**CS230 SKRIPTA – BY OTO 5460**

**L01**

1. **Šta je centralizovani računarski sistem?**

Centralizovani računarski sistem je arhitektura u kojoj je sav softver, podaci i kontrola koncentrisana u jednom centralnom računaru ili serveru. Svi korisnici pristupaju resursima preko tog centralnog čvora.

1. **Koje su prednosti decentralizovanih računarskih sistema u odnosu na centralizovane?**

Decentralizovani računarski sistemi nude bolju skalabilnost i otpornost na otkaze u poređenju sa centralizovanim sistemima. Svaki čvor u decentralizovanom sistemu može funkcionisati nezavisno, što doprinosi većoj pouzdanosti i smanjenju rizika od kvara.

1. **Šta je distribuirani računarski sistem i koja je njegova osnovna karakteristika?**

Distribuirani računarski sistem je arhitektura u kojoj se procesi i resursi računarskog sistema nalaze na više fizički razdvojenih računara koji su povezani mrežom. Osnovna karakteristika distribuiranih sistema je transparentnost, tj. sposobnost da se korisnicima i aplikacijama sakrije da se procesi i resursi nalaze na različitim računarima.

1. **Kako se postiže transparentnost u distribuiranim sistemima?**

Transparentnost u distribuiranim sistemima postiže se kroz upotrebu middleware-a, sloja softverske infrastrukture koji pruža standardizovan interfejs aplikacijama. Middleware omogućava aplikacijama da pristupaju procesima i resursima distribuiranog sistema na isti način bez obzira na njihovu fizičku lokaciju.

1. **Koje su četiri vrste transparentnosti u distribuiranim sistemima?**

Četiri vrste transparentnosti u distribuiranim sistemima su: transparentnost pristupa, lokacije, relokacije i migracije. Transparentnost pristupa se odnosi na sakrivanje razlika u načinima pristupa resursima. Transparentnost lokacije se odnosi na sakrivanje fizičke lokacije resursa. Transparentnost relokacije se odnosi na pomeranje objekata bez uticaja na korisnike. Transparentnost migracije podržava mobilnost procesa i resursa bez prekida komunikacije.

1. **Kako se ostvaruje transparentnost pristupa u distribuiranim sistemima?**

Transparentnost pristupa ostvaruje se kroz standardizaciju interfejsa za pristup resursima distribuiranog sistema. To znači da aplikacije pristupaju resursima na isti način bez obzira na njihovu fizičku lokaciju ili tehničke specifikacije.

1. **Koja je uloga midlvera u distribuiranim sistemima?**

Midlver je sloj softverske infrastrukture koji se nalazi između transportnog i aplikativnog sloja distribuiranog sistema. Njegova uloga je da pruži standardizovan interfejs aplikacijama i omogući transparentan pristup procesima i resursima distribuiranog sistema.

1. **Šta je transparentnost replikacije i zašto je važna u distribuiranim sistemima?**

Transparentnost replikacije se odnosi na sakrivanje činjenice da postoje više kopija resursa ili da više procesa radi u zaključanom režimu, omogućavajući da jedan preuzme kontrolu ako drugi ne uspe. Ovo je važno jer omogućava veću pouzdanost i dostupnost sistema kroz redundanciju, a korisnici ne moraju biti svesni ove složenosti.

1. **Šta je transparentnost konkurencije i kako se postiže u distribuiranim sistemima?**

Transparentnost konkurencije se odnosi na to da korisnici ne primete da drugi koriste iste resurse na kojima oni rade istovremeno. To se postiže kroz mehanizme zaključavanja koji omogućavaju da se resursi koriste na kooperativan način, bez konflikata ili primetnog ometanja korisnika.

1. **Kakva je uloga maskiranja kvarova u distribuiranim sistemima i koje su poteškoće u ostvarivanju ovog cilja?**

Maskiranje kvarova je ključna za održavanje pouzdanosti distribuiranih sistema, omogućavajući da korisnici ili aplikacije ne primete kada deo sistema ne radi kako treba, i da se sistem automatski oporavi od otkaza. Glavna poteškoća je razlikovanje između mrtvih i sporo reagujućih procesa, kao i nemogućnost preciznog utvrđivanja uzroka kvara u složenim distribuiranim okruženjima.

1. **Šta znači otvorenost distribuiranih sistema i zašto je važna?**

Otvorenost distribuiranih sistema se odnosi na sposobnost sistema da pruži komponente koje se lako mogu koristiti ili integrisati u druge sisteme, kao i da podržava komunikaciju sa komponentama van samog sistema. Ovo je važno jer omogućava fleksibilnost, interoperabilnost i mogućnost prilagođavanja sistema promenljivim zahtevima i tehnološkim trendovima.

1. **Šta podrazumeva pouzdanost distribuiranih sistema i kako se može meriti?**

Pouzdanost distribuiranih sistema se odnosi na sposobnost sistema da kontinuirano pruža usluge u skladu sa očekivanjima, čak i u prisustvu kvarova ili otkaza. Pouzdanost se može meriti kroz veličine kao što su srednje vreme do otkaza (MTTF), srednje vreme do oporavka (MTTR) i srednje vreme između otkaza (MTBF), što omogućava procenu performansi i otpornosti sistema na greške.

1. **Koje su ključne tehnike skaliranja distribuiranih sistema i kako se razlikuju?**

Ključne tehnike skaliranja distribuiranih sistema uključuju vertikalno skaliranje (povećanje kapaciteta postojećih resursa) i horizontalno skaliranje (dodavanje novih resursa radi proširenja kapaciteta sistema). Vertikalno skaliranje se odnosi na nadogradnju postojećih komponenti, dok horizontalno skaliranje podrazumeva dodavanje novih čvorova u sistem radi distribucije opterećenja i povećanja performansi.

------------------------------------------------------------------------------------------------------------------------------------------

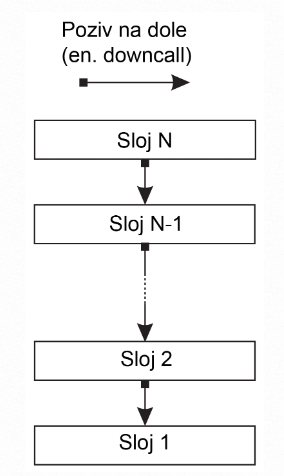
------------------------------------------------------------------------------------------------------------------------------------------

**L02**

**SLOJEVITA ARHITEKTURA**

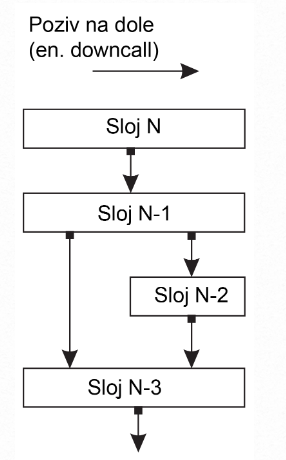
1. **Kako je organizovana jednosmerna arhitektura bez preskakanja sloja i u kojem kontekstu je često upoređena?**

Jednosmerna arhitektura bez preskakanja sloja organizovana je tako da komponente komuniciraju isključivo ka sledećem nižem sloju, slično kao u mrežnoj komunikaciji. Ova organizacija omogućava jednostavnu strukturu gde se komponente oslanjaju na komunikaciju samo sa slojem ispod sebe, često upoređujući se sa standardnom organizacijom u mrežnoj komunikaciji.



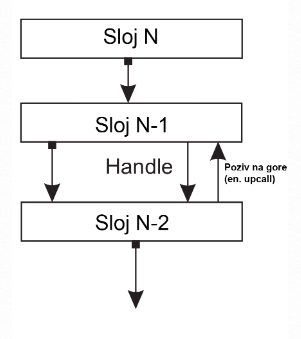
1. **Kako se koristi jednosmerna arhitektura sa preskakanjem sloja i koja je uloga slojeva u takvoj organizaciji?**

Jednosmerna arhitektura sa preskakanjem sloja koristi se kada aplikacija koristi specijalizovane biblioteke koje su implementirane korišćenjem drugih biblioteka. Slojevi su organizovani tako da aplikacija koristi biblioteke na nižim slojevima, ali može direktno komunicirati sa specijalizovanim bibliotekama bez prolaska kroz sve slojeve. Na primer, aplikacija A se implementira na sloju N−1, specijalizovana matematička biblioteka na sloju N−2, dok je zajednička OS biblioteka na sloju N−3.



1. **Kako se definiše dvosmerna arhitektura i u kojim situacijama se može primeniti?**

Dvosmerna arhitektura se javlja kada niži sloj inicira zahtev prema višem sloju. Ovo se često dešava kada operativni sistem ili neki drugi niži sloj signalizira događaj, nakon čega viši sloj izvršava odgovarajuću operaciju koju je prethodno preneo referencu. Na primer, kada operativni sistem signalizira pojavu događaja, može pozvati korisnički definisanu operaciju koja je prethodno preneta referencom, obično nazvanom ručkom (handle).

****

**MONOLITNA ARHITEKTURA**

1. **Kako se može definisati monolitna arhitektura i kako funkcioniše (primer)?**

Monolitna arhitektura je tradicionalni pristup razvoju aplikacija u kojem su svi delovi aplikacije integrisani u jednu celinu. U ovom modelu, aplikacija se sastoji od jednog softverskog modula koji obuhvata sve funkcionalnosti, poput korisničkog interfejsa, poslovne logike i pristupa podacima.

Na primeru veb sajta za rezervaciju hotela, monolitna arhitektura podrazumeva da se svi delovi aplikacije, poput pretrage hotela, procesa rezervacije, interakcije sa bazom podataka i eksternim servisima, nalaze u jednom softverskom modulu. Svaki zahtev korisnika prolazi kroz sve ove delove aplikacije unutar istog sistema.

1. **Kako se prednosti monolitne arhitekture manifestuju u procesu razvoja aplikacija?**

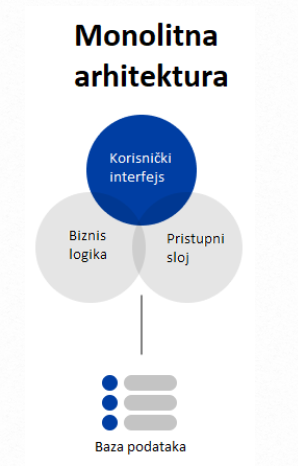
Prednosti monolitne arhitekture uključuju jednostavnost razvoja, organizaciju koda na jasan i strukturiran način, jednostavno skladištenje podataka u jednu bazu, lako raspoređivanje aplikacije i relativnu skalabilnost, što omogućava jednostavno održavanje aplikacije od strane jednog tima.

1. **Koje su glavne izazove i nedostatke monolitne arhitekture u procesu razvoja i održavanja aplikacija?**

Glavni izazovi monolitne arhitekture uključuju poteškoće pri izvođenju velikih promena, nestabilnost aplikacije pri promenama u velikom obimu, kolateralnu štetu koja može biti generisana od manjih promena, nekontrolisani rast aplikacije i organizaciju koda, kao i poteškoće u održavanju i skaliranju kako aplikacija postaje sve kompleksnija.

1. **Koja je glavna mana monolitne arhitekture?**

Monolitna arhitektura se suočava s izazovom rastuće složenosti i skaliranjem aplikacije jer sve funkcije i komponente aplikacije deluju unutar istog koda. To znači da kako aplikacija postaje veća i kompleksnija, teže je održavati čist i organizovan kod. Skaliranje monolitne aplikacije takođe postaje izazovno jer je teško dodavati nove resurse ili instance aplikacije bez povećanja kompleksnosti i rizika od grešaka.



**MIKROSERVISNA ARHITEKTURA**

1. **Koje su ključne karakteristike mikroservisne arhitekture u pogledu komunikacije i organizacije servisa?**

Mikroservisi su najčešće procesi koji komuniciraju putem mreže, koristeći protokole nevezane za određeni tip tehnologije. Takođe, organizovani su oko funkcionalnosti i mogu se posebno isporučivati, što ih čini lako zamenjivim.

1. **Kako se aplikacije zasnovane na mikroservisnoj arhitekturi razlikuju od tradicionalnih monolitnih aplikacija?**

Aplikacije zasnovane na mikroservisnoj arhitekturi prirodno se nameću modularnom strukturom i podržavaju kontinuiranu isporuku softvera. Za razliku od monolitnih aplikacija, izmena jednog dela aplikacije zahteva ponovno bildovanje i isporuku samo jednog ili malog broja servisa. Takođe, mikroservisne aplikacije se pridržavaju principa sitno granuliranog interfejsa, poslovno orijentisanog razvoja, poliglotnog programiranja i decentralizovane kontinuirane isporuke, što ih čini agilnijim i skalabilnijim.

1. **Kako razdvajanje dužnosti u mikroservisnoj arhitekturi doprinosi brzini projekta i kontroli evolucije aplikacije?**

Razdvajanje dužnosti u mikroservisnoj arhitekturi omogućava posebnim timovima da nezavisno razvijaju svaki mikroservis, što ubrzava proces projekta. Svaki servis se fokusira na jednu ulogu, pridržavajući se principa jedne odgovornosti. Ovo rezultira labavo povezanim servisima koji omogućavaju lokalizovan uticaj promena unutar konkretnog servisa, dok ostatak aplikacije ostaje stabilan i netaknut.

1. **Koji su glavni izazovi i nedostaci mikroservisne arhitekture?**

Glavne mane mikroservisne arhitekture obuhvataju nelogično razdvajanje, više mrežnih interakcija, problem skladištenja i deljenja podataka, kao i složenost testiranja. Nelogično razdvajanje može dovesti do kompleksnosti i težine održavanja, dok više mrežnih interakcija može povećati kompleksnost i smanjiti performanse aplikacije. Skladištenje i deljenje podataka mogu biti izazovni zbog potrebe za održavanjem nezavisnosti između mikroservisa, a testiranje aplikacija može biti složeno zbog velikog broja komponenti. Upravljanje ovim izazovima zahteva pažljivo planiranje, implementaciju pouzdanih mehanizama komunikacije, sinhronizacije podataka i testiranja, kako bi se osigurala stabilnost i efikasnost mikroservisne arhitekture.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L03**

**RESTFUL ARHITEKTURA (REPRESENTATIONAL STATE TRANSFER)**

1. **Koje su ključne karakteristike RESTful arhitekture i zašto je postala popularna?**

Ključne karakteristike RESTful arhitekture obuhvataju jedinstvenu šemu imenovanja resursa, isti interfejs za sve servise, potpunu opisnost poruka i izvršenje bez stanja. Postala je popularna zbog svoje jednostavnosti i širokog prihvatanja za veb integraciju servisa.

1. **Koji su prednosti i nedostaci RESTful arhitekture u poređenju sa drugim pristupima poput SOAP-a?**

Prednosti RESTful arhitekture uključuju jednostavnost i fleksibilnost, dok nedostaci mogu obuhvatiti komplikacije kod distribuiranih transakcija i kompleksnih komunikacionih šema. SOAP, s druge strane, definiše komunikacioni protokol i propisuje format poruka, što može rezultirati većim brojem parametara po operaciji.

1. **Kako se razlikuju RESTful i SOAP interfejsi, kao i koje su implikacije tih razlika na funkcionalnost aplikacija poput Amazon S3?**

RESTful interfejsi obično nude samo osnovne operacije poput GET, PUT, POST i DELETE, dok SOAP interfejsi mogu imati više operacija i ograničenje u broju parametara po operaciji. Ove razlike mogu uticati na kompleksnost i efikasnost komunikacije između aplikacija. Na primer, u slučaju Amazon S3, REST interfejs se oslanja na URI-je za opis resursa, dok SOAP interfejs ima ograničenje u broju parametara po operaciji.

**PRETPLATNICKE ARHITEKTURE**

1. **Kakve su osnovne karakteristike koordinacije u pretplatničkim arhitekturama?**

Pretplatničke arhitekture karakteriše koordinacija procesa putem komunikacije i saradnje, gde se procesi posmatraju kao autonomno operativne jedinice. Koordinacija može biti direktna, putem poštanskog sandučeta, zasnovana na događajima ili sa deljenim prostorom za podatke.

1. **Kako se razlikuju sistemi sa deljenim prostorom za podatke od sistemima zasnovanih na događajima?**

Sistemi sa deljenim prostorom za podatke omogućavaju procesima komunikaciju putem struktuiranih podataka, tzv. torki, koje se smeštaju u zajednički prostor. S druge strane, sistemi zasnovani na događajima omogućavaju procesima objavljivanje obaveštenja o događajima na koje su zainteresovani, a pretplatnici se pretplaćuju na određene događaje.

1. **Koje su prednosti i izazovi pretplatničkih arhitektura?**

Pretplatničke arhitekture nude fleksibilnost i skalabilnost u distribuiranim sistemima, omogućavajući efikasnu komunikaciju između procesa. Međutim, izazovi se javljaju u implementaciji efikasnog uparivanja pretplata sa obaveštenjima, što može postati usko grlo, naročito kada su u pitanju sigurnost i privatnost podataka.

1. **Sta predstavlja magistrala dogadjaja (event-bus)?**

Magistrala događaja (event bus) je mehanizam koji omogućava komunikaciju između različitih delova softverskog sistema putem razmene događaja. To je kanal preko kojeg se događaji mogu emitovati (publikovati) od strane jednog dela sistema i primati (pretplatiti se) od strane drugih delova sistema koji su zainteresovani za te događaje.

**ORGANIZAIJCA MIDLVERA**

1. **Kakvu ulogu ima midlver u distribuiranim sistemima i kako se organizuje?**

Midlver predstavlja sloj softvera iznad operativnih sistema računara u distribuiranim sistemima, omogućavajući efikasno deljenje i raspodelu resursa preko mreže. Organizuje se tako da sakriva razlike u hardveru i operativnim sistemima od aplikacija i pruža sredstva za komunikaciju između komponenti različitih distribuiranih aplikacija.

1. **Koje usluge pruža midlver u distribuiranim sistemima i kako se razlikuju od usluga operativnih sistema?**

Midlver pruža usluge za međuaplikacionu komunikaciju, zaštitu, maskiranje i oporavak od grešaka, i druge. Glavna razlika u odnosu na usluge operativnih sistema je to što su usluge midlvera prilagođene mrežnom okruženju. Većina tih usluga je korisna za mnoge aplikacije, čime midlver funkcioniše kao kontejner zajedničkih komponenti i funkcija koje aplikacije više ne moraju implementirati odvojeno.

1. **Zašto je midlver važan za razvoj distribuiranih aplikacija?**

Midlver je važan jer omogućava efikasno deljenje resursa i komunikaciju između komponenti distribuiranih aplikacija, čime olakšava razvoj složenih sistema. Takođe, midlver pruža standardizovane usluge kao što su međuaplikaciona komunikacija i zaštita, čime smanjuje potrebu za ponovnim implementiranjem tih funkcionalnosti u svakoj aplikaciji pojedinačno.

1. **Koja je svrha omotača (wrappera) u organizaciji midlvera i kako rešavaju problem nekompatibilnih interfejsa?**

Omotači, ili adapteri, su posebne komponente koje nude interfejs prihvatljiv za klijentsku aplikaciju, transformišući funkcije komponente u one koje su pogodne za korišćenje u aplikaciji. Oni rešavaju problem nekompatibilnih interfejsa omogućavajući aplikacijama da koriste funkcionalnosti postojećih komponenti.

**OMOTACI (WRAPPERI) I BROKERI**

1. **Šta je brokerski sistem poruka i kako funkcioniše?**

Brokerski sistem poruka je tip midlvera koji se sastoji od logički centralizovane komponente, brokera, koji upravlja komunikacijom između različitih aplikacija. Aplikacije šalju zahteve brokeru koji ih obrađuje i prosleđuje odgovarajućim aplikacijama, kombinuje i transformiše odgovore, i vraća rezultate početnoj aplikaciji.

1. **Kako brokerski sistem poruka smanjuje potrebu za razvojem omotača?**

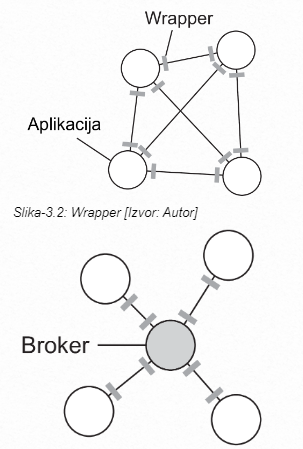
Brokerski sistem poruka smanjuje potrebu za razvojem omotača tako što omogućava da svaka aplikacija komunicira samo sa brokerom, umesto direktno sa drugim aplikacijama. Ovo smanjuje broj potrebnih omotača sa O(N^2) na O(N), što čini sistem skalabilnijim i efikasnijim.

1. **Kako omotači doprinose proširivosti sistema i otvorenosti?**

Omotači omogućavaju proširivost sistema tako što omogućavaju integraciju postojećih komponenti sa novim aplikacijama, transformišući funkcionalnosti i interfejse komponenti u interfejse prihvatljive za korišćenje u novim kontekstima. Time se postiže otvorenost sistema za nove zahteve i promene.

1. **Zašto razvoj omotača po potrebi može postati neefikasan i nepraktičan?**

Razvoj omotača po potrebi može postati neefikasan i nepraktičan jer sa porastom broja aplikacija, broj potrebnih omotača raste kvadratno u odnosu na broj aplikacija. To znači da bi sa N aplikacija, bilo potrebno razviti O(N^2) omotača, što nije skalabilno.



**INTERCEPTORI**

1. **Kako interceptor doprinosi prilagođavanju midlvera specifičnim potrebama aplikacije?**

Interceptor prekida uobičajeni tok kontrole i omogućava izvršenje aplikacijski specifičnog koda, što omogućava prilagođavanje midlvera specifičnim potrebama aplikacije. To je ključno za postizanje otvorenosti i fleksibilnosti sistema.

**VISESLOJNE KLIJENT SERVER ARHITEKTURE**

1. **Šta predstavlja osnovnu klijent-server arhitekturu?**

Osnovna klijent-server arhitektura je oblik distribuirane arhitekture u kojoj se procesi u sistemu dele na dve glavne grupe: servere, koji implementiraju određenu uslugu, i klijente, koji zahtevaju tu uslugu slanjem zahteva serveru i čekanjem odgovora.

1. **Kako se ostvaruje komunikacija između klijenta i servera u osnovnoj klijent-server arhitekturi?**

Komunikacija između klijenta i servera može biti ostvarena jednostavnim protokolom bez veze, gde klijent šalje zahtev serveru i čeka odgovor, bez održavanja stalne veze.

1. **Šta su idempotentni zahtevi u kontekstu klijent-server komunikacije?**

Idempotentni zahtevi su oni zahtevi čije ponavljanje neće promeniti rezultat ili stanje sistema. Ovo je važno za tretiranje izgubljenih ili dupliciranih poruka u klijent-server komunikaciji.

1. **Koja je prednost pouzdanog protokola orijentisanog ka uspostavljanju veze u odnosu na protokole bez veze u klijent-server arhitekturi?**

Pouzdan protokol orijentisan ka uspostavljanju veze omogućava bolje upravljanje izgubljenim ili oštećenim porukama, što je ključno za pouzdan rad u nepouzdanim mrežnim okruženjima.

1. **Koje su dve glavne organizacije u višeslojnim arhitekturama?**

Dve glavne organizacije u višeslojnim arhitekturama su podela na klijentsku i serversku mašinu i podela na slojeve korisničkog interfejsa, procesiranja i podataka.

**PEER TO PEER ARHITEKTURA**

1. **Koja je osnovna karakteristika peer-to-peer (P2P) sistema?**

Osnovna karakteristika P2P sistema je da su svi čvorovi jednaki i da svaki čvor istovremeno deluje kao klijent i server.

1. **Koje su dve vrste preklapajućih mreža u P2P sistemima?**

Dve vrste preklapajućih mreža u P2P sistemima su strukturisane i nestruktuirane mreže.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L04**

1. **Koja je uloga niti u distribuiranim sistemima?**

Niti igraju ključnu ulogu u distribuiranim sistemima jer omogućavaju paralelno izvršavanje zadatka. To znači da više niti može istovremeno izvršavati različite delove programa ili zadatke, što doprinosi efikasnosti i skalabilnosti sistema.

1. **Šta je virtuelni procesor i kako se koristi u operativnom sistemu?**

Virtuelni procesor je apstrakcija fizičkog procesora koja omogućava operativnom sistemu da pokreće više virtuelnih procesa istovremeno na jednom fizičkom procesoru. On se koristi za efikasno upravljanje resursima i omogućava multitasking i paralelno izvršavanje više procesa.

1. **Kako funkcioniše tabela procesa u operativnom sistemu i koji su ključni elementi koje sadrži?**

Tabela procesa je struktura podataka koja se koristi u operativnom sistemu za praćenje svih pokrenutih procesa. Ključni elementi koji se obično nalaze u tabeli procesa uključuju identifikator procesa (PID), stanje procesa, prioritete, informacije o resursima koje proces koristi (poput memorije i fajlova), informacije o pristupu resursima, informacije o procesorskom vremenu i informacije o roditeljskom procesu. Ova tabela omogućava operativnom sistemu efikasno upravljanje procesima, njihovo praćenje i pružanje resursa koje procesi zahtevaju za izvršavanje.

1. **Šta obuhvata kontekst procesa i zašto je važan u operativnom sistemu?**

Kontekst procesa obuhvata sve informacije o trenutnom stanju procesa koje su neophodne za njegovo privremeno zaustavljanje i kasnije nastavljanje izvršavanja. To uključuje vrednosti registara CPU-a, stanje memorije, otvorene datoteke i ostale resurse koje proces koristi. Kontekst procesa je važan jer omogućava operativnom sistemu da efikasno upravlja više procesima tako što im omogućava da se privremeno zaustave i ponovo pokrenu, čime se postiže multitasking i paralelno izvršavanje.

1. **Sta je interprocesna komunikacija?**

Interprocesna komunikacija (IPC) je proces razmene podataka ili informacija između različitih procesa koji se izvršavaju na istom računarskom sistemu ili čak na različitim računarima u mreži. Ova komunikacija omogućava procesima da sarađuju, razmenjuju informacije ili sinhronizuju svoje aktivnosti radi postizanja zajedničkog cilja.

1. **Koje su glavne metode interprocesne komunikacije i kako one omogućavaju komunikaciju između različitih procesa?**

Glavne metode interprocesne komunikacije (IPC) obuhvataju mehanizme kao što su razmena poruka (message passing) i deljenje memorije (shared memory).

Razmena poruka omogućava procesima da komuniciraju slanjem poruka jedan drugom. Poruke mogu biti asinhrono ili sinhrono dostavljene, a mehanizmi za to uključuju redove poruka, cevovode (pipes), signale i sockets.

Deljenje memorije omogućava procesima da pristupaju zajedničkom prostoru u memoriji, omogućavajući im da dele podatke. Ovo se obično postiže kroz deljene memorijske segmente ili mapiranje fajlova u memoriju.

Obe metode omogućavaju komunikaciju između različitih procesa tako što omogućavaju slanje podataka ili sinhronizaciju između njih, čime se olakšava saradnja i deljenje resursa. Međutim, svaka metoda ima svoje prednosti i mane, a izbor metode zavisi od specifičnih zahteva aplikacije i ograničenja operativnog sistema.

1. **Koji su glavni nedostaci many-to-one threading modela u implementaciji niti na nivou korisničkog prostora?**

Glavni nedostaci many-to-one threading modela uključuju činjenicu da više niti mapira na jedan entitet koji može biti raspoređen, što znači da blokirajući sistemski pozivi mogu blokirati ceo proces i sve druge niti u tom procesu. Ovo ograničenje onemogućava efikasno korišćenje višenitnih aplikacija, posebno u situacijama kada nekoliko niti istovremeno radi na različitim zadacima.

1. **Koje su prednosti one-to-one threading modela u odnosu na many-to-one threading model kada je reč o implementaciji niti?**

Prednosti one-to-one threading modela uključuju to da svaka nit predstavlja jedinicu koja može biti raspoređena, što omogućava da blokiranje jedne niti ne utiče na ostale niti. Ovaj model takođe omogućava efikasnu upotrebu višeprocesorskih sistema, jer svaka nit može biti dodeljena različitom procesoru ili jezgru radi paralelnog izvršavanja. Osim toga, ovaj model olakšava sinhronizaciju i upravljanje nitima, jer operacije sa nitima, poput kreiranja, brisanja i sinhronizacije, mogu biti obavljene direktno od strane jezgra operativnog sistema.

**VISENITNI KLIJENTI**

1. **Šta je vreme ukupnog puta (round-trip time) i zašto je važno u kontekstu distribuiranih sistema, posebno veb pretraživača?**

Vreme ukupnog puta (round-trip time) u kontekstu distribuiranih sistema predstavlja vreme potrebno da se poruka ili zahtev pošalje sa izvora do odredišta, a zatim da se odgovor vrati nazad na izvor. Ovo vreme uključuje vreme potrebno za slanje podataka preko mreže, obradu na odredišnom uređaju i povratnu putanju podataka do izvora.

1. **Šta je strategija rotacije (round-robin) i kako se koristi u kontekstu distribuiranih sistema?**

Strategija rotacije (round-robin) je jedan od načina balansiranja opterećenja u distribuiranim sistemima. Ova strategija funkcioniše tako što se zahtevi ili konekcije ravnomerno raspoređuju među dostupnim serverima ili resursima, jedan za drugim, u krug. Na primer, kada klijent šalje zahtev za veb stranicu, strategija rotacije će proslediti taj zahtev prvom dostupnom serveru, a zatim sledeći zahtev proslediti drugom serveru, i tako dalje, u krug.

U kontekstu distribuiranih sistema, strategija rotacije omogućava efikasno iskorišćenje resursa jer se opterećenje ravnomerno raspoređuje među serverima. Ovo pomaže u izbegavanju preopterećenja pojedinačnih servera i osigurava bolju dostupnost usluge za klijente. Sa višenitnim klijentom, ova strategija omogućava uspostavljanje paralelnih konekcija sa različitim replikama servera, što doprinosi bržem prenosu podataka i smanjenju vremena potrebnog za prikazivanje veb stranica.

1. **Kako višenitni klijent funkcioniše u kontekstu veb pretraživača i koje su prednosti ovog pristupa?**

Višenitni klijent u kontekstu veb pretraživača funkcioniše tako što omogućava istovremeno obavljanje više zadataka, kao što su preuzimanje HTML datoteke, slika, ikona, itd., sa veb servera. Kada korisnik otvori određenu veb stranicu, glavna HTML datoteka se prvo preuzima, a zatim prikazuje korisniku. Istovremeno, posebne niti se aktiviraju da preuzmu ostale delove stranice, kao što su slike, ikone i drugi elementi.

Ovaj pristup omogućava što brže prikazivanje delova veb stranice korisniku, dok se ostali delovi još uvek preuzimaju. Dok korisnik čita tekst i interaguje sa stranicom, pretraživač nastavlja sa preuzimanjem preostalih delova stranice. Kako se delovi preuzimaju, prikazuju se korisniku u realnom vremenu, što doprinosi boljem korisničkom iskustvu.

Prednosti višenitnog klijenta u veb pretraživaču uključuju efikasnije iskorišćenje vremena tokom preuzimanja veb stranica, brže prikazivanje sadržaja korisnicima, kao i mogućnost otvaranja više konekcija istovremeno što može poboljšati performanse, posebno u slučaju kada je server opterećen ili spor.

**VISENITNI SERVERI**

1. **Kako se dispečer, radne niti (working threads) i endpointovi koriste u arhitekturi višenitnog servera, posebno u kontekstu obrade zahteva u distribuiranim sistemima?**

Dispečer je komponenta višenitnog servera koja prima dolazne zahteve od klijenata i prosleđuje ih odgovarajućim radnim nitima na obradu. Radne niti (working threads) su niti koje izvršavaju zadatke koje im je dodelio dispečer, kao što su operacije nad datotekama ili bilo koja druga obrada zahteva. Endpointovi predstavljaju tačke na kojima klijenti mogu pristupiti serveru i slati zahteve. Ovi zahtevi se prosleđuju serveru putem endpointa, gde ih dispečer distribuira radnim nitima na obradu. Ova arhitektura omogućava efikasnu obradu više zahteva istovremeno u višenitnom okruženju

1. **Kako se višenitni server realizuje kao mašina konačnog stanja (finite state machine)?**

Višenitni server realizuje se kao mašina konačnog stanja tako što svaka nit reaguje na dolazne zahteve kao na događaje. Kada zahtev stigne, nit pregleda trenutno stanje i u skladu sa tim reaguje na zahtev. Ukoliko je moguće, zahtev se obrađuje iz keš memorije. U suprotnom, nit može zakazati asinhronu operaciju, kao što je operacija čitanja sa diska, i nastaviti sa procesiranjem drugih zahteva. Kada se operacija završi, nit se obaveštava i nastavlja sa završetkom obrade zahteva. Ovaj pristup omogućava efikasno upravljanje resursima i smanjenje vremena čekanja, čime se postižu bolje performanse servera.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L05**

**VIRTUELIZACIJA**

1. **Šta podrazumeva virtuelizacija računarskih resursa?**

Virtuelizacija podrazumeva apstrakciju računarskih resursa, što uključuje razdvajanje resursa računara u više zasebnih radnih okruženja kroz tehnike poput hardverskog ili softverskog particionisanja, timesharinga, mašinske simulacije, emulacije i drugih.

1. **Koja je svrha virtuelizacije u IT infrastrukturi?**

Svrha virtuelizacije u IT infrastrukturi je efikasnije korišćenje računarskih resursa putem omogućavanja pokretanja više virtuelnih mašina na jednoj fizičkoj mašini, što omogućava deljenje resursa na nekoliko razdvojenih okruženja i istovremeno pokretanje različitih operativnih sistema i aplikacija.

1. **Kako se virtuelizacija realizuje na fizičkom računaru?**

Virtuelizacija se realizuje na fizičkom računaru kroz izradu virtuelnih mašina (VM), softverskih implementacija računara koje izvršavaju programe na isti način kao prava mašina. Kroz ovaj proces, fizički resursi računara se deluju na način da podrže rad više izolovanih radnih okruženja, svako sa svojim operativnim sistemom i aplikacijama.

1. **Šta je Virtual Machine Monitor (VMM)?**

Virtual Machine Monitor (VMM) je komponenta softvera koja omogućava kreiranje i upravljanje virtuelnim mašinama na fizičkom računaru. VMM je odgovoran za izolaciju virtuelnih mašina jednih od drugih, kao i od fizičkog hardvera, omogućavajući im da deluju kao nezavisni entiteti sa sopstvenim operativnim sistemima i aplikacijama.

1. **Kako se definišu pojmovi host (domaćin) i guest (gost) u kontekstu virtuelizacije?**

U kontekstu virtuelizacije, host (domaćin) je osnovni računar na kojem se izvršava virtualizacija, dok je guest (gost) virtuelna mašina koja se pokreće unutar tog osnovnog računara.

1. **Koja je osnovna razlika između emulacije i virtuelizacije?**

Osnovna razlika između emulacije i virtuelizacije leži u načinu na koji se pristupa hardverskim resursima. Emulacija podrazumeva apstrakciju hardvera i arhitekture virtuelne mašine, dok se kod virtuelizacije jednostavno izoluje virtuelna mašina unutar postojećeg hardvera.

1. **Šta podrazumeva emulacija u kontekstu virtuelnih mašina?**

Emulacija u kontekstu virtuelnih mašina znači da se emulira hardverska struktura, na primer x86 arhitektura, i koristi dodatni sloj za prevođenje i preusmeravanje na nivou gostujuće virtuelne mašine.

1. **Kako virtuelizacija funkcioniše u odnosu na host računar i gostujuću virtuelnu mašinu?**

Kod virtuelizacije, host računar propušta izvršavanje gostujuće virtuelne mašine na svom postojećem hardveru. Gostujuća virtuelna mašina funkcioniše izolovano unutar hosta, bez potrebe za dodatnim slojevima prevođenja i preusmeravanja, što omogućava bolje performanse i brži rad u poređenju sa emulacijom.

**X86 VIRTUELIZACIJA**

1. **Kako se x86 virtuelizacija realizuje?**

X86 virtuelizacija postiže se dodavanjem virtuelnog sloja između hardvera i operativnog sistema.

1. **Koje su prednosti x86 virtuelizacije?**

X86 virtuelizacija omogućava konkurentan rad više instanci operativnog sistema na istom računaru, dinamičku raspodelu resursa poput CPU-a, skladišta, memorije i I/O uređaja, te unapređenje konsolidacije servera, agilnosti baza podataka i kontinuiteta poslovanja.

1. **Koji su osnovni karakteristike privilegovanog i neprivilegovanog pristupa u kontekstu računarskih sistema?**

U računarskim sistemima, privilegovani pristup omogućava softveru korišćenje svih hardverskih instrukcija, dok neprivilegovani pristup ograničava softver na izvršavanje samo neprivilegovanih instrukcija. Privilegovani režim se koristi za izvršavanje operacija koje zahtevaju visok nivo privilegija, poput upravljanja hardverom, dok se neprivilegovani režim koristi za izvršavanje korisničkih programa i operacija koje ne zahtevaju visok nivo privilegija.

**VIRTUELNE MASINE**

1. **Koja je uloga hipervizora u virtuelizaciji?**

Hipervizor je specijalizovani softver koji omogućava kreiranje i upravljanje virtuelnim mašinama na fizičkom računaru. On emulira hardverske resurse i omogućava svakoj virtuelnoj mašini da radi kao zaseban operativni sistem, nezavisno od drugih virtuelnih mašina i fizičkog hardvera. Na taj način, hipervizor omogućava deljenje hardverskih resursa između više virtuelnih mašina, što povećava iskorišćenje hardvera i olakšava upravljanje računarskim resursima.

1. **Koje su osnovne razlike između hipervizora tipa 1 i hipervizora tipa 2?**

Osnovna razlika između hipervizora tipa 1 i hipervizora tipa 2 je u tome gde se svaki od njih izvršava i kako pristupa hardveru. Hipervizor tipa 1, poznat kao "bare metal" hipervizor, radi direktno na hardveru, dok hipervizor tipa 2 radi kao aplikacija unutar postojećeg operativnog sistema. To znači da hipervizor tipa 1 ne zahteva host operativni sistem, dok hipervizor tipa 2 zahteva host operativni sistem na koji se instalira.

1. **Kako radi hipervizor tipa 1 u odnosu na hipervizor tipa 2 u pogledu performansi i fleksibilnosti?**

Hipervizor tipa 1 pruža bolje performanse i veću fleksibilnost u poređenju sa hipervizorom tipa 2. To je zbog toga što hipervizor tipa 1 radi direktno na hardveru, što smanjuje overhead i omogućava efikasnije korišćenje resursa. Sa druge strane, hipervizor tipa 2 mora raditi kroz host operativni sistem, što može dovesti do većeg overheada i ograničenja u performansama.

1. **Šta je paravirtuelizacija i kako se razlikuje od tradicionalnih metoda virtuelizacije?**

Paravirtuelizacija je proces postizanja virtuelizacije pri čemu se trajno uklanjaju osetljive instrukcije, umesto da se emuliraju. Umesto toga, hipervizoru se upućuju pozivi za hipervizor, što omogućava efikasnije izvršavanje i bolje performanse. Paravirtuelizacija se razlikuje od tradicionalnih metoda virtuelizacije koje pokušavaju da emuliraju kompletnu hardversku arhitekturu, što može dovesti do većeg overheada i gubitka performansi.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L06**

**UMREZENI KORISNICKI INTERFEJS**

1. **Koje su karakteristike i prednosti umreženih korisničkih interfejsa?**

Umreženi korisnički interfejs omogućava da se sve obradi i skladišti na serveru, što čini klijentsku mašinu terminalom bez potrebe za lokalnim skladištenjem. Ovakav pristup rezultira aplikaciono neutralnim rešenjem, što olakšava upravljanje sistemom i omogućava lakšu podršku različitim uređajima. Ova tehnologija postaje sve popularnija sa sve većom dostupnošću interneta i povećanim korišćenjem mobilnih uređaja.

**X WINDOW SYSTEM (X11)**

1. **Šta je X Window System (X11) i kakvu ulogu ima u kontroli korisničkog interfejsa?**

X Window System, ili X11, je protokol i softverski sistem koji omogućava kontrolu korisničkog interfejsa, uključujući monitor, tastaturu i miš. On se koristi za bit-mape terminala, podržavajući tradicionalne i moderne uređaje poput pametnih telefona i tableta.

1. **Kako X jezgro funkcioniše i koje zadatke obavlja u X Window System-u?**

X jezgro obavlja upravljačke zadatke specifične za terminal, pružajući interfejs za kontrolu ekrana i hvatanje događaja tastature i miša.

1. **Koja je uloga Xlib biblioteke u X Window System-u i kako aplikacije komuniciraju sa X jezgrom?**

Xlib je biblioteka koja omogućava aplikacijama da komuniciraju sa X jezgrom putem X protokola, slanjem zahteva za kreiranje prozora, postavljanje boja i drugih operacija.

**ANATOMIJA SERVERA**

1. **Kako se organizuju serveri u slučaju iterativnog servera?**

U slučaju iterativnog servera, sam server obrađuje zahtev i, ako je potrebno, vraća odgovor klijentu koji je poslao zahtev.

1. **Šta karakteriše konkurentni server?**

Konkurentni server ne obrađuje zahtev sam, već ga prosleđuje zasebnoj niti ili drugom procesu, nakon čega odmah čeka sledeći dolazni zahtev.

1. **Koja je alternativna implementacija konkurentnog servera?**

Alternativna implementacija konkurentnog servera je kreiranje novog procesa za svaki novi dolazni zahtev, što se često koristi u mnogim Unix sistemima.

1. **Šta je krajnja tačka (endpoint) u kontekstu kontaktiranja servera?**

Krajnja tačka (endpoint) je mesto na koje klijenti šalju zahteve, takođe nazvano port, na mašini na kojoj server radi. Svaki server sluša određenu krajnju tačku.

1. **Kako klijenti obično saznaju krajnju tačku servisa?**

Jedan pristup je globalno dodeljivanje krajnjih tačaka za dobro poznate servise, poput FTP-a koji koristi TCP port 21 ili HTTP-a koji koristi TCP port 80. Ove krajnje tačke su dodeljene od strane Internet Assigned Numbers Authority (IANA), i klijentu je potrebno samo da pronađe mrežnu adresu mašine na kojoj server radi. Alternativno, servisi koji ne zahtevaju unapred dodeljene krajnje tačke mogu koristiti dinamički dodeljene krajnje tačke.

1. **Kako se rešava problem pronalaženja krajnje tačke za servise koji koriste dinamički dodeljene krajnje tačke?**

Jedno rešenje je imati poseban proces nazvan daemon koji radi na svakoj mašini na kojoj su serveri. Demon prati trenutnu krajnju tačku svakog servisa koji implementira lokalno postavljeni server. Klijent će prvo kontaktirati daemona, zatražiti krajnju tačku, a zatim kontaktirati specifični server.

1. **Koje su dve osnovne metode za prekid servera prilikom prekida komunikacije sa klijentom?**

Dve osnovne metode za prekid servera prilikom prekida komunikacije sa klijentom su: izlazak korisnika iz klijentske aplikacije, što automatski prekida vezu sa serverom, ili slanje "out-of-band" (OOB) podataka koji signaliziraju serveru da je potrebno prekinuti vezu.

1. **Šta su "out-of-band" (OOB) podaci i kako se koriste za prekid servera?**

"Out-of-band" (OOB) podaci su podaci koji se obrađuju od strane servera pre bilo kojih drugih podataka od tog klijenta. Koriste se za prekid servera tako što se serveru šalje posebna krajnja tačka na koju klijent šalje OOB podatke ili se OOB podaci šalju preko iste veze preko koje klijent šalje originalni zahtev, na primer, putem TCP-a.

1. **Kako server može osluškivati OOB podatke za prekid veze sa klijentom?**

Server može osluškivati OOB podatke tako što omogući posebnu krajnju tačku na koju klijent šalje OOB podatke i istovremeno osluškuje (sa nižim prioritetom) tačku kroz koju prolaze normalni podaci. Alternativno, OOB podaci se mogu slati preko iste veze preko koje klijent šalje originalni zahtev, a server reaguje na primljene OOB podatke prekidom veze.

1. **Koja je bitna razlika između opšteg objektnog servera i klasičnijih servera?**

Bitna razlika je što objektni server sam po sebi ne pruža određenu uslugu, već usluge implementiraju objekti koji se nalaze na serveru. Server pruža samo način za pozivanje lokalnih objekata na osnovu zahteva sa udaljenih klijenata.

1. **Kako objekti funkcionišu unutar objektnog servera?**

Objekti u objektnom serveru se sastoje od dva dela: podataka koji predstavljaju njihovo stanje i koda za izvršavanje njihovih metoda. Implementacija metoda i razdvajanje podataka zavisi od konkretnog objektnog servera.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L07**

**OSI REFERENTNI MODEL**

1. **Šta predstavlja OSI referentni model?**

OSI referentni model je dizajniran da omogući otvorenim sistemima da komuniciraju preko mreže. To je standardni model koji identifikuje različite nivoe uključene u komunikaciju, daje im standardna imena i ukazuje koji nivo treba da obavi određeni posao.

1. **Koje su dve vrste komunikacionih usluga prema OSI modelu?**

Prema OSI modelu, postoje dve vrste komunikacionih usluga - usluga orijentisana ka povezivanju (connection-oriented service) i usluga bez povezivanja (connectionless service).

1. **Koji je primer usluge orijentisane ka povezivanju prema OSI modelu?**

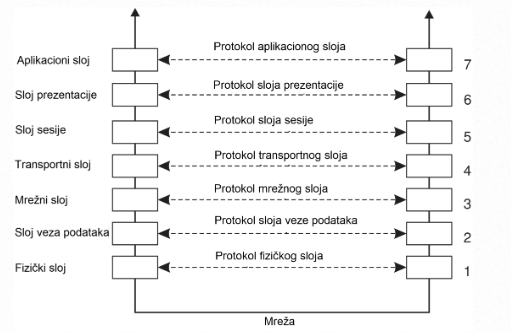
Primer usluge orijentisane ka povezivanju prema OSI modelu je telefon, gde pre razmene podataka pošiljalac i primalac eksplicitno uspostavljaju vezu pre nego što razmene podatke.

1. **Koji je primer usluge bez povezivanja prema OSI modelu?**

Primer usluge bez povezivanja prema OSI modelu je stavljanje pisma u poštansko sanduče, gde pošiljalac jednostavno šalje prvu poruku kada je spreman, bez potrebe za prethodnim podešavanjima.

1. **Koji su sedam slojeva OSI referentnog modela?**

Sedam slojeva OSI referentnog modela su: Fizički sloj, Sloj veze podataka, Mrežni sloj, Transportni sloj, Sloj sesije, Sloj prezentacije i Aplikacioni sloj.



1. **Koje komunikacione usluge pruža fizički sloj prema OSI modelu?**

Fizički sloj se bavi standardizacijom načina kako su dva računara povezana i kako se 0 i 1 reprezentuju.

1. **Koje su osnovne funkcije sloja veze podataka prema OSI modelu?**

Sloj veze podataka pruža sredstva za detektovanje i možda ispravljanje grešaka u prenosu, kao i protokole za održavanje sinhronizacije između pošiljaoca i primaoca.

1. **Šta radi mrežni sloj prema OSI modelu?**

Mrežni sloj sadrži protokole za rutiranje poruke kroz računarsku mrežu, kao i protokole za upravljanje zagušenjem.

1. **Koje usluge pruža transportni sloj prema OSI modelu?**

Transportni sloj pruža protokole koji direktno podržavaju aplikacije, kao što su oni koji uspostavljaju pouzdanu komunikaciju ili podržavaju realno vreme tokom prenosa podataka.

1. **Koje komunikacione usluge pruža sloj sesije prema OSI modelu?**

Sloj sesije pruža podršku za sesije između aplikacija, uključujući uspostavljanje, održavanje i prekid sesija.

1. **Koja je osnovna uloga sloja prezentacije prema OSI modelu?**

Sloj prezentacije propisuje kako se podaci predstavljaju na način koji je nezavisan od računara na kojima se izvršavaju aplikacije koje komuniciraju, uključujući enkripciju, kompresiju i formatiranje podataka.

1. **Šta obuhvata aplikacioni sloj prema OSI modelu?**

Aplikacioni sloj obuhvata protokole za e-poštu, pristup vebu, prenos datoteka i druge aplikacije koje korisnici direktno koriste za komunikaciju preko računarske mreže.

1. **Kako proces P komunicira sa udaljenim procesom Q prema OSI modelu?**

Proces P formira poruku i prosleđuje je sloju aplikacije kroz interfejs sloja aplikacije. Poruka se zatim prosleđuje kroz različite slojeve modela prema sloju fizičkog prenosa.

1. **Kako svaki sloj obrađuje poruku tokom njenog prenosa prema OSI modelu?**

Svaki sloj dodaje ili ispituje svoje vlastito zaglavlje na poruku koja prolazi kroz njega, pri čemu se zaglavlje koristi za informacije o protokolu sloja i procesu komunikacije.

**MIDLWARE PROTOKOLI**

1. **Kako se midlver definiše u kontekstu OSI referentnog modela, i koje karakteristike ima?**

Midlver je aplikacija koja logički živi, uglavnom, u OSI sloju aplikacije, ali koja sadrži mnogo protokola opšte namene koji zaslužuju svoje sopstvene slojeve, nezavisno od drugih, specifičnijih aplikacija.

**TIPOVI KOMUNIKACIJE (POSTOJANA & NEPOSTOJANA) (SINHRONA & ASINHRONA)**

1. **Kako se komunikacija razlikuje između postojane (persistant) i nepostojane (transient) komunikacije u kontekstu midlvera u računarstvu klijent-server?**

Postojana komunikacija podrazumeva da se poruka koja je poslata za prenos čuva od strane komunikacionog middleware-a sve dok se ne dostavi primaocu, čime se omogućava da aplikacija za slanje ne mora nastaviti sa izvršavanjem nakon slanja poruke.

Suprotno tome, kod nepostojane komunikacije, poruka se čuva u komunikacionom sistemu samo dok aplikacije za slanje i primanje izvršavaju, i ako komunikacioni sistem ne može da dostavi poruku zbog prekida u prenosu ili neaktivnosti primaoca, poruka će biti odbačena.

1. **Kako se razlikuju asinhrona i sinhrona komunikacija u kontekstu računarstva, i koje su karakteristične osobine svake?**

Asinhrona komunikacija je karakterizovana time da pošiljalac nastavlja odmah nakon slanja svoje poruke za prenos, dok je poruka privremeno smeštena od strane middleware-a.

Sa druge strane, sinhrona komunikacija zahteva da pošiljalac bude blokiran dok se njegov zahtev ne završi ili ne dobije potvrdu.

**RPC (REMOTE PROCEDURE CALL)**

1. **Šta je udaljeni poziv procedura (RPC) i kako funkcioniše u distribuiranim sistemima?**

Udaljeni poziv procedura (RPC) je koncept koji omogućava programima da pozivaju procedure koje se nalaze na drugim mašinama u distribuiranim sistemima. Kada proces na mašini A pozove proceduru na mašini B, pozivajući proces na A je suspendovan, a izvršavanje pozvane procedure se odvija na B. Informacije se mogu prenositi od pozivaoca do pozvane procedure u parametrima, a rezultati se mogu vratiti izvršitelju poziva. RPC omogućava programerima da izbegnu eksplicitno rukovanje komunikacijom između procesa, čime se postiže transparentnost pristupa u distribuiranim sistemima.

1. **Koja je uloga server stub-a u udaljenom pozivu procedura (RPC) i kako funkcioniše na strani servera?**

Server stub je komponenta u sistemu udaljenog poziva procedura (RPC) koja se nalazi na strani servera. Njegova uloga je da prima poruke koje stižu preko mreže, raspakuje parametre iz tih poruka i poziva odgovarajuće server procedure kako bi izvršio zahtev klijenta. Server stub transformiše zahteve koji dolaze preko mreže u lokalne pozive procedura na serveru, omogućavajući serveru da izvrši zahtevane operacije kao da su direktno pozvane od strane klijenta. Nakon završetka izvršenja, server stub pakuje rezultat u poruku i šalje ga nazad klijentu preko mreže.

1. **Šta je marshaling parametara i zašto je bitno u udaljenom pozivu procedura?**

Marshaling parametara se odnosi na proces upakivanja parametara funkcije ili procedure u format koji je pogodan za prenos preko mreže. Ovaj proces je važan u udaljenom pozivu procedura jer omogućava klijentskom stub-u da pravilno upakuje parametre u poruku koja će biti poslata serveru.

------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------------------------------------------------------------

**L08**

**KOMANDA MAKE – SINHRONIZACIJA SATOVA U DISTRIBUIRANOM SISTEMU**

1. **Šta je osnovni cilj komande make?**

Osnovni cilj make-a je automatski odrediti koje delove velikog programa treba ponovo kompajlirati i izdati neophodne komande za njihovo ponovno kompajliranje.

1. **Kako make određuje koje delove programa treba ponovo kompajlirati?**

Make određuje koje delove programa treba ponovo kompajlirati tako što pregleda vremenske oznake (timestamps) kada su poslednji put modifikovani svi izvorni i objektni fajlovi i upoređuje ih. Ako je izvorni fajl noviji od odgovarajućeg objektnog fajla, potrebno je ponovno kompajliranje.

1. **U kojim operativnim sistemima je komanda make posebno popularna?**

Komanda make je posebno popularna u Unix, Linux i drugim operativnim sistemima sličnim Unix-u.

1. **Da li bi bilo moguće sinhronizovati sve satove u distribuiranom sistemu?**

Iako je teoretski moguće postići određeni stepen sinhronizacije, praksa pokazuje da zbog različitih faktora kao što su kašnjenja u mreži i razlike u brzini rada satova na različitim računarima, apsolutna sinhronizacija svih satova u distribuiranom sistemu je veoma teško ostvariva.

1. **Šta može biti posledica nesinhronizovanih satova u distribuiranom sistemu?**

Nesinhronizovani satovi mogu dovesti do toga da build sistem kao što je make ne prepozna da je izvorni fajl izmenjen, jer bi mu mogao biti dodeljen stariji vremenski pečat (timestamp) nego odgovarajućem objekt fajlu. Kao rezultat toga, make neće pozvati kompilator, što može dovesti do mešavine starih i novih objektnih fajlova u izvršnom binarnom programu. Ovo može uzrokovati pad programa, što će programeru otežati identifikaciju problema u kodu.

**FIZICKI CASOVNICI**

1. **Šta je računski tajmer i kako funkcioniše?**

Računski tajmer je precizno izrađen kvarcni kristal koji oscilira na dobro definisanoj frekvenciji kada je pod tenzijom. Uz kristal idu dva registra: brojač i registar za čuvanje. Svaka oscilacija smanjuje brojač za jedan, a kada brojač dođe do nule, generiše se prekid i brojač se ponovo učitava iz registra za čuvanje. Na ovaj način, tajmer može generisati prekide na željenoj frekvenciji.

1. **Kako se održava tačnost softverskog sata na računaru?**

Kada se sistem prvi put pokrene, korisnik unosi trenutnu vremensku zonu ili vreme, koje se zatim konvertuje u broj taktova posle nekog poznatog početnog datuma i sačuva u memoriji. Većina računara ima baterijom podržanu CMOS RAM memoriju koja čuva datum i vreme, tako da se ne moraju unositi prilikom svakog sledećeg pokretanja. Procedura za obradu prekida dodaje jedan u vreme sačuvano u memoriji prilikom svakog taktovanja sata, čime se softverski sat održava ažurnim.

1. **Šta uzrokuje odstupanje sata (clock skew) u distribuiranim sistemima?**

Odstupanje sata u distribuiranim sistemima uzrokuje to što svi n kristali rade sa malo različitim brzinama, što dovodi do toga da softverski satovi postepeno izlaze iz sinhronizacije i daju različite vrednosti kada se očitavaju.

1. **Zašto je sinhronizacija satova važna u distribuiranim sistemima, čak i ako nije potrebno tačno stvarno vreme?**

Sinhronizacija satova je važna jer omogućava čvorovima u distribuiranom sistemu da se slože oko trenutnog vremena ili relativnog vremena događaja. Na primer, za pokretanje make-a je važno da se čvorovi slože da je input.o zastareo u odnosu na novu verziju input.c. Ovo omogućava praćenje događaja međusobno, što je ključno za ispravno funkcionisanje sistema.

**LOGICKI CASOVNICI**

1. **Šta su logički časovnici (logical clocks) i zašto su važni u distribuiranim sistemima?**

Logički časovnici su koncept koji se koristi u distribuiranim sistemima da bi se pratio redosled događaja između čvorova. Važni su jer omogućavaju da se događaji (kao što je stvaranje nove verzije fajla) prate i sinhronizuju bez potrebe za tačnim stvarnim vremenom, omogućavajući doslednost i koordinaciju u sistemu.

1. **Šta predstavlja relacija "desi se pre" (happens-before) i kako se primenjuje?**

Relacija "desi se pre" (happens-before) koju je definisao Lamport znači da se svi procesi slažu da se događaj a dešava pre događaja b. Relacija "desi se pre" se direktno primenjuje u dva slučaja:

* Ako su a i b događaji u istom procesu, i a se dešava pre b, tada je relacija a → b istinita.
* Ako je a događaj slanja poruke od jednog procesa, a b je događaj prijema poruke od drugog procesa, tada je a → b takođe istinito. Poruka ne može biti primljena pre nego što bude poslata, jer je potrebno konačno, nenulto (pozitivno) vreme da stigne.

1. **Kako funkcioniše softverski brojač u kontekstu distribuiranog sistema i kako se vrednosti brojača razlikuju između procesa?**

U distribuiranom sistemu, svaki proces se izvršava na različitim mašinama, svaka sa svojim sopstvenim časovnikom implementiranim kao softverski brojač. Ovaj brojač se povećava za određenu vrednost svakih T vremenskih jedinica. Međutim, vrednost kojom se povećava brojač razlikuje se po procesu. Na primer, časovnik u procesu P1 povećava se za 6 jedinica, u procesu P2 za 8 jedinica, a u procesu P3 za 10 jedinica. Ove razlike uzrokuju da se softverski satovi postepeno izlaze iz sinhronizacije, što može dovesti do različitih vrednosti vremena kada se očitavaju, i potencijalno do problema u koordinaciji događaja među procesima.

1. **Kako funkcioniše Lamportov algoritam za sinhronizaciju satova u distribuiranim sistemima?**

Lamportov algoritam funkcioniše tako što koristi relaciju "desi se pre" (happens-before) da sinhronizuje satove među procesima u distribuiranom sistemu. Kada proces pošalje poruku, ona nosi vreme slanja prema satu pošiljaoca. Kada poruka stigne kod primaoca, ako sat primaoca pokazuje vrednost pre vremena kada je poruka poslata, primalac pomera svoj sat napred tako da bude za jedan veći od vremena slanja. Na ovaj način se osigurava da redosled događaja bude očuvan. Na primer, ako je poruka m3 poslata u 60, mora stići u 61 ili kasnije, pa primalac pomera svoj sat na 61 ili više. Slično, poruka m4 koja je poslata kasnije stiže u 70, čime se izbegavaju nemoguće situacije sa negativnim vremenima putovanja poruke.

**MEDJUSOBNO ISKLJUCENJE**

1. **Šta je međusobno isključenje i zašto je ključno za operativne sisteme i distribuirane sisteme?**

Međusobno isključenje je osnovni koncept koji osigurava da konkurentni procesi ili niti ne mešaju kada pristupaju deljenim resursima, kao što su promenljive, datoteke ili hardverski uređaji. U operativnim sistemima, međusobno isključenje je neophodno kako bi se sprečili konflikti koji mogu dovesti do oštećenja podataka ili neusaglašenosti. Mehanizmi međusobnog isključivanja, kao što su zaključavanja, semafori ili muteksi, nameću pravilo prema kojem samo jedan proces može pristupiti deljenom resursu u jednom trenutku, dok ostali moraju čekati da se resurs oslobodi. Ovo promoviše redovno izvršavanje i održava integritet sistema, što je ključno za stabilnost i pouzdanost operativnih sistema. U distribuiranim sistemima, međusobno isključenje je takođe važno kako bi se sprečilo oštećenje ili nekonzistentnost deljenih resursa, što je ključno za konkurentno i pouzdano izvršavanje procesa u takvim okruženjima.

1. **Šta su locks (zaključavanja) i zašto su važna za operativne sisteme i distribuirane sisteme?**

Locks (zaključavanja) su mehanizmi koji omogućavaju kontrolu pristupa deljenim resursima tako što omogućavaju samo jednom procesu ili niti da pristupi resursu u određenom trenutku. Oni su važni za operativne sisteme jer sprečavaju konflikte i neusaglašenost prilikom pristupa deljenim resursima, što doprinosi stabilnosti i pouzdanosti sistema. U distribuiranim sistemima, locks se koriste na sličan način kako bi se osiguralo da samo jedan proces može pristupiti deljenim resursima u isto vreme, čime se sprečava konflikt i oštećenje podataka.

1. **Kako se koriste semafori (semafores) u upravljanju međusobnim isključenjem u višenitnim i višeprocesnim okruženjima?**

Semafori (semaphores) su mehanizmi međusobnog isključenja koji omogućavaju kontrolu pristupa deljenim resursima kroz koncept brojača. Oni se koriste u višenitnim i višeprocesnim okruženjima tako što procesima omogućavaju da čekaju na slobodan resurs i da ga zatim oslobode kada ga završe koristiti. Ovaj mehanizam pomaže u sprečavanju sukoba i trkaćih uslova, čime se osigurava da resursi budu pravilno korišćeni i da se integritet podataka održava.

1. **Šta su mutexi (muteksi) i kako se koriste za obezbeđivanje međusobnog isključenja u operativnim sistemima?**

Mutexi (muteksi) su mehanizmi međusobnog isključenja koji omogućavaju samo jednom procesu ili niti da pristupi deljenom resursu u jednom trenutku. Oni se koriste u operativnim sistemima za sprečavanje trkaćih uslova i oštećenja podataka prilikom pristupa deljenim resursima. Mutexi obezbeđuju siguran pristup resursima tako što procesima omogućavaju da ih zaključaju prilikom korišćenja i otključaju kada završe, čime se osigurava redosled i integritet podataka.

1. **Šta su race conditions (uslovi trke) i zašto su problematični u višenitnim i višeprocesnim okruženjima?**

Race conditions (uslovi trke) su problematične situacije koje se javljaju u višenitnim i višeprocesnim okruženjima kada više procesa ili niti pokušava pristupiti deljenom resursu istovremeno, što može rezultirati neodređenim ili nekonzistentnim stanjem resursa. Race conditions su problematični jer mogu dovesti do oštećenja podataka ili neusaglašenosti, te stoga zahtevaju korišćenje mehanizama međusobnog isključenja kao što su locks, semafori ili mutexi radi sprečavanja ovakvih situacija.

1. **Kako funkcioniše centralizovan algoritam za međusobno isključenje u distribuiranim sistemima, i koje su karakteristike ovog pristupa?**

Centralizovan algoritam za međusobno isključenje u distribuiranim sistemima funkcioniše tako što se jedan proces bira kao koordinator. Kada neki proces želi pristupiti deljenom resursu, šalje zahtevnu poruku koordinatoru, navodeći koji resurs želi da pristupi. Koordinator zatim proverava da li je taj resurs trenutno dostupan. Ako resurs nije zauzet, koordinator odobrava zahtev i šalje odgovor procesu tražiocu, omogućavajući mu pristup resursu. U suprotnom, ako resurs već koristi drugi proces, koordinator odbija zahtev ili ga blokira dok resurs ne postane dostupan. Ovaj pristup centralizovanog međusobnog isključenja je jednostavan za implementaciju i održavanje, ali može biti nepouzdan u slučaju kvara ili preopterećenja koordinatora.

1. **Kako centralizovani algoritam za međusobno isključenje u distribuiranim sistemima obavlja oslobađanje resursa i kako procesi nastavljaju sa radom nakon oslobađanja?**

Kada proces završi sa korišćenjem deljenog resursa, šalje poruku koordinatoru kako bi oslobodio svoj ekskluzivni pristup resursu. Koordinator, po prijemu ove poruke, uzima prvu stavku sa liste odloženih zahteva za pristup resursu. Zatim, šalje poruku odobrenja tom procesu, omogućavajući mu pristup resursu. Ako je proces bio blokiran, tj. ako je ovo prva poruka koju prima nakon blokade, proces se odblokira i može pristupiti resursu. Ako je već poslata eksplicitna poruka koja odbija dozvolu, proces će morati periodično da proverava dolazni saobraćaj ili će kasnije biti blokiran. Kada proces primi odobrenje od koordinatora, može nastaviti sa radom koristeći deljeni resurs.

1. **Kako distribuirani algoritam koristi Lamportove logičke časovnike i kako funkcioniše njegovo rešenje za međusobno isključenje u distribuiranim sistemima?**

Distribuirani algoritam koristi Lamportove logičke časovnike kako bi omogućio raspoređivanje svih događaja u sistemu i jednoznačno odredio redosled događaja. Kada proces želi pristupiti deljenom resursu, šalje poruku koja sadrži ime resursa, broj svog procesa i trenutno (logičko) vreme svim drugim procesima. Kada proces primi zahtevnu poruku od drugog procesa, akcija koju preduzima zavisi od njegovog trenutnog stanja u vezi sa resursom. Ako proces ne pristupa resursu i ne želi mu pristupiti, šalje poruku OK pošiljaocu. Ako već ima pristup resursu, ne odgovara na zahtev već stavlja zahtev u red. Ako želi pristupiti resursu ali ga još uvek nema, upoređuje timestamp dolazne poruke sa timestampom poruke koju je sam poslao. Ako dolazna poruka ima niži timestamp, proces šalje nazad poruku OK. Ako njegova sopstvena poruka ima niži timestamp, proces stavlja dolazni zahtev u red i ne šalje ništa.

1. **Kako funkcioniše algoritam prstena za postizanje međusobnog isključivanja u distribuiranim sistemima?**

Algoritam prstena koristi logički prsten u kojem su procesi raspoređeni tako da svaki proces zna ko je sledeći u redu. Kada se prsten inicijalizuje, procesu P0 se daje žeton (token). Token cirkuliše oko prstena i prenosi se od jednog procesa do sledećeg u sekvenci. Kada proces dobije token, proverava da li treba da pristupi deljenom resursu. Ako treba, koristi resurs, obavlja potrebne zadatke, i zatim prosleđuje token dalje. Proces ne može odmah ponovo koristiti resurs sa istim tokenom. Ako proces ne treba resurs, jednostavno prosleđuje token sledećem procesu. Token cirkuliše kontinuirano, osiguravajući da samo jedan proces može pristupiti resursu u bilo kom trenutku, čime se garantuje međusobno isključivanje. Redosled u kojem token cirkuliše sprečava da bilo koji proces bude zanemaren.