OPERACIJSKI SISTEMI:
Oeracijski sistem je množica programov, ki rač. ponujajo kontrolne in podporne funkcije in nam tako omogočajo nadzor in komunikacijo s programsko opremo(wind,linux).Aplikacijska programska oprema naredi računalnik bol uporaben,sama ne komunicira z V/I napravam, to delaOS.Aplikacijska prog. oprema kumonicira z OS z jezikom, ki je specifično za vsak OS.
VRSTE OPERACIJSKIH SISTEMOV:
Enoopravilni(DOS):V določenem trenutku opravljajole eno opravilo.Ne znajo zaščititi enega programa pred drugim, ne more omejiti aplikacije in tako lahko uniči drugo aplikacijo,ki teče.
Večopravilni(win):so sposobni navidezno istočasno izvrševat več opravil.To se doseže tako, da se en proces izvaja določeno št. časa, potem priklopi in izvaja drugi proces.Če je to opravljeno dovolj hitro, uporabnik ne opazi ker preklopi v 1/100 sek(čas med priklopom).
Mnogouporabniški(Linux): Isti rač. lahko istočasno uporabljavečje število uporabnikov.Na isti rač. se prijavi več uporabnikov(nap.preko mreže).Bistvena zahteva je večja varnost in večja stabilnost kot pri navadnih večopravilnih.
Večprocesorski: Pospirajo paralelnost izvajanja procesov. To podpirajo vsi(unix,linux,win),So večopravilni mnogouporabnišk pri procesorsko potratnih zadevah(grafika) in pri strežnikih. Poznamo **Simetrični:**procesi so razporejeni enakomerno preko vseh CPUjev.**Asimetrični:** en procesor za OS in komunikacijoV/I enotami,ostali procesorji za uporabniške programe.
Navidezni stroji: Aplikacije so namenjene postavljanju navideznih rač. na istem rač.
Mrežni OS: temeljijo na ideji terminalskega dela.OS naložen na centralnem strežniku, uporabniki uporabljajo terminale, kjer terminali skrbijo le za vnos in prikaz podatkov.
STROKTURA OPERACIJSKEGA SISTEMA:
Opravljanje procesov(od nje je odvisna stabilnost in učnikovitost OS.),upravljanje s primarnim pom(RAM), upravljanje s sekundarnim pom(trdi disk), upravljanje z V/I napravami,upravljanje z datotečnim sistemom(logična organizacija podatkov na sekunadrnem pomn.), sistem zaščite(stabilnost sistema –kateri uporabniki dostopi do aplikacij..), sodelovanje v mreži(omogoča prerazporeditev dela), interpretirenje ukazov(omogoča pogovor med uporabinkom in OS).
Prednosti in slabosti: Uporabnik nima dostopa do hardwere-a.Programi željo po uporabi hardvere-a samo posredujejo OS, ta potem to uresniči ali pa tudi ne..Stvar je počasnejša.Stabilnejši sistem: ne pride do sporov med programi oz. Da programi ne zahtevajo kaj, kar bi povzročilo nestabilnosti.
UPORABNIŠKI IN SISTEMSKI REŽIM: Uporabniški prog. ne smejo uporabljati vseh ukazov,ki jih je CPU zmožna izvršiti.**Sistemski režim** je močnejši in nevarnejši,lahko pa uporablja vse možne ukaze za CPE.Uporabniški program klice servis in izvrši hardware.Preklop men uporabniškim in sistemskim režimom pride zaradi treh razlogov.1.sistemski klic,2 prekinitev-mehanizem za vključevanje V/I naprav.3.Pasti: izvor je v programih, obnašajo se kot navadne prekinitve.
SODOBNEJŠA IZVEDBA OS.:
Je bolj stabilen,bolj fleksibilen in malce počasnejši.Težnja proizvajalcev je v to smer, samo napaka v mikrojedru zamaje celoten sistem, napake v posameznih strežnikih lahko odpravimo brez ponovnega zagona celotnega sistema.Servisi niso vezani eden na drugega.
Ukazni interpretor: Del OS viden uporabnik.Omogoča uporabniku zagon, ustavitev aplikacij, preklope med njimi ter kreiranje komunikacije med aplikacijami.Obstaja textovni način(konzola) in Grafični vmesniknapr. Windows(kliki miše se pretvarjajo v ukaze,ki jih ukazni interpretor zna razbrat in jihz tudi izvrši)
ZAGON OPERACIJSKEGA SISEMA:
Vklop računalnika: Zagon BIOSA (pomn. ki ohrajnja svojo vsebino),POST(Preveri hardwarskw kompunente),Pričetek **iskanja** zagonskega programa (najprej na disk,ko se ta

najde se prenese v RAM in ta program povzame nalogo zagona OS),ko se OS naloži nam je predano opravljanje s sistemom.To poteka enako za Linuxe in Wind., le da za linuxe se nalagajo drugačne dat. **V LINUXu** ti da možnost zagona OS, možnost izbire.Jedro operacijskega sistema se naloži v pomnilnik, v tem času so neke pikice na enkranu.Ko naloži, se začne pogleda hardware(auto-probing).Naprej init- zažene strežniške procese,pregleda če je disk vreden(če ne scandisk).Strežniški procesi ne berejo s tipkovnice,nič ne izpisujejo,a vseeno delajo, so vedno v pomnilniku.Potem je prijava v sistem(getin+login).

PROCES:

Programski proces je pravzaprav odvijanje nekega programa skupaj z okoljem, ki je za to odvijanje potrebno(programske strokture,podatki,sistemske zmogljivosti).

Paralelni procesi: So lahko odvisni ali neodvisni drug do drugega(potrebujejo občasno sinhronizacijo).Problematični kader potrebujejo za svoje delovanje iste zmogljivosti-konkorenčni procesi.Nap. 2 procesa želita hkrati brati isti del diska.

STANJE PROCESA:

New(sprejeto)->Propravljen(zaključen dogodek)->Čakajoč(čakaj dogodek)<- Tekoč(iztop)->Končan (med Pripravljen in tekoč je (->uvrstitev razvrčevalnika, <- prekinitev))

Nov: zagon procesa.Programi v delovnem pom.Proces dobi PID.

Pripravlnej: PID-proces ID; Procesi ki bi se lahko izvajali a nimajo na voljo procesorskega časa.

Uvrstitev razvrčevalnika: Sistemski proces katerega naloga je da vzame tekoč proces, ga da v stanje pripravljenosti in da drugi pripravljeni proces v izvajanje.

Tekoč:je proces katerega trenutno izvaja procesorLahko ga proces zaključi (**Končan:** proces sprosti resurse,katere je ta proces uporabljal).Lahko pa pride do časovnega prekinitvenega signala (nazaj v pripravljen, kjer čaka ponovno na procesorski čas, pri čemer si zapomni kje je ostal).Procs zahteva resurs, ki mu ni na voljo, se ne more izvajati naprej in ni pripravljen, ker na nekaj čaka. Da se v stanje **Čakajoč** kjer ostane dokler razloga za to stanje niso rešeni, ko se reši gre nazaj med pripravljene.**Problem stradanja:** Za resurse, katere mnogi potrebujejo! Če se postaviš na konec vrste, lahko da ga ne bo več ko prideš do njega.Zato včasih gre direkt iz čakajoč v teko.

RAZVRČANJE PROCESOV:

Pri tem se upošteva več parametrov:**Poštenost:**Obravnava vse procesorje enako(v teoriji),nekateri procesi so pomembnejši.**Učinkovitost:** Skušamo čim koristneje izrabit sistemske resurse in zamenjati čas, ki ga sistem porabi sam zase.**Odzivnost:** Preklopi naj bodo hitri, da imamo občutek da se proces izvaja brez prekinitev (1/100 sek).

SISTEMI:

ROUN ROBIN:je osnova za večino ostalih načinov.Proces čaka na izvajanje, če se ne izvede do konca, se vrne na začetek in ponovno čaka na izvajanje.Zelo pošten način,vse procese obravnava enako.

Večnivojsko algoritemsko razvrščanja: Procesi na ravni se izvajajo le če so vse ravni nad njegovo prazne.daljši programi tako izgubljajo na prioriteti in spustijo naprej krajše programe.Tako se krajši prog. prej izvršijo,daljši programi pa se le malce dlje izvajajo.
Problem: Problem stradanja: Če dokaj pomemben a malce daljši proces pade na nižjo raven.Če se pojavljajo novi, kratki procesi, bo daljši proces dolgo čakal.Problem se odpravi tako, da se meri čas, koliko časa proces čaka.Če je ta proces predolgi, se program prestavi v višjo raven.

NITI:

Ima svoj sklad, svoje registre a si z ostalimi nitmi skupnega procesa deli skupni naslovni prostor in globalne spremenljivke.Imajo dosto do skupnih datotek, imajo pravice dostopa do vsega znotraj tega procesa.S tem se poveča prepustnost sistema.Več zahtevkov se lahko hkrati izvaja.Programer mora urediti, da se dve niti ne zaletitahkrati hočeta uporabljati isti

resurs.Vsaka nit zahteva določene sistemske resurse.Sistemski klici nam omogočajo opravljanje z niti.
Implementacija niti: Statistična (niti se kreirajo v fazi prevajanja), Dinamična (niti se kreirajo v fazi izvajanja).
STATIČNA:jedro izbira med procesi, run time sistem pa je prekapljal med nitmi procesa,ki je bil na vrsti za preklop.Pozitivna lasnost je da se preklopi v uporabniškem režimu (so bitni). Slaba lasnost je pa da če ima proces le eno nit, ni preklapljanja.
DINAMIČNA:Jedro preklaplja med niti,traja dlje časa(priklopi iz uporabniškega v strežniški način delovanja).Preklopi možni med vsemi nitmi, ne dodeljuje se čas proces, ampak niti. Osnova za dodeljevanje procesorskega časa so niti.Vsak prog. ima vsaj eno nit, kreiramo iz funkcije main.

PORAZDELJENI SISTEMI:

Vzamemo več delovnih postaj, jih povezat v omrežju in jih pripravit do sedolovanja na dolocenem procesu.To naj bi zagotavljalo transparentnost,feksibilnost(če odstraniš eno postaja,sistem ni prizadet), zanesljivost in učinkovitost.
Mikrojedro: Prenaša klice med aplikacijskimi in strežniki, delno upravlja s pomnilnikom.
SISTEMSKI klici sedaj preko mrežne tečejo na druge rač. Pri tem se uporabnik ne zaveda, da sistemski klici potujejo drugam, zato da bo software mislil da še vedno dela na eni mašini.
Uporaba decentraliziranih algoritmov: Noben rač. nima inf. O celotnem stanju sistema. Izpad enega rač. ne blokira algoritma.Rač se odloča na podlagi inf. Ki jih trenutno ima.
Sistemski klici v normalnih sistemih: Ko kličemo funkcijo, se v sklad shrani vrednost spremenljivke in vrnitveni naslov, kamor naj se proces vrne ko zaključi z funkcijo.
Sistemski klici v porazdeljenih sistemih: Argumenti se zapakirajo v štrclju odjemalca, posredujejo mikro jedro, ta ga preko mreže pošlje jedru strežnika, ki ga posreduje štrcelju strežnika, ta pa argumente pošlje klicani proceduri.Nazaj se proces ponovi.

DVOSLOJNA ARHITEKTURA:

Se pojavi z podatkovno revolucijo.Včasih so se programi pisali za specifičen primer.En program je vseboval vse potrebne podatke kot tudi uporabniški vmesnik.Ko potrebuje več ljudi iste podatke, je te treba izločiti-SUPB omogoča dostop do istih podatkov večim ljudem.**Problem** se pojavi ko pride do spremembe zahtev nap. v podjetju je potrebno dosti dela za prilagoditev podatkovne baze.Ko pride nova verzija je potrebno nadgraditi aplikacijo vseh odjemalcev.Več odjemalcev, bolj obremenjene komunikacijske poti.

TROSLOJNA / VEČSLOJNA ARHITEKTURA:

To so prezentacijska logika, poslovna logika, podatki.Rešitev za problem dvoslojne aritekture je prav ta.Obstaja poleg poslovne logike še aplikacijski strežnik.Aplikacija je na strežniku, kdorkoli jo bo lahko uporabljal, vedno bo zadnja verzija za vse.

OPRAVLJANJE Z PRIMARNIM POMNILNIKOM:

Asociacija v času prevajanja: Programer mora določit kje v pomn. se bo stvar nahajala.Se uporablja v nižjih programskih jezikih.Uporabnost: ne če bo program deloval na sistemu, kjer teče operacijski sistem.To naredi OS. Ko pišeš programe za sisteme baz OS-razni mikrokontrolerji,ki komunicirajo razne naprave.
Asociacija v času nalaganja: Je največkrat uporabljena.Nalagalnik določi lokacijo,kam bo naložil program in ustrezno spremenil dele kode programa, da bo ta še vedno deloval. Ob preverjanju se označijo potrebni deli in po defontu so ti naslovi postavljeni pod predpostavko da bo program na lokaciji 0. Loader vrednosti naslova priševa dehansko začetno lokacijo.
Asocijacija v času izvajanja: Program se lahko premika po naslovnem prostoru tudi v času izvajanja.Logični pomiki povzročeni s spremembo logičnega naslova pomnilniške lokacije. Navidezni pomn. ki dela z navideznimi naslovi, prirejeni fizičnim naslovom.Za transormacijo naslovov skrbi preslikovalnik, ki omogočapomike naslovov tudi v času izvajanja.CPU dela z

navideznimi naslovi, v pomn. so fizični naslovi. Preslikovalnik določi navidezni naslov vsakemu fizičnemu naslovu in z temi dela CPU.
Dinamično nalaganje in povezanje: Ne potrebujemo istočasno vsega v pomn.Več procesov lahko želi uporabljati iste module. Pznamo **Klasično:** exe file:zaporedje ukazov (vsebuje vse ukaze z knjižnicami in se v celoti naloži ko ga zaženeš).**Dinamični:**Tu imaš samo osnovo ostalo je pa v modulih,ki se naložijo samo kadar se potrebujejo.Namesto modula se na njegovo mesto naloži štrcelj, ki povzroči da program misli da je modul že ima.Prvič ko ta modul želimo uporabit, se ta tudi dejansko naloži in naslednjič bo modul že pripravljen.Vsakič ko se knjižnica nalaga, štrcelj preveri če ta že obstaja na disku.Tako lahko več programov uporablja isto knjižnico.V štrclju piše tudi katera verzija knjižnice je potrebna.V knjižnici so podatki o verziji in združljivosti za nazaj, informacija o verzijah omogoča sobivanje različnih verzij na istem rač. Nova verzija ne povozi stare.