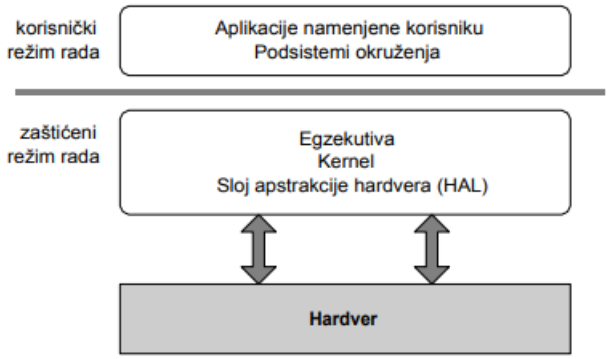
Osnovne karakteristike Windows OS

Danas je Windows **najpopularniji** operativni sistem, uživajući ogromnu nadmoć **na tržištu stonih PC računara**. Značajno rasprostranjen i u segmentu malih i srednjih servera u primjenama kao što su mrežni serveri ili serveri baza podataka.



Nastao je kao grafička nadogradnja MS DOS operativnog sistema. Današnje verzije se baziraju na prvobitnoj NT verziji Poslije Windows-a 95 ove dvije linije razvoja su integrisane u jednu, a od Windows-a ne predstavlja više samo grafičko okruženje već uistinu potpuni operativni sistem. (Do Windows-a 3.0 Windows-i su bili „samo“ shell, a DOS je bio kernel operativnog sistema.
Windows OS radi na računarima zasnovanim na procesorima firme Intel i njima sličnim. Oznaka za takve procesore je h86 kompatibilni. Kao značajan proizvođač prosecora koji su hardverska platforma ovog operativnog sistema (sem Intela) je AMD (Posljednjih godina je po nekim performasama čak i nadmašio Intel.) Postoje ili su postojale varijante koje rade na procesorima DEC Alpha, MIPS i PowerPC. Postoje varijacije za procesore sa 32, 64 i 128 bita.

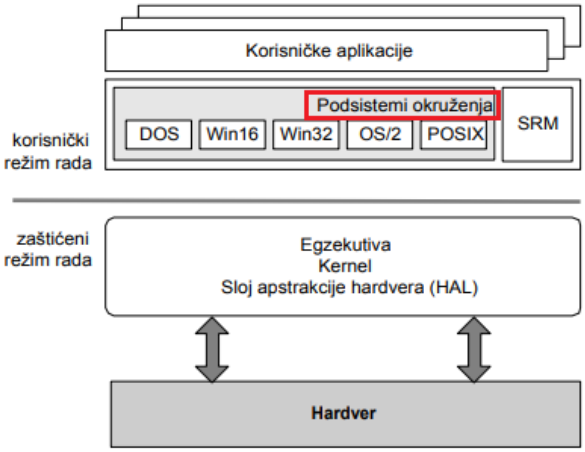
Windows je višeprocesni operativni sistem sa pretpražnjenjem, sposoban da iskoristi prednosti višeprocesorske arhitekture.



Pojednostavljena arhitektura Windows OS

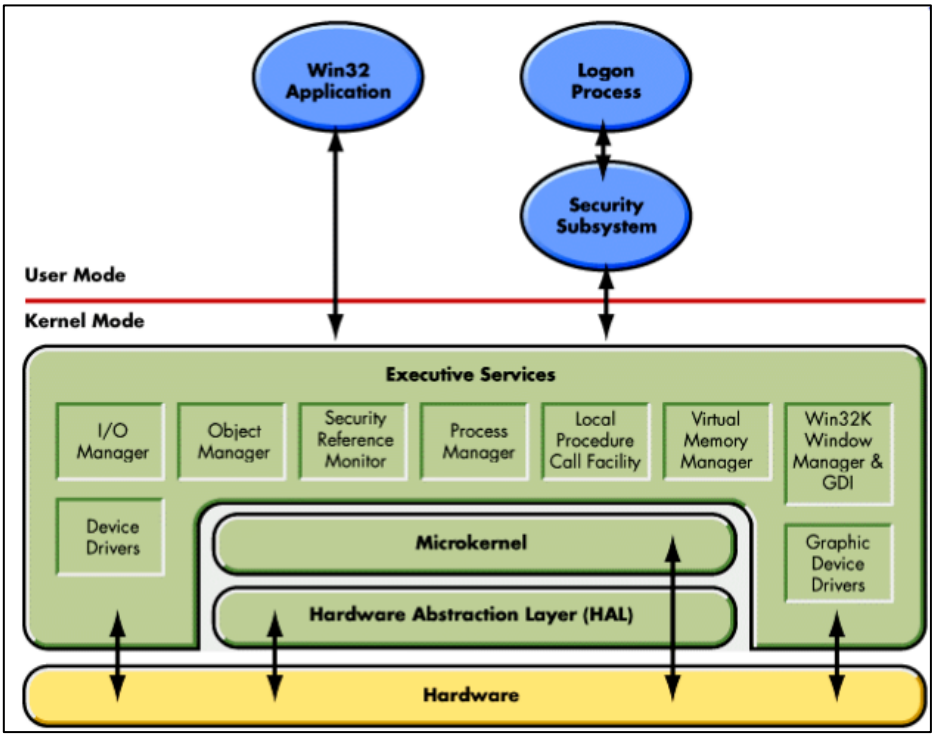
Windows jezgra (Windows kernel) je **hibridnog dizajna** budući da je dio jezgrinih modula implementiran u vidu korisničkih aplikacija, no smatra se u osnovi monolitnom jezgrom.
Govrimo o hibridu jer se mikrokernel nalazi unutar kernela.

Podsjećamo da je HAL sloj posrednik između kernela i hardvera, koji višim slojevima operativnog sistema obezbeđuje podršku za rad sa procesorom i simetrično multiprogramiranje.
Keš za instrukcije, keš za podatke i sve funkcije vezane za tajmer, prekide, konkretan firmware i rukovanje greškama, zavise od konkretne računarske arhitekture i kao takve se na nivou HAL-a prevode u API (*s kojim se možete detaljni upoznati malo kasnije*) razumljiv višim slojevima operativnog sistema.



Podsistemi okruženja su programi koji rade u korisničkom režimu rada i pružaju servise ka aplikacijama.
Ovi servisi simuliraju usluge koje pruža specifičan operativni sistem.
Podsistemi okruženja implementirani su u Windows kao klijent/server model; aplikacija koja traži servis je klijent, a podsistem okruženja server.

Ovakav koncept sa neznatnim izmjenama zadržan je i u aktuelnim verzijama Windowsa. Izmjene su se odnosile na poboljšanja vezana za mrežne aplikacije i višenitno upravljanje procesima omogućeno višejezgrenim mikroprocesorima.



Blok šema koja odgovara Windowsu XP

Upravljanje objektima Izvršni servisi: Egzekutiva

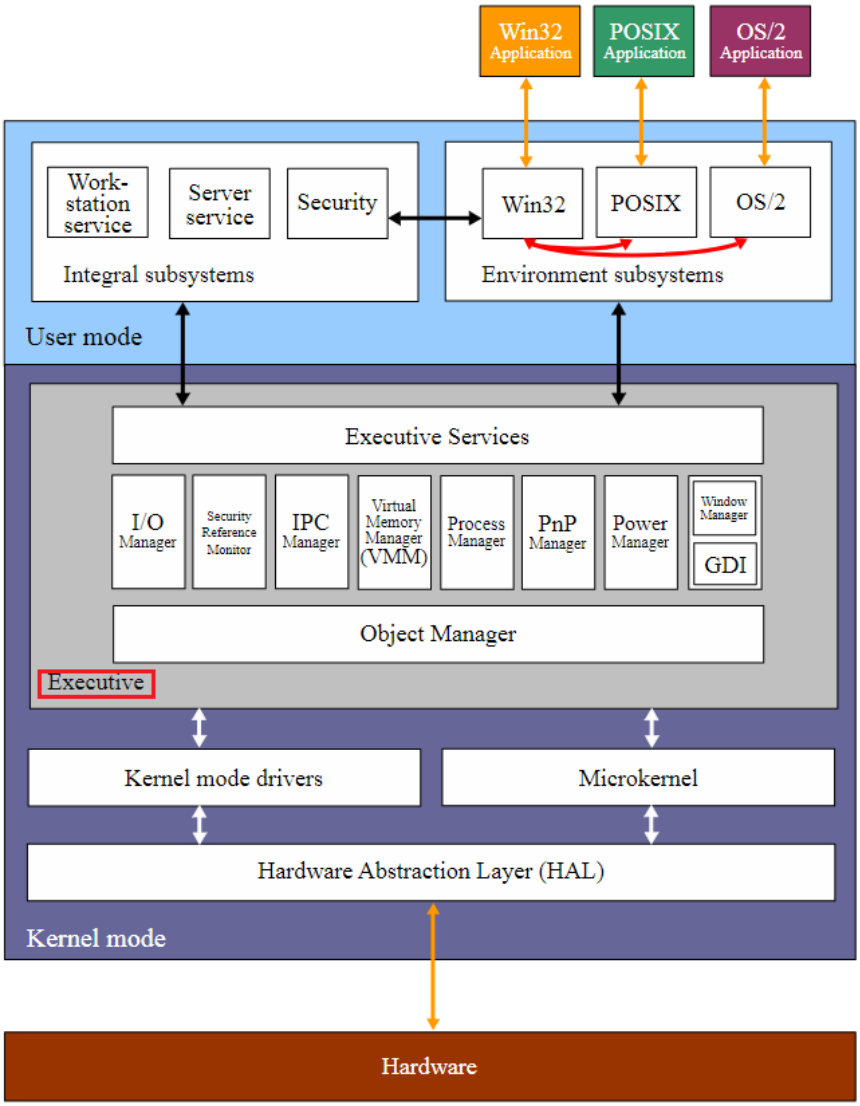
Za upravljanje objektima Windows koristi objekte za sve svoje usluge (servise) i entitete.
Windows kernel je hibridno jezgro; arhitektura se sastoji od jednostavnog kernela, sloja apstrakcije hardvera (HAL), drajvera **i niza usluga (zajedno nazvanih Executive)**, koji svi zajedno rade u kernelu.

Upravljač (manager) objekata nadzire korišćenje objekata:
• generiše handle objekta,
• provjerava sigurnost i sigurnosna ograničenja i
• vodi računa tj. prati i evidentira koji procesi koriste koje objekte.

Manipulacija objektima se obavlja standardnim skupom metoda, a to su: create, open, close, delete, query name, parse i security.

Egzekutiva dozvoljava da svakom objektu bude dato ime, koje može biti stalno tj. permanentno ili privremeno. Imena objekata su struktuirana kao imena putanja u MS-DOS operativnom sistemu ili Unixu.

Proces ostvaruje pristup određenom kernel objektu pozivanjem funkcije koja otvara handle ka tom objektu. Hande je jedinstven za proces.
Proces može zatvoriti odgovarajući handle pozivanjem CloseHandle funkcije.
Operativni sistem može obrisati objekat ako broj procesa koji ga koriste padne na 0.
Proces može dobiti handle objekta ukoliko kreira novi objekat ili deli postojeći objekat sa drugim procesom.



Windowsi koriste tri načina za djeljenje objekata između procesa:
• otvori postojeći objekat (proces daje objektu ime prilikom kreiranja, a zatim drugi proces otvara objekat po tom imenu),
• proces nasleđuje handle objekta od procesa roditelja i
• dobije handle duplikat od drugog procesa (duplikat se kreira funkcijom DuplicateHandle).

Svaki objekat je zaštićen listom za kontorlu pristupa (access control list).

Programi i procesi kod Windows OS

Da dodatno razjasnili razliku između procesa i programa, napravimo jednu analogiju. Zamislimo kuhara koji peče rođendarsku tortu svojoj kćeri. On ima recept i kuhinju opremljenu sa svim sastojcima: brašnom, jajima i slično.

U svijetu OS-a kuhar bio bi procesor, a recept bio program, dok bi sastojci bili ulazni podaci. Zamislimo sad da kuharova kći utrči u kuhinju plačući jer ju je ubola pčela. Kuhar tada zapamti gdje je stao u svom receptu, i pruži pomoć svojoj kćeri.
U svijeti OS-a vidimo da je procesor bio prebačen sa jednog procesa na proces višeg prioriteta, od kojih svaki ima različiti program. Nakon pružene pomoći kuhar će nastaviti tačno od mjesta gdje je stao.
Ključna ideja je da je proces aktivnost neke vrste. On ima program, ulaz, izlaz i stanje. Pojedini procesor može dijeliti nekoliko procesa, zajedno sa nekim algoritmom raspoređivanja rada tih procesa na tom procesoru.

Windows program je uobičajno „event driven“ (vođen događajima). Glavni program čeka da se neki događaj desi pa onda poziva proceduru koja rukuje tim događajem. Tipični događaji su pritisak tastera, pokret miša... Kada se nešto tako desi poziva se handler da obradi događaj, ažurira sliku na monitoru i ažurira interno stanje programa.

Programi pod Windows mogu da se izvršavaju u dva osnovna moda:
- kernel mod-kad se izvršava sistemski kod i
- korisnički mod-kad se izvršavaju aplikacije.

U okviru Microsoft Windows operativnog sistema process je centralni objekat.
Podsjećamo proces je program u izvršenju.
32-bitni Windows operativni sistem je multitasking operativni sistem koji podržava više niti koje se izvode unutar procesa.

Windows nudi veliki broj Win32 API poziva za sve module koje sadrži.

Tako, npr. iako je veći dio upravljanja memorijom za programera apstrakcija, programer preko API poziva može koristiti osobinu da je proces u stanju da ubaci fajl u njegovu virtuelnu memoriju i da dijelove fajla tretira kao memorijske riječi.

Takođe je interesantan I/O fajlova. Fajl se u Windows-u tretira kao linearna sekvenca bajtova.
Postoje oko 60 Win32 API poziva za kreiranje fajlova i direktorijuma, zatvaranje, podešavanje atributa, itd.

Win32 takođe ima listu poziva sigurnosti, tj. zaštite. Svaki proces ima svoj ID na osnovu kojeg ga objekat identifikuje da li taj proces ima pravo pristupa tom objektu ili ne. Objekti imaju liste pristupa koje ukazuju koji sve korisnici mogu da im pristupe

Tehnički gledano, u svim ovim slučajevima, neki postojeći proces sistemskim pozivom kreaira novi proces.
U Windowsima (pritom se misli na WindowsAPI), **koristimo API poziv CreateProcess**, koji obavlja i kreiranje novog procesa i učitavanje novog programa u novokreirani proces.

Primjer 1: simuliranje komandne ljuske:
U pseudo-jeziku opis ljuske bi mogli dati na ovaj način:

```
ponavljaj {
        ispiši_ prompt;
        učitaj_naredbu(N, P);
        kreiraj_novi_proces();
```

```
        roditelj: čekaj dok proces dijete ne završi;
        dijete: pokreni naredbu N sa parametrima P;
}
```

Create Process (nalazi se u zaglavlju Windows.h) ima 10 parametara:

Pokazivač na ime modula koji se želi izvršiti.
Pokazivač na string koji predstavlja komandu liniju koja će se izvršiti.
Pokazivač na strukturu za sigurnosnim atributima procesa.
Pokazivač na strukturu za sigurnosnim atributima za inicijalnu dretvu.
Bit koji kazuje da li novi proces naslijeđuje handle-ove procesa koji ga je kreirao.
Različiti flagovi (npr. error mode, prioritet, i sl.)
Pokazivač na environment.
Pokazivač na radni direktorij novog procesa.
Pokazivač na strukturu koja postavlja inicijalne vrijednosti prozora za novi proces.
Pokazivač na strukturu koja vraća podatke o novo kreiranom procesu kao što su njegov process id i handle novog procesa.

Vraća 0 u slučaju greške.

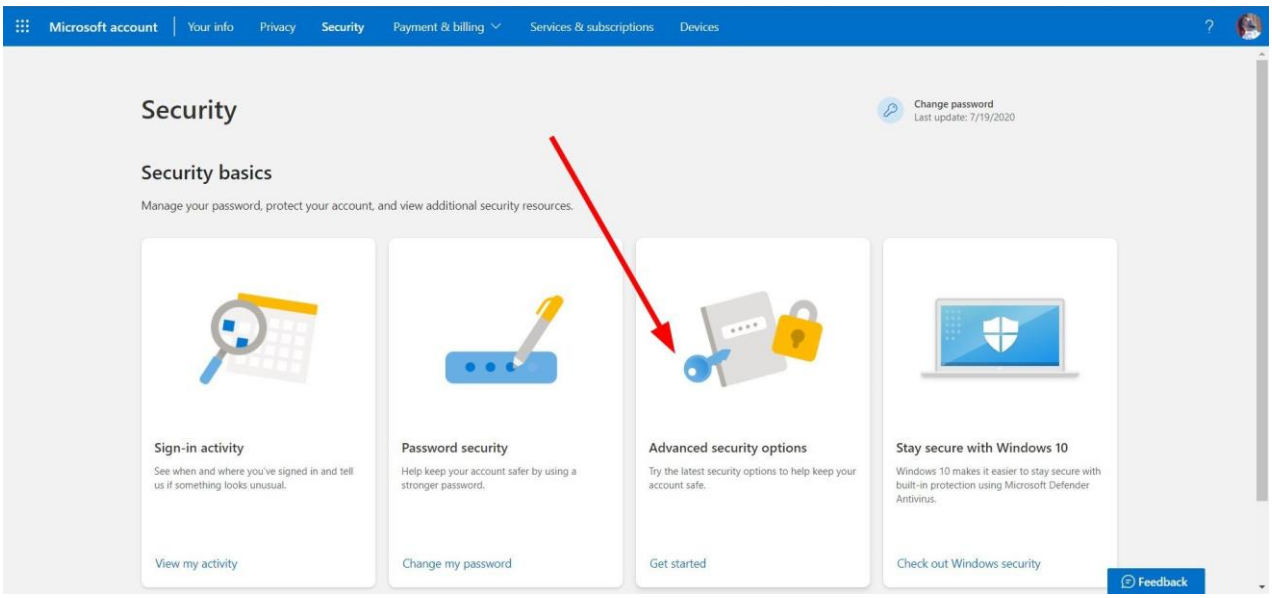
Umrežavanje

Windowsi podržava obje standardne vrste umrežavanja, tzv. peer-to-peer i klijent-server. Za obje poseduje odgovarajući skup alata za upravljanje mrežom. Za to koristi
• NDIS (Network Device Interface Specification) - odvaja mrežni adapter od transportnog protokola tako da bilo koji od njih može biti izmenjen bez uticaja na onaj drugi i
• TDI (Transport Driver Interface) - omogućava bilo kojoj komponenti nivoa sesije da koristi bilo koji transportni mehanizam.
Windowsi implementiraju transportne protokole kao drajvere koji mogu biti učitani tj. napunjeni (loaded) i ispražnjeni (unloaded) iz sistema dinamički.

Sigurnosni podsistem

Podsistem za prijavljivanje (logon) i sigurnosni podsistem (Security Subsystem) autentifikuju korisnike koji se prijavljuju na Windows sisteme.
Od korisnika se zahtjeva da imaju ime naloga i lozinke. Paket za autentifikaciju, autentifikuje korisnike uvijek kada pokušaju da pristupe nekom objektu u sistemu.

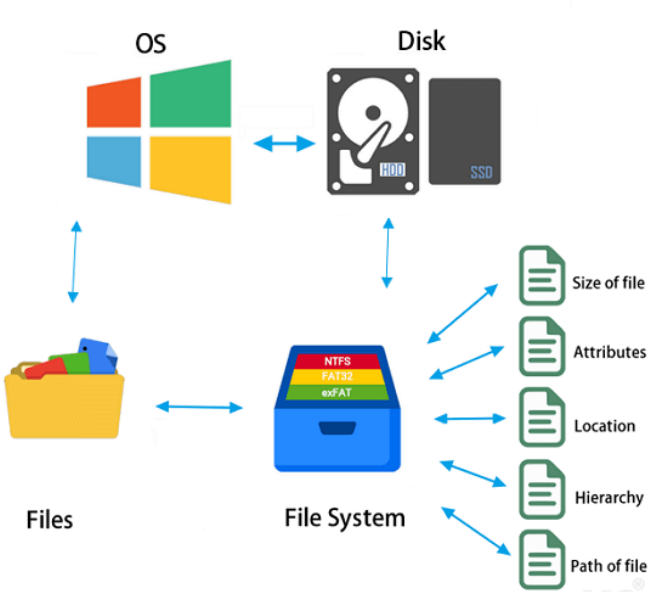
Operativni sistem Windows implementira podrazumjevani skup protokola za autentifikaciju, uključujući Kerberos, NTLM, Transport Layer Security/Secure Sockets Layer (TLS/SSL) i Digest, kao dio proširive arhitekture.
Osim toga, neki protokoli se kombinuju u pakete za autentifikaciju kao što su Negotiate i Credential Security Support Provider.

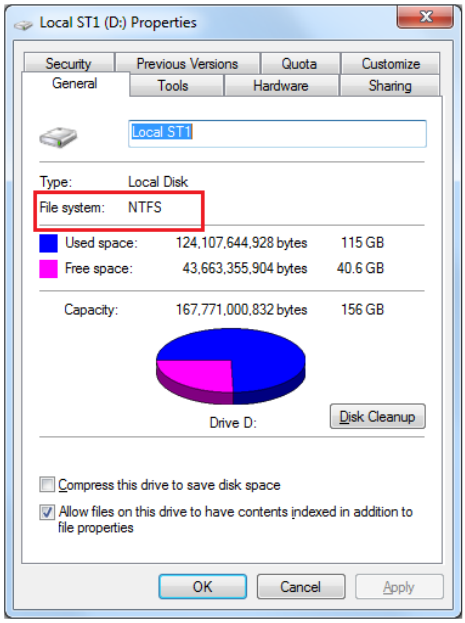


Ovi protokoli i paketi omogućavaju autentifikaciju korisnika, računara i usluga; proces autentifikacije, zauzvrat, omogućava ovlaštenim korisnicima i servisima pristup resursima na siguran način.

NTFS sistem datoteka

*Windows 11, 10, 8.1 i drugi MS Server operativni sistemi prvenstveno koriste NTFS (**New Technology File System**) sistem datoteka.*

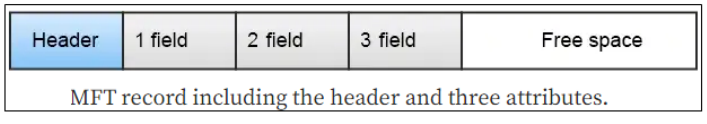




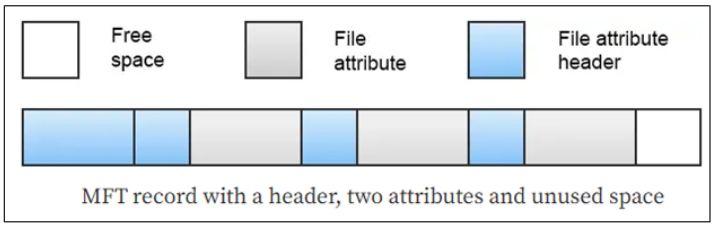
Osnovna struktura NTFS sistema datoteka je volumen (volume), zasnovan na logičkoj disk particiji. Volumen može zauzeti dio diska, cio disk, a može se i rasprostrati na više diskova.
Osnovne jedinice za dodjelu prostora na disku su klasteri (clusters), odnosno grupe koje čine 2n sektora diska, koje se adresiraju korišćenjem logičkih brojeva klastera

NTFS sistem, po svojoj strukturi, podsjeća na relacionu bazu podataka koja organizuje podatke na principu B+ stabla. Prostor imena (NTFS name space) je organizovan hijerarhijom direktorijuma, pri čemu index root sadrži vrh (korjen) B+ stabla.
Datoteka u NTFS sistemu datoteka je struktuirani objekat koji se sastoji od atributa.

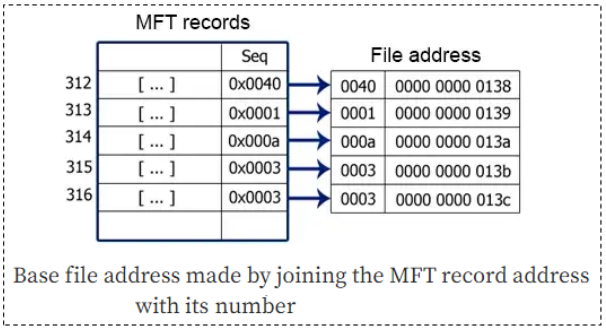
NTFS pohranjuje informacije o datotekama i direktorijumima u glavnoj tablici datoteka (**Master File Table: MFT**). Ova tabela datoteka sadrži informacije o svakoj datoteci i direktorijumu navedenim u sistemu datoteka. Svaka datoteka ili direktorij ima najmanje jedan zapis u MFT-u.
MFT predstavlja prvu liniju svake NTFS particije, i ima svoju rezervnu kopiju. Ova redundansa osigurava oporavak sistemskih datoteka u slučaju oštećenja.
Pokazivači u boot sektoru volumena opisuju lokaciju na kojoj se nalazi MFT. Svaka datoteka na jednom NTFS volumenu ima svoj jedinstveni 64-bitni identifikator (unique ID) koji se zove referenca datoteke (engl. file reference).
Datoteke i direktorijumi su objekti NTFS sistema datoteka i slične su strukture (NTFS tretira direktorijum kao specijalnu datoteku). Svaki objekat ima svoje ime i sadržaj. Sadržaj datoteke su konkretni podaci. Sadržaj direktorijuma čine indeksi relativnih lokacija, veličina direktorijuma i bitmapirana reprezentacija strukture direktorijuma. NTFS podržava Unicode imena datoteka, dužine do 255 karaktera. Ovaj način imenovanja doprinosi jednostavnijoj internacionalizaciji operativnog sistema.



MFT record including the header and three attributes.



MFT record with a header, two attributes and unused space



Base file address made by joining the MFT record address with its number

Format MFT zapisa je izuzetno jednostavan. Svaki zapis je veličine tačno 1 KB. Prva 42 bajta u zaglavlju imaju fiksnu strukturu, dok se ostatak zapisa koristi za pohranjivanje atributa kao što su naziv datoteke ili sistemski atributi. Broj atributa kao i veličina svakog atributa mogu varirati.

Jedinstvena za NTFS je mogućnost pohranjivanja malih datoteka na licu mjesta. Cijeli sadržaj male datoteke može se pohraniti kao atribut u MFT zapisu, uvelike poboljšavajući performanse čitanja i smanjujući izgubljeni prostor na disku („slack“ prostor).

Sigurnost NTFS volumena je izvedena iz Windows objektnog modela. Svaki fajl objekat ima sigurnosni opis (securiy descriptor) koji je smešten u MFT. Ovaj atribut sadrži pristupni žeton vlasnika datoteke i pristupnu kontrolnu listu (access control list) koja utvrđuje privilegije koje su dodjeljene svakom korisniku koji ima pristup datoteci.

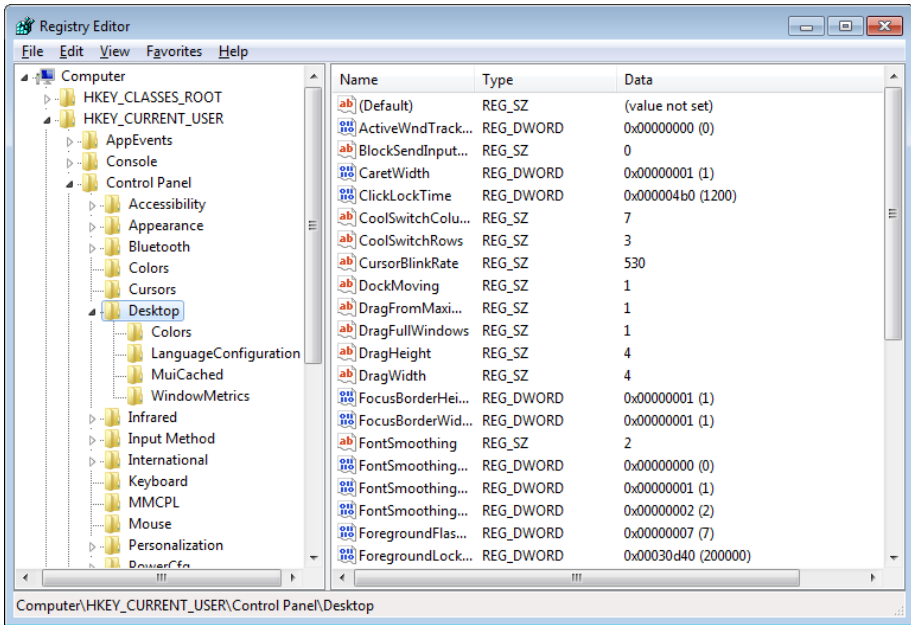
Jedna od mogućnosti NTFS sistema datoteka je i kompresija datoteka radi smanjenja potrebnog prostora za smeštanje.
NTFS dijeli podatke iz datoteke na jedinice kompresije (engl. compression unit) koje su u suštini blokovi od 16 kontinuiranih klastera.
Za „raštrkane“ i „oskudne“ datoteke, tj. datoteke čiji se podaci nalaze u više manjih dijelova na različitim mjestima na disku ili koji imaju kontinuirane blokove istovjetnih podataka, NTFS koristi drugu tehniku da uštedi prostor:
• klasteri koji sadrže sve nule u suštini nisu alocirani na disku,
• umjesto toga, praznine (engl. gap) su zadržane u sekvenci virtuelnih klastera smeštenih u MFT zapisu datoeke,
• kada se učitava fajl, ako se nađe praznina u virtuelnim klasterima, NTFS će samo napuniti nulama taj deo pozivačeovog bafera.

Informacije o Windows OS – Windows registar - Windows Registry

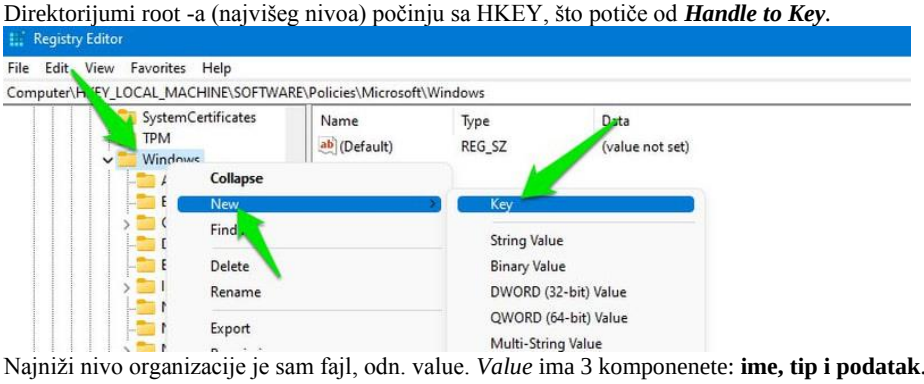
Registri predstavljaju dio Windows-a koji je teži za razumijevanje zbog organizacije podataka i kriptovane nomenklature.
Windows registar je baza podataka u kojoj su zabilježena sva podešavanja operativnog sistema i korisničkog okruženja, podaci o instaliranim programima, računarskoj opremi i svi drugi podaci vezani za pravilno funkcionisanje računarskog sistema.
Gubitak informacija u registru ima za posljedicu potrebu za reinstalacijom čitavog softvera.

Nekada su za aplikacije postojale .ini fajlovi u koje su programi pamtili sopstvena podešenja. Često su se nalazile na zajedničkoj lokaciji što je, u slučaju više korisnika, onemogućavalo da svaki ima svoja podešenja. Suprotno tome, registar pamti sva aplikacijska podešenja u jedan središnji repozitorijum i to u standardizovanom obliku.

Registri baza se sastoje od veoma malih fajlova koji su organizovani po direktorijumima i poddirektorijumima. Ti mali fajlovi sadrže informacije o hardveru, softveru i korisnicima. Ovi fajlovi se nazivaju **values**.
Pregled ove tabele na računaru dobija startovanjem *regedit* ili *regdt32*. (A kod Win 11 sa Windows key.)



Direktorijum se, u slučaju registara, naziva **key** (ključ).
Direktorijumi root -a (najvišeg nivoa) počinju sa HKEY, što potiče od ***Handle to Key***.



Najniži nivo organizacije je sam fajl, odn. value. *Value* ima 3 komponenete: **ime, tip i podatak**.
Postoje 11 standardnih tipova value-a. Tip može biti takav da ukazuje na drugi value ili neki drugi direktorijum baš kao što to rade prečice na desktopu. Ovakav tip se naziva ***simbolic link***.

U sljedećoj tabeli, dat je pregled najvažnijih ključeva (i nekih podključeva) sa opisom podataka koji se u njima smještaju.

Ključ	Opis
HKEY_LOCAL_MACHINE	Osobine hardvera i softvera
HARDWARE	Opis hardvera veza između hardvera i drrajvera
SAM	Informacije za logovanje
SECURITY	Generalne informacije o sigurnosti
SOFTWARE	Podaci o instalisanim aplikacijama
SYSTEM	Podaci o butovanju sistema
HKEY_USERS	Informacije o korisnicima
HKEY_PERFORMANCE_DATA	Brojači koji monitorišu sistemske performanse
HKEY_CLASSES_ROOT	Link: HKEY_LOCAL_MACHINE\SOFTWARE\CLASSES
HKEY_CURRENT_CONFIG	Link ka trenutnom profilu hardvera
HKEY_CURRENT_USER	Link ka profilu aktuelnog korisnika

Interesantno je da ovakav pregled prikazuje samo 5 ključeva dok se za ključ HKEY_ PERFORMANCE_DATA mora startovati jedan od programa: *pfmon*, *pview* ili *perfmon*.
Na osnovu navedenog se vidi da u stvari postoje samo 3 ključa a ostala 3 su linkovi na već postojeće direktorijume.
Tako, npr. HKEY_CLASSES_ROOT se linkuje sa direktorijumom koji rukovodi COM (Componenet Object Model) objektima i povezuje programe sa ekstenzijama fajlova.
Kada korisnik duplo klikne na ikonu fajla sa, recimo .doc ekstenzijom, program koji hvata klik miša treba da odluči koji će program da startuje i, pregledom na bazu podataka o ekstenzijama u ovom registru, odlučuje se za Microsoft Word.

Baza registra je u potpunosti pristupačna Win32 programeru. Zato je važno napomenuti da, pošto registri predstavljaju dio sistema od vitalnog značaja, **gubitak informacija na registrima ima za posljedicu potrebu za reinstalacijom čitavog softvera.**

Promjena -editovanje Windows registra

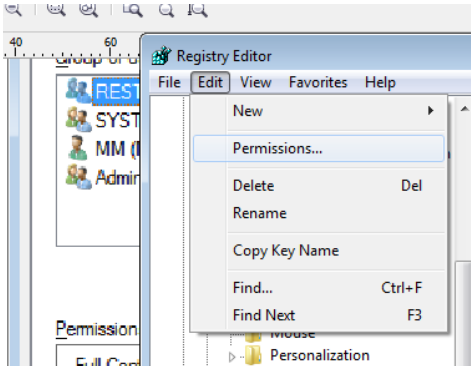
Editovanje se pokreće komandom Run: Regedit,

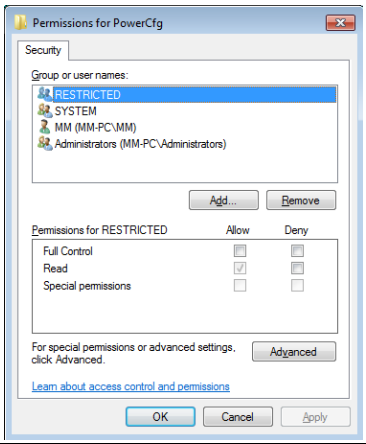
Vrlo je bitno istaći važnost pažljivog rukovanja ovim alatom. **Jedna-dvije, nepromišljene promjene uništavajau kompletan sadržaj računara.**

Pojedini elementi koje vidite na lijevoj strani Regedita mogu se izvesti u zaseban tekstualni fajl. Rezultujući zapis se potom lako može menjati iz tekst editora, ukoliko za to postoji potreba, a njegovo pridruživanje bazi postiže se dvostrukim klikom. Ukoliko želite da dio baze odstranite ponovo putem REG fajlova, dovoljno je da ispred konkretne linije jednostavno dodate znak minus, ali ovakvu praksu treba izbegavati.

Uzmimo za **primjer** jednu od relativno čestih potreba za modifikovanja Registryja – **odstranjivanje unosa koje su deinstalirani programi zaboravili**. Zamislimo fiktivnu situaciju u kojoj ste se odlučili da famozni ACDSee pošaljete u zasluženu penziju. Rutina za odstranjivanje programa kao i obično nije posao odradila do kraja, te prvo treba posjetiti i obrisati lokaciju „HKEY_LOCAL_MACHINE\ SOFTWARE\ ACD Systems\ ACDSee\" i identičan ključ pod „HKCU". Zamislimo takođe da je naziv programa zaostao u Add/Remove Programs, pa se treba zaputiti na „HKLM\ SOFTWARE\ Microsoft\ Windows\ CurrentVersion\ Uninstall", gdje ćete nesumnjivo naći neke unose davno zaboravljenih i odstranjenih programa. Birate ključ s lijeve strane, Delete i – voila, kontrolna tabla ponovo je čista i uredna.

Primjer za pravilno postavljanje dozvola znatno je kompleksnije. Win OS se zasniva na korisnicima **sa** administratorskim privilegijama **i bez** njih i, kao što smo već pomenuli, potpunoj kontroli nad pojedinim granama, prostom čitanju ili specijalnim slučajevima koji su još složeniji.
Promjena dozvola određenog ključa vrši se putem **Permissions** iz menija desnog klika, izborom konkretne grupe ili svih korisnika, pa obilježavanjem odgovarajućih polja za (ne)odobravanje pristupa.
Osnovnoj promjeni vrijednosti, bilo dodavanjem, izmjenom ili brisanjem postojećih elemenata, može se pristupiti i direktno iz komandne linije.
Ukoliko, na primjer, želite da pridružite zasebni REG fajl postojećoj bazi, neophodno je da unesete „regedit.exe /s naziv_fajla.reg". Na ovaj način spajanje sadržaja fajla sa bazom biće u pozadini, dok je izostavljanjem parametra „/s" za to neophodno izdati dozvolu. Sufiks „/c" služi za kreiranje novog unosa, „/d" za odstranjivanje postojećih, „/e" je za eksportovanje, a s obzirom na to da se ovim putem ne postiže ništa više nego regularnim pristupom Regeditu, nećemo se dalje zadržavati na ovom segmentu editovanja.





Kreiranja procesa i Windows model programiranja

Programi pisani za tradicionalne operativne sisteme koriste proceduralni model u kojem se programi izvršavaju od vrha prema dnu. Putanja izvršavanja programa može zavisi o ulaznim parametrima, ali uglavnom je prilično predvidljiva.
Unix i Windows se fundamentalno razlikuju što se tiče modela programiranja. Dok se UNIX program sastoji od koda koji nešto izvršava ili pravi pozive sistema, Windows program je upravljan događajima. Glavni program čeka da se desi neki događaj, kao što je pritisnuta taster na tastaturi, pomak miša, pritisnuto neko od dugmadi na mišu, ubačena fleš memorija ili slično, i onda zove proceduru koja dalje preuzima kontrolu nad tim događajem. Ti handleri procesiraju događaj, ažuriraju ekran, ažuriraju unutrašnje stanje programa.

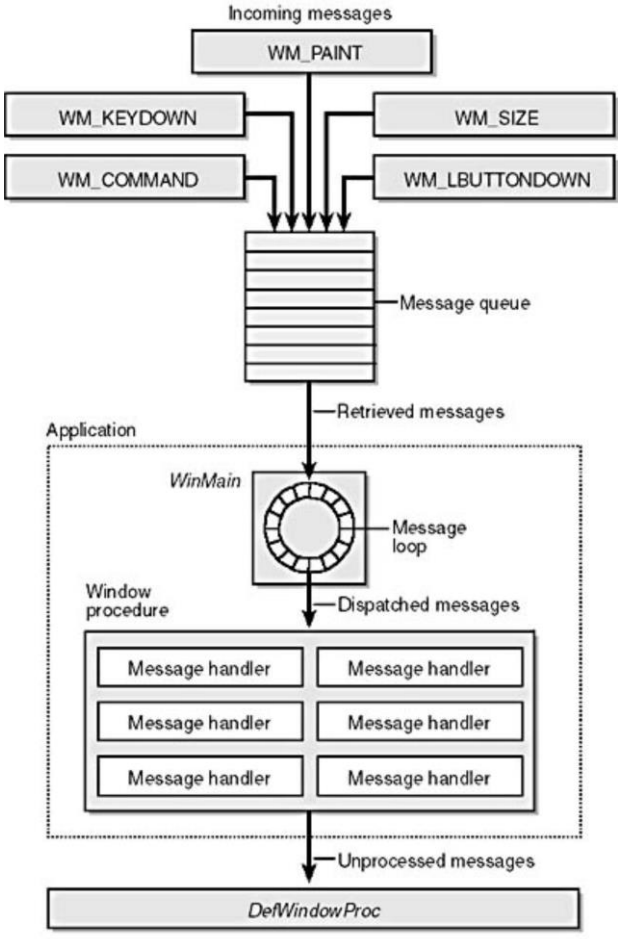
Windows programi koriste model događaja prikazan na slici u kojem aplikacija odgovara na događaje obrađujući poruke koje šalje operativni sistem. Događaj može biti pritisak tastera, pomak miša, ili neka druga stvar.

Ulazna tačka Windows programa je funkcija **WinMain**, ali većina posla se obavlja u funkcijama koje nazivamo window procedure. One obrađuju poruku koja je poslana prozoru.

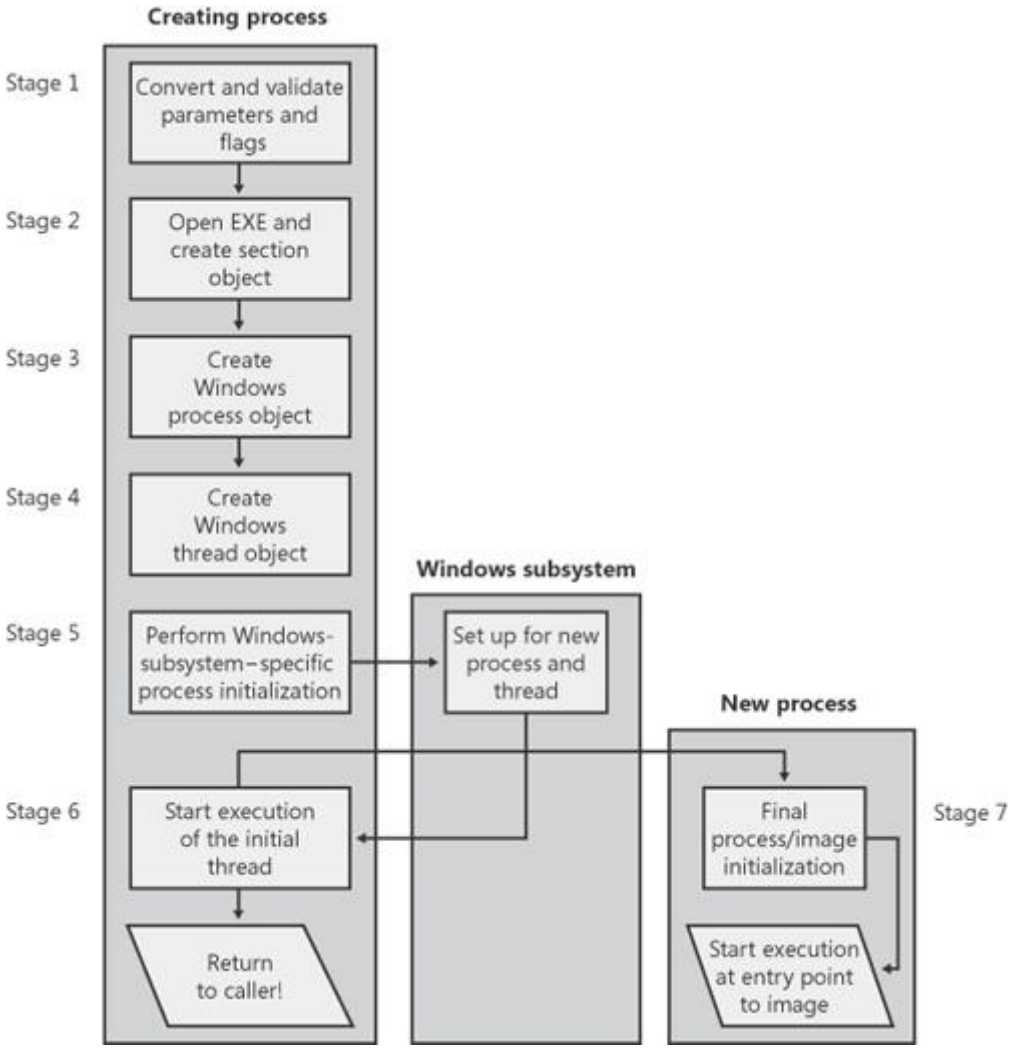
WinMain kreira taj prozor i onda ulazi u petlju poruka (**message loop**) gdje se poruke prihvaćaju i šalju prikladnim window procedurama.

Petlja poruka završava dolaskom WM_QUIT poruke kojom aplikacija završava sa radom. Ova se poruka obično pojavljuje jer je korisnik odabrao Exit iz File MENIJA ili kliknuo X u gornje desnom uglu.

Završetkom petlje poruka, WinMain završava i aplikacija završava sa radom.



Osnovna funkcija upravljanja Windows procesima je CreateProcess, koja kreira proces sa jednom niti. Navedite ime izvršne programske datoteke kao dio poziva CreateProcess.



Dijagram koji pokazuje korake i stanja pri kreiranju Windows procesa

Uobičajeno se govori o roditeljskim i podređenim procesima (*parent/child processes*), što nije baš korektno. CreateProcess kreira novi proces s jednom primarnom niti (koja može stvoriti dodatne niti).

Komunikacija između procesa

Win32 aplikacije ostvaruju interprocesnu komunikaciju dijeljenjem kernel objekata.
Da bi se sinhronizovao konkurentni pristup deljenim objektima od strane niti, kernel obezbeđuje objekte sinhronizacije, kao što su semafori i mutexi.
Jedno alternativno značenje interprocesne komunikacije je prosljeđivanje poruka, koje je posebo popularno za Windows GUI aplikacije. Jedna nit šalje poruku drugoj niti ili prozoru, pri čemu se sa porukom mogu poslati i podaci.
Svaka Win32 nit ima svoj vlastiti ulazni red iz koga nit dobija poruke. *Ovo je mnogo pouzdanije nego djeljivi ulazni red 16-bitnog prozora zato što sa posebnim redovima jedna zakočena aplikacija ne može blokirati ulaz za ostale aplikacije.*

DLL: Biblioteke za dinamičko povezivanje

DLL je tip datoteke koji sadrži naredbe koje mogu pokrenuti drugi programi da bi učinili nešto. Na ovaj način više programa može dijeliti mogućnosti koje su programirane u jednoj datoteci, pa čak i istovremeno. Drugim rječima DLL je izvršna datoteka, koja izvršava određene funkcije, i koja ne može pokretati sama sebe nego se pokreće od izvršne exe datoteke.
Biblioteke za dinamičko povezivanje (Dynamic Link Libraries) su jedan od najvažnijih strukturnih elemenata Windows operativnog sistema. DLL datoteke nisu direktno izvršne i ne primaju poruke.
To su posebne datoteke s funkcijama koje se pozivaju iz programa ili drugih DLL-ova zbog obavljanja određenog posla. **DLL sadrži funkcije koje drugi programi mogu upotrebljavati i koristi se tek kada ga aplikacija pozove**. DLL obično ima ekstenziju .dll ali može imati i druge nastavke.
Dynamic link znači da se DLL linka (povezuje) sa našim programima po potrebi ***na zahtjev*** *i njihovo dinamičko linkovanje, (dok static link povezuje program sa bibliotekom prije izvođenja).*
Tako kodovi u blibliotekama mogu da se ažuriraju (transparentno za aplikacije), a zatim se uklanjaju iz memorije ukoliko više nisu potrebni.
Za razliku od standardne programske biblioteke, čije se funkcije linkuju sa aplikacijama u vrijeme kompajliranja koda aplikacije, aplikacija koja koristi DLL linkuje se sa ovim DLL funkcijama u vrijeme izvršenja. Otud opis dinamička u imenu.

Najveća prednost DLL je što ga može koristiti više aplikacija pa ne mora svaka aplikacija imati svoju biblioteku. DLL se može smatrati kao dodatak našem programu ili cijelim Windowsima jer ga svi programi mogu koristiti.

Postoje dvije vrste biblioteka, 'object' i 'import'.

Object library je obična biblioteka dok import library služi da pokaže našem .exe programu gdje se nalaze funkcije u dll-u. Znači svaki DLL mora imati svoju **import library**.

Kako kreirati DLL?

Naučite C++ i onda slijedite uputstvo korak po korak:
To create a new dynamic link library (DLL) project

1. From the **File** menu, select **New** and then **Project…**.
2. From the **Project types** pane, under **Visual C++**, select **Win32**.
3. From the **Templates** pane, select **Win32 Console Application**.
4. Choose a name for the project, such as **MathFuncsDll**, and enter it in the **Name** field. Choose a name for the solution, such as **DynamicLibrary**, and enter it in the **Solution Name** field.
5. Press **OK** to start the Win32 application wizard. From the **Overview** page of the **Win32 Application Wizard** dialog, press **Next**.
6. From the **Application Settings** page of the **Win32 Application Wizard**, under **Application type**, select **DLL** if it is available or **Console application** if **DLL** is not available. Some versions of Visual Studio do not support creating a DLL project using wizards. You can change this later to make your project compile into a DLL.
7. From the **Application Settings** page of the **Win32 Application Wizard**, under **Additional options**, select **Empty project**.
8. Press **Finish** to create the project.

Sistemski pozivi i API kod Windowsa (Windows API/WinAPI)

Microsoft je definisao skup nazvan Win32 API procedura (Aplikacioni programski interfejs) koje programeri treba da koriste da bi dobili odgovarajuće usluge operativnog sistema. Ovaj interfejs je (djelimično) podržan od svih verzija Windows-a počev od Windows-a 95.
Broj Win32 API poziva je vrlo velik i mjeri se hiljadama. Mnogi od tih poziva koriste da bi se ažurirali pozivi sistema, ali daleko veći broj poziva se potpuno izvršava u korisničkom prostoru. Kao posljedica toga je činjenica da je **nemoguće vidjeti koji pozivi su pozivi sistema (tj. ono što odrađuje kernel), a koji nisu.**

Štaviše, ono što je poziv sistema u jednoj verziji Windows-a, može se odraditi u korisničkom prostoru drugog, i obrnuto.
Win32 API ima velik broj poziva za upravljanje prozorima, geometrijskim figurama, tekstom, fontovima, skrol barovima i slično, što se može (a najčešće i jeste) podvesti pod tipično korisničko okruženje.

WinAPI je Microsoftov osnovni skup interfejsa za programiranje aplikacija dostupnih u operativnim sistemima Microsoft Windows.
To je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operativnog sistema kao standardne biblioteke rutina (funkcija, procedura, metoda), struktura podataka, objekata i protokola. API je nezaobilazan u stvaranju novih aplikacija. Umjesto da se programi pišu novi iz temelja, programeri svoj rad nastavljaju i baziraju na radu drugih.
Podrška za programere je dostupna u obliku kompleta za razvoj softvera, Microsoft Windows SDK, koji pruža dokumentaciju i alate potrebne za izgradnju softvera zasnovanog na Windows API-ju i povezanim Windows interfejsima.

• Win16 je API za prve, 16-bitne verzije Microsoft Windows-a. Oni su se u početku nazivali jednostavno Windows API, ali su kasnije preimenovani u Win16 u nastojanju da se razlikuju od novije, 32-bitne verzije Windows API-ja. Funkcije Win16 API-ja nalaze se uglavnom u osnovnim datotekama OS-a: kernel.exe (ili krnl286.exe ili krnl386.exe), user.exe i gdi.exe. Uprkos ekstenziji datoteke exe, ovo su zapravo biblioteke sa dinamičkim vezama.

• Win32 je 32-bitni interfejs za programiranje aplikacija (API) za 32-bitne verzije Windows-a (NT, 95 i novije verzije). API se sastoji od funkcija implementiranih, kao i kod Win16, u sistemskim DLL-ovima. Osnovni DLL-ovi Win32 su kernel32.dll, user32.dll i gdi32.dll. Win32 je predstavljen sa Windows NT. Verzija Win32 koja se isporučuje sa Windowsom 95 prvobitno se nazivala Win32c, a c znači kompatibilnost. Ovaj termin je kasnije Microsoft napustio u korist Win32.

• Win32s je ekstenzija za Windows 3.1x familiju Microsoft Windows-a koja je implementirala podskup Win32 API-ja za ove sisteme. "s" označava "podskup".

• Win64 je varijanta API-ja implementirana na 64-bitne platforme Windows arhitekture (od 2021., x86-64 i AArch64). I 32-bitna i 64-bitna verzija aplikacija se još uvijek može kompajlirati iz jedne baze koda, iako su neki stariji API-ji zastarjeli, a neki od API-ja koji su već bili zastarjeli u Win32 su uklonjeni. Svi memorijski pokazivači su po defaultu 64-bitni (model LLP64), tako da se izvorni kod mora provjeriti kompatibilnost sa 64-bitnom aritmetikom pokazivača i po potrebi ponovo napisati.

Win32 API može da pravi sistemske pozive. Ovo znači da je programer u mogućnosti da (indirektno, preko API funkcije) kreira kernel objekte (fajlove, procese, thread-e, itd.) Svaki poziv koji kreira neki objekat, mora da vrati "palicu" (*handle*) procesu koji je napravio poziv i taj hendle se može koristiti naknadno za vršenje operacija nad tim objektom. Handle je vezan za proces koji je napravio objekat. Ovo ima za posljedicu da se taj handle za rukovanje tim objektom ne može direktno proslijediti drugom procesu. U pojedinim slučajevima se omogućava pristupanje i drugog procesa objektu, što se radi dupliciranjem handle-a.
Kada pričamo o objektima, treba napomenuti da **ne prave svi sistemski pozivi objekte** i da pored objekata kernela postoje i korisnički objekti. Jedini kernel objekti su oni koji treba da se imenuju, zaštite ili na neki način dijele.

Ako koristite Win OS je jedini način manipulisanja objektima je preko Win32 API poziva čime se izvrši određena akcija nad handle-om objekta. Ali , treba napomenuti da Windows 2000 istovremeno zapostavlja neke važne karakteristike objektno orijentisanih sistema a to su: polimorfizam i donekle naslijeđivanje.

Za potrebe Windows 32bitnog aplikativnog programiranja Microsoft je razvio **API - Application Programming Interface. Za 32-bitne platforme koristi se Microsoft®Win32®API**.
API sadrži skup funkcija, poruka (messages) i struktura koje su smještene u sljedećim dll- dinamičkim bibliotekama:
Kernel32.dll, Advapi32.dll, GDI32.dll, User32.dll, Crtdll.dll i shell32.dll.

API je realizovan kao set DLL-ova

Microsoft je dokumentovao svoje DLL fajlove, **što znači da je objavio šta rade i način kako ih prozvati** i to je sigurno jedan od razloga masovnog korištenja Windows-a, jer je relativno lako praviti aplikacije koje će optimalno koristiti mogućnosti Windows-a.
API funkcije su pisane u jeziku C. (Što logično znači da se mogu pozivati iz njega.)
Pojednostavljeno API je set potprograma koji omogućava direktan priključak na Windows.

Pogledajmo osnovne kategorije i funkcije WinAPI-ija.

API Funkcije

1.) Upravljanje prozorima (Window Management)

Funkcije za upravljanje prozorima namenjene su za stvaranje i korišćenje prozora za interakciju sa korisnikom u aplikaciji. Većina aplikacija će kreirati barem jedan prozor.

2.) Rad sa grafičkim uređajima (GraphicsDeviceInterface GDI)

Funkcije za rad sa grafičkim uređajima omogućavaju aplikacijama kreiranje grafičkog izlaza odnosno prikazana na displeju (monitor),stampacu i drugim uređajima. Korišcenjem GDI funkcija, mogu se crtati linije, tekst, bitmapslike (.bmp) i sl.

3.) Sistemski Servisi

Funkcije za rad sa sistemskim servisima pružaju aplikaciji pristup resursima računara i osobinama operativnog sistema, kao što su **memorija, fajl sistem (file systems) i procesi**.
Aplikacija koristi funkcije za rad sa sistemskim servisima za upravljanje resursima računara koji su potrebni aplikaciji za rad.
Na primjer

- funkcije za upravljanje memorijom (memory management functions) aplikacija koristi za alociranje i oslobađanje potrebne memorije,
- funkcije za upravljanje procesima i sinhronizaciju(process management and synchronization functions) za pokretanje i koordiniranje rada više aplikacija ili višestrukih niti (multiple threads of execution) unutar iste aplikacije.

4.) Multimedija

Multimedijske funkcije omogućavaju aplikaciji rad sa audio i video elementima.
Korišcenjem ovih funkcija, aplikacija može stvarati dokumente koji uključuju u sebi muziku, zvučne efekte, video klipove…

5.) Remote Procedure Calls (Pozivi udaljenih procedura)

RPC se koristi kod distribuiranih aplikacija.
Primjeri distribuiranih aplikacija su djeljene baze podataka (shared databases), udaljeni fajl serveri (remote file servers), udaljeni printer serveri (remote printer servers).

Kategorije API servisa

Funkcionalnost omogućena od strane Windows API-ja se **može** grupisati u sedam kategorija:

1.) **Osnovni servisi**

Omogućavaju pristup fundamentalnim resursima dostupnim Windows operativnim sistemu. Služe za rukovanje stvarima kao što su **fajl sistem, procesi(processes) i niti(threads), pristup Windows registriju(windows registry) i rukovanje greškama (error handling)**.
Sve funkcije za rukovanje navedenim su sadržane u **kernel.exe**, **krnl286.exe** ili **krnl386.exe** na 16 bitnim Windows operativnim sistemima i **Kernel32.dll** i **Advapi32.dll** na 32 bitnim Windows operativnim sistemima.

Kernel32.DLL je 32 bitna dinamička biblioteka u Windows 95,98 i Me. Dio je kernela operativog sistema. Ova biblioteka služi za rukovanje memorijom , ulazno izlaznim operacijama i interuptima (prekidima). Kada se startuje Windows operativni sistem, Kernel32.dll se smješta u zaštićeni dio memorijskog prostora, da slučajno neki drugi program ne bi mogao da se prepiše preko njega u memoriji.

2.) **Graphics Device Interface**

Pruža mogućnosti ispisa grafičkog sadržaja na standardne izlaze (monitore, štampače i ostale izlazne uređaje). Nalazi se u **Gdi.exe** na 16 bitnim Windows operativnim sistemima, i **Gdi32.dll** na 32 bitnim Windows operativnim sistemima.

3.) **User Interface**

Pruža mogućnosti kreiranja i upravljanja prozorima (screen windows) i najosnovnije kontrole nad njima, kao što su dugmad i skrolbarovi, prihvatanje sadržaja od miša i tastature i ostale osnovne funkcije povezane sa GUI dijelom Windows operativnog sistema.
Nalazi se **User.exe** na 16 bitnim Windows operativnim sistemima, i **User32.dll** na 32 bitnim Windows operativnim sistemima. Od pojave WindowsXP operativog sistema nalazi se u **Comctl32.dll** zajedno sa CCL-om(Common Control Library).

4.) **Common Dialog Box Library**

Omogućava aplikacijama standardne dialog prozore za otvaranje i snimanje fajlova, biranje boje i fonta itd…
Nalazi se u **Commdlg.dll** na 16 bitnim Windows operativnim sistemima i u **Comdlg32.dll** na 32 bitnim Windows operativnim sistemima. Pripada User Interface kategoriji API-ja.

5.) **Common Control Library**

Omogućava aplikacija pristup nekim naprednijim kontrolama omogućenim od strane operativnog sistema. Ovo uključuje status barove, progres barove, toolbarove i tabove.
Biblioteka se nalazi u dll fajlu **Commctrl.dll** na 16 bitnim Windows operativnim sistemima i u **Comctl32.dll** na 32 bitnim Windows operativnim sistemima. Pripada User Interface kategoriji API-ja.

6.) **Windows Shell**

Komponenta Windows API-ja koja dozvoljava aplikacijama da koriste funkcije omogućene od strane Shell-a operativnog sistema.
Nalazi se u **shell.dll** na 16 bitnim i u **shell32.dll**, **shlwapi.dll** na 32 bitnim Windows operativnim sistemima.Pripada User Interface kategoriji API-ja.

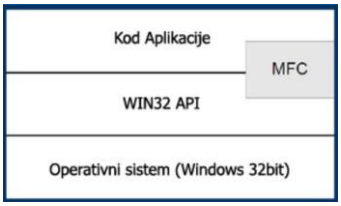
7.) **Network Services**

Omogućavaju pristup raznim mrežnim mogućnostima operativnog sistema. Kao pod-komponente sadrži NetBIOS, Winsock, NetDDE, RPC…

MFC

Windows programiranje osim korišćenjem API funkcija može vršiti i korišcenjem **Microsoft Foundation Class biblioteke (MFC)**.

MFC spadaju u klasu grafičkih biblioteka, uveden je u upotrebu davne 1992. godine. Microsoft je u međuvremenu razvio mnogobrojne alate i frejmvorke za razvoj, ali MFC i dalje ostaje u upotrebi. Uopšteno, MFC sadrži C++ klase koje opisuju objekte kao što su prozori, dijalog prozori (dialog boxes) i slični objekti važni za grafički korisnički interfejs.



MFC funkcije članice klasa pozivaju API funkcije i znatno olakšavaju kreiranje aplikacija u odnosu na direktno pozivanje API funkcija.

Mnoge od MFC funkcija dijele svoja imena s odgovarajućim API funkcijama. Jedna od karakteristika MFC-a je upotreba "Afx" kao prefiksa za mnoge funkcije, makroe i standardno prethodno kompajlirano ime zaglavlja "stdafx.h".

Komentar uz prethodni Windows i naredni Android OS

U ovom priručniku obrađeni su principi i funkcije operativnih sistema.
Date su i tri implementacije realizacije operativnih sistema: DOS, Windows i Android.
Namjera je da se na konkretnim primjerima ilustruju principi i omogući da se lakše shvati kako su operativni sistemi implementirani u praksi.
I kod Windows i kod Android OS mogu da posluže kao uvodna analiza za korisničke intervencije i pokažu put kako bi pristupili razvoju pojedinih dijelova ili aplikacija koje podržavaju operativni sistemi.

Android

Android je operativni sistem za mobilne uređaje, kao što su mobilni telefoni, tablet računari i laptopi sa ekranima osjetljivim na dodir (touch screen), prije svega za smartfone "pametne" telefone.

Android je zapravo mnogo više od toga: skup open source softwarea koji uključuje OS, middleware i ključne aplikacije, zajedno sa API library-ma za pisanje mobilnih aplikacija koje mogu oblikovati izgled i funkciju mobilnih uredaja. Google definiše Android kao prvu potpuno otvorenu platformu za mobilne uredaje, sav software potreban za moblni uređaj ali bez vlasničkih (kopirajt) ograničenja i zapreka koje koče razvoj i inovacije.

Android razvija Open Handset Alliance (podržana i finansirana od strane u Googla), i baziran je na Linux kernelu i GNU softveru.

Ovaj operativni sistem je 2007. razvila firma Android Inc. (koju je poslije kupio Google) i kasnije je proširen na.

Prema broju prodatih jedinica uređaja baziranih na Android OS, ovaj operativni sistem je na prvom mjestu u svijetu, ne samo na "smartphone" uređajima već i na svim računaraskim platformama uopšte. (2019. je pretekao Windowse.) Android je najprodavaniji OS u svijetu na pametnim telefonima od 2011., a na tabletima od 2013. Od maja 2021. imao je preko tri milijarde aktivnih korisnika mjesečno, najveću instaliranu bazu bilo kojeg operativnog sistema, a od januara 2021. , Google Play prodavnica je sadržavala preko 3 miliona aplikacija. Android 12.1/12L uključuje poboljšanja specifična za sklopive telefone, tablete, ekrane veličine desktopa i specijalno za laptope bazirane na Google pretraživaču Chrome tzv. Chromebookove. Posljednja verzija Android 13, objavljena je 15. avgusta 2022.

Glavna prednost Android operativnog sistema je što radi s otvorenim kodom (open source), što znači da svako može stvarati aplikacije. Svako može preuzeti njegov izvorni kod, urediti ga prema svojim zahtjevima i pokrenuti vlastiti prilagođeni (custom) ROM.

Činjenica da je njegov kôd otvoren, omogućava i niz drugih prednosti, prije svega mogućnost da se greške mogu brže pronaći i popraviti.

Kod Androida su sve aplikacije ravnopravne i sve se mogu izbrisati, zamjeniti i nadograditi. Ne postoje granice medu aplikacijama - aplikacije bez ograničeja, komuniciraju međusobno i razmjenjuju podatke. *Android sve aplikacije tretira jednako, dok Windows Mobile i Appleov iPhone, koji su bazirani na vlasničkom software-u, daju veći prioritet svojima (native) aplikacijama i ograničavaju komunikaciju izmedu native i third party aplikacija*.

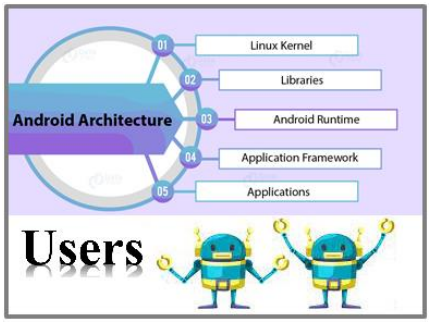
Na zvaničnoj Android stranici možete pročitati: Android je otvoren za sve: programere, dizajnere i proizvođače uređaja. To znači da više ljudi može eksperimentirati, zamišljati i stvarati stvari koje svijet nikada nije vidio.

> Da se ne pomisli da je Google "stvarni pristalica" besplatnih stvari:
> Većina Android uređaja se isporučuje s unaprijed instaliranim dodatnim vlasničkim softverom, prije svega Google Mobile Services (GMS) koji uključuje osnovne aplikacije kao što su Google Chrome, platforma za digitalnu distribuciju Google Play i pridruženu razvojnu platformu Google Play usluga.

Android je open i free, objavljen pod open source Apache licence i svi ga mogu slobodno downloadati i modifiikovati. Otvorenost je velika prednost Androida; prihvatile su ga velike kompanije takozvanog OHA: Open

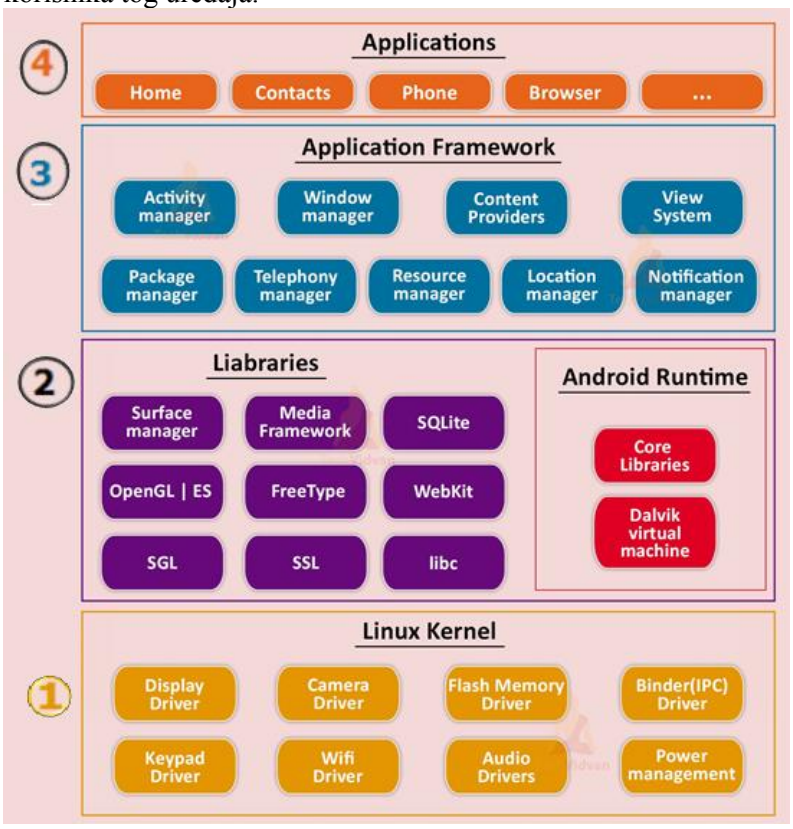
Handset Alliance (udruženje od preko 30 tehnoloških kompanija uključujući Motorolu, HTC, TMobile i ostale koje razvijaju uređaje koji koriste Android).

Android arhitektura – 5 komponenti Android arhitekture

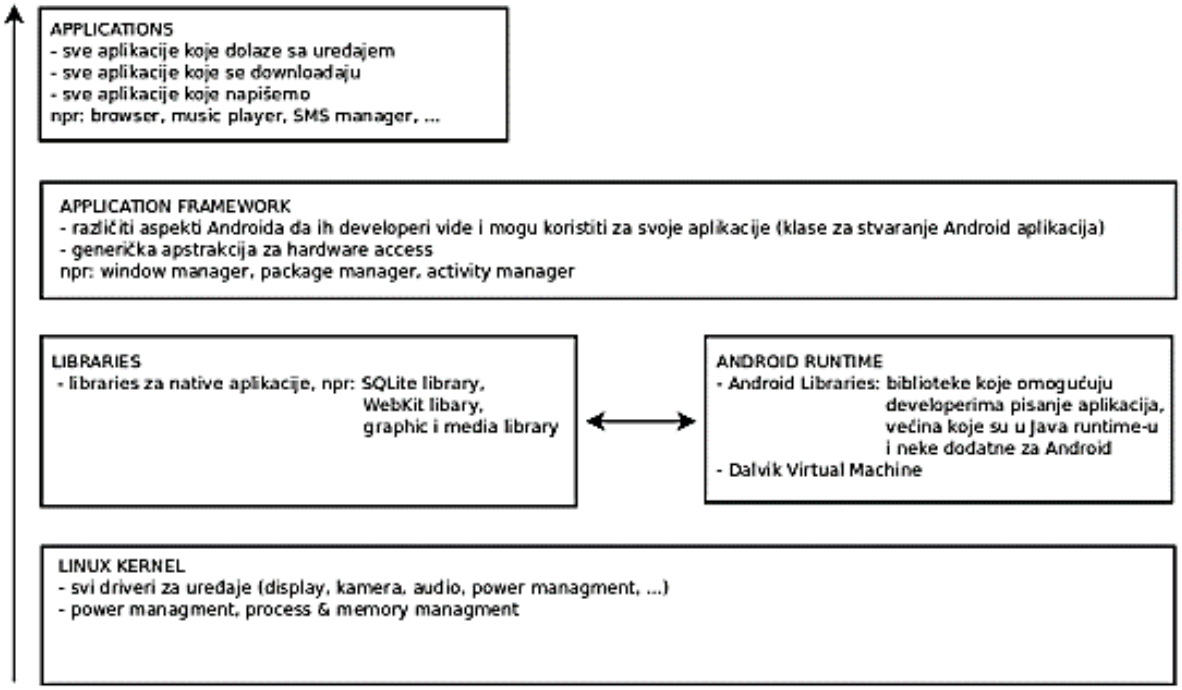


Android arhitektura odgovara Linux operativnom sistemu i kao podlogu ima sopstvenu verziju Linux Kernela.

U osnovi to je hibridni slojeviti operativni sistem koji se od 5 djelova koji su smješteni u 4 sloja za podršku svim potrebama Android uređaja i korisnika tog uređaja.

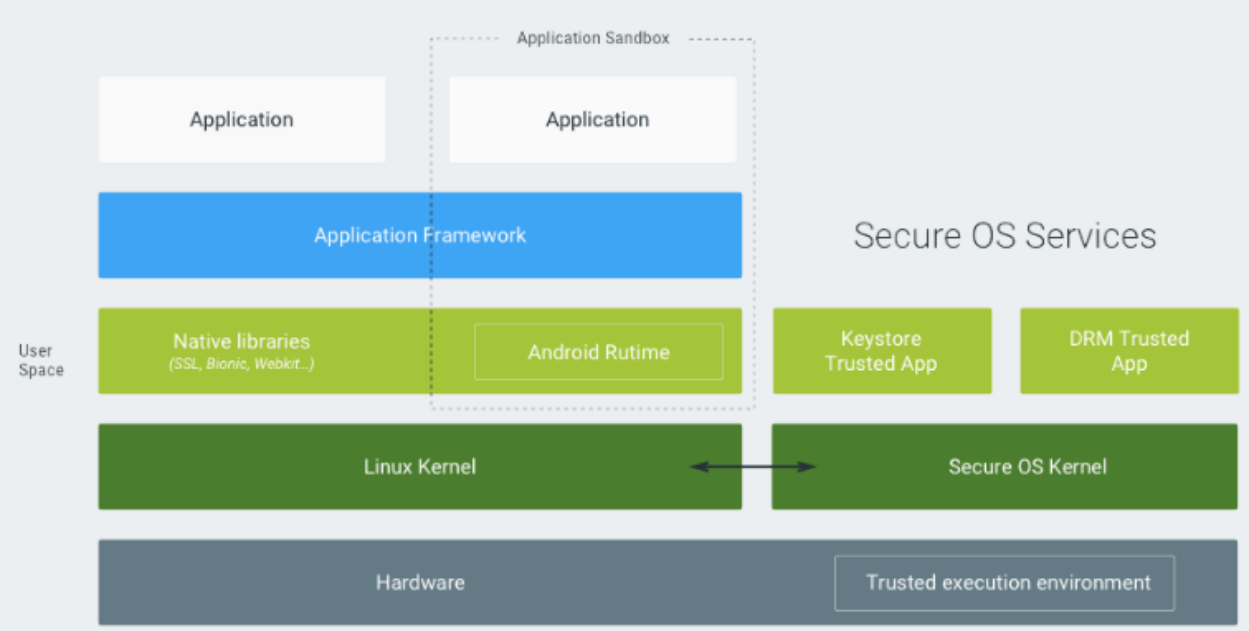


Komponente Android arhitekture



Arhitektura Android sistema prikazana po slojevima

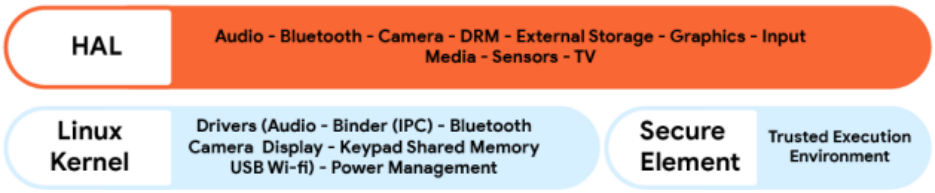
Na predhodnim slikama data je „uobičajena“ i standardna ilustracija arhitekture Android operativnog sistema, a malo drukčiju i zvaničnu ilustraciju (preuzetu sa oficijelnog Android sajta) daje ilustracija ispod:



Na njoj je naglasak na sigurnosti ovog sistema, što i jeste jedan od glavnih problema (ne samo ovog) operativnog sistema.

Linux kernel

Linux kernel je **najniži i najvažniji sloj** Android arhitekture i osnovni je dio Android arhitekture.



Ovaj sloj obezbjeđuje :

- Sigurnost
- Upravljanje procesima
- Upravljanje memorijom
- Upravljanje uređajima
- Multitasking

Poznat je i kao **apstraktni sloj androida** i u osnovi objedinjuje kernel i HAL interfejs. Odgovoran je za nivo apstrakcije između hardvera uređaja i gornjih slojeva Android arhitekture.

Sastoji se od drajvera uređaja kao što su kamera, fleš memorija, ekran, tastatura, Wifi itd.

Biblioteke-Libraries

Drugi sloj je podjeljen na dva dijela i sastoji od skupa biblioteka (Native Libraries)) i Android Runtime.

Android komponenta je napravljena korištenjem izvornih kodova i zahtijeva izvorne biblioteke, koje su napisane na C/C++, a većina biblioteka su biblioteke otvorenog koda.

Ovaj sloj obrađuje podatke koji su specifični za hardver. Evo pregleda 10 najvažnijih biblioteka koje su uključene u Android OS:

1) Osnovne biblioteke- Core Libraries

Grupa osnovnih biblioteka koje su prisutne u Android operativnom sistemu.

2) MediaFramework

Android izvorno podržava popularne medijske kodeke, što olakšava aplikacijama kreiranim na Android platformi da koriste/reprodukuju multimedijalne komponente bez upotrebe posebnih drajvera i aplikacija za njih.

3) SQLite

Android ima SQLite bazu podataka koja omogućava aplikacijama da imaju vrlo brzu funkciju izvorne baze podataka bez potrebe za bibliotekama trećih strana.

4) Slobodni tip-Freetype

Android dolazi s unaprijed instaliranim brzim i fleksibilnim mehanizmom za fontove. To omogućava programerima aplikacija da stiliziraju komponente svoje aplikacije.

5) OpenGL

Android dolazi sa OpenGL grafičkim sistemom. To je C biblioteka koja pomaže Androidu da koristi hardverske komponente u realnom vremenu prikazivanja 2D i 3D grafike.

6) SSL

Android dolazi s ugrađenim sigurnosnim slojem koji omogućava sigurnu komunikaciju između aplikacija na Androidu i drugih uređaja kao što su serveri, drugi mobilni uređaji, ruteri 6.

7) SGL

Android dolazi sa grafičkom bibliotekom implementiranom u kodu niskog-mašinskog nivoa koji efikasno renderuje grafiku za android platformu. Radi sa komponentama višeg nivoa Android grafičkog pajplajna-cjevovoda Android okvira.

8) Libc

Jezgro Androida sadrži biblioteke napisane na C i C++, koji su jezici niskog nivoa namijenjeni za ugrađenu upotrebu koji pomažu u maksimiziranju performansi. Libc pruža sredstva za izlaganje funkcija sistema niskog nivoa kao što su Threads, Sockets, IO i slično ovim bibliotekama.

9) Webkit

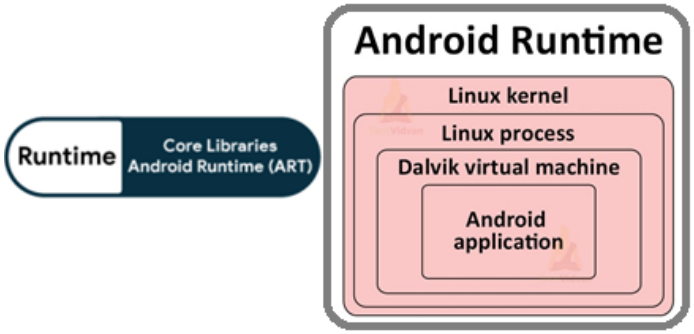
Ovo je motor pretraživača otvorenog koda koji se koristi kao osnova za pravljenje pretraživača. Zadani Android pretraživač prije verzije 4.4 KitKat ga koristi za renderiranje web stranica. Omogućava programerima aplikacija da renderuju web komponente u sistemu prikaza koristeći WebView. Ovo omogućava aplikacijama da integriraju web komponente u svoju funkcionalnost.

10) Surface Manager

Upravitelj površina je odgovoran za osiguravanje glatkog prikazivanja ekrana aplikacija. To radi sastavljanjem 2D i 3D grafike za renderiranje. Dalje to omogućava vršenjem baferovanja van ekrana.

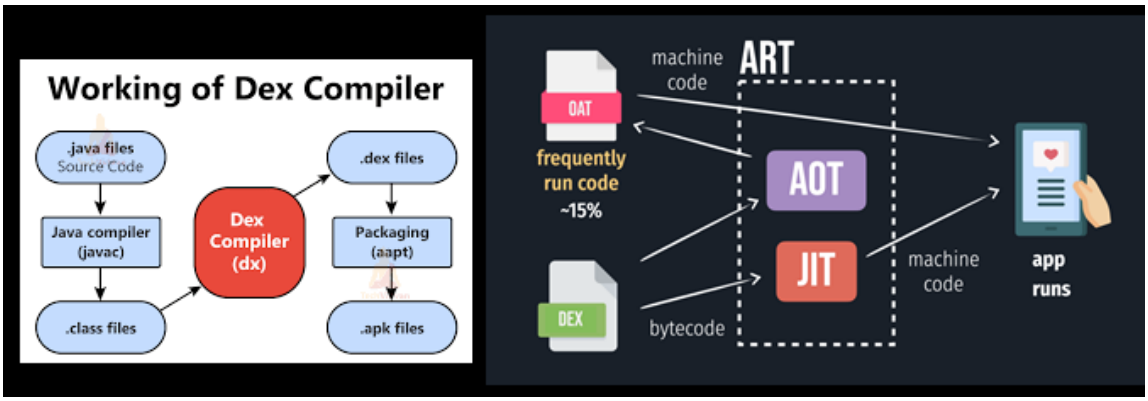
Android biblioteka je **strukturno ista kao i modul Android aplikacije**. Može uključivati sve što je potrebno za izradu aplikacije, uključujući izvorni kod, datoteke resursa i Android manifest. Međutim, umjesto kompajliranja u APK koji radi na uređaju, Android biblioteka se kompilira u Android arhivu (AAR) datoteku koju možete koristiti kao vezu za modul Android aplikacije.

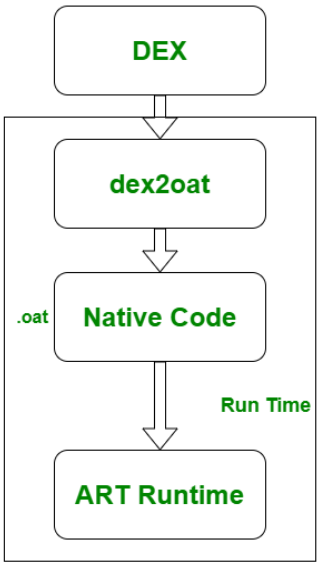
-ART- Android Runtime- Rantajm kontrola izvršavanja programa



Android runtime (ART) je upravljano vrijeme izvođenja koje koriste aplikacije i neke sistemske usluge na Androidu.

ART i njegov prethodnik Dalvik prvobitno su kreirani posebno za Android projekat. ART i Dalvik su kompatibilna vremena izvođenja koja pokreću Dex bajtkod, tako da bi aplikacije razvijene za Dalvik trebale raditi kada se pokreću s ART-om. ART uvodi kompilaciju unaprijed (AOT) koja može poboljšati performanse aplikacije. ART također ima strožu verifikaciju vremena instalacije nego Dalvik.





Ako znate programirati u Javi onda možete koristiti ovaj paket (ART) za kreiranje koda biblioteka.

Možda ste čuli za termine ***odeksiran i dedeksiran***. Virtualna mašina prevodi Java datoteke u Dex format koji **virtualna mašina izvršava i prevodi u binarni kod te na taj način može raditi na Linux kernelu da upravlja resursima hardvera, softvera, procesa i memorije**. Ono što se radi u ovom slučaju je da *uzmete mali dio aplikacije i zatim ga prekompajlirate, oni mogu nastaviti i napraviti dio svoje aplikacije optimiziran za rad na njihovom uređaju.*

Bajt kod generisan od strane Java kompajlera mora biti konvertovan u .dex datoteku Višestruke datoteke klasa se kreiraju kao jedna .dex datoteka i komprimirana .jar datoteka je veća od nekomprimovane .dex datoteke.

Android 5.0 "Lollipop" je prva verzija u kojoj je ART jedini uključeni runtime. Ono što **ART** sada radi je da **donosi aplikacije koje su u potpunosti kompajlirane kada se instaliraju na uređaj**. Dakle, veće performanse jer nema potrebe za konvertovanjem koda u bajt kod, a zatim kompajliranjem. Ali nedostatak je što vam treba više prostora za pohranu i malo duže za instalaciju zbog kompilacije tokom instalacije što znači da mora stalno biti aktivan (živjeti) na uređaju.

Okvir aplikacije -Application Framework-

Aplikacioni okvir prevodi kod koji su napisali programeri u mašinski kod koji vrši proračune i koristi komponente android okvira za kreiranje i povezivanje sa drugim android objektima i programima u funkcionalnu i efikasnu cjelinu.

Aplikacioni okvir izgrađen na sloju izvorne biblioteke pruža nam interfejs za programiranje aplikacije i usluge višeg nivoa. Takođe, karakteristike Android operativnog sistema su nam dostupne preko API-ja napisanih u obliku JAVA klasa. A Android programeri koriste ove usluge visokog nivoa za pravljenje aplikacija.

Android aplikacijski okvir je softverski alat koji omogućava programerima aplikacija da sastave gotov proizvod koji ispunjava zahtjeve vlasnika. Framework pruža osnove aplikacije, koje treba dopuniti grafikom, animacijom, posebnim karakteristikama i funkcionalnošću.

Takođe se sastoji od Android sloja apstrakcije hardvera (HAL) koji omogućava Android aplikacijskom okviru da komunicira sa drajverima uređaja specifičnim za hardver. Djeluje kao interfejs za implementaciju proizvođača hardvera. Android aplikacija koristi HAL API-je za dobivanje naredbi s različitih hardverskih uređaja.

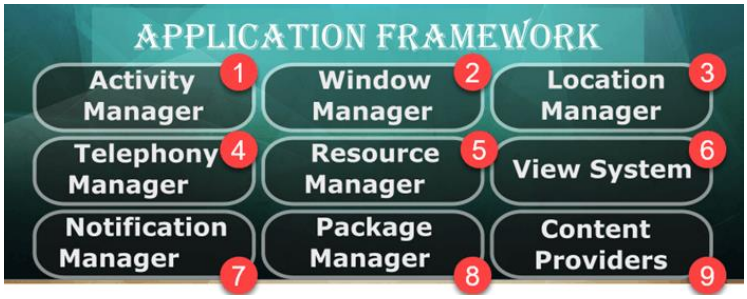
Okvir aplikacije sastoji se od sljedećih ključnih usluga:
Upravitelj aktivnosti: Metoda u ovoj klasi koristi metode testiranja i otklanjanja grešaka.

Dobavljač sadržaja: Pruža podatke iz aplikacije u druge slojeve.

Resource Manager: Omogućava pristup resursima koji nisu kodirani.

Upravitelj obavijesti: Korisnici dobijaju obavještenje o svim radnjama koje se dešavaju u pozadini.

Sistem pogleda: djeluje kao osnovna klasa za widgete i odgovoran je za rukovanje događajima.



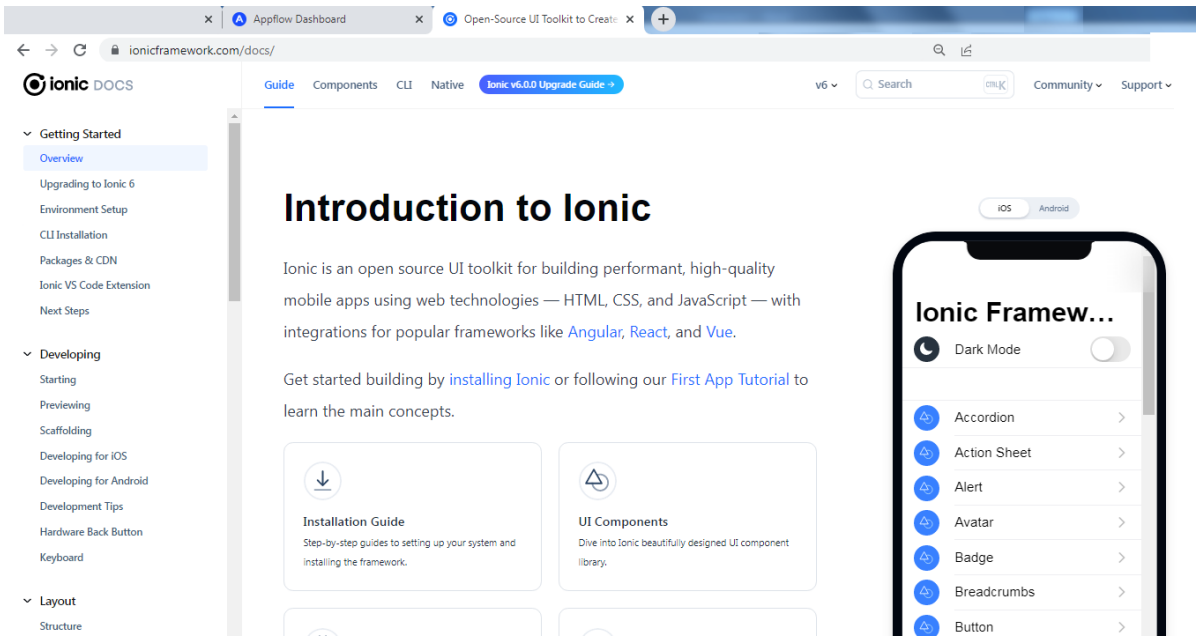
9 standardnih alata koje sadrže aplikacioni okviri

Aplikacijski okviri su dizajnirani da pojednostave proces razvoja aplikacije i olakšaju upravljanje, modificiranje i ispravljanje grešaka u budućnosti. Android framework ima strukturu komponenti i interfejs koji pruža mogućnost ponovnog korištenja dizajna i koda aplikacije i centraliziranja osnovnog koda na jednoj lokaciji. Osnovna ideja Android framework-a da pomogne programerima da kreiraju strukturu aplikacije i napišu kod mnogo brže.

Formalno bi bilo potrebno razlikovati SDK (komplet za razvoj softvera) i okvire. SDK u sebe uključuje okvir (kao obično najvažiju koponentu, pa se najčešće koriste kao sinonimi.

Postoje destine (a možda i stotine) razvojnih okvira koji podržavaju razvoj android aplikacija Na slici možete vidjeti 5 preporučenih (kao najboji) od strane businessofapps.com-a za 2022. godinu:

Kao najkorištenije (bar prema većini izvora) je Iconic. Napravljeno je oko 4 miliona aplikacija baziranih na Ionic, sa više od 5 miliona programera u preko 200 zemalja širom svijeta. Iconic je integrisan mnogim uslugama, uključujući Google Play, Instagram i druge platforme.



Aplikativni nivo -Applications- Programiranje Android aplikacija

Ovo je najviši sloj Android arhitekture.

Ovaj sloj se sastoji od izvornih Android aplikacija i instaliranih aplikacija trećih strana. Oni su u paketu za Android i sve aplikacije koje treba instalirati su napisane samo u ovom sloju.

Aplikacije mogu biti sistemske. ili korisničke, ili od strane OEM proizvođača uređaja (telefona). Aplikacije se obično objavljuju na Google Playju.

Android, prema zadanim postavkama, dolazi sa skupom aplikacija koje čine android uređaje upotrebljivim od početka i kupovine. Evo pregleda tih aplikacija:

1. Početna-Home: Početna stranica na Androidu se sastoji od ikona pokretača za najčešće korištene aplikacije kojima bi krajnji korisnik mogao željeti brz pristup. Možete pokrenuti aplikacije klikom na pokretače ovih aplikacija. Na samom vrhu ekrana imate widgete koji prikazuju mrežu, nivo baterije, datum i vrijeme.
2. Kontakti- Contacts: Android, prema zadanim postavkama, pruža način za pohranjivanje i preuzimanje kontakata. Kontakt informacije se dijele u drugim aplikacijama radi poboljšanja funkcionalnosti.
3. Poruk- Messages e: Android pruža mogućnost slanja i primanja SMS poruka.
4. E-pošta-Email: Android dolazi sa izvornom podrškom za usluge e-pošte. Za postavljanje Android uređaja potreban je Gmail račun. Postavljanje Gmail-a aktivira druge komponente zavisne od e-pošte na Android uređajima. Neke funkcije zavisne od e-pošte uključuju mehanizme sigurnosti i oporavka. Još jedna funkcija zavisna od e-pošte je pristup Play Store-u, tržištu za Android aplikacije.
5. Pregledač-Browser: Android dolazi sa zadanim pretraživačem.

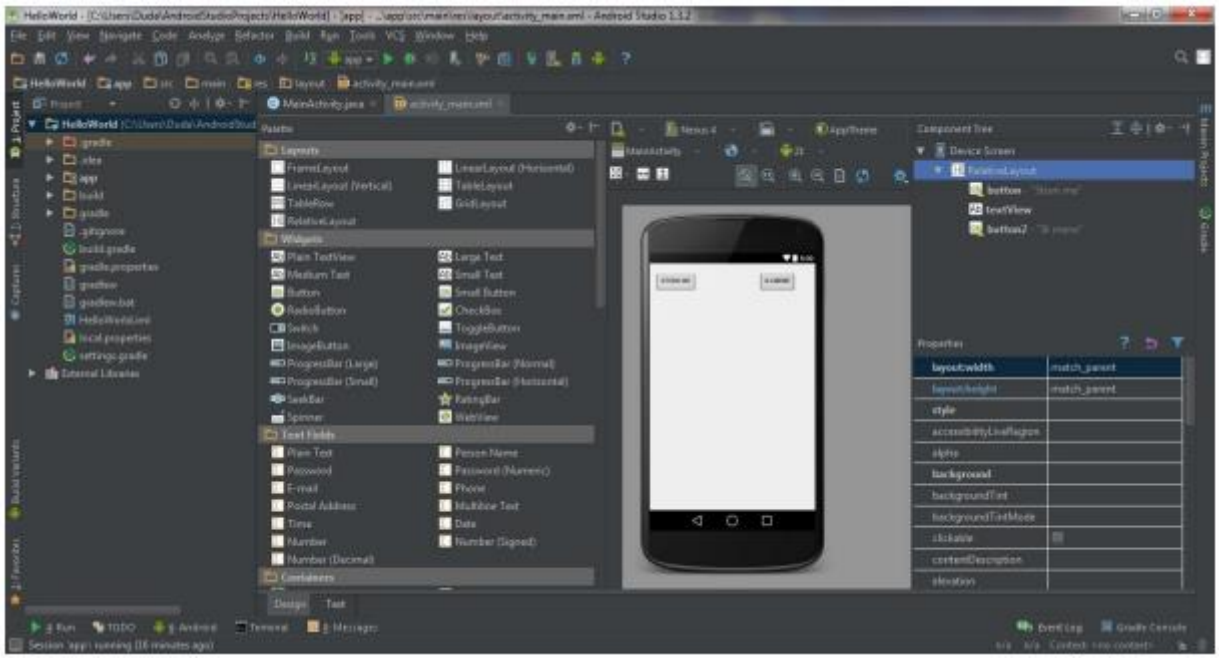
6. Ladica za obaveštenja- Notification Drawer: Prevlačenjem nadole po ekranu otkriva se fioka sa obaveštenjima. Pruža događaje aplikacije kojih bi korisnik trebao biti svjestan. Iznad obavještenja nalazi se skup prečica do nekih često korištenih postavki uređaja koje korisnici mogu mijenjati.

Ove postavke uključuju prekidače za uključivanje i isključivanje za različite hardverske komponente kao što su Bluetooth i Wifi. Dug pritisak na ove prekidače-događaje omogućava nam da se krećemo do njihove stranice sa konfiguracijama.

Za kreiranje Android aplikacije može se koristi neki Aplikacioni okvir (koji ima uključenu opciju razvoja aplikacija), ili preporučeno neki namjenski editor i razvojno okruženje.

Možete pronaći reklame softverskih paketa koje kažu da je moguće kreirati android aplikaciju bez poznavanja programiranja (npr. korištenjem Tu-App.net). Jeste moguće, ali za kreiranje ili krajnje jednostavne, ili loše aplikacije. Poželjno je da poznajete objekto orjentisano programiranje, Javu odnosno C++, da bi imali kontrolu nad kodom koji ste kreirali.

Kao razvojno okruženje **ranije** se preporučeno koristila **Eclipse**. Eclipse se i dalje može koristiti, ali je izgubio razvojnu podršku (i update) Google-a. Projekti izrađeni u Eclipse-u se lako mogu prenijeti u **Android Studio**, koji je oficijelno IDE za Android platformu potpuno besplatan putem Apache licence 2.0 i više platformi (Microsoft Windows, macOS i GNU / Linux).



Tipičan interfejs Android Studia

Android Studio omogućuje ne samo editovanje-uređivanje koda, već uklanjanje grešaka, alate za dizajn i fleksibilan sistem kompilacije, kao i trenutno stvaranje, postavljanje i testiranje aplikacije.

Za razvoj aplikacija Android Studio zahtjeva instalaciju Java Development Kit-a (JDK).

Aplikativni nivo je **sloj sa kojim krajnji korisnici komuniciraju**. Na ovom sloju programeri aplikacija objavljuju svoje aplikacije za pokretanje.

Aplikativni sloj se često naziva nivoom korisnika, za razliku od slojeva ispod koji su uglavnom podešeni za razvoj sistemskih aplikacija. Mi smo ovdje IPAK izvršili dodatnu diferncijaciju i izdvojili kao poseban korisnički nivo.

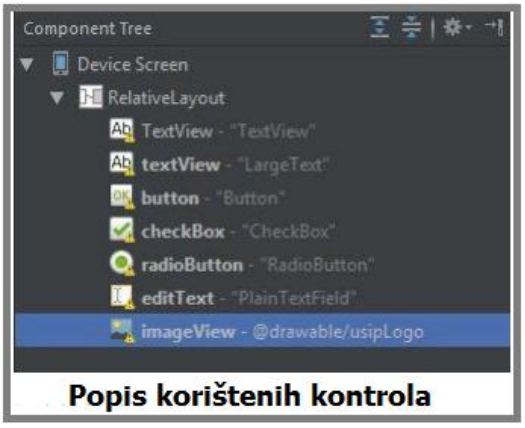
Editor interfejsa Android Studia

Sredinom Android Studia dominira njegov WYSIWYG editor interfejsa aplikacija. Sastoji se od palete s mnogim kontrolama (kontrole rasporeda, prikazi teksta, slika, Web stranica, sata, kontrole izbora itd.) koje je moguće mišem odvući na virtualni pregled interfejsa desno od palete. Na tom pregledu se kontrole pozicioniraju u odnosu na ostale kontrole i rubove ekrana te im se mogu uređivati osnovni atributi.

Istovremeno se u pozadini generiše XML kod koji opisuje raspored elemenata i njihove atribute. (Gassner, Developing Android Apps Essential Training).

Dostupne su kontrole unutar palete organizirane u nekoliko kategorija:

- Layouts – kontejnerski objekti koji diktiraju razmještaj kontrola na ekranu. Korijen interfejsa je layout koji može sadržavati druge layoute stvarajući hijerarhiju.
- Widgets – najčešće kontrole koje se nalaze u većini dizajna. U njih spadaju gumbi, pregledi teksta, slika i Web stranica, elementi formulara, trake napretka i slično.
- Text Fields – polja za unos teksta koja se razlikuju po tipu podataka za koje su predviđeni.
- Containers – kontejneri koji grupiraju slične kontrole koje imaju zajednička svojstva.
- Date & Time – sve vezano za datum i vrijeme (sat, kalendar, zaporni sat…).
- Expert – specijalizirane kontrole koje se koriste samo u posebnim slučajevima.
- Custom – sadrže kontrole vlastite izrade, često iz drugih projekata.



Virtualnom uređaju za pregled interfejsa moguće je odabrati model, orijentaciju njegovog zaslona i verziju Android sistema za koji će se prikazivati pregled.

Ponuđeno je nekoliko modela uređaja s najčešćim veličinama i rezolucijama ekrana te je čak moguće odabrati više uređaja odjednom kako bi se osigurao što bolji prikaz izgleda interfejsa aplikacije na što većem broju različitih uređaja.

Desno od pregleda dizajna interfejsa nalazi se popis svih kontrola koje se koriste, zajedno s njihovim najvažnijim atributom (npr. prikazani tekst ili putanja do slike).

Ispod popisa korištenih kontrola nalazi se editor svojstava pomoću kojega se za odabranu kontrolu može odrediti mnoga svojstva bez potrebe za ručnim pisanjem XML koda. Ipak, programeri češće koriste XML način editora i ručno defišu željene atribute.

Prateći programi Android Studia

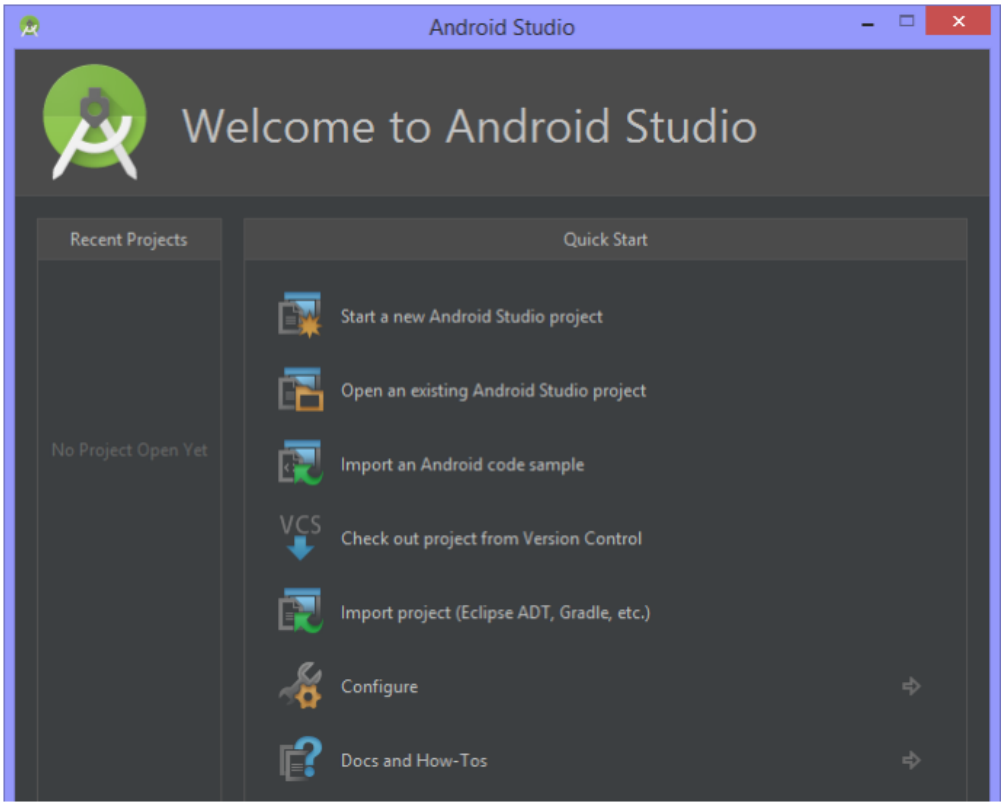
Kao i Eclipse prije njega, Android Studio dolazi s nekoliko posebnih programa koji olakšavaju česte radnje u programiranju. Neki od njih su isti oni programi koji su pratili prethodnika, neki su unaprijeđeni, a nekima je funkcionalnost integrirana u Android Studio.

Najvažniji su

- SDK Manager, kojim je moguće pregledati instalirane Android platforme, nadograditi ih, instalirati nove platforme ili posebne komponente. npr. za podršku Google Play usluga
- AVD Manager je alat koji služi za upravljanje virtualnim Android uređajima koji se pokreću uz pomoć Android emulatora. Omogućuje pregled, stvaranje i brisanje virtualnih uređaja.
- Device Monitor koje je moguće pronaći u Tools (alati) meniju pod stavkom Android. To je je alat za pregled stanja uređaja koji su spojeni na računalo. Omogućuje praćenje memorije i pokrenutih procesa na uređaju te korištenje mreže. Prikazuje se i LogCat prozor kojeg je moguće filtrirati te tako pratiti željenu aplikaciju.

Stvaranje Android aplikacije

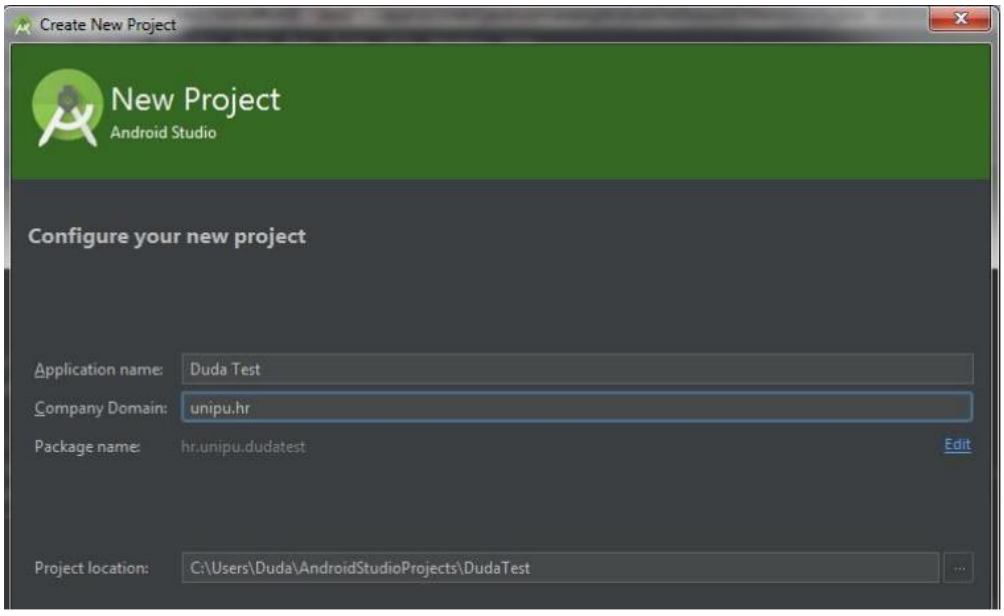
Stvaranje nove aplikacije najčešće započinje stvaranjem novog projekta u Android Studiu.



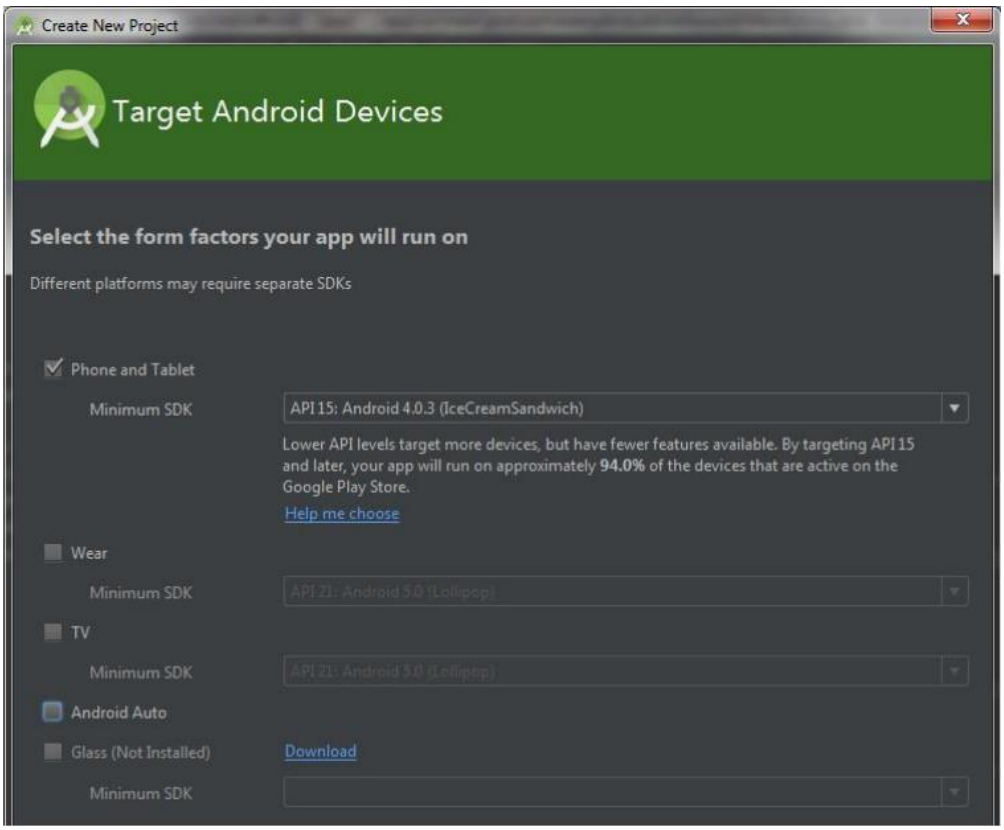
Početni ekran sa dobrodošlicom u Android Studio

Već na početnom ekranu dobrodošlice programer može odabrati želi li započeti novi projekt, otvoriti postojeći projekt ili ga stvoriti uvozom već postojećih projektnih datoteka iz drugih izvora (npt. Već pominjane Eclipse).

Čarobnjak za izradu novog projekta traži ispunjavanje određenih polja kako bi stvorio i podesio novu aplikaciju. Sam čarobnjak je prvi korak ka stvaranju nove aplikacije i prikazat će se u cjelosti na sljedećim slikama uz prateći tekst.



Prvi korak čarobnjaka za izradu novog projekta



Drugi korak čarobnjaka za izradu novog projekta

Ako želite da nastavite sa izradom Android aplikacije možete koristiti Wizard, ali preporučujemo da se dodatno edukujete i nastavite sa programiranjem u Android okruženju.

Aktivnosti - Activity- kod Android Studia

Prije kraja ovog uvodnog upoznavanja sa programiranjem Android aplikacija korištenjem Amdroid Studia, par rječi o osnovnim aktivnosti koje definišu redosljed i način izvršenja događaja unutar prozora.

Stvaranje Android aplikacije obično počinje stvaranjem glavne aktivnosti (Activity). Nakon toga postavljaju se servisi (Services), primatelji emitiranja (BroadcastReceivers) i pružatelji sadržaja (ContentProviders) i svi se zajedno povežu pomoću namjera (Intents). Zajednički naziv za navedene djelove je komponenta. Navedene komponente su one koje se najčešće koriste. Aktivnosti su odgovorne za korisničke interfejse, a servisi implementiraju operacije koje se pokreću u pozadini.

Primatelji emitiranja čekaju događaje sistema, dok pružatelji sadržaja spremaju podatke aplikacije.

Aktivnost je obično jedan ekran kojeg korisnik vidi na uređaju. Aplikacija se obično sastoji od više aktivnosti pa se one izmjenjuju kako korisnik koristi aplikaciju.

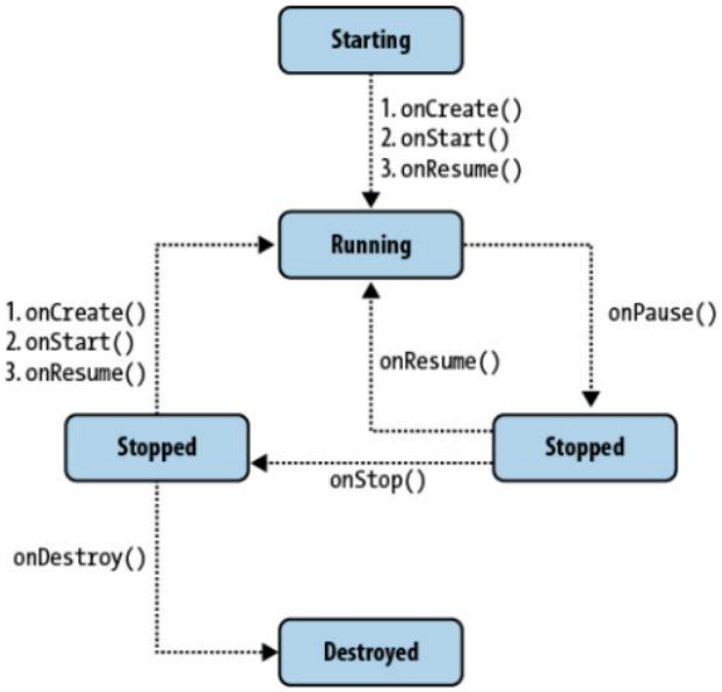
To čini aktivnost vizualne najistaknutijom komponentom.

Kao što se Web projekt sastoji od više stranica, tako se aplikacija sastoji od više aktivnosti.

Dok Web projekt ima početnu stranicu, aplikacija ima glavnu (main) aktivnost koja se prikazuje po pokretanju. Kao što je moguće prebacivanje s jedne Web stranice na drugu putem navigacije i poveznica, tako je moguće i mijenjanje prikazane aktivnosti na mobilnom uređaju.

Aktivnosti mogu imati nekoliko stanja i na programeru je prepušteno da odredi kako će se aplikacija ponašati zavisno o stanju određene aktivnosti.

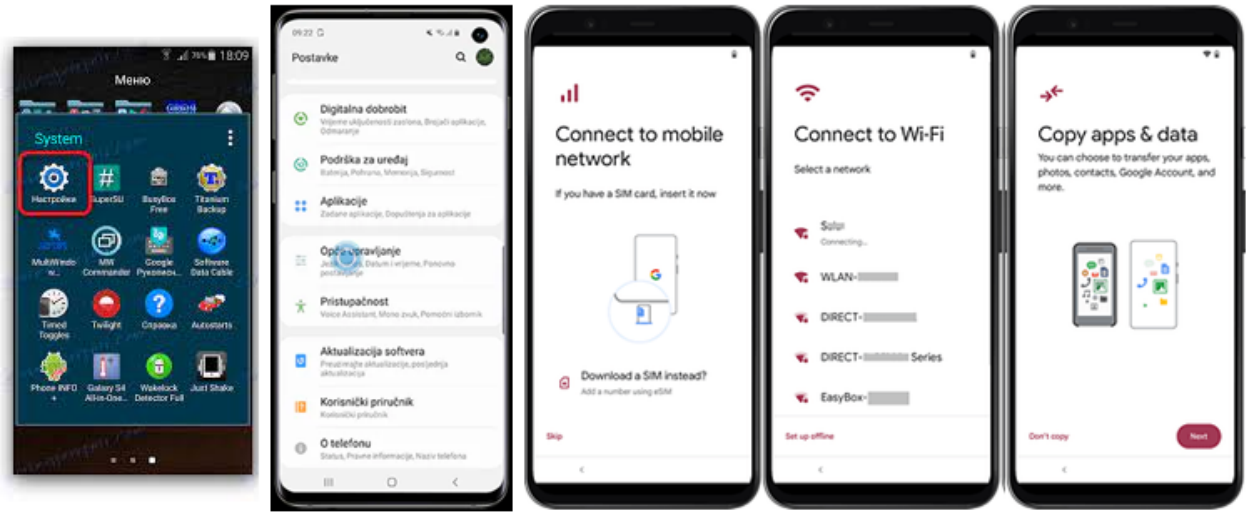
Životni ciklus aktivnosti i tranzicijske metode koje se javljaju prikazani su na slici ispod.



Životni ciklus aktivnosti
(uporedi sa konceptom i principima ranije opisanih procesa)

Korisnički nivo

Da biste koristili (ili dizajnirali i inovirali) predhodne slojeve potrebno je da se bavite programiranjem. Nije neophodno da budete profesionalac, ali neophodno je znanje koje prosječan korisnik operativnog sistema Android implementiranog na mobilni telefon NEMA. Za prvi sloja potrebno je i poznavanje hardvera, a za ostale softvera i programiranja i barem osnovnih hardverskih komponenti ili platformi na kojima je realizovan. Preporučeno bi bilo korištenje Java programskih alata (a moguće je i nekih drugih, recimo C++), nekog Android frejmvorka.
A za korisnički nivo, kome pristupa nekoliko milijardi ljudi, **dovoljno je da kupite telefon**.



„Obični“ korisnici komuniciraju sa aplikativnim slojem. Kad kupe i aktiviraju telefon (pristupe nekoj od mreža mobilne telefonije) lako će pronaći standardne alate za podešavanja i korištenje koje ovdje nećemo objašjavati. Početna stranica na Androidu se sastoji od ikona pokretača za najčešće korištene aplikacije kojima bi krajnji korisnik možda želio brz pristup. *Njihov osnovni set dat je u predhodnom poglavlju (i sivom okviru).*
Možete pokrenuti aplikacije klikom na pokretače ovih aplikacija. Na samom vrhu ekrana imate widgete koji prikazuju mrežu, nivo baterije, datum i vrijeme...

Ma vidjeli ste, Androd je stvarno jednostavan, da ne kažemo user friendly.

Izvori i reference

Materijali preuzet sa
https://razno.sveznadar.info/
http://software.sveznadar.info
https://www.suse.com/suse-defines/definition/modular-operating-system/
https://www.researchgate.net/profile/Dragan-Pleskonjic-2/publication/325766392_Operativni_sistemi_koncepti/links/5ff9b46a45851553a02f11c8/Operativni-sistemi-koncepti.pdf
http://perun.pmf.uns.ac.rs/old/budimac/os/geller-html/index.html
https://web.math.pmf.unizg.hr/~karaga/android/android_skripta.pdf
http://www.stsmihajlopupin.edu.rs/dokumenta/Memorija.pdf
http://www.stsmihajlopupin.edu.rs/dokumenta/Sistemi_datoteka.pdf
http://www.znanje.org/abc/tutorials/operatingsystems/01/operatingsystems.htm
(Autori: Eugenija Mihal i Bojan Stojković)
http://es.elfak.ni.ac.rs/Papers/DBarac%20-%20RTOS%20za%20male%20embedded%20sisteme.pdf
www.programmer.co.me/USB.../USB%20Monitor%20Document.doc
http://etsracunari.wordpress.com/operativni-sistemi/ciljevi-i-zadaci-operativnih-sistema/
www.mginformatika.com/
http://www.infoteh.rs.ba/rad/2012/RSS-6/RSS-6-10.pdf
http://www.informatika.buzdo.com/s734.htm
http://www.richardgoyette.com/SEMINIXSecureComputingBootProcess.html
http://bs.wikipedia.org/wiki/Operativni_sistem
http://bs.wikipedia.org/wiki/Windows_7
http://hr.wikipedia.org/wiki/Microsoft_Windows
http://en.wikipedia.org/wiki/Unix
http://itc.wikidot.com/istorija-operativnih-sistema
http://itc.wikidot.com/slojeviti-operativni-sistemi-layered-systems
http://carstvolokvanja.com/knjige/programiranje/programski.pdf
http://msdn.microsoft.com/en-us/library/aa450748.aspx
http://windows.microsoft.com/en-US/windows/home
http://technet.microsoft.com/en-us/magazine/2008.03.kernel.aspx?pr=blog
https://srednjaskolabrus.edu.rs/2020/10/26/arhitekture_op_sistema77356/
https://hr.admininfo.info/programaci-n-en-dispositivos-m-viles-con-android
https://core.ac.uk/download/pdf/197587391.pdf
https://www.geeksforgeeks.org/difference-between-dalvik-and-art-in-android/
https://data-flair.training/blogs/android-architecture/
https://techvidvan.com/tutorials/android-architecture/
https://www.android.com/what-is-android/
https://www.guru99.com/android-architecture.html