OPERATIVNI SISTEMI – SIIT

**1. NITI.PPT**

**1. Sta je proces?**

- jedinica izvrsavanja i rasporedjivanja (procesor izvrsava instrukcije procesa isprepletano sa drugim procesima, a OS rasporedjuje te procese)

- jedinica kojoj se dodeljuje resurs (proces sadrzi virtuelni adresni prostor gde je slika procesa, i moze imati vlasnistvo nad ui resursom ili fajlom)

**2. Koja su stanja procesa?**

* Novi - proces upravo stvoren, nije ga jos OS prihvatio u red spremnih procesa
* Spreman - spreman za izvr, ali trenutno procesor ne izvrsava njegove instr
* Izvrsavanje - proces cije instrukcije trenutno procesor izvrsava
* Blokiran - proces koji se ne moze izvrs dok se ne pojavi neki dogadjaj npr kraj UI radnje
* Izlaz - proces koji je OS izbacio iz reda spremnih procesa jer je zavrsio rad

**3. Sta je visenitna obrada?**

- sposobnost OSa da podrzi vise konkurentnih putanja izvrsavanja jednog procesa

- nit je jedan tok izvrsavanja u okviru procesa

**4. Sta je proces u visenitnoj obradi?**

- jedinica dodele resursa (virt adr prostor gde je smestena slikom proc)

- jedinica zastite resursa (zasticen pristup: procesoru, drugim procesima (medjuprocesna komunik, fajlovima i U/I resursima)

**5. Od cega je nit sacinjena?**

* stanje izvrsavanja - izvrsavanje, spremna..
* kontekst niti (sacuvano stanje procesorskih registara dok je blokirana ili spremna)
* stek izvrsavanja niti
* pristup memoriji i resursima svog procesa (deljeno sa drugim nitima istog procesa)

Svaka nit ima svoj upravljacki blok i stek

Sve niti dele adresni prostor i imaju pristup istim podacima

**6. Koje su koristi visenitne obrade?**

* pozadinski posao - dugacke ili blokirajuce pozadinske oper se izvrsavaju u posebnim nitima
* takodje, interakcija sa korisnikom je moguca i dok druga nit nije zavrsila operaciju
* asinhrona obrada - naredba predaje posao drugoj niti
* nit koja je predala posao moze da nastavi druge naredbe dok se prethodna nije jos zavrsila
* brzina izvrsavanja - dok je 1 nit blokirana, druge mogu da rade
* deljenje resursa unutar apl - niti istog procesa dele istu memoriju i laksa i brza je komunik izmedju niti nego izmedju procesa
* ekonomicnost (brze napraviti nit od procesa, brza komunikacija izm niti nego procesa )
* iskoriscenje viseprocesne arhitekture - ubrzanje ako se niti izvrsavaju na razlicitim procesima
* modularna strukt programa - organizovanje koda u celine daje logicke delove programa te j jednostavnija implem i odrz

**7. Koja su moguca stanja niti?**

* spreman,
* izvrsavanje
* i blokiran

**8. Kojim se akcijama menja stanje niti?**

* kreiranje nove niti (spreman)
* blokiranje - kada nit mora da ceka neki dogadjaj te se proces prebacuje na neku drugu spremnu nit i cuva se sadrzaj proc. registara
* deblokiranje - desio se dogadjaj na koji je blokirana nit cekala
* i zavrsavanje

**9. Zasto je bitno sinhronizovati niti?**

-sve niti jednog procesa dele isti adresni prostor te je potrebno sinhornizovati pristupe ovim deljenim resursima zbog potencijalnih data race i nepravilnog rada programa

resursi mogu doci u nekonzistentno stanje

**10. Koji su tipovi niti?**

- User Level Threads (ULT)

* + sav posao upravljanja vrsi aplikacija
  + kernel nije svestan postojanja niti
  + niti se kreiraju koriscenjem biblioteke niti
  + biblioteka sadrzi kod za: pravljenje/unistavanje niti, prosldjivanje poruka i podataka izm niti i rasporedjivanje niti

(-) sve aktivnosti su unutar JEDNOG procesa i kada je on blokiran sve niti su blokirane sto onemogucava multiprocesiranje jer su sve niti 1 proces pa je on na jednom procesoru

(+) prednosti u tome sto ne zahteva prelazak u rezim kernela i aplikacija ima potpunu kontrolu nad rasporedjivanjem niti. mogu se izvrsavati na bilo kom os

KERNEL rasporedjuje procese i dodeljuje im stanje

Niti imaju svoje stanje kao logicka kategorija i nije direktno povezana sa stanjem procesa

Stanje niti oznacava status niti unutar procesa

npr nit je u stanju izvrsavanje:

znaci da je aktivna nit u okviru procesa a ne mora nuzno da znaci i da se trenutno izvrs na procesoru

kada procesor krene da izvrsava proces, izvrsavace se nit u stanju izvrsavanje iz tog procesa

- Kernel Level Threads (KLT)

* + aplikacija inicira kreiranje niti dok kernel upravlja njima i rasporedjuje ih
  + kernel odrzava podatke za ceo proces i za pojedinacne niti
  + (+) prednost u tome sto se moze iskoristiti multiprocesiranje i rasporediti vise niti istog procesa na vise procesora!
  + ako je jedna nit procesa blokirana, kernel moze da rasporedi drugu nit istog procesa
  + (-) mane su sto se zahteva prelazak u rezim kernela da bi se preslo sa jedne na drugu nit, kao i to sto je komutacija za jedan red velicine sporija od ULT
* Kombinovani pristup jeste da se veci deo upravljanja i rasporedjivanja odvija u korisnickom prostoru, a niti su preslikane na manji ili jednak deo njih u kernelu.

**2. KONKURENTNOST.PPT**

**1. Sta sve moze da omoguci OS prilikom upravljanja procesima?**

* Multiprogramiranje - vise procesa u jednoprocesorskom sistemu
* Multiprocesiranje - vise procesa unutar multiprocesora
* Distribuiranu obradu - vise procesa na vise distribuiranih racunara

**2. Sta je konkurentnost?**

- Konkurentnost je vazan aspekt pri svakom upravljanju visestrukim procesima

- Tice se pitanja medjusobne interakcije i zavisnosti procesa.

**3. Gde se javlja konkurentnost?**

- Svuda gde se vrsi preplitanje procesa jer je scenario i trenutak preplitanja nepredvidiv!

- kod multiprogramiranih sistema razliciti procesi pristupaju istim resursima

- jedna aplikacija moze imati vise konkurentnih procesa-niti

- funkcije OSa su implementirane kao vise procesa-niti

**4. Koje probleme donosi konkurentnost?**

- Deljenje globalnih resursa - stetno preplitanje - dva procesa vrse upis i citanje nad istom globalnom promenljivom a ne zna se redosled tih procesa.

- Komplikovanija dodela resursa - resurs moze biti dodeljen procesu koji ga ne koristi, procesi mogu biti medjus blokiranju zbog cekanja na resurse

- Detekcija gresaka - teze utvrditi zasto dolazi do neocekivanog ponasanja - rez nisu deterministicki

**5. Kako se prevazilaze problemi konkurentnosti?**

* Medjusobna iskljucivost. - obavljanje akcije bez negativnog uticaja drugih procesa
* Sinhronizacija. - uskladjivanje ponasanja procesa sa aktivnoscu drugih procesa

**6. Sta je data race?**

- Rezultat programa zavisi od scenarija preplitanja i redosleda izvrsavanja instrukcija izmedju procesa, dakle nema deterministickog ponasanja.

- Desava se kada ostoji deljeni resurs i pristup njemu nije sinhronizovan.

- Ponovna izvrsavanja istog koda ne daju isti rezultat!!!

**7. Kada dolazi do uzajamne iskljucivosti procesa?**

- Kada vise procesa treba da koristi isti resurs.

- Za ispravan rad je potrebno da u jednom trenutku jedan proces pristupa \*kriticnom resursu.

**8. Sta je kriticna sekcija?**

- Deo programa u kojem se pristupa kriticnom resursu.

- Za ispravan rad, kod u kriticnoj sekciji procesi moraju da izvrsavaju jedan po jedan

- Samo jedan proces u jednom trenutku moze biti u kriticnoj sekciji.

**PONASANJE DRUGIH PROCESA AKO JE JEDAN PROCES VEC U KRITIČNOJ SEKCIJI:**

**9. Sta je busy waiting?**

- Proces koji ne moze da udje u kriticnu sekciju ostaje aktivan i konstantno proverava da li moze da udje.

- Ovo resava problem, ali nije efikasno jer konstantno trosi procesorsko vreme cak i kada ne mora da radi.

**10. Sta je blokiranje procesa?**

- Proces koji ne moze da udje u kriticnu sekciju odlazi u stanje BLOKIRAN.

- Sistem ga obavestava o tome kada moze da udje -> Znatno efikasniji pristup.

**11. Sta je propusnica?**

- Predstavlja kljuc deljenih resursa.

- Proces koji pokusavju da udju u k.s. pokusavaju da zakljucaju deljene resurse.

- Prvi koji uspe da zakljuca deljeni resurs ulazi u k.s. a ostali prelaze u stanje blokiran.

- Kada proces zavrsi rad, otkljucava resurs i neki od blokiranih procesa uspeva da ga zauzme i zakljuca, te tada taj ulazi u k.s....

**12. Koji su moguci pristupi hardverske podrske za uzajamnu iskljucivost?**

- Onemogucavanje prekida - proces se izvrsava do prekida

ne moze doci do preplitanja ali je neefikasno i ne radi na multiproc arh

- Specijalne masinske instrukcije koje obavljaju atomicne operacije.

Proces u jednom nedeljivom koraku proverava da li moze da zauzme propusnicu i ako moze postavlja fleg da je zauzeo.

Primeri su instrukcije Compare&Swap i Exchange\*\*\*\* pogledati slajdove ☹

**13. Sta je deadlock?**

- Situacija kada svi procesi uzajamno cekaju da drugi procesi oslobode deljeni resurs.

- Jedni druge blokiraju i niko ne napreduje niti pušta resurs koji je nekome drugom potreban

**14. Sta je livelock? \* pogledati online**

- Situacija u kojoj dva ili vise procesa nisu blokirani ali menjaju stanja tako da se medjusobno "blokiraju", odnosno onemogucavaju napredak izvrsavanja.

- Nijedan proces nije blokiran ali ni ne napreduje u vrsenju korisnog rada

**15. Sta je gladovanje?**

- Situacija u kojoj proces, iako je spreman, nikad ne dobija procesor od rasporedjivaca.

**\* Zahtevi za medjusobnu iskljucivost?**

-Medjusobna iskljucivost se mora sprovesti ako u jednom trenutku sme samo jedan proces da koristi neki resurs

-Ne sme se desiti uzajamno blokiranje ili gladovanje tj. proces ne sme beskonacno da ceka ulaz u k.s.

-U k.s. se ulazi bez odlaganja (odmah) ukoliko nijedan drugi proces nije u k.s.

-Ne mogu se praviti predpostavke o relativnim brzinama procesa

-Proces mora izaci iz k.s. u konacnom vremenu

**16. Sta je sinhronizacija?**

- Uvodjenje kriticne sekcije moze da obezbedi sekvenc pristup resursima

- Sinhronizacija je uvodjenje eksplicitnog redosleda izvrsavanja odredjenih procesa u cilju pravilnog izvrsavanja programa.

**17. Koji su koncepti viseg nivoa za sinhronizaciju?**

* Semafori
* Uslovne promenljive
* Monitori

**\* Metode semafora**

* Semafor je deljeni brojac, metode su mu atomicne! \*slajd!
* semWait ili P ili down - smanjuje brojac, sacekaj da brojac bude >= 0
* semSignal ili V ili up - povecava brojac

**18. Koje su varijante semafora prema tipu brojaca?**

* Brojacki/Opsti semafor - brojac moze dobiti proizvoljnu celobrojnu vrednost
* Binarni - brojac moze imati vrednost 0 ili 1 (ponasa se kao mutex)

\*pogledaj primer semafora u cpp

**19. Koje su varijante semafora po nacinu rasporedjivanja?**

- Jak semafor - FIFO princip uvezivanja procesa, garantuje da nece doci do gladovanja

- Slab semafor - nema odredjenog redosleda koji ce proces postati spreman, ne garantuje da nece doci do gladovanja

**20. Koji su problemi sa Semaforom?**

- Cesto je algoritmima potrebno vise semafora

- Pozivi semSignal i semWait nisu upareni

- Neispravan rad ako redosled operacija wait i signal nije odgovarajuci

- Tesko je debagovati takve greske

- Koristi se istovremeno za obezbedjivanje medjusobne iskljucivosti i sinhronizacije, sto je pozeljno da bude odvojeno.

**21. Sta je uslovna promenljiva?**

- Mehanizam za sinhronizaciju putem signalizacije izmedju niti.

- Niti imaju USLOV na koji moraju cekati da bude ispunjen i za koji nit moze obavestiti druge niti da je ispunjen.

**22. Koje operacije ima mehanizam uslovne promenljive?**

- Wait - cekaj dok god uslov nije ispunjen

- Signal - obavesti jednu nit da je uslov ispunjen (notify\_one)

- Broadcast - obavesti SVE niti da je uslov ispunjen (notify\_all)

c++ - condition\_variable klasa, metode se pozivaju unutar k.s. zasticene unique\_lock objektom

**\*. wait metoda ?**

**POGLEDAJ SVESKU!**

-Nit se blokira i ceka dok neka druga nit nad ovim cv objektnom ne pozove notify ili notify all

-Pre blokiranja niti, propusnica se otkljucava, nit pred blokiranje pusta propusnicu

-Da bi izasla iz wait, mora se sacekati na propusnicu

-Nakon izlaska iz wait, mora se ponovo proveriti uslov jer se mozda promenio dok je nit dobila propusnicu

--zato koristimo wait unutar while (uslov) petlje

-Lazno budjenje - spurious wakeup - nit moze izaci iz cekanja cak iako nije notificirana \*google

**23. Sta je monitor?**

- Softverski sablon koji ENKAPSULIRA podatke, propusnicu, operacije nad podacima i uslovne promenljive za sinhronizaciju.

- Ponasa se kao black-box koji pomocu metoda za pristup podacima obezbedjuje interfejs za rad sa njima.

- Monitor garantuje ispravan rad i sinhronizaciju metoda.

-Procesi vide monitor kao black box i pristupaju podacima putem njegovih metoda

-Te metode se delom izvrsavaju sekvencijalno jer je potrebno zauzeti propusnicu

-Te metode implementiraju internu logiku sinhronizacije i mogu blokirati nit

-Niti ne moraju da vode racuna o sinhr, to radi monitor!

**24. Koji su tipovi monitora?**

- Hoare monitori - operacija signal odmah aktivira signaliziranu nit, a nit koja je izvrsila signalizaciju se blokira

- Mesa monitori - signal samo prevodi nit u stanje spreman,

a nit koja je izvrsila signalizaciju nastavlja izvrsavanje.

signalizirana nit ceka propusnicu i kada se aktivira mora ponovo proveriti uslov!

C++ notify radi po ovom principu

**3. UZAJAMNO BLOKIRANJE.PPT**

**1. Sta je uzajamno blokiranje?**

- Trajno blokiranje skupa procesa koji se nadmecu za isti skup resursa.

- Svaki proces u skupu je blokiran čekajući na dogadjak koji može aktivirati neki drugi blokirani proces skupa

**2. Kakav model sistema moze omoguciti dolazak do uzajamnog blokiranja?**

- Sistemi koji koriste resurse na sledeci nacin:

Zahtev resursa uz cekanje -> Upotreba resursa -> Otpustanje resursa nakon koriscenja.

**3. Koji su uslovi za uzajamno blokiranje?**

- Uzajamno iskljucivanje - u jednom trenutku moze samo jedan proces koristiti resurs

- Drzanje i cekanje - proces drzi resurse dok ceka dodeljivanje drugih resursa

- Bez prekida - ne moze se nasilno oduzeti resurs od procesa koji ga drzi

- Kruzno cekanje - potencijalni rezultat prva 3 uslova, stvara se zatvoren krug procesa koji drze resurse potrebne ostalima.

Prva tri su uslovi za mogucnost uzajamnog blokiranja, a sva cetiri su za postojnje uzajamnog blokiranja!!!

**4. Koji su pristupi resavanja problema uzajamnog blokiranja?**

* Sprecavanje - mehanizmi koji eliminisu uslove koji dovode do u.b.
* Izbegavanje - dinamicka raspodela resursa takva da ne dolazi do u.b.
* Otkrivanje - mehanizmi za oporavak od u.b. nakon sto se u.b. desi

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\* Sta je ideja sprecavanja?**

-Projektovati sistem tako da nije moguce doci do u.b. eliminisanjem uslova koji dovode do u.b.

-Ideja je spreciti pojavu jednog od 4 uslova za u.b.

-Ne moze se spreciti prvi uslov uzajamnog iskljucivanja jer deljenim resursima procesi moraju pristupati 1 po 1.

-Moze se spreciti uslov DRZANJE I CEKANJE, ukinuti uslov BEZ PREKIDANJA ili spreciti KRUZNO CEKANJE!

**5. Koje su mane sprecavanja uslova DRZANJE I CEKANJE?**

- Moguce je ukoliko proces trazi sve potrebne resurse odjednom

-> neefikasno jer ih zakljucava iako neke ne koristi, plus on ceka da svi ti potrebni resursi budu slobodni, te moze doci do gladovanja tog procesa

- proces je blokiran sve dok SVI resursi koje trazi ne postanu free, cak ni ne koristi one koje moze da koristi dok ne dobije sve

**6. Kako funkcionise odbacivanje uslova BEZ PREKIDANJA? DVE VARIJANTE!**

- a) Vrsi se naredba procesu da otpusti resurse ukoliko mu je odbijen zahtev za zauzimanje nekog resursa.

- b) Kada neki proces zatrazi resurs a drugi proces ga drzi, taj drugi proces se prekida i otpusta mu resurse

\* Primenljivo je samo kod resursa cije se stanje moze sacuvati (npr. proc registri).

**7. Kako funkcionise sprecavanje kruznog cekanja?**

- Ukoliko se uradi linearno slaganje tipova resursa, svaki resurs dobija indeks.

- Resursu Rj moze pristupiti proces koji je zauzeo Ri samo ako je j > i.

- Moze biti neefikasno jer se procesu mozda nepotrebno zabranjuje zauzimanje resursa.

- Ako programer ne postuje predvidjeni redosled opet moze doci do u.b.

\***REZIME SPRECAVANJA U.B.**

-Sprecavanje u.b. je isuvise restriktivno!

-Slaba je iskoriscenost resursa i smanjena je propusna moc sistema

-Manje restriktivni pristup koji dozvoljava vise konkurentnosti je IZBEGAVANJE U.B.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**8. Koje su karakteristike pristupa izbegavanja uzajamnog blokiranja?**

- Dinamicki se vrsi raspodela resursa tako da ne dodje do u.b.

- Dozvoljeno je postojanje tri neophodna uslova u.b.

- Prave se razumni izbori tako da nikad ne dodje do u.b.

- Dinamicki se donosi odluka o tome da li ce tekuci zahtev dodele resursa dovesti do u.b.!!!

- Omogucuje vise konkurentnosti i manje je restriktivan u odnosu na sprecavanje u.b.

**9. Koje su varijante u pristupu izbegavanja uzajamnog blokiranja?**

a) Ne pokretati ceo proces ukoliko njegovi zahtevi mogu dovesti do u.b.

b) Ne obradjivati zahtev procesa za resursom ako ta dodela moze dovesti do u.b.

**10. Sta sve stanje sistema dinamicke raspodele resursa sadrzi?**

- Vektor R - ukupna kolicina svakog resursa sistema

- Vektor V - raspolozivo - ukupna kolicina svakog resursa sistema koja nije dodeljenja nijednom procesu

- Matricu zahteva C - MORA BITI UNAPRED ODREDJENO ZA SVAKI PROCES

- Matricu dodele A - trenutna dodeljenost resursa procesima

- Ogranicenja:

Rj = Vj + Sum(i=1, n)Aji (svi resursi su raspolozivi ili dodeljeni)

Cij <= Rj - resurs ne moze zahtevati vise od ukupne kolicine R

Aij <= Cij - resursu ne moze biti dodeljeno vise od zahtevanog

Proces nece biti pokrenut ako njegovi zahtevi za resursima mogu dovesti do u.b.

**11. Uslov za pokretanje novog pocesa.**

Rj >= C(n+1)k + Sum(i=1, n)Cj

- u prevodu: Pokreni proces samo ako se mogu zadovoljiti maksimalni zahtevi tekucih procesa uvecani za zahteve novog procesa

- Suvise je restriktivno - pretpostavlja se da ce svi procesi istovremeno koristiti sve resurse

**12. Na cemu se zasniva odluka o dodeli resursa?**

- Na osnovu toga da li je sistem u bezbednom ili nebezbednom stanju.

**13. Kada je sistem u bezbednom stanju?**

- Sistem je u bezbednom stanju ako moze ispuniti zahteve za resursima svakog procesa u nekom redosledu koji ne dovodi do u.b.

- To znaci da postoji sekvenca u kojoj bi zahtevi trebali biti obradjivani, odnosno resursi dodeljivani, tako da svaki proces ceka konacno vreme!!!

- Kada je stanje nebezbedno moze, ali ne mora doci do u.b.

- U bezbednom stanju nema u.b.

**14. Kako se proverava da li je sistem u bezbednom stanju?**

- Ako postoji sekvenca procesa u kojoj vazi da za svaki Pi, resursi koje zahteva Pi mogu biti ispunjeni sa njegovim trenutnim raspolozivim resursima plus resursima koje koriste procesi prethodni u odnosu na njega (Pj gde je j<i).

- Tada jeste bezbedno stanje

**15. Koje su karakteristike algoritma za izbegavanje uzajamnog blokiranja?**

- Utvrdjuje da li postoji sekvenca u kojoj procesi zauzimaju i oslobadjaju resurse tako da su zahtevi svih ispunjeni.

- Cilj je da sistem bude u bezbednom stanju

- Kada proces napravi zahtev, simulira se da je on odobren i proverava se kakvo je novnonastalo stanje sistema.

- Ukoliko je sistem u bezbednom stanju zahtev je odobren, u suprotnom proces se blokira.

**16. Opisati korake bankarevog algoritma - algoritam utvrdjivanja bezbednog stanja.**

1. Kreiramo vektor W koji predstavlja simulaciju stanja raspolozivih resursa u nekom buducem trenutku. Inicijalno je W == V (trenutno raspolozivo stanje resursa).

2. Pronalazimo proces ciji se zahtevi trenutno mogu ispuniti. Cij - Aij <= Wj za svako j=1, ..., m Ako ne uspemo algoritam se prekida

3. Simuliramo da je proces zauzeo resurse, zavrsio rad i oslobodio ih. Wj += Aij

4. Vratimo se na 2. i ukoliko uspemo da prodjemo kroz sve procese i za sve simuliramo dodelu resursa, onda je stanje bezbedno.

**17. Navesti prednosti izbegavanja uzajamnog blokiranja.**

- Manje je restriktivno od sprecavanja

- Nije neophodno prekinuti i vratiti unazad procese kao kod OTKRIVANJA uzajamnog blokiranja

**18. Navesti mane izbegavanja uzajamnog blokiranja.**

- Maksimalni zahtevi za resursima svakog procesa se moraju unapred definisati

- Proces koji se razmatra mora biti nezavisan od drugih procesa (redosled izvrsavanja nije sinhronizovan)

- Mora postojati fiksan broj resursa za dodelu

- Proces ne moze da otpusti resurs

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**19. Koje su karakteristike pristupa otkrivanja uzajamnog blokiranja?**

- Kada se uzajamno blokiranje desi, vrse se mehanizmi za oporavak.

- Ne ogranicava se pristup resursima.

- Procesima se dodeljuju zahtevani resursi.

- Povremeno se vrsi provera da li je doslo do kruznog cekanja

- Ako da, vrsi se oporavak sistema

**20. Opisati korake algoritma za otkrivanje uzajamnog blokiranja.**

- Cilj je da oznacimo procese koji nisu uzajamno blokirani, ostali jesu

1. Inicijalizujemo W = V

2. Proveravamo da li postoji proces P koji ima sve nule u matrici dodela - on onda sigurno nije u u.b. i oznacavamo ga

3. Pokusavamo da nadjemo P ciji zahtevi su manji ili jednak sa W, ako ga ne nadjemo kraj algoritma

4. Ukoliko smo ga pronasli, oznacavamo ga, i W = W + Pi (simuliramo da se proces izvrsio i oslobodio je svoje resurse)

5. Vracamo se na korak 3, trazimo da li jos neki proces ispunjava taj uslov

Preostali neoznaceni procesi su uzajamno blokirani!!!!

**21. Opisati strategije za oporavak od uzajamnog blokiranja.**

1. Prekini sve u.b. procese - najjednostavnije, velika je cena ovoga jer ponistava sve sto je proces uradio do tad

2. Vrati sve u.b. procese na neku raniju kontrolnu tacku i ponovo ih pokreni - moze opet doci do u.b., ali zbog nedeterministicnosti u ponovnom izvrsavanju mozda ne bude u.b.

3. Redom prekidaj sve u.b. procese dok vise ne bude blokiranja - treba uspostaviti kriterijum po kom ce se vrsiti redosled prekidanja (prioritet, utrosak procesorskog vremena, predvidjeno preostalo vremen izvrsavanja, tip koriscenog resursa, kolicina resursa potrebna za zavrsetak)

4. Redom oduzimaj resurse od procesa i dodeljuj ih drugim procesima dok vise ne bude blokiranja - treba izabrati resurs i proces tako da se minimizuju troskovi.

Proces kojem je oduzet resurs mora se vratiti na neko prethodno stanje u kom je bio pre zauzimanja tog resursa.

Mora se spreciti potencijalno gladovanje (ne smeju se resursi oduzimati uvek istom procesu).

**22. Kakve su strategije za uzajamno blokiranje koriscene u savremenim os?**

- Tipicno ne pruzaju zastitu od u.b.

- Ignorise se mogucnost pojave u.b.

- OS se implementira sa namerom da ne ulazi u deadlock, a korisnicki programi su odgovornost programera

**23. Problem 5 filozofa? Slajd**

**4. OS PREGLED.PPT**

**1. Sta je operativni sistem?**

- Program koji upravlja izvrsavanjem aplikacionih programa i sluzi kao interfejs izmedju programa i hardvera racunara.

**2. Koji su ciljevi OS?**

- Treba da obezbedi pogodnost, efikasnost i mogucnost izvrsavanja novih korisnickih aplikacija. Tj, omoguciti dodavanja novih funkcionalnosti sistema nezavistno od ugradjenih osnovnih servisa

**3. OS kao interfejs izmedju korisnika i racunara?**

- Korisnik vidi racunarski sistem kao skup aplikacija.

- Te aplikacije mogu biti implementirane kao skup masinskih instrukcija koje su u potpunosti odgovorne za upravljanje hardverom racunara, ali to bi bilo pretesko za implementirati.

- Zato se obezbedjuje skup sistemskih programa, poput OS, koji obavljaju interakciju sa hardverom.

- Aplikacije se obracaju sistemskom programu da obavi odredjene funkcije

- Najvazniji sistemski program je OS

**4. Koji sve tipovi interfejsa postoje u racunarskom sistemu?**

- API - interfejs aplikativnog programa - servisi koje aplikacija moze pozivati (skup funkcija i klasa u nekoj lib npr)

- ABI - binarni interfejs aplikacije - API nizeg nivoa (veza sa OS). Definise format komunikacije sa OS (sistemski pozivi, nacin predstavljanja tipova podataka, konvencija poziva fja...)

- ISA (Instruction Set Architecture) - *Arhitektura skupa instrukcija* - skup instrukcija masinskog jezika koje hardver podrzava

**5. Koja su dva osnovna nacina komunikacije izmedju korisnika i racunara?**

- Komandna linija (kao veza ka OSu)

- GUI

**6. U okviru cega se interpreter komandi moze nalaziti?**

- Moze biti ugradjen u kernel

- Moze biti implementiran kao poseban program u okviru OS (shell) - windows i unix

**7. Koja su dva nacina za interpretiranje komandi?**

- Interpreter komandi sadrzi u sebi kod za izvrsavanje komande

- Komande se izvrsavaju pozivom sistemskih programa - cesce

Komandna linija ne razume komande nego samo sluzi sa identifikaciju fajla koji ce biti ucitan i izvrsen

“ls test” komanda startuje program ls i prosledjuje test kao parametar

**8. Nabroj standardne usluge OS:**

- Izvrsavanje programa - upravlja resursima koje program koristi, ucitavanje instrukcije i podataka u glavnu mem, inicijalizacija UI uredjaja..)

- Pristup U/I uredjajima - OS je interfejs za pristup UI uredjajima - programer zahteva ui oper kroz interfejs, a dalje os vodi komunikaciju s ui uredjajem

- Pristup fajlovima - OS vodi racuna o internoj strukturi uredjaja za skladistenje i nacinu skladistenja i obezbedjuje kontrole pristupa fajlovima

- Deljeni pristup sistemu - OS obezbedjuje zastitu resursa od neovlascenog pristupa i resava konflikte kod takmicenja za resurse

- Upravljanje greskama - OS obezbedjuje reakciju na hardverske i softverske greske (prekidi programa, prijava greske, ponavljanje opercije..)

- Nadzor sistema - OS vrsi statistiku upotrebe resursa i nadgleda performanse

**\*OS kao upravljac resursa?**

-Racunar = skup resursa za prenos, skladistenje i obradu podataka

-OS odgovoran za upravljanje ovim resursima

Funkcionise na isti nacin kao i ostali softver!

Os je program ili skup programa koje izvrsava procesor

**Razlika u odnosu na drugi softver je NAMENA - softver OSa sadrzi instrukcije kojim se upravlja resursima sistema (rasporedjivanje aplikacija, zauzimanje memorije..)**

slika slajd 17!!!!

**9. Koji pristupi razvoja OSa su postojali kroz istoriju?**

* Serijska obrada
* Jednostavni sistem paketne obrade
* Multiprogramirani sistemi paketne obrade
* Sistemi sa deljenjem vremena

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\*Karakteristike serijske obrade**

-Racunari nisu imali OS, nego su korisnici direktno pristupali hardveru

-Samo jedan korisnik je u jednom trenu imao pristup racunaru

**10. Koji su problemi serijske obrade?**

- Korisnici se upisuju na spisak za rezervisanje racunara i tesko je organizovati taj raspored ako izvrsavanje traje duze od planiranog, a ako je krace onda je neracionalno koriscenje!

- Postavljanje programa je uzimalo znacajan deo vremena.

- Direktno se upravlja hardverskim resursima.

- Nema istovremenog izvrsavanja vise programa.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\*Karakteristike jednostavnih sistema paketne obrade?**

-Korak ka efikasnijem koriscenju racunara

-Nema pristupa racunaru direktno

-Korisnici predaju poslove operatoru, koji ih kao skup programa postavlja na racunar

-Pripremljen skup programa izvrsava poseban softver - MONITOR - program sa namenom da upravlja izvrsenjem drugih programa

**11. Sta je monitor u jednostavnom sistemu paketne obrade?**

- Program sa posebnom namenom da upravlja izvrsavanjem drugih programa.

- Izvrsava jedan po jedan program

- Programi se nakon izvrsavanja vracaju na monitor

- Nakon toga se moze ucitati naredni program.

**\*Funkcije monitora?**

-Monitor kontrolise sekvencu programa

-Poseban deo memorije je namenjen smestanju programa monitora

-Monitor ucitava korisnicki program i predaje mu kontrolu

-Korisnicki program vraca kontrolu monitoru kada zavrsi rad

**12. Koje su funkcije procesora u jednostavnom sistemu paketne obrade?**

- Inicijalno izvrsava instrukcije iz dela memorije gde je monitor smesten.

- Kada monitor ucita program procesor ce izvrsavati instrukcije grananja i krenuti u izvrsavanje korisn.prog. sve dok ne dodje do kraja programa ili greske.

- Predati kontrolu korisnickom programu znaci da procesor prelazi da izvrsava instrukciju i delu memorije gde je korisn. program

- Vracanje kontrole monitoru znaci da procesor narednu instrukciju izvrsava iz programa monitora

**13. Koji su tipovi hardverske podrske za monitor?**

- Zastita memorije - korisn. program ne sme pristupati delu memorije gde je monitor

- Vrem brojac - program ne sme predugo koristiti procesor

- Privilegovane instrukcije - deo instrukcija moze da izvrsi samo monitor

- Prekidi - daju vise fleksibilnosti jer omogucavaju predaju i oduzimanje upravljanja programu

**14. Koji su rezimi izvrsavanja u jednostavnom sistemu paketne obrade?**

* Korisnicki rezim - za korisnicke programe, ogranicen pristup memoriji, nije moguce neke instrukcije izvrsavati
* Kernel rezim - za izvrsavanje monitora, privilegovane instr + neogrnicen pristup memoriji

**15. Koja ideja stoji iza multiprogramiranih sistema paketne obrade?**

- Da se programi vise ne delegiraju monitoru sekvencijalno, vec je vise programa istovremeno postavljeno za izvrsavanje, ali i dalje u jednom trenutku procesor moze izvrsavati samo jedan program.

**16. Koji se problemi javljaju iz toga sto je procesor posvecen samo jednom programu?**

- Slabo iskoriscenje procesora. Npr dok se ceka izvrsenje U/I operacije procesor je slobodan i ceka jer U/I uredjaji su dosta sporiji od procesora.

**\*Jednoprogramiranje?**

-Procesor izvrsava instrukcije dok ne dodje do UI instr, tada mora da ceka kraj UI oper da bi nastavio rad

**17. Sta je multiprogramiranje?**

- Viseprocesna obrada podataka.

- Pored OS u mem se smesta vise korisnickih programa.

- Dok jedan program ceka U/I, procesor moze da pređe da izvrsava neki drugi program iz memorije.

-> veca upotreba procesora, manje ukupno vreme, veca propusna moc, manje vreme odziva..

**18. Koji su tipovi hardverske podrske za multiprogramiranje?**

- U/I prekidi

- DMA - direktan pristup memoriji (direct memory access)

- Procesor izdaje naredbu za U/I operaciju i nastavlja izvrsavanje drugog posla.

Kontroler za U/I obradjuje U/I operaciju i nakon izvrsavanja postavlja prekid i predaje kontrolu programu za obradu prekida OS.

OS ce predati kontrolu nekom drugom spremnom poslu.

**19. Sta OS treba omoguciti da bi se moglo razviti multiprogramiranje?**

- Upravljanje memorijom – poslovi (procesi) koji se izvrsavaju moraju biti u memoriji istovremeno

- Rasporedjivanje posla - algoritam rasporedjivanja koji odlucuje koji se posao sledeci izvrsava i koliko dugo

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**20. Koja je ideja u sistemima sa deljenjem vremena?**

- Podrska VISEKORISNICKOM pristupu racunarskim resursima

- Procesor se deli izmedju vise korisnika.

- Korisnici istovremeno pristupaju sistemu preko terminala a OS preplice izvrsavanje korisnickih programa

- Posto je vreme reakcije ljudi relativno sporo, gotovo je neprimetna razlika u odnosu na sisteme sa jednokorisn rezimom.

**\*Uporedi multiprogramiranu paketnu obradu i deljenje vremena?**

-MPO obezbedjuje multiprogramiranje a DV visekorisnicki rad

-MPO ima cilj sto vece iskoriscenosti procesora a DV smanjenje vremena odgovora

**21. Navedi primer realizacije sistema sa deljenjem vremena:**

- CTSS (Compatible Time Sharing System)

- Ucitava i izvrsava jedan po jedan korisnIcki program

- Na svake 0.2 sekunde se pravi prekid

- Pri svakom prekidu OS preuzima upravljanje i moze dodeliti procesor drugom korisniku

**22. Koji su zahtevi za savremene OS?**

- Multiprogramiranje i Deljenje vremena su uveli standardne zahteve za dizajn savremnih OS

- Multiprogramiranje uvelo zahtev deljenja resursa i konkurentni pristup, takmicenje za resurse, zastitu od stetnog uplitanja

- Deljenje vremena uvelo zahtev visekorisnickog pristupa resursima i zastitu i rasporedjivanje

**23. Koja su glavna dostignuca savremenih OS?**

* Procesi
* Upravljanje memorijom
* Zastita informacija i bezbednost
* Rasporedjivanje i upravljanje resursima

**24. Sta je proces?\*\***

- Jedinica aktivnosti koju karakterisu naredbe za izvrsavanje, tekuce stanje i dodeljeni skup sistemskih resursa.

- Program u izvrsavanju.

- Primerak programa koji se izvrsava na racunaru

-Entitet koji se moze dodeliti i izvrsavati na procesoru

**25. Navedi tri dela koja cine proces:**

- Izvrsni program - instrukcije koje procesor treba da izvrsi

- Podaci - pridruzeni podaci koji su potrebni programu za izvrsavanje - promenljive

- Kontekst izvrsavanja (stanje procesa) - interni podaci: stanje registara, prioritet procesa, info o tome da li ceka na U/I ...

**\* Model upravljanja procesom**

SLIKA SLAJD 56!

-U memoriji ima lista procesa gde za svaki proces ima ukazivac na blok memorije gde se nalazi

-Kontekst procesa moze biti smesten sa samim procesom ili odvojeno

-Proces koji se trenutno izvrsava na procesoru

-Naredna instrukcija u aktivnom procesu

-Pocetna adresa bloka u kojem je proces za relativno referenciranje programskom brojaca i podataka

-Duzina bloka u kom je proces za zaštitu pristupa memoriji - ne sme se referencirati lokacije van granica!

**\*Visenitna obrada**

-Proces je podeljen na *niti* koje se izvrsavaju konkurentno

Nit je izvršna jedinica posla, sadrzi kontekst i sopstvenu oblast podataka

Izvrsava se sekvencijalno i moze se prekinuti

Proces je skup 1 ili vise niti i dodeljenog sistemskog resursa

**26. Koje su odgovornosti OS pri upravljanju memorijom?**

- izolacija procesa - ne sme pristupiti memoriji koja je rezervisana za neki drugi proces

- automatsko dodeljivanje i upravljanje - aplikacija ne mora direktno da vodi racuna o upravljanju memorijom

- zastita i kontrola pristupa - razliciti korisnici mogu na razlicite nacine da pristupe odredjenim delovima mem

- dugotrajna memorija - OS mora da omoguci programima da trajno skladiste info (fajl sistem)

**27. Sta je virtuelno adresiranje?**

- Mehanizam koji omogucuje da programi vrse adresiranje sa logicke tacke gledista.

- Program ne vodi racuna o kolicini fizicke memorije.

- Program ne vodi racuna o stvarnoj adresi u glavnoj memoriji vec se ona odredjuje tokom izvrsavanja, a ne u toku pisanja.

**28. Sta omogucuje sistem stranicenja?**

- Proces je podeljen na neki broj blokova fiksne duzine - stranica

- Program adresira rec pomocu virtuelne adrese koja se sastoji od broja stranice i pomeraja unutar nje

- Stranica moze biti bilo gde u glavnoj mem

- Stranicenje pruza dinamicko mapiranje virtuelne adrese koja se koristi u programu u fizicku (realnu) adresu

**29. Kako se vrsi adresiranje virtuelne mem?**

- Ne moraju sve stranice procesa biti u glavnoj mem sve vreme, sve su na disku

- Ako je referencirajuca stranica na disku, ucitava se u glavnu mem.

- Ukoliko je glavna memorija puna, mora se izbaciti neka stranica nazad na disk.

- OS odredjuje politiku dodeljivanja: koje i koliko stranica ce biti u memoriji, algoritam zamene stranica, algoritam prebacivanja str iz glavne mem nazad na disk...

**30. Sta zastita informacija i bezbednost moraju obezbediti?**

- Raspolozivost - zastita sistema od prekida rada.

- Poverljivost - pravo pristupa odredjenim podacima

- Integritet podataka - zastita od nedozvoljene izmene

- Autenticnost - provera identiteta i validanosti poruka

**31. Sta OS treba da obezbedi u kontekstu upravljanja resursima i rasporedjivanja?**

- Nepristrasnost - poslovi slicnog tipa koji se takmice za resurs treba da dobiju isti i fer pristup tom resursu

- Razlicitost odziva- namerna pristrasnost odredjenom procesu ako to omogucava bolje ispunjenje celokupnih zahteva

- Efikasnost - pokusava da uveca propusnu moc, a smanji vreme odziva - treba naci balans

**32. Koji su elementi OS za rasporedjivanje?**

- Kratkorocni red procesa - red procesa spremnih za izvrsavanje, dodeljivanje se moze vrsiti kruzno ili prema prioritetu

- Dugorocni red procesa - red procesa koji cekaju da udju u kratkorocni red

- Za svaki U/I uredjaj postoji red procesa koji cekaju na upotrebu tog uredjaja

- Po opsluzivanju prekida ili sistemskog poziva, kratkorocni rasporedjivac odredjuje proces koji ce naredni da se izvrsi

**33. Sta je SMP (Symmetric Multiprocessing)?**

- Simetricna VIŠEPROCESNA obrada.

- Vise procesora deli istu glavnu memoriju i U/I uredjaje.

- Svi procesori mogu da obavljaju istu funkciju.

- Vise procesa moze paralelno da se izvrsava

- OS vodi racuna o rasporedjivanju niti ili procesa na procesore i sinhronizaciju izm njih

**34. Koje su prednosti viseprocesne obrade - SMP?**

- Performanse - paralelno obavljanje posla -> bolje perform

- Raspolozivost - Otkaz jednog procesora nece zaustaviti sistem, samo ce biti manje perform

- Postepeno poboljsanje sistema - moguce je poboljsati perform sistema dodavanjem procesora

- Skaliranje - cena racunara bazirana na broju procesora i performansi

**35. Opisati dizajn OS za SMP?**

- simultani konkurentni procesi ili niti - kernel treba da podrzi konkurentni pristup od strane vise procesa koji se izvrsavaju istovremeno na razlicitim procesorima

- rasporedjivanje - svaki procesor moze da sprovodi rasporedjivanje sto znatno komplikuje stvari

- sinhronizacija - medjusob iskljucivost i signaliziranje vise aktivnih procesa istovremeno koji imaju pristup deljenim adresnom prostoru i UI resursima

- upravljanje memorijom - iskoristiti paralelizam hardvera za bolju performansu

- pouzdanost i otpornost na greske - OS treba da reaguje na gubitak procesora smanjenjem performanse i ponovnim rasporedjivanjem

**36. Sta su sistemski pozivi?**

- API servisi koje OS pruza i pomocu kojih se izvrsavaju odredjene funkcionalnosti OSa.

- Mogu se pozvati direktno ili koriscenjem sistemske biblioteke (indirektni).

**\*Direktni i indirektni pozivi**

- Kod direktnih sistemskih poziva korisnicka aplikacija zahteva uslugu OSa slanjem sistemskog poziva

- OS u rezimu kernela izvrsava taj zadatak i vraca odgovor aplikaciji

-indirektni: Sistemska biblioteka sadrzi kod za upucivanje sistemskog poziva OSu i sluzi kao srednji sloj izm aplik i OSa -> Korisnicka aplik ne mora da zna nista o specifikacij sistemskog poziva!

**37. Koji su tipovi sistemskih poziva?**

- Upravljanje procesima - zavrsi, prekini, ucitaj, izvrsi, kreiraj......

- Upravljanje fajlovima - kreiraj i obrisi, otvori i zatvori.......

- Upravljanje uredjajima - zatrazi i otpusti uredjaj, procitaj i upisi....

- Evidencija informacia - preuzmi i postavi sistemsko vreme i datum

- Komunikacija - kreiraj i obrisi komunikacionu vezu, primi i posalji poruku

**38. Koji su tipovi strukture kernela OS?**

* Monolitno jezgro - u jednom velikom jezgru sve funkcionalnosti OSa
* Slojevita struktura - funkcionalnosti su podeljene u hijerarhijski org slojeve. Svaki sloj poziva operacije koje pruza nizi nivo, a pruza operacije koje poziva visi nivo.

Problemi: bidirekciona komunikacija i sporo je jer se prolazi kroz niz sist poziva kroz slojeve

* Mikrokernel - jezgro sadrzi samo osnovne funkcije, a ostali servisi su implementirani u korisnickom rezimu.
  + - Pojednostavljena implementacija i pruzena fleksibilnost, ali smanjene performanse jer servisi iz korisnickog moda moraju preko sistemskih poziva da obavljaju operacije OSa
* Modularna struktura - funkcionalnosti su podeljene u module koji se dinamicki ucitavaju. Kernel uvek sadrzi kljucne funkcije. Sistem podeljen u nezavisne celine koje mogu medjusobno komunicirati.
* Hibridna struktura - kombinacija prethodno navedenih

**39. Kako se vrsi pokretanje OS?**

- Bootstrap loader je program koji se prvi pokrece i koji je odgovoran za ucitavanje OSa u memoriju i njegovo pokretanje

- Ucitavanje je realizovano u dva nivoa: Jednostavni bootstrap loader zapisan u ROM mem koji se prvi pokrece (BIOS, UEFI) i on je odgovoran za ucitavanje kompleksnijeg loadera koju ucitava OS.

**40. Sta omogucuje virtuelizacija?**

- Da se na jednom racunaru istovremeno izvrsava vise razlicitih OS ili instanci jednog OSa.

- Moguce je na taj nacin na istoj platformi izvrsavati vise aplikacija koje rade na razlicitim OSima

- Host operativni sistem moze da podrzi vise virtuelnih masina

- Svaka VM ima karakteristike posebnog OS i u nekim verzijama imaju posebne hardverske platforme

**41. Objasniti ideju VM:**

- Host OS moze da podrzi izvrsavanje vise instanci OS-a preko VM koriscenjem VMM ( Virtual Machine Monitor) koji preko host OS-a obezbedjuje komunikaciju VM -> Host OS -> Hardver

**42. Pojmovi masine:**

- Iz persp aplikacije - svojstva jezika visokog nivoa i sistemski biblioteticki pozivi - definise je API

- Iz persp Procesa - virtuelni mem prostor, registri, skup mas instr.. - definise je ABI

- Iz persp OS - fajl sistem i U/I resursi, stvarna mem.. - ISA predstavlja interfejs izm OS i masine

**43. Koji su tipovi VM?**

- Procesna VM - Java VM, Microsoft .NET framework - kreira se kad i proces i ukida kad se zavrsi, portabilnost - Virtuelna platforma za izvrsavanje jednog procesa

- Sistemska VM - VMware - virtualni softver je host odredjenom broju gostujucih OS.

jedna hardverska platforma moze da podrzi vise izolovanih gostujucih OS

**5. PROCESI.PPT**

**\*Definicija procesa**

-Program u izvrsavanju, Primerak programa koji se izvrsava u racunaru

-Entitet koji se moze dodeliti i izvrsavati u procesoru

-Jedinica aktivnosti koju karakterisu naredbe za izvrsavanje, tekuce stanje i dodeljeni skup sistemskih resursa

**1. Koji su zahtevi OS pri upravljanju procesima?**

- Preplitanje izvrsavanja vise procesa

- Dodela resursa procesima i zastita dodeljenih resurs od nekontrolisanog pristupa drugih procesa

- Komunikacija izmedju procesa

- Sinhronizacija izmedju procesa

**2. Koji su elementi procesa?**

* Programski kod - sta treba da radi - moze biti deljen izmedju vise instanci istog procesa
* Skup podataka nad kojima se izvrsava posao – promenljive npr
* Atributi procesa - dodatne info koje OS skladisti da bi upravljao procesom

**3. Koji su atributi procesa?**

* Id - unique
* Stanje procesa
* Prioritet procesa
* programski brojac - adresa sledece instrukcije u kodu
* pokazivaci na mem blokove - pokaz na kod i podatke vezane za proces
* kontekstni podaci - podaci iz registara procesora vezani za trenutno stanje procesa
* U/I statusne informacije - neobradjeni UI zahtevi, dodeljeni UI uredjaji, lista fajlova koju koristi proces
* Racunovodstvene informacije - korisceno proc vreme, vremenska ogranicenja...

**4. Opisati UBP (Upravljacki Blok Procesa):**

- UBP sadrzi atribute procesa i omogucuje multiprogramiranje:

-- tako sto sadrzi informacije koje omogucavaju da se proces prekine i kasnije nastavi svoje izvrsavanje kao da nije ni doslo do prekida.

-- kada dodje do prekida, snimaju se vrednosti procesorskih registara u UBP

-- kada nastavlja, snimljene vrednosti se ucitavaju iz UBP u procesorske registre

**5. Sta je trag procesa?**

- Niz instrukcija koje proces izvrsava.

**6. Kako se ponasanje procesora moze predstaviti?**

- Kao niz isprepletanih tragova razlicitih procesa.

**7. Sta je rasporedjivac- dispatcher?**

- Program koji prebacuje procesor sa jednog na drugi proces.

- Programski kod rasporedjivaca je u gl mem.

**\* Stanja procesa?**

* Izvrsavanje
* Spreman
* Blokiran - u cekanju
* Novi
* Izlaz

**8. Kako se stvara proces?**

- OS kreira strukture podataka za upravljanje procesom i ucitava ga u radnu memoriju, ceo proces ili deo njega.

- Proces se kreira prilikom pokretanja programa ili ako program roditelj zatrazi od OS kreiranje novog procesa potomka.

**9. Koji tipovi indikatora za zavrsavanje procesa postoje?**

Mora postojati mehanizam da OS zavrsi proces na osnovu indikatora dobijenog od procesa:

* korisnicka akcija - npr izlazak iz aplikacije
* greska nastala u programu
* instrukcija roditeljskog procesa za zavrsetak procesa potomka
* zavrsavanje roditeljskog procesa

OS je zaduzen da oslobodi resurse koje je proces koristio

**\*Prelazi stanja procesa?**

* Null -> Novi (kreiranje procesa)
* Novi -> Spreman (kad je os spreman da prihvati novi proces jer ne moze puno aktivnih odjednom da ima)
* Spreman -> Izvrsavanje (rasporedjivac pokrece 1 proces)
* Izvrsavanje -> Izlaz (proces daje znak kraja ili prekida s radom)
* Izvrsavanje -> Spreman (kada mu istakne max vreme, ili se pojavi proc viseg prioriteta, ili dobrovoljno oslobodi procesor)
* Izvrsavanje -> Blokiran (kad mora da ceka na neku oper koju zahteva npr Ui)
* Blokiran -> Spreman (docekao je dogadjaj zbog kog je bio blokiran)
* Spreman -> Izlaz i Blokiran -> Izlaz (potomak moze biti prekinut od roditelja dok je spreman ili blokiran)

**\*Raspredjivanje s dva reda?**

1. Red spremnih procesa - iz kojeg se bira jedan za izvrsavanje

2. Red blokiranih procesa - kada se pojavi dogadjaj proces ide u red spremnih

- kada se pojavi dogadjaj OS mora da proveri ceo red

**\*Rasporedjivanje s vise redova?**

-Za svaki dogadjaj odvojen red blokiranih procesa

-Kad se dogadjaj pojavi, svi procesi iz tog reda se prebacuju u red spremnih

**10. Sta OS treba da zna o procesu da bi mogao da upravlja njime?**

- lokaciju procesa

- atribute procesa (id, stanje etc..)

**11. Od cega je sacinjena fizicka slika procesa?\*\***

* Upravljacki blok (kolekcija atributa) - id, info o stanju procesora, info za upravljanje procesom!
* Stack - lokalne prom, povratne vrednosti fja...
* Heap - dinamicki alocirane promenljive
* Podaci - globalne promenljive
* Programski kod - instrukcije koje proces izvrsava

**\*Upravljacki blok procesa ima koje grupe informacija?**

* identifikatori procesa
* informacije o stanju procesora
* informacije za upravljanje procesom

**\*Lokacije slike procesa?**

-Proces je podeljen u stranice koje se sve nalaze u sekundarnoj memoriji

-U svakom trenu deo stranica je u glavnoj memoriji

-Stranice ne moraju biti u fizicki susednim lokacijama

**12. Koji tipovi id procesa postoje?**

* Jedinstveni id procesa - OS referencira proces pomocu njega- +ceo broj
* id roditeljskog procesa - koji je stvorio proces
* korisnicki id - id korisnika koji je odgovoran za proces

**13. Sta se nalazi u info o stanju procesora?**

* Predstavljaju sadrzaj registara procesa - kada se proces prekine, potrebno je sacuvati stanje procesorskih registara
* Registri vidljivi korisniku - mogu se referencirati iz masinskog koda programa
* Upravljacki i statusni registri - programski brojac, uslovni kodovi (rez poslednje oper), info o statusu
* Pokazivaci na stack - vrh steka procesa

**14. Sta se nalazi u info za upravljanje procesom?**

* Informacije za rasporedjivanje - stanje, prioritet, dodatne info za rasporedjivanje (npr vreme cekanja na procesor), dogadjaj na koji se ceka...
* Strukturiranje podataka - struktura koja medjusobno povezuje procese, npr red cekanja procesa istog prioriteta
* Medjuprocesna komunikacija – markeri, signali i poruke za komunikaciju izmedju procesa
* Privilegije procesa - kojim delovima mem moze da pristupi i koje instr moze da izvrsi
* Upravljanje memorijom - pokazivaci na tabelu stranica za VM
* Vlasnistvo nad resursima - koje resurse proces kontrolise

**15. Koji su rezimi izvrsenja procesora?**

* Korisnicki rezim - manje privilegija, za korisnicke programe
* Rezim kernela - mogu i privileg. instrukcije, potpuna kontola nad procesorom i svim njegovim instrukcijama, registrima i memorijom, OS se izvrsava u ovom modu

**16. Kako se vrsi promena rezima izvrsavanja?**

- Menja se bit u statusnom registru koji ukazuje na to da li je korisnicki rezim ili rezim kernela.

- Sistemski pozivi prevode sistem iz korisnickog u rezim kernela.

**17. Opisati korake stvaranja novog procesa:**

1. Dodeljuje mu se unique ID

2. Dodeljuje se prostor za sve elemente slike procesa

3. Inicijalizuje se upravljacki blok (programski brojac na prvu instrukciju, ostale vrednosti na default)

4. Azuriraju se strukture sa vezama procesa (npr. uvezuje se u listu spremnih procesa)

5. Kreiraju se ili azuriraju druge strukture pod (npr kreira se fajl sa podacima o aktivnosti procesa)

**18. Sta je komutiranje procesa?**

- Promena trenutno aktivnog procesa koji se izvrsava na procesoru.

- Prekida se proces koji se izvrsava, OS drugom procesu daje stanje izvrsavanje i njemu predaje kontrolu

- Prethodno se sacuva stanje registara

19. Kada moze doci do komutacije procesa?

1. Kada se desi prekid:

* prekid generatora takta - isteklo dozvoljeno vreme - time slice
* U/I prekid - OS prebacuje procese iz blokiran u spreman jer se desio UI dogadjaj na koji su cekali
* Greska memorije - refrencirala se stranica koja nije u glavnoj mem - OS mora da preuzme kontrolu da bi je dobacio sa diska u gl mem

2. Kada dodje do greske u izvrsavanju programa

* tada OS preuzme kontrolu da bi reagovao na gresku

3. Kada se desi sistemski poziv (poziv operacije OSa iz procesa)

- npr proces oce da pristupi nekom fajlu i OS preuzima kontrolu da bi izvrsio UI akciju- ne podrazumeva nuzno prelazak na izvrs drugog procesa, ali se moraju sacuvati stanja registara

**20. Opisi korake promene stanja procesa:**

1. cuva se sadrzaj procesora - programskog brojaca i drugih registara
2. azurira se upravljacki blok procesa - promena stanja na Spreman ili Blokiran
3. prebacivanje upravljackog bloka u odgov red - u red spremnih ili blokiranih procesa
4. izbor narednog procesa za izvrsavanje
5. azurira se upravljacki blok izabranog procesa - promena stanja na Izvrsavanje
6. azurira se strukture podataka za upravljanje mem - potencijalno ucitavanje stranica ili izmene u gl mem
7. postavljanje sadrzaja registara procesora u skladu sa sacuvanim stanjem iz upravljackog bloka izabranog procesa za novo izvrsavanje

**21. Kada je proces nezavisan?**

- Ako na njegovo izvrsavanje ne utice izvrsavanje drugog procesa i on ne deli podatke ni sa jednim drugim procesom.

**22. Kada je proces zavistan?**

- Kada njegovo izvrsavanje zavisi od ishoda zavrsavanja drugog procesa i/ili deli podatke sa njim.

- Medjusobno zavisni procesi moraju da saradjuju putem medjuprocesne komunikacije

**23. Koji su razlozi za medjuprocesnu komunikaciju?**

* Deljenje informacije - razlicitim procesima trebaju isti podaci-resursi
* Ubrzanje izracunavanja nad delovima istih podataka - npr izracunavanje matrice - ubrzanje na multiprocesorskoj arh!
* Modularnost - logicki tokovi izvrsavanja koji pristupaju istim podacima

**24. Koja su dva nacina medjuprocesne komunikacije?**

1. Deljena memorija

* Uspostavlja se prostor deljene memorije.
* Dva procesa komuniciraju upisom i citanjem iz tog prostora.
* Format podataka, iskljucivanje i sinhronizacija su odgovornost procesa.

2. Razmena poruka

- procesi ne moraju eksplicitno da kreiraju region deljene memorije

- komunikacije pozivima operacija SEND i RECEIVE

**25. Koji su nacini razmene poruka?**

- Blokirana (dok primalac ne primi ili dok on ne saceka poruku) ili neblokirana(nastavlja izvrsavanje nakon slanja ili poziva prijema) komunikacija

- Direktno (direktno preko id-a procesa, samo dva proc komuniciraju) ili indirektno (indirektno se komunicira pomocu sanducica, vise procesa mogu medjusobno komunicirati) adresiranje

- Baferovani (u sanduce moze stati 0..N ili neogranicen broj poruka u redu) ili nebaferovani zapis

**\* Poredjenje tipova komunikacije?**

-Deljena memorija je brza jer ne zahteva sistemske pozive za pristup podacima

-Razmena poruka je pogodnija za distribuirane sisteme gde nema deljenje memorije

**26. Koja su dva slucaja vlasnistva nad sanducicem indirektne medjuprocesne komunikacije?**

* Vlasnik je proces - na kraju procesa se sanduce unistava, moraju se drugi procesi obavestiti, ILI
* Vlasnik je OS - pruza mehanizam za kreiranje/brisanje sanduceta kao i primanje i slanje poruka

**28. Sta omogucuje Pipes mehanizam?**

- Komunikaciju izmedju dva procesa po principu proizvodjac-potrosac.

- Proizvodjac upisuje na jedan kraj a potrosac cita sa drugog.

**29. Koji tipovi pipes mehanizma postoje?**

- jednosmerna ili dvosmerna komunikacija

-- dvosmerna: full duplex(u isto vreme mogu u oba smera da teku podaci) ili half duplex (u jednom trenutku samo jedan smer)

- relacioni roditelj-potomak

- da li je lokalna komunikacija ili moze preko mreze

**30. Koja su 3 pristupa izvrsavanja OS-a?\*\***

Os je skup programa koje izvrsava procesor i radi kao i ostali softver

1. Jezgro OS nije proces, ono se izvrsava u privilegovanom rezimu kao zaseban entitet, van bilo kog procesa, a procesi su samo korisnicki procesi.
2. Softver OS se izvrsava unutar korisnickih procesa, kada je potrebno da se izvrsi sistemski poziv, proces se samo prebaci u rezim kernela.
3. OS zasnovan na procesima, funkcije jezgra su organizovane u procese i izvrsavaju se u rezimu kernela.

**6. UPRAVLJANJE MEMORIJOM.PPT**

Da bi se podrzalo multiprogramiranje, OS mora da cuva vise procesa u glavnoj memoriji

Memorija se deli izmedju vise procesa

**\*Šta podrazumeva izvršavanje instrukcije?**

-Procesor preuzima instrukcije iz memorije na osnovu vrednosti brojaca instrukcije

-Instrukcija se dekodira i moze doci do dobavljanja operanada iz mem

-Nakon izvrsenja instrukcije na operandima, moze doci do smestanja rezultata oper nazad u mem

**1. Sta podrazumeva upravljanje memorijom?**

- Prebacivanje blokova podataka izmedju masovne i glavne memorije.

- Potrebno je optimizovati trenutke i kolicinu podataka koji se razmenjuju jer je kapacitet RAM memorije ogranicen i memorijski U/I uredjaji su spori.

**2. Sta racunarski hardver i OS moraju da podrze?**

* Relokaciju
* Deljenje
* Zastitu

**3. Sta je relokacija?**

- Relokacije je ubacivanje i izbacivanje procesa, ili njegovih delova, u/iz glavne memorije iz/u masovne memorije.

- Proces u toku izvrsavanja moze menjati lokaciju u glavnoj mem

- Moze biti vracen na neku drugu lokaciju u gl mem nakon swappinga iz diska

- Nije moguce znat unapred gde ce proces zavrsiti u gl mem!!!!!

**4. Sta se podrazumeva pod deljenjem memorije?**

- Vise procesa pristupaju istim delovima memorije.

- Npr. vise instanci istog programa dele programski kod.

- Sistem za upravljanje memorijom treba da omoguci kontrolisani pristup deljenim podrucjima u memoriji bez ugrozavanja zastite.

**5. Sta se podrazumeva pod zastitom memorije?**

- Proces ne sme da pristupa bez dozvole lokacijama u koje su smesteni drugi procesi.

- Zastita mora biti sprovedena tokom run time (izvrsavanja) jer zbog relokacije nisu unapred poznate lokacije procesa i promenljive su.

- Zato ne moze u toku prevodjenja da se sprovede zastita (compile time)

**6. Objasni kako funkcionise hardverska podrska za relokaciju i zastitu.**

- Svaki proces mora da ima odvojen svoj memorijski prostor, odnosno skup adresa kojima on moze da pristupa.

- Postoje dva procesorska registra – base (prva lok u koju je proces smesten) i limit (poslednja adresa pocevsi od 0 kojoj proces pristupa).

- Ukoliko adresa koju proces pokusava da referencira nije izmedju base i base+limit, desava se adressing error.

**7. Navedi tri nacina definisanja adresa u programu.**

1. adrese poznate u compile time

* u toku kompajliranja se zna gde je program u memoriji
* u samom programu su stvarne adrese
* nije moguce znati lokaciju programa u toku kompajliranja ako treba podrzati multiprogramiranje

2. adrese poznate tokom ucitavanja programa u mem

* stvarne adrese se upisuju u toku ucitavanja programa u mem
* nije upotrebljivo ako program moze biti pomeran tokom izvrsavanja

3. adrese poznate tokom izvrsavanja

* jedina varijanta koja podrzava relokaciju
* ako nije unapred definisana lokacija na koju ce program biti ucitan i ako se u toku izvrsavanja ta lokacija menja
* tek u toku izvrsvanja svake instrukcije se moze odrediti stvarna adresa kojoj se pristupa

**8. Opisati razliku izmedju logickih i fizickih adresa.**

-Adresa upisana u programu je logicka (relativna)

-Stvarna adresa kojoj se pristupa je fizicka (apsolutna)

-Logicke adrese su definisane relativno u odnosu na blok u koji je proces trenutno smesten

-U toku izvrsavanja ove relativne adrese se prevode u fizicke

**9. Navedi 3 nacina upravljanja memorijom:**

1. Alociranje susednih mem lokacija - deljenje memorije na particije i partnerski sistem

2. Stranicenje

3. Segmentacija

**\*Objasni deljenje memorije na particije**

Ovo je nacin upravljanja memorijom koriscen pre pojave VM, danas slabo

Kompletan proces se smesta u uzastopne memorijske lokacije

Postoji fiksno deljenje memorije na particije ISTE velicine u koje se smestaju celi procesi

Posto su svi jednaki, zauzima se prva raspoloziva particija

**10. Koji su problemi fiksnog deljenja na particije sa jednakim velicinama particija?**

- Program moze biti prevelik za particiju.

- Neefikasno koriscenje memorije jer se zauzima cela particija (INTERNA FRAGMENTACIJA!).

- Ogranicen broj aktivnih procesa (jednak broju predefinisanih particija)

- Veliki broj malih procesa nece efikasno koristiti memoriju

**\* Fiksno deljenje na particije NEJEDNAKIH velicina?**

-Manji programi se smestaju u manje particije i time se smanjuje interna fragmentacija

ALGORITMI:

1. Procesu se dodeljuje najmanja particija u koju on moze da stane, svaka particija ima red cekanja - nije efikasno jer proces ceka cak i kad ima slobodnih vecih particija!

2. Efikasniji nacin je da ima jedan red cekanja za sve particije i da se bira najmanja raspoloziva particija

**\*Dinamicko deljenje na particije?**

Promenljiv broj particija promenljivih velicina

Procesu se daje onoliko memorije koliko zahteva

Lose strane: eksterna fragmentacija

**11. Koje probleme fiksnog deljenja na particije prevazilazi dinamicko deljenje na particije?**

- Promenljiv je broj particija - ukida se ogranicenje broja procesa

- Particije su promenljive velicine - procesu se dodeljuje tacno onoliko koliko mu treba

**12. Objasniti pojavu eksterne fragmentacije:**

- Vremenom se javlja mnogo malih rupa u memoriji. Postoji dovoljno slobodne memorije ali niije u sukcesivnim lokacijama.

- Moze se resiti sazimanjem.Pomeraju se procesi tako da budu susedni, slobodna mem na kraj, DUGO TRAJE I TROSI VREME!

**13. Objasniti sazimanje:**

- Povremeno se procesi pomeraju tako da se nalaze u sukcesivnim lokacijama na jednom kraju a sva slobodna lokacija na drugom kraju.

**14. Koja su 3 algoritma za smestanje kod dinamickog deljenja na particije:**

1. najbolji odgovarajuci - trazi se blok koji najbolje odgovara zahtevu - sporo, mora cesto sazimanje jer ostane puno malih blokova
2. prvi odgovarajuci - od pocetka mem trazi se prvi blok koji odgovara zahtevu - rezultuje u puno malih fragmenta na pocetku koji se ponovo moraju pretrazivati
3. sledeci odgovarajuci - od proslog smestanja se trazi prvi odgovarajuci blok koji odgovara zahtevu - cesto se smestaju na kraj memorije gde je najveci slobodan blok, on ce biti izdeljen na male fragmente i javlja se potreba za cestim sazimanjem

**\* Problemi deljenja na particije**

-Interna i eksterna fragmentacija

**\* Partnerski sistem?**

Celokupna memorija se tretira kao blok velicine 2^U

Ako se zahteva blok velicine s, 2U-1 < s <= 2U, tada s ceo blok alocira

U suprotnom se blok deli na jednaka dva bloka

Postupak se rekurzivno ponvalja dok god se ne stvori najmanji blok koji je veci ili jednak s

**15. Navesti prednosti i mane partnerskog sistema:**

- Prednosti: zauzima najbolji odgovarajuci blok zahteva i sazimanje je jednostavno izmdju dva susedna bloka

- Mane: interna fragmentacija

**16. Objasniti ideju iza stranicenja:**

- Memorija se deli na male jednake delove koji se nazivaju okviri

- Procesi se dele na stranice koje su jednake velicine kao i okviri

- Stranice se mogu dodeljivati okvirima koji ne moraju biti susedni

**17. Navesti prednosti stranicenja:**

- Nema eksterne fragmentacije kao ni potrebe za sazimanjem

- Interna fragmentacija samo u poslednjoj stranici procesa

**18. Koja je uloga tabele stranica?**

- Mapira stranicu procesa na okvir memorije u kojem je ona smestena.

- Logicka adresa se na osnovu broja stranice i relativnog pomeraja u okviru stranice pretvara u apsolutnu

**\*RELATIVNA VS LOGICKA ADRESA – ne kontam lol**

Relativna - pomeraj u odnosu na pocetak

Logicka - broj stranice i pomeraj unutar nje

**\*Segmentacija**

-Kod segmentacije, programer ne vidi program kao niz adresiranja linearnih lokacija nego kao skup segmenata razlicitih duzina

-Segmentacija je način upravljanja memorijom koji se bazira na posmatranju memorije iz ugla programera

-Logicki adresni prostor se deli u segmnte promenljive duzine

-Slicno dinamickom deljenju na particije

-Nema interne fragmentacije

-ADRESA: BROJ SEGMENTA + POMERAJ UNUTAR SEGMENTA

-Postoji tabela segmenata za svaki proces

**19. Navesti karakteristike segmentacije:**

- Nacin upravljanja memorijom koji se bazira na posmatranju memorije iz ugla programera.

- Logicki adresni prostor se deli u segmente promenljive velicine.

- Nema interne fragmentacije.

**7-8. VIRTUELNA MEMORIJA.PPT**

**1. Koje dve karakteristike moraju biti prisutne da se upravlja memorijom i da ne bi bilo neophodno da svi delovi procesa budu ucitani u glavnu memoriju?**

1. Proces moze biti izdeljen na manje celine koje ne moraju zauzimati sukcesivne memorijske lokacije.
2. Memorijske reference su logicke adrese koje se dinamicki prevode u fizicke u run time (u toku izvrsavanja)

\*proces moze biti ubacivan i izbacivan iz gl mem u toku izvrsavanja i da bude smesten u razlicite lokacije u memoriji u toku vremena

Ako su obe karakteristike prisutne tada ne moraju svi delovi procesa da budu istovremeno u gl mem u toku izvrsavanja jer se ne koriste svi delovi jednako i jednako cesto

**\*Razlika izmedju stvarne i virtuelne memorije?**

Stvarna je glavna memorija a virtuelna je mehanizam upravljanja memorijom gde je deo diska namenjen za smestanje delova onih procesa koji se trenutno izvrsavaju

Omogucuje efikasno multiprogramiranje i smanjuje ogranicenja glavne memorije

**2. Koje su prednosti nove strategije upravljanja memorijom?**

- Vise procesa se moze naci u glavnoj memoriji jer su ubaceni samo delovi a ne celi procesi

- Sa vecim brojem procesa veca je sansa da ce u svakom trenu bar neki biti SPREMAN

- Proces sada moze biti i veci od glavne memorije jer ne mora ceo biti u mem.

- Korisnik vidi virtuelni adresni prostor sa kontinualnim logickim adresama koje idu od nule

- Strategija je transparentna korisniku

- Heap i Stack memorija dodeljena procesu se dinamicki menja

- Procesi mogu deliti deo memorije tako sto se ista stranica mapira u virtuelni adresni prostor vise procesa (sistemske lib koje deli vise procesa npr)

--Putem deljenje mem mogu komunicirati procesi, dele isti kod instance istog procesa

**3. Sta cini rezidentni skup procesa?**

- Rezidentni skup procesa cine delovi procesa koji su smesteni u glavnoj memoriji u odredjenom trenutku. Mora postojati indikator za svaku stranicu koji ukazuje na to da li je ona smestena u glavnu memoriju ili ne - bit.

**4. Kako se izvrsava proces?**

- Ukoliko se referencirani deo procesa nalazi unutar glavne memorije on se izvrsava,

a ukoliko nije ucitan to je greska stranice -page fault koji odredjuje hardver pomocu invalid bit-a stranice

Signalizira se OS-u da dobavi referenciranu stranicu u glavnu memoriju i onda nastavlja izvrsavanje.

Koraci:

1. Proces generise prekid kada se referencira nevalidna stranica
2. Šalje se zahtev OSu da dobavi tu stranicu a proces se postavlja u stanje blokiran
3. OS šalje zahtev disku da učita tu potrebnu stranicu - čekanje na redu na uređaj
4. Disk kada prebaci tu stranicu u gl mem će postaviti prekid
5. Azurira se info o stranici (bit upotrebe npr) i postavlja se proces u stanje spreman
6. Čeka se na procesor da opet pokrene taj proces i ponovo se izvrsava instrukcija koja je prethodno izazavala page fault

**6. Kakva je efikasnost virtuelne memorije?**

- Efikasnost VM zavisi od verovatnoce pojave i ucestalosti page fault-a.

- Ako zelimo dobre performanse, trebala bi nam ucestalost od 1/400.000 referenciranja po greski stranice.

- Dobavljanje stranice sa diska je vremenski zahtevno!

**7. Sta je pojava brbljanja (thrashing)?**

- Kada sistem provodi najveci deo vremena razmenjujuci delove izmedju stvarne i VM umesto da izvrsava instrukcije.

**8. Kako se izbegava brbljanje?**

- Tako sto OS izbacuje stranicu iz glavne memorije za koju smatra da je najmanja verovatnoca da ce biti uskoro referencirana.

To se predvidja na osnovu nedavne istorije.

**9. Sta je princip lokalnosti?**

- Pojava da program i reference podataka unutar procesa teze da se grupisu. U kratkom intervalu procesor uglavnom radi sa malim i ogranicenim skupom podataka. Samo nekoliko delova procesa je potrebno u kratko vremenskom periodu

**10. Navedi 4 razloga zbog kojih postoji princip lokalnosti:**

1. Izvrsavanje programa je skoro uvek sekvencijalno - naredna adresa je ona koja sledi trenutnoj, izuzeci su grananja i pozivi fja!
2. U kratkom periodu program radi na odredjenoj fji - pristupa lokalnim promenljivima ili malim podskupom globalnih
3. Postoje iterativni blokovi koda sa malim brojem instrukcija koji se izvrsavaju vise puta - PETLJE - obrada je ogranicena na mali deo susednih adresa
4. Podaci se cesto cuvaju u strukturi koja cuva podatke u sukc lokacijama (npr niz)

**11. Zbog cega je princip lokalnosti dobar za VM?**

- Zbog prisustva principa lokalnosti mozemo lako odrediti koje stranice, odnosno delovi procesa, ce biti potrebni u skorijem toku izvrsavanja.

- Tako se moze izbeci brbljanje

**12. Koja su 2 tipa podrske za VM?**

1. Hardverska - podrzano adresiranje stranica/segmenata

2. Softverska - OS mora da pomera stranice/segmente iz glavne u sek memoriju i obratno

**13. Sta se nalazi u tabeli stranica?**

- Svaki proces ima svoju tabelu stranica koja vodi evidenciju o tome koja se stranica procesa nalazi u kom okviru memorije.

**\*Stranicenje je?**

-Proces je podeljen na stranice koje su sve smestene na disku

-Deo tih stranica je u glavnoj memoriji

**14. Sta sadrzi svaka stavka tabele stranica?**

-Svaki proces ima svoju tabelu stranica koja sluzi za evidenciju u kojim okvirima memorije je koja stranica. Svaka stavka tabele stranica procesa sadrzi:

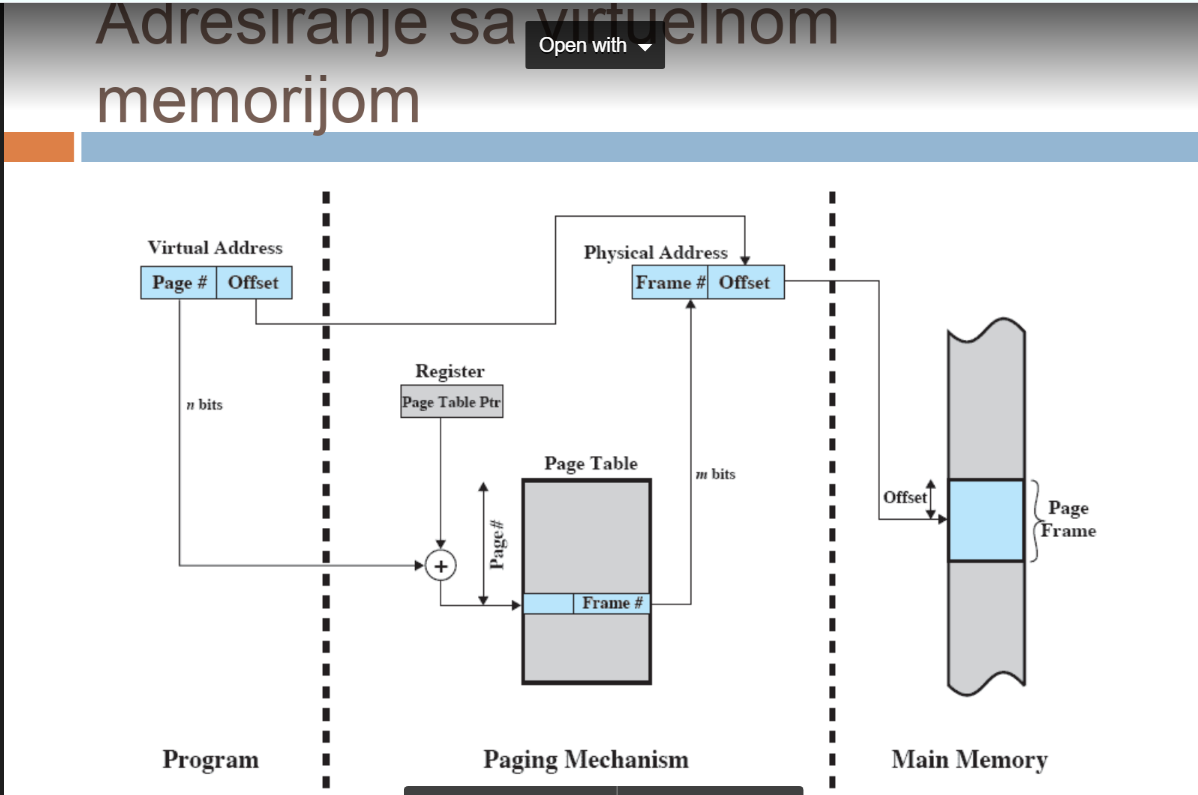
* Broj okvira u kojoj je stranica smestena -- ako je stranica validna (u gl. Mem je)
* P - da li je stranica prisutna u mem - valid invalid bit
* M - da li je stranica menjana od kako je ucitana (ako jeste, treba upisati izmenu na disk)

**15. Gde se skladisti pocetna adresa tabele stranica?**

- Skladisti se u upravljackom bloku procesa i upisuje se u procesorski registar prilikom komutacije.

**!!!!!!! 16. Kako se vrsi konverzija log adrese stranice u fizicku adresu?**

- Broj stranice iz virtualne adrese se koristi za indeksiranje tabele stranica i dobavljanje broja okvira memorije gde je smestena str. Zatim se broj okvira memorije kombinuje sa brojem pomeraja i dobija se fizicka adresa.



**17. Sta je TLB?**

- Translation Lookaside Buffer

- Ako se koristi VM pri svakom referenciranju podataka treba dva pristupa memoriji:

1. Jedan da se preuzme stavka iz tabele stranica

2. Jedan da se procita vrednost sa izracunate fizicke adrese

- Da bi se prevazisao ovaj problem koristi se TLB bafer

- TLB bafer sadrzi stavke tabele stranica koje su najskorije koriscene, u vidu brze kes memorije

- U savremenim procesorima je organizovan u vise hijerarhijskih nivoa, svaki nivo je veci ali sporiji

- obicno postoje odvojeni TLB za adrese instrukcija i podataka

**18. Kako se odvija referenciranje stranice uz TLB?**

Procesor kada pokusava da referencira stranicu, najpre proveri TLB i ukoliko se nalazi u TLB formira se fizicka adresa,

U suprotnom se gleda tabela stranica. Ako se stranica ne nalazi u mem, generise se page fault i OS je duzan da je ubaci sa diska u gl mem.

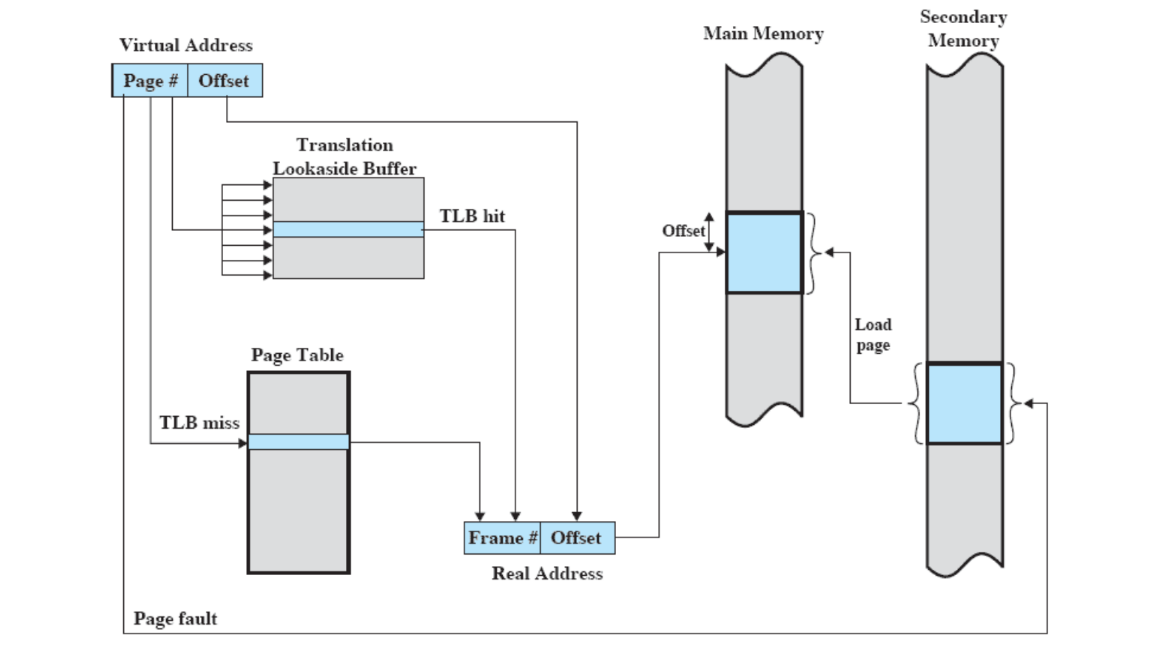
TLB se azurira tako da ukljuci info o ovoj upravo referenciranoj stavki

TLB sadrzi samo neke stavki tabele stranice.

Ne moze se stavka jednostavno pronaci indeksiranjem na osnovu broja stranice

Svaka stavka TLBa mora da sadrzi broj stranice i kompletnu stavku tabele stranice

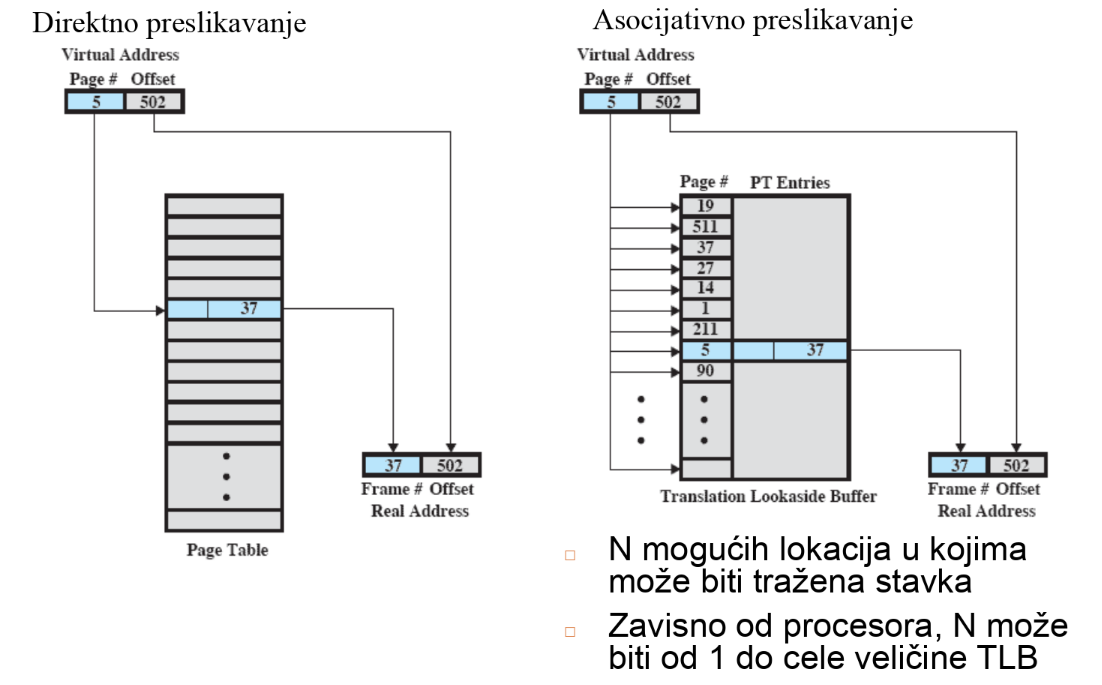
Procesor moze istovremeno da ispita vise stavki TLBa da pronadje odgovarajucu stranicu - ASOCIJATIVNO PRESLIKAVANJE



**19. Koja je razlika izmedju direktnog (tabela stranica) i asocijativnog preslikavanja stranice u TLB?**

- Direktno se u tabeli stranica indeksiranjem dobavlja stranica na osnovu broja stranice

- Asocijativno se iterira kroz TLB i proveravaju se vise stavki da bi se pronasla stranica, mana toga je sto je kompleksnost O(n)



**20. Koja su dva moguca scenarija stavki TLB?**

1. Stavke TLB se mogu odnositi na vise razlicitih procesa - tada je bitno da stavka sadrzi info o tome kojem procesu stranica pripada
2. Stavke TLB se mogu odnositi samo na jedan proces - tada je obavezno isprazniti TLB prilikom komutacije procesa

**21. Koji je cilj TLB?**

- da se ostvari sto veci broj pogodaka (pronadjenih stavki u TLBu), odnosno sto manja potreba za jos jednim pristupom memoriji radi pristupa tabeli stranica

- sto je manja potreba pristupati tabeli stranica, ubrzava se pristup referenciranoj adresi

**22. Sta se ubraja u vreme pristupa memoriji?**

- Ukupno vreme potrebno za upis/citanje iz memorije.

U to spada i pristup tabeli stranica ukoliko se ne pronadje u TLB.

Prosecno efektivno vreme pristupa se odnosi na to koliki je procenat pogotka u TLB.

**23. Sta ako je tabela stranica prevelika da bi se cuvala u glavnoj mem?**

- Jedno resenje je da se ona smesti u VM i da se ona podeli u stranice

Tada pri izvrsavanju procesa deo tabele stranica mora da bude u gl mem jer je potrebno pristupati stavci koja se odnosi na str koja se trenutno referencira

- Drugo resenje je Hijerarhijska Tabela Stranica, za prevelike tabele

**24. Koje su karakteristike hijerarhijske tabele stranica?**

- Varijanta za prevelike tabele stranica

- Tabela stranica je realizovana u vise nivoa

- Postoji mala osnovna tabela stranica - outer page table

- Stavke te tabele su pokazivaci koji pokazuju na stranicu u drugoj tabeli stranica na drugom nivou

- U toj drugoj tabeli su podaci o stranicama procesa (ako je tabela u dva nivoa)

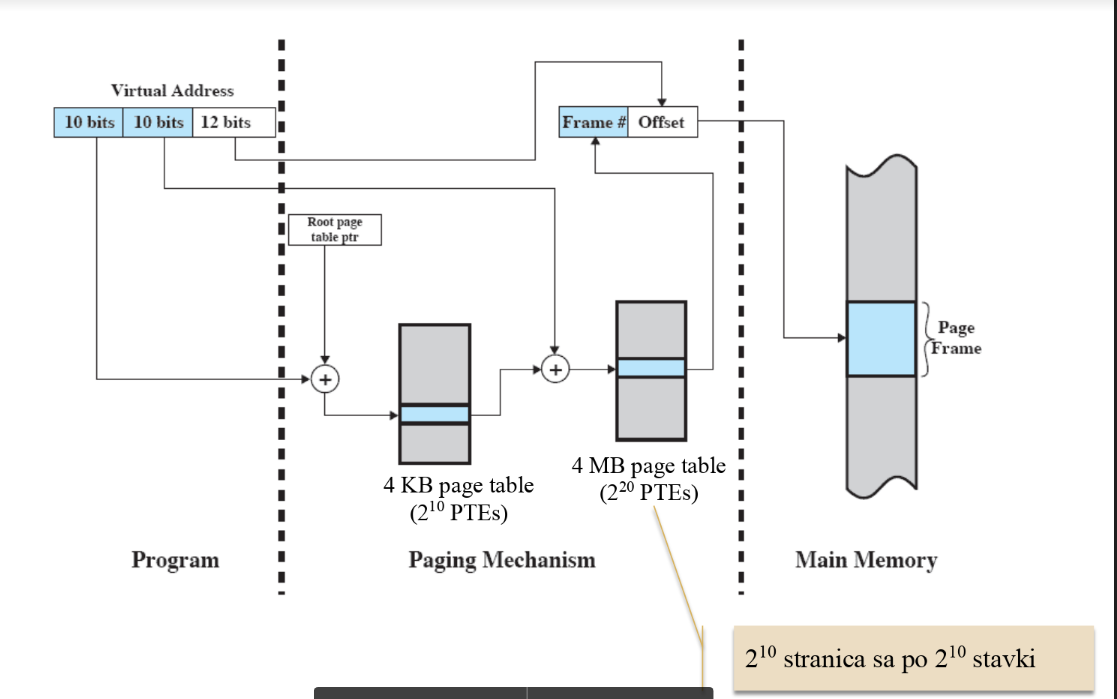
- Moze se realizovati u N nivoa - najcesce 3

- svaki nivo ukazuje na referenciranu stranicu na sledecem nivou

- tabele stranica na svim nivoima se cuvaju u RAM mem (radnoj mem)

- OS ne kreira sve delove tabele stranica na nizim nivoima

- npr ne kreiraju se stavke tabele stranica za stranice iz heap memorije koje se ne koriste i time stedi RAM u odnosu na klasicnu tabelu u jednom nivou



**25. Kako se odvija ref stranice koriscenjem hijerarhijske tabele stranica?**

- Referencira se osnovni nivo tabele stranica koji ukazuje na dalje nivoe koji su veci i u sebi skladiste info o okvirima stranica koje se ref.

- Npr osnovni nivo ima 2na10 stavki (pokazivaca) a svaki od tih 2na10 pokazuje na 2na10 stavki iz narednog nivoa - naredni nivo ima 2na20 stavki!

**26. Koja je mana klasicne tabele stranica u kontekstu velicine memorije?**

- Velicina klasicne tabele stranica proporcionalna je velicini virtuelnog adresnog prostora.

**27. Koja je alternativa tabeli stranica koja prevazilazi manu velicine mem?**

- Invertovana tabela stranica koja ima jednu stavku za svaki okvir.

- Svaka stavka invertovane tabele govori koja je stranica kog procesa trenutno smestena u koji okvir.

- Kod linearne invertovane tabele stranica, mora se linearno prolaziti kroz okvire i traziti da li je u njima trazena stranica - SPORO

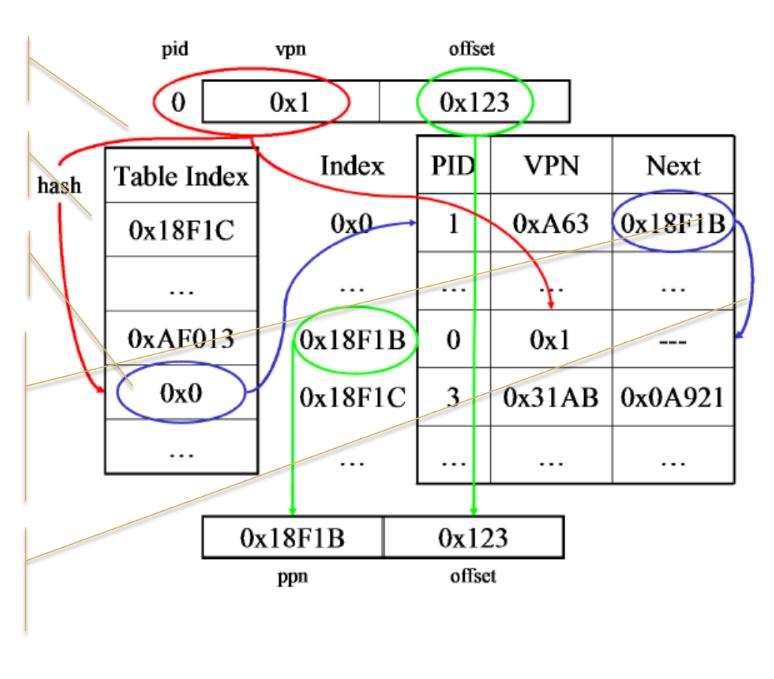
**28. Koja je alternativa invertovanoj tabeli stranica koja prevazilazi manu linearnog trazenja?**

- Hesirana invertovana tabela stranice.

- Indeks stavke je broj stranice konvertovan hes funkcijom

- Sadrzaj stavke je okvir u kojem je stranica smestena

- Vise stranica moze hes funkcijom da se preslika u istu stavku. Zato se ulancavaju stranice sa istim hesom, te se prolazi se kroz lanac i nalazi se stranica koja je referencirana!!



**29. Kako je implementirana zastita pristupa memoriji u VM?**

- Da li je lokacija samo za citanje, citanje i upis, ili samo izvrsavanje znamo pomocu dodatna 3 bita u stavci tabele stranica

- Zabranjen je pristup stranicama koje ne pripadaju procesu - implicitno je ugradjeno u logiku rada virtuelne memorije

- Proces pristupa samo stavkama u VM, tj referencira samo virtuelne adrese, a hardver ih mapira na fizicke adrese.

- Proces nema mehanizam da pristupi drugim mem lok van njehovog adresnog prostora, on vidi samo svoj virtuelni prostor

**\*Deljenje stranice?**

Stranicenje omogucava deljenje zajednickog koda koji se smesta samo jednom u gl mem

Vise razl procesa u svojoj tabeli stranica referenciraju isti okvir sa deljenim kodom

Primer - tri instance istog editora kojima se razlikuje samo sekcija za podatke

**!!!!!! \*Velicina jedne stranice?**

-Sto je velicina stranice manja, bice manje interne fragm, vise stranica ce biti u glavnoj mem

-Vise stranica u radnoj mem => te stranice ce biti bliske skorasnjim referencama

-Ali ce biti vise stranica po procesu => vece tabele stranica => tabele zauzimaju vise radne mem

-Ili je deo tabele u VM pa su ceste greske stranica za pristup tabeli

-Sekundarna memorija efikasnije prenosi vece blokove - prednost vecim stranicama

-Ako su stranice vece, pojedinacne stranice koje su trenutno u radnoj mem ce sadrzati lokacije sve dalje od skorasnjih reference, dakle raste broj gresaka stranice

-Ako se velicina stranice priblizava celoj velicini procesa, broj gresaka stranice opada

-Ako se ceo proces smesti u jednu stranicu - nema gresaka stranice!

-Ako ima vise stranica jednog procesa u mem, veca je ucestalost gresaka stranica za druge procese!

**30. Objasniti parametre koji uticu na ucestalost gresaka stranice:**

- velicina stranice - broj gresaka ce biti manji sto je manja stranica

- broj dodeljenih okvira - sto se vise stranica procesa ubaci u glavnu mem, to je manja verovatnoca greske, ali sto se vise mem zauzme stranicama jednog procesa to je veca verovatnoca greske stranica za druge procese.

**\*Segmentacija?**

Programer vidi program kao skup segmenata sa odredjenm funkcionalnoscu

Segmentacija je nacin upravljanja memorijom koji podrzava ovakav korisnikov pogled na memoriju

-Omogucuje programeru da vidi memoriju kao da se sastoji od vise adresnih prostora (segmenata)

-Segmenti mogu biti nejednake, dinamicke velicine!

-Proces je podeljen na segmente koji su svi na disku, a deo u gl mem

-Svaki segment ima stavku tabele stranica

**31. Sta sadrzi stavka tabele stranica za svaki segment?**

- Adresa pocetka segmenta - BASE

- Duzina segmenta - LIMIT

- P - da li je segment u gl mem

- M - da li je segment menjan od kako je ucitan u gl mem - ako da, treba pre izbacivanja izmenu sacuvati na disk

**32. Koje su prednosti segmentacije a koje stranicenja?**

- Segmentacija je vidljiva za programera, podrzava modularnost, deljenje i zastitu i eliminise unutrasnju fragmentaciju

- Stranicenje je transparentno za programera, omogucava lako upravljanje mem jer su svi delovi iste velicine i uklanja spoljasnju fragmentaciju

**33. Objasni karakteristike kombinovanog pristupa stranicenja i segmentacije:**

- Adresni prostor se deli u odredjeni broj segmenata

- Svaki segment podeljen u stranice

- Adresa sadrzi broj segmenata i broj stranice unutar segmenta

- Svaki proces zahteva tabelu segmenta a svaki segment tabelu stranica

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

**34. Koji su preduslovi OS podrske za upr mem?**

- Da harver koristi tehnike VM i podrzava segmentaciju i/ili stranicenje

Softver je odgovoran za algoritme koji se primenjuju pri upravljanju memorijom - OS donosi odluku kada kako i koju stranicu zamenjuje

**35. Koje su 3 politike za koje OS mora da vodi racuna?**

* Politika donosenja - kada ucitati stranicu u glavnu mem
* Politika smestanja - gde u glavnoj mem smestiti stranicu
* Politika zamene - koju stranicu izbaciti iz glavne mem da bi se mogla smestiti nova

**36. Koja su dva pristupa politike donosenja?**

1. Stranicenje po zahtevu

* stranica se donosi u glavnu mem samo kada se napravi referenca na lokaciju u toj stranici
* rezultuje u puno page fault-ova na pocetku rada programa ali taj broj opada vremenom

2. Predstranicenje

* stranice se unapred donose u glavnu mem cak i ako nisu refrencirane
* efikasnije je odjednom ucitati vise stranica iz sekund. mem ako su u sukcesivnim mem lokacijama
* nepotrebno moze doneti stranicu koja se nece koristiti

**37. Koja je politika zamene stranica?**

- Kada su svi okviri u glavnoj mem zauzeti potrebno je da se donese nova stranica i odredi koju ce da smeni

- Cilj je da se minimizuje broj page fault tako sto ce se izbaciti stranice za koje je najmanja verovatnoca da ce biti opet referencirane.

- Vecina politika pokusava da predvidi buduce ponasanje na osnovu ponasanja u proslosti

- Princip lokalnosti moze pomoci da se utvrdi koja stranica je najmanje verovatna da ce uskoro biti referencirana

**\*Ako referencirana stranica nije u gl mem?**

Ako nema slobodnih okvira, neophodno je dva puta pristupiti disku

Jednom da se izbacena stranica sacuva, drugi put da se nova referencirana ucita

Ako se uvede u tabelu stranica još jedan bit koji označava da li je stranica izmenjena,

pri zameni se na disk upisuju samo izmenjene stranice ?

**38. Zasto se uvodi zakljucavanje okvira?**

- Okvir koji je zakljucan sprecava zamenu stranice u njemu.

- U zakljucanim okvirima se cuvaju kernel OS, glavne upr strukture, U/I bufferi.

Takodje se tek donesene stranice mogu zakljucati kako ne bi doslo do njihove zamene pre prvog koriscenja.

Potreba za indikatorom na to da li je okvir zakljucan ili ne.

Zakljucavanjem se smanjuje broj raspolozivih okvira za smestanje stranica!

**39. Koje su politike za izbor stranice za zamenu?**

1. Optimalna
2. LRU - last recently used- najmanje skoro koriscenja
3. FIFO
4. Casovnik

**40. Kako se porede politike za izbor stranice za zamenu?**

- Po broju page faulta u nizu referenci.

**41. Koje su karakteristike optimalne politike?**

- Bira se ona stranica za koju je vreme do sledece reference najduze.

- Nije moguce implementirati je jer zahteva od OS da savrseno poznaje dalji splet dogadjaja, i samo predstavlja standard za procenu daljih algoritama.

**42. Koje su karakteristike najmanje skoro koriscene politike - LRU?**

- Bazira se na evidenciji proslih dogadjaja i referenciranja.

- Zamenjuje stranicu koja je najduze bila referencirana pod pretpostavkom da ni uskoro nece biti.

- Po principu lokalnosti trebala bi da bude zamenjena stranica za koju je najmanje vrv da ce biti ref u bliskoj buducnosti.

- Radi skoro jednako kao optimalna politika.

- Teska je za implementaciju

- Dve varijante impl: BROJAC I STEK

**43. Objasniti dve varijante implementacije LRU politike:**

1. Brojac - procesor vodi racuna o broju otkucaja svake stranice i na osnovu toga upisuje trenutni takt prilikom svakog ref stranice. Zamenjuje se stranica sa najmanjim trenutkom vremena

2. Stek - stranice se prilikom refrenciranja smestaju na vrh steka, ukoliko se vec nalaze unutar steka, brisu se sa tog mesta i smestaju na vrh. Smenjuju se stranice sa dna steka.

**44. Koje su mane LRU politike?**

- Ne moze se implementirati u realnim sistemima. Bilo koja impl usporava sistem jer se prilikom svakog referenciranja mora azurirati takt ili stek -> veliko usporenje

**45. Objasniti karakteristike FIFO politike zamene:**

- Tretira okvire kao kruzni bafer, pokazivac na sledeci okvir koji treba izbaciti

- Najjednostavnija za implementaciju

- Bice zamenjena stranica koja je najduze u memoriji sto je cesto pogresno

- Losije performanse od LRU

- Stranice koje se cesto referenciraju ce svaki cas biti izbacivane i ponovo ubacivane!!

- Za razliku od LRU ne prepoznaje stranice koje su cesce referencirane od ostalih, vise gresaka od LRU

**46. Koja je alternativa LRU? – pogledaj opet slajd za 8bit**

- Politike koje pokusavaju da dobiju rez LRU a da izbegnu usporenje koje LRU donosi

- Ideja je da OS evidentira 8bitnu rec upotrebe za svaku stranicu tako da u regularnim intervalima OS smesta bit upotrebe na najvisu poziciju reci, a ostale bitove pomera u desno.

- Bit na najmanjoj poziciji ispada iz reci

- Na taj nacin dobijamo info o tome koja stranica je odavno bila referencirana

- Stranica sa najmanjom dekardnom vrednoscu je najmanje referencirana i bice zamenjena

**47. Koje su karakteristike politike casovnika?**

- Slicno kao alternativni LRU samo sto se evidentira samo jedan bit upotrebe za svaku stranicu

- Kada se stranica donese u mem iili biva referencirana, bit = 1

- Kada je potrebno pronaci stranicu za zamenu, kruznim kretanjem kroz okvire se svaki okvir sa bitom upotrebe 1 postavlja na 0 sve dok se ne naidje na okvir sa bitom 0, njega menjamo.

- Naredno pretrazivanje ce krenuti od okvira koji je poslednji zamenjen.

- Posto ide u krug kroz okvire, u najgorem slucaju ce se tek u drugom prolasku pronaci okvir kojem je u prvom prolasku stavljen bit = 0

**48. Koje se unapredjenje uvodi na politiku casovnika?**

Povecava se broj bitova koje algoritam koristi - svaka stranica ima bit promene i algoritam moze i njega uzeti u obzir. Prednost u zameni daje ona stranica koja nije menjana jer se rezultati izmene te stranice ne moraju smestati nazad u sek memoriju pre zamene.

**\*Poredjenje algoritama**

FIFO < CLOCK < LRU < OPTIMAL

Zbog principa lokalnosti, nakon odredjene granice, dalje povecavanje broja okvira dodeljenih procesu ima manji uticaj

**49. Zasto se uvodi baferovanje stranica?**

- Deo glavne memorije se odvoji da radi kao keš stranica, zamenjena str se dodaje u keš

- Ako nije menjana dodaje se u listu slobodnih str, ako jeste u listu menjanih str

- Zamenjene stranice se nalaze u baferu i ako se nadje potreba za njima mogu se brzo ucitati

- Postojanje liste izmenjenih stranica implicira to da se izmenjene stranice mogu grupno upisati nazad u sekundarnu memoriju sto je dosta brze!

**\*Upravljanje rezidentnim skupom**

-Sto je manje memorije dodeljeno jednom procesu, to vise procesa moze istovr da bude u mem

-Sto je manji broj okvira dodeljen procesu, bice vise page faultova

-Zbog principa lokalnosti, povecavanje broja dodelj okvira preko neke granice nece imati zapazen efekat na broj page faultova

**50. Koji su parametri odredjivanja minimalne velicine rezidentnog skupa?**

* Procesu dodeliti bar onoliko okvira koliko svaka pojedinacna instrukcija moze da ref
* Procesu dodeliti onoliko okvira da ne bude previse gresaka stranica

**51. Koje su politike velicine rezidentnog skupa?**

* Fiksno dodeljivanje - proces dobija fiksan broj okvira za smestanje stranica, moze biti jednak svim procesima ili proporcionalno broju stranica procesa
* Promenljivo dodeljivanje - procesi sa velikim brojem page fault-ova dobijaju vise okvira dok se procesima sa manjim brojem smanjuje – tj. broj dodelj okvira se menja u toku zivotnog veka procesa

**52. Koji su opsezi zamene?**

* Lokalni - biraju se stranice samo onog procesa koji je izazvao page fault
* Globalni - bilo koja nezakljucana stranica svih procesa su kandidati za zamenu - cesce

**\*Fiksno dodeljivanje, lokalni opseg**

-Unapred se odredi broj okvira koji su dodeljeni procesu

-Ako se dodeli malo, bice puno gresaka stranice

-Ako se dodeli previse, bice malo procesa u gl memoriji - cesto ni jedan nece biti spreman

-dakle -> nefleksibilno je

**\*Promenljivo dodeljivanje, globalni opseg**

-OS odrzava listu slobodnih okvira

-Procesu se dodeljuje slobodni okvir kada se desi greska stranice

-Ako nema slobodnih, zamena neke od nezakljucanih okvira u mem

-Procesu kojem oduzmemo stranicu ima manji rezidentni skup i moze mu smanjiti performanse

**\*Promenljivo dodeljivanje, lokalni opseg**

Novom procesu se dodeli odredjeni broj okvira

Pojavi se greska -> zamena neke stranice tog procesa

Analiza broja okvira dodeljenih procesu s vremena na vreme i po potrebi smanjenje ili povecanje broja okvira radi boljih performansi

**53. Sta je radni skup W?**

W(t, A) - U trenutku vremena t skup stranica koje su bile ref u poslednjih A jedinica vremena cine radni skup

- jedno referenciranje predstavlja jednu jedinicu vremena A

*-* vreme se meri referenciranjem stranica

- radni skup procesa zbog principa lokalnosti ima relativno stabilne periode kao i periode drasticne promene zbog prelaska na novu lokalnost

**54. Opisati strategiju radnog skupa:**

- Nadgledati radni skup svakog procesa

- Periodicno uklanjati stranice koje nisu u radnom skupu

Problem: merenje radnog skupa je zahtevno sa stanovista performansi

**55. Sta je PFF?**

- Page Fault Frequency

- Umesto nadgledanja radnog skupa nadgledati ucestalost gresaka stranica

-Ako je PFF ucestalost manja od nekog praga - smanjiti rezidentni skup procesu

-Ako je veca - povecati rez skup procesu

**56. Opisati algoritam ucestalosti greske stranice PFF:**

* Umesto nadgledanja radnog skupa nadgleda se ucestalost greske
* Svaka stranica ima bit upotrebe
* Brojac referenci broji koliko se referenci desilo od prosle greske
* Pri svakoj gresci vrednost brojaca se poredi sa nekim pragom F
* ukoliko je manja od F rez skup se prosiruje novom stranicom
* ukoliko je veca od F rez skup se smanjuje za stranice sa bitom upotrebe 0
* pri svakoj gresci resetuju se bitovi upotrebe na 0 svakoj stranici u rez skupu

**57. Koja je mana algoritma ucestalosti greske stranice?**

- Ne radi dobro u toku prelaznih perioda kada se prelazi na novu lokalnost.

- Pri prelasku u novu lokalnost veliki broj gresaka ce povecavati rez skup ne izbacujuci odmah stranice stare lokalnosti

- Kasnije ce se opet smanjivati jer se stranice stare lokalnosti nece referencirati

- Da bi se str izbacila iz radnog skupa potrebno je da prodje F jedinica virt vremena

**58. Sta je VSWS?--?**

- Variable-interval Samplet Working Set

- definise se interval uzrokovanja

- u toku intervala stranice koje prouzrokuju page fault se smestaju u rezid skup

- nakon intervala se izbacuju stranice sa bitom upotrebe 0

- Tokom izvrsavanja proces meri virt vreme. Ako je prosao maks period od poslednjeg usrokovanja, suspenduje se proces i skeniraju bitovi upotrebe.

- Ukoliko je u toku int broj gresaka stranice presao neki prag:

- ako je prosao min period uzrokovanja, vrsi se uzrokovanje

- ako nije prosao ceka se da istekne i tek onda se vrsi uzrokovanje

**59. Opisati politiku ciscenja:**

- Odlucuje kada se izmenjena stranica upisuje nazad u sek mem

* Ciscenje po zahtevu - stranica se upisuje odmah prilikom zamene
* Predciscenje - stranice se upisuju grupno

- Najbolji pristup je baferovanje stranica - lista izmenjenih i neizmenjenih

**60. Koji su problemi premalo procesa u glavnoj mem, a koji problemi sa previse?**

- Premalo procesa u glavnoj mem uzrokuje to da ce vecina proc biti blokirani a procesor besposlen

- Previse procesa u glavnoj mem uzrokuje to da je rez skup premali i da ce biti puno page fault-ova

**61. Koje su varijante izbora procesa koji ce biti suspendovan?**

- Proces najnizeg prioriteta

- Proces koji je izazvao gresku

- Proces koji najduze nije bio aktivan

- Proces sa najmanjim rez skupom ?

- Najveci proces (dobija se najvise slobodnih okvira)

- Proces za koji je najmanje hitno da bude zavrsen

**9. JEDNOPROCESORSKO RASPOREDJIVANJE.PPT**

**1. Koji su ciljevi procesorskog rasporedjivanja?**

* Dodeljivanje procesoru/procesorima procese za izvrsavanje
* Obezbedjivanje fer deljenja vremena medju procesima
* Sprecavanje gladovanja procesa
* Obezbedjivanje efikasnog koriscenja procesora
* Uspostavljanje privilegija medju procesima
* Pomenute ciljeve sprovede uz sto manje dodatnog opterecenja
* Treba da se ispune ciljevi sistema – dobro vreme odziva, velika propusna moc, efikasnost procesora

**2. Koji su tipovi procesorskog rasporedjivanja?**

1. Dugorocno

* stvaranje novog procesa
* odlucuje se da li da se proces doda skupu procesa koji konkurisu za izvrsavanje (na osnovu FIFO ili nekog kriterijuma npr prioritet..)
* upravlja stepenom multiprogramiranja
* sto je vise procesa, manji je procenat vremena u kojem procesa moze da se izvrsava

2. Srednjerocno

* deo funkcije razmene
* odlucuje da se proces doda broju procesa koji su delimicno ili potpuno u glavnoj mem
* odredjuje kada proces treba da bude zamenjen na disk
* rasporedjivanje zadovoljava ciljni stepen multiprogramiranja

3. Kratkorocno (dispecer)

* najcesce se izvrsava
* odlucuje instrukcije kog procesa ce procesor da izvrsava
* aktivira se kada se desi dogadjaj (prekid generatora takta, U/I, poziv os....)

\* ukratko dugorocno rasp odredjuje da li novi proces ide u red spremnih ili suspendovanih

srednjerocno odlucuje koji proces iz suspendovanih ide u spremni i koji iz suspendovanih ide u blokiran

kratkorocno odlucuje koji spreman ide u izvrsavanje

**3. Koji su kriterijumi kratkorocnog rasporedjivanja, na osnovu kojih se ocenjuju politike?**

1. Korisnicki orijentisani kriterijumi - ponasanje programa kako ga vidi korisnik npr. vreme odziva - vreme proteklo od podnosenja zahteva do dobijanja izlaza
2. Sistemski orijentisani - usresredjeni na efikasno koriscenje procesora npr. propusna moc - broj obradjenih procesa u jedinici vremena

**4. Razlike izmedju kriterijuma koji su povezani i onih nepovezanih sa performansom:**

- Oni koji su povezani sa performansom su kvantitativni i merljivi (vreme odziva, propusna moc...)

- Oni koji nisu povezani sa performansom su kvalitativni i teze merljivi (predvidljivost - da usluga prikazuje iste karakteristike tokom vremena)

**5. Navedi karakteristike korisnicki orij krit vezanih za perform, i ostalih:**

- Korisnicki orijentisani vezani za performansu:

* vreme prolaska zadatka - vreme od podnosenja zahteva do zavrsetka procesa
* vreme cekanja - vreme cekanja procesa u redovima cekanja
* vreme odziva - vreme od podnosenja zaheva do pocetka primanja odziva
* rokovi - proces se mora zavrsiti u zadatom roku

- Ostali:

* predvidljivost - posao se treba zavrsiti priblizno istim vremenom i troskovima bez obzira na opterecenje sistema

**6. Navedi karakteristike sistemski orijentisanih kriterijuma vezanih za perform, i ostalih:**

- Vezani za perf:

* propusna moc - broj zavrsenih procesa u jedinici vremena - cilj: uvecati propusnu moc
* iskoriscenje proc - koliko % je procesor zauzet - cilj: sto vise da radi

- Ostali:

* pravicnost- ravnopravno deljenje procesors. vremena od strane procesa i negladovanje procesa
* primena prioriteta - procesi sa visim prioritetom imaju prednost
* uravnotezenje resursa - optimizuje upotrebu resursa (prednost ide onima koji manje koriste preopterecene resurse)

**\*Dva rezima u kojima se vrsi izbor narednog procesa kojem ce biti dodeljen procesor?**

1. Bez prekidanja - proces se izvrsava dok god se ne zavrsi ili se sam blokira da ceka UI uslugu OSa
2. Sa prekidanjem - OS prekida proces koji se izvrsava i stavlja ga u stanje spreman - moze prekinuti proces na osnovu prekida generatora takta npr. ako je isteklo max dovzoljeno vreme

**7. Kako moze biti definisan prioritet procesa?**

Obicno je brojna vrednost

1. Interno - prioritet proistice iz podataka procesa
2. Eksterno - prioritet je definisan u procesu izvan OS (npr. vaznost procesa, tip procesa..)

Raspoređivač bira uvek proces sa višim prioritetom kao sledeći za izvršavanje i to iz različitih redova čekanja za svaki nivo prioriteta.

Pojava GLADOVANJA - kada procesi sa nizim prioritetom ne dobijaju procesor ako ima onih sa visim. Resenje: izmena prioriteta u zavisnosti od starosti procesa i istorije izvrsavanja

**8. Sta je CPU-burst?\*\***

- Vreme proteklo od trenutka dobijanja procesora do gubljenja procesora.

- CPU burst se zavrsava blokiranjem procesa ili vracanjem u red spremnih zbog isteka max dozvoljenog vremena – time slice?

- Proces u toku zivota dobija i gubi vise puta procesor - CPU burst mu se menja!!

- Razlikujemo dva tipa procesa:

* Procesi sa intenzivnom upotrebom procesora - imaju dugacak CPU burst
* Procesi sa intenzivnim UI operacijma - kratak CPU burst

**9. Navedi politike rasporedjivanja procesa:**

* 1. Po redosledu dolaska - FCFS - FIFO
  2. Kruzno dodeljivanje – Round robin
  3. Najkraci proces sledeci - SPN
  4. Najkrace preostalo vreme
  5. Sledeci sa najvecim odnosom odziva
  6. Povratna sprega

**10. Koji su parametri da testiranje politika rasporedjivanja procesa?**

* trenutak dolaska - trenutak kad je postao spreman proces
* trenutak zavrsetka - trenutak kad je izvrsavanje instrukcija kompletirano
* vreme usluge (Ts) - vreme potrebno procesoru da izvrsi instr procesa
* vreme prolaska (Tr) - vreme koje proces provodi u sistemu = usluga + cekanje
* normalizovano vreme prolaska Tr/Ts - relativno kasnjenje procesa - u najb sluc je 1.0, inace > 1.0

**\*First come first served - FCFC politika?**

Politika bez prekidanja u kojoj se proces stavlja u red spremnih procesa

Kada tekuci proces prestane da se izvrsava bira se sledeci koji je najduze cekao – prvi u redu

**11. Koje su mane FCFS (First Come First Served) politike?**

- Kratki procesi dugo cekaju na izvrsavanje

- los odnos vremena prolaska Tr i vremena usluge Ts

- procesi koji imaju dosta U/I dugo cekaju na ponovno izvrsavanje nakon U/I

- dobro prolaze procesi koji intenzivno koriste procesor jer nema prekida kao za UI

- efektivna samo u kombinaciji sa prioritetima procesa

**\*Round Robin - kruzno dodeljivanje?**

Prekidna politika gde svaki proces dobija isecak vremena pre nego sto bude prekinut od strane generatora takta

Ako je istekao isecak trenutni proces se stavlja u red cekanja

Sledeci spremni proces se bira na osnovu FCFS FIFO principa

**12. Mane RR (Round Robin) politike?**

- procesi sa puno U/I kratko koriste vremenski isecak jer se zbog UI blokiraju pre kraja isecka, i nakon U/I opet cekaju u redu spremnih

- bolje prolaze procesi sa intenzivnom upotrebom procesora jer iskoriste ceo vremenski isecak i onda se stave u red spremnih procesa

**13. Kakva bi trebalo da bude velicina isecka u RR politici?**

- ne treba biti premali jer ce biti puno prekida i komutacije -> sporo

- ne treba biti preveliki jer onda RR postaje FCFS

- isecak treba biti neznatno veci od prosecnog vremena za interakciju sa procesorom - na ovaj nacin proces uglavnom obavi posao ili se blokira zbog U/I pre isteka isecka.

- U vecini modernih sistema izmedju 10 i 100 ms

-Procenjeno je da isecak treba da bude takav da 80% CPU bursts bude krace od isecka

**14. Kako VRR (Virtual RR) unapredjuje klasican RR?**

- Resava nepravedan tretman procesa sa dosta UI oper

- Nakon U/I oper proces se ubacuju u poseban FCFS red cekanja koji ima veci prioritet od regularnog red cekanja.

- Tom procesu se vreme isecka umanjuje za onoliko koliko je pre U/I prekida bio izvrsavan.

**\*Najkraci sledeci ? SPN – shortest process next**

Politika bez prekidanja gde se bita proces sa najmanjim ocekivanim CPU burstom

Prikladnije ime: shortest cpu burst next

Kratak proces skace na celo reda ispred drugih procesa

**15. Koje su mane SPN (Shortest Process Next) politike?**

- Smanjuje se predvidljivost vremena prolaska, narocito duzim procesima

- Duzim procesima preti potencijalno gladovanje jer kraci procesi stalno uskacu

- Nije pozeljna za sisteme sa deljenjem vremena jer ne podrzava prekide

**16. Koje su dve varijante procene ocekivanog CPU-burst-a?**

- Prva je da se nadje prosek trajanja svih prethodnih CPU-bursts-a procesa.

- Druga varijanta je da se veca tezina da skorijim instancama - eksponencijalno usrednjavanje. Sto je posmatranje starije to se manje uzima u obzir u proracunu srednje vrednosti.

Eksponencijalno usrednjavanje bolje predvidja stvarno trajanje procesa

**\*Najkrace preostalo vreme - SRT - shortest remaining time**

- Politika s prekidanjeM - prekidna varijanta SPN - shortest process next

- Kada se pojavi novi proces u redu spremnih procesa dobice procesor umesto TEKUCEG procesa ako ima manji ocekivani CPU burst

**17. Koje prednosti uvodi SRT (Shortest Remaining Time) politika?**

- Prekid se desava jedino pojavom novog procesa koji ima kraci CPU-burst u odnosu na onaj koji se trenutno izvrsava.

- Samo novi proces moze ispuniti taj uslov - samo novi proces u redu moze da ima kraci CPU burst od tekuceg procesa

- Radi veoma dobro ukoliko se dobro predvidja CPU-burst.

**18. HRRN (Highest Response Ratio Next) politika:**

- Politika bez prekidanja gde se za svaki proces se racuna proporcija R = (w+s)/s

* w je vreme koje je proces proveo u cekanju na procesor
* s je CPU-burst

- Kada se tekuci proces zavrsi ili blokira, bira se sledeci sa najvecim R

**19. Koje su prednosti HRRN politike?**

- Dobar je tretman za procese sa malim CPU-burstom jer je s malo i R onda veliko

- Dobar je tretman za procese koji dugo cekaju jer w je veliko i R je onda veliko

- Izbegava gladovanje

- Priblizno isto ce se tretirati duzi i kraci procesi

**20. Koje su karakteristike politike povratne sprege?**

- Prekidna politika

- svaki put kada se proces prekine stavlja se u FCFS red cekanja NIZEG prioriteta

- novi procesi brze dobijaju procesor i kraci procesi brze prolaze

1. Kvantum ako je svima isti -> dugi procesi mogu gladovati
2. Kvantum obrnuto proporcionalan prioritetu -> procesi manjeg prioriteta kad dobiju procesor mogu duze da ga koriste

**21. Koje su karakteristike FSS (Fair-Share Scheduler) politike?**

- procesi se klasifikuju u grupe (npr procesi iste aplikacije)

- rasporedjivanje uzima u obzir i grupu procesa kojoj pripada

- prioritet se racuna na osnovu prioriteta samog procesa, dotadasnjeg vremena koriscenja procesora tog procesa i dotadasnjeg vremena koriscenja procesora od strane grupe kojoj proces pripada.

**22. Karakteristike linux rasporedjivaca su:\*\***

- completely fair scheduler

- procesima se dodeljuje procenat procesorskog vremena zavisno od trenutnog broja procesa u sistemu i prioriteta procesa

- vremenski isecak se DINAMICKI dodeljuje kako bi se postigla planirana proporcija u koriscenju ukupnog procesorskog vremena

- u sistemu sa N procesa istog prioriteta - svaki proces bi dobio 1/N procesorskog vremena

- prioriteti sluze kao tezinski faktor pri rasporedeli procesorskog vremena

- nice vrednost definise prioritet - procesi sa vecom vrednoscu imaju manji prioritet jer su vise obzirni prema drugim procesima

-- opseg nice vrednosti od -20 do 19 (default je 0)

- definise maksimalno kasnjenje - konfigurabilna vrednost koja odredjuje duzinu vremenskog intervala u kojem svaki proces mora dobiti procesor bar jednom\*\*\*\*

* + (npr. dva procesa odnosa prioriteta 4:1 sa maks dozv kasnjenjem od 10ms, prvi dobija procesr na 8ms a drugi na 2ms)
  + ako ima previse procesa isecak bi postao jako mali, a troskovi komutacije preveliki

- definise minimalnu granularnost - vrednost koja odredjuje minimalnu duzinu isecka bez obzira na broj procesa

* ovim se za veliki broj procesa gubi ravnopravnost po ceni upotrebljivosti sistema

-nacin odabira sledeceg procesa se ne zasniva na redovima procesa

-procesima se dodeljuje brojna vrednost - virtuelno vreme izvrsavanja

-virtuelno vreme izvrsavanja racuna se na osnovu svih prioriteta u sistemu i dotadasnjeg koriscenja procesora

- virtuelna vremena izvrsavanja se organizuje u binarno stablo pretrage,

sledeci proces je onaj sa najmanjim virtuelnim vremenom izvrsavanja – krajnji levi proces u krajnjem levom listu stabla

**10. MULTIPROCESORSKO RASPOREDJIVANJE.PPT**

**1. Kako se klasifikuju multiproc sistemi?**

1. Labavo spregnuti - autonomni, svaki procesor ima svoju memoriju i U/I kanale
2. Funkcionalno specijalizovani - asimetricni multiproc, jedan glavni procesor opste namene koji rukuje skupom procesora koji imaju specijalizovanu ulogu
3. Cvrsto specijalizovani multiprocesor - simetricni multiprocesor, skup ravnopravnih procesora koji dele memoriju, njima upravlja OS

**2. Na koja 3 vazna pitanja se mora dati odgovor u kontekstu rasporedjivanja za multiproc?**

* Dodeljivanje procesa procesorima
* Upotreba multiprogramiranja na pojedinim procesorima - da li da vise procesa konkurise za izvrsavanje na istom procesoru
* Stvarno rasporedjivanje procesa - nacin izbora sledeceg procesa za izvrsavanje na procesoru

**3. Koje su karakteristike STATICKOG dodeljivanja procesa procesorima?**

- proces se predefinisano dodeli nekom procesoru

- manja je rezija oko rasporedjivanja

- moguce je rasporediti grupu procesa na isti namenski procesor - GRUPNO RASPOREDJIVANJE

- mana je sto jedan procesor moze biti besposlen dok je drugi preopterecen

**4. Koje su karakteristike DINAMICKOG dodeljivanja procesa procesorima?**

- Proces se u razlicitim trenucima moze izvrsavati na razlicitim procesorima

- Jedno resenje je zajednicki opsti red cekanja na bilo koji raspolozivi procesor

- Drugo resenje je odvojen red cekanja za svaki procesor \*cesce je ovo resenje u savremenim os

**5. Koje su tri varijante pristupa raspodele pocesa na procesore?**

I kod statickog i kod dinamickog dodeljivanja potreban je softver koji rasporedjuje procese na procesore

* 1. Glavni- pomocni (master slave)
  2. Jednaka ravnopravnost (Peer)
  3. Pristupi koji kombinuju prethodna 2

**6. Opisati glavni-pomocni pristup – master slave:**

- Kljucne funkcije kernela se izvrsavaju na jednom glavnom procesoru

- Glavni procesor vrsi rasporedjivanje procesa na pomocne procesore

- Pomocni procesori izvrsavaju korisnicke programe i mogu se obratiti glavnom kada im recimo zatreba U/I oper

- Mane su to sto je glavni procesor preopterecen

-- ako glavni procesor rikne -> riknjava ceo sistem!!!!!

**7. Opisati pristup jednake ravnopravnosti - peer:**

- Kernel se moze izvrsavati na svakom procesoru

- Svaki procesor vrsi rasporedjivanje

- Veci je pritisak na OS jer se mora voditi racuna o tome da dva procesora ne uzmu isti proces

**8. Zasto se nastoji na tome da proces ne menja procesor na kojem se prethodno izvrsavao - SKLONOST PROCESA NA PROCESOR?**

- Jer se podaci o izvrsavanom procesu cuvaju u kes memoriji procesora.

- Cilj je da naredno rasporedjivanje procesa moze da iskoristi ranije kesirane podatke

- Zato se nastoji da proces ne menja procesor na kom se prethodno izvrsavao

**9. Kako se realizuje raspodela opterecenja?**

- Vazno je raporediti procese na procesore tako da se poveca iskoriscenog svakog procesora

- Raspodela opterecenja je potrebno u sistemima gde svaki procesor ima poseban red spremnih procesa

- Raspodela opterecenja NEGATIVNO utice na sklonost procesa za procesor

- operacije push i pull

* push - poseban deo OS gleda opterecenje na procesorima i shodno tome prebacuje procese iz reda cekanja jednog u drugi red manje opterecenog procesora
* pull - nezaposleni procesor povlaci kod sebe neki proces iz reda cekanja opterecenog procesora

- push i pull se medjusovno ne iskljucuju ?

**\* Procesor sa vise jezgara - multi core cpu?**

- Vise procesorskih jezgara na istom cipu

- Svako jezgro je odvojen procesor

**10. Sta su hardverske niti procesorskog jezgra?**

- Usporenje memorije: procesor pri izvrsavanju instrukcije dosta vremena provede cekajuci na dostupnost podataka

-- Ako ne pronadje podatak u kesu, mora da pristupa nizim nivoima kesa ili radnoj memoriji

-- Dok taj podatak ne bude ucitan, procesor moze da izvrsava druge instrukcije

-- Kreiraju se posebne hardverske niti na jednom jezgru procesora

-- Dok jedna nit ceka na memoriju, druga se izvrsava

- Svaka hardverska nit se ponasa kao logicki procesor.

- OS vidi procesor sa 2 jezgra i 2 hardverske niti kao 4 procesora.

- OS vrsi rasporedjivanje softverskih niti procesa na hardverske niti jezgra

**\* Rasporedjivanje softverskih niti?**

Aplikacija se moze organizovati kao skup niti koje saradjuju medjusobno i izvrsavaju se konkurentno na istom adresnom prostoru.

U multiprocesorskom sistemu dolazi do boljih performansi jer se niti istovremeno izvrsavaju na razlicitim procesorima

**11. Navedi 4 pristupa za multiprocesorsko rasporedjivanje niti:\***

* deljenje opterecenja - preko centralizovanog rasporedjivaca se niti ravnomerno rasporedjuju procesorima (mana- malo je vrv da ce prekinuta nit nastaviti izvrsavanje na istom procesu kao i to da ce niti jedne aplikacije istovremeno dobiti pristup procesorima)
* grupno rasporedjivanje - grupe povezanih niti se rasporedjuje na skup procesa - poboljsava performanse jer su komutacije redje
* dodeljivanje namenskog procesora - u startu se za svaku nit odredi procesor na kojem ce se izvrsavati do kraja - smanjeno je iskoriscenje procesora ali se mogu drasticno ubrzati performanse pogotovo kod paralelnih sistema
* dinamicko rasporedjivanje - aplikacija dinamicki menja broj niti zavisno od broja procesora koji su joj trenutno dodeljeni - nije zazivela u savr sistemima

**12. Deljenje opterecenja?**

- Niti se ne dodeljuju posebnom procesoru nego se preko centralizovanog rasporedjivaca rasporedjuje kada procesor postane raspoloziv

- Red cekanja se moze organizovati po nekom algoritmu za jednoprocesorsko rasporedjivanje

- Opterecenje se ravnomerno rasporedjuje po procesorima

- MANE:

* Malo je verovatno da ce prekinuta nit nastaviti na istom procesoru - neefikasan rad sa kes memorijom
* Malo je verovatno da ce niti jedne app istovremeno dobiti pristup procesorima - nedovoljno iskorisc prednosti paralelizma

**13. Grupno rasporedjivanje?**

* Skup povezanih niti se rasporedjuje za izvrsavanje u isto vreme na skup procesora
* Moze poboljsati performanse jer su komutacije redje
* Niti se brze sinhronizuju sa drugom niti ukoliko se obe izvrsavaju
* Nije potrebna komutacija da bi druga nit dobila procesor

**14. Dodeljivanje namenskog procesora?**

- Esktremni oblik grupnog rasporedjivanja

- U startu se svakoj niti odrediti procesor na kojem ce se od pocetka do kraja izvrsavati

- Nema multiprogramiranja - neki od procesora mogu biti besposleni

- Moze drasticno ubrzati performanse jer odsustvo komutacije dolazi do ubrzanja

- U veoma paralelnim sistemima (npr racunanja) vaznija je efektivnost od iskoriscenja procesora

**15. Dinamicko rasporedjivanje?**

- Broj niti u procesu se menja u toku izvrsavanja u zavisnosti od broja procesora koji su dodeljeni aplikaciji

- Nije zazivelo

**11. UI UPRAVLJANJE.PPT**

**1. Koje su kategorije U/I uredjaja?**

1. Za komunikaciju sa korisnikom – mis, tastatura, stampac
2. Za komunikaciju sa hardverom – diskovi, senzori, kartice
3. Za komunikaciju sa udaljenim uredjajem – modemi, mrezne kartice

**2. U cemu se ogledaju razlike izmedju U/I uredjaja?**

* brzina prenosa podataka
* uloga u sistemu - utice na podrsku OS za uredjaj
* slozenost upravljanja - jednostavno upravljanje (mis, stamp) ili slozeno (disk)
* jedinica prenosa podataka - orIjentisani na 1) blokove ili 2) tokove bajtova (jedinice prenosa su blokovi ili tokovi bajt po bajt)
* nacin predstavljanja podataka - nacin kodiranja - format, parnost
* upravljanje greskama - razlicit nacin reakcije na gresku i skup kodova gresaka
* nacin pristupa - sekvencijalni (salje i prima podatke u fiksnom redosledu - ruter, stampac..) ili proizvoljan (upucuje se zahtev uredjaju da se pozicionira na bilo koju lokaciju sa podacima - disk)
* podrska za upis i citanje - upis i citanje (disk), samo upis (stampac), samo citanje (tastatura/mis)

**3. Kako uredjaj razmenjuje podatke sa racunarom?**

- Preko prenosnog medijuma kao sto su kabel ili vazduh.

**4. Kako uredjaj moze biti povezan sa racunarom?**

* Preko porta - kada jedan uredjaj koristi komunikacione kanale
* Preko magistrale - kada vise uredjaja deli isti skup kanala.

**5. Sta je kontroler?**

- Hardver koji upravlja portom, magistralom ili uredjajem

**6. Sta je host adapter?**

- Kontroler u racunaru zaduzen za komunikaciju sa uredjajem.

**\* Sta je device controller?**

- Kontroler u uredjaju namenjen za upravljanje tim uredjajem

**7. Koje su dve varijante komunikacije procesora sa U/I hardverom?**

1. Procesor izvrsava specijalnu U/I instrukciju koja adresira registre kontrolera i upisuje/cita u njih.
2. Memory mapped I/O - Procesor izvrsava standardnu instrukciju koja pristupa adresama u radnoj memoriji, a koje su namapirane na registre kontrolera U/I uredjaja.

**8. Koji tipovi U/I registra postoje?**

* Data-in - iz njega procesor preuzima podatke
* Data-out - u njega procesor upisuje podatke
* Statusni reg - info o statusu operacije za procesor (da li je gotova, kod greske...)
* Upravljacki registar - u njega procesor upisuje komandu za U/I uredjaj

**9. Koji su tipovi organizacija U/I funkcije?**

1. Programirani U/I - procesor na zahtev procesa zadaje U/I komandu. Proces aktivno proverava da li je U/I operacija zavrsena - zaposleno cekanje - busy waiting
2. U/I pokretan prekidom - procesor na zahtev procesa izdaje U/I komandu. Ukoliko je instrukcija blokirajuca, proces prelazi u stanje blokiran, a u suprotnom procesor nastavlja izvrsavanje instrukcija procesa. UI uredjaj postavlja prekid kada zavrsi operaciju. Rutina za obradu prekida se izvrsava kao reakcija na zavrsenu U/I operaciju.

**10. Kada je potrebno izvrsiti iskljucivanje prekida?**

- Dok traje kriticna operacija koja ne sme biti prekinuta – npr. dok se obradjuje prethodni prekid.

- Procesor ima bit koji oznacava da li su prekidi ukljuceni.

- Privilegovanom instrukcijom se moze postaviti vrednost ovog bita.

- Taj bit se odnosi samo na standardne (maskable) prekide,

dok se specijalni (nonmaskable) prekidi ne mogu iskljuciti (osim u slucaju fatalnih gresaka).

**\* Sta kada uredjaj postavi prekid?**

- Kada uredjaj postavi prekid, potrebno je izvrsiti rutinu za obradu prekida

- U drajveru se nalazi kod za obradu prekida

- Svaki drajver registruje svoju rutinu-

- Pri signalizaciji prekida uredjaj postavlja broj - adresu - koji ukazuje na tip prekida - tako procesor zna koji je uredjaj postavio prekid!

**11. Sta je vektor prekida?**

- Niz adresa koje ukazuju na rutine za obradu prekida.

Kada uredjaj postavi prekid, on salje adresu koja govori koji je tip prekida i na osnovu te adrese se preko vektora prekida indeksira odredjeni element

- Obicno vektor prekida ima manje elem nego sto ima uredjaja: tada element vektora prekida sadrzi adresu liste obradjivaca prekida i skeniraju se svi elementi liste da se nadje odgovarajuci obradjivac

**12. Objasniti kako funkcionise DMA (direktni pristup memoriji - direct memory access) prenos.**

- Procesor moze da delegira izvrsavanje IO operacija specijalizovanom procesoru - dma kontroleru

- DMA kontroler upravlja razmenom podataka izmedju glavne memorije i UI modula!

- Procesor salje DMA modulu zahtev za prenos bloka podataka: src adr, dest adr i broj bajtova za prenos

- DMA kontroler vrsi prebacivanje podataka

- kada se DMA prenos zavrsi DMA kontroler postavlja prekid i procesor tako zna da je zavrsen

**\*Koraci u dma prenosu? – slika slajd engleski**

1. Drajveru uredjaja se trazi transfer podataka sa diska na baferu na adresi X

2. Drajver uredjaja obavestava kontroler diska da prebaci C bajtova sa diska u bafer na adresi X

3. Kontroler diska pokrece DMA transfer

4. kontroler diska salje svaki bajt DMA kontroleru

5. DMA kontroler salje bajtove u bafer na adresi X, povecava adresu i decr C dok god je C>0

6. Kada C==0 DMA salje prekid procesoru da signalizira kraj transfera

**13. Koje su razvojne faze U/I funkcije?**

* 1. procesor direktno upravlja U/I uredjajem
  2. uvodi se kontroler - UI modul - koriste se programirani UI bez prekida
  3. uvode se prekidi - efikasnije jer proces ne mora stalno da proverava da l se zavrsilo
  4. ui modul direktno upravlja memorijom preko dma - procesor moze da radi druge stvari dok dma prenosi blokove podataka
  5. ui modul postaje poseban procesor - citav niz ui aktivnosti obavlja ui procesor bez da se uplice glavni procesor
  6. ui modul dobija i sopstvenu memoriju - ui modul kao odvojen racunar koji samostalno upravlja ui uredjajem uz minimum angaz procesora

**14. Koji su ciljevi projektovanja U/I podsistema?**

- Efikasnost - ui uredjaji su spori u odnosu na glavnu mem i procesor, multiprog dozvoljava izvrsavanje drugog procesa dok prethodni ceka na obradu ui, procesi se razmenjuju izmedju glavne i sek memorije preko ui

- Opstost - radi jednostavnosti pozeljno je da se upravlja svim ui uredjajima na jedinstven nacin. to se postize apstrakovanjem zajednickih operacija poput: citaj pisi otvori i zatvori.

- rutine nizeg nivoa obavljaju operacije nad konkretnim uredjajem i sakrivaju detalje uredjaja

- drajveri uredjaja implementiraju te operacije zasebno za svaki uredjaj.

- Zato se uvodi hijerarhijska sturkura realizovana slojevima, svaki sloj se oslanja na onaj sledeci u hijerarhiji.

izmena u jednom sloju ne bi trebala da izazove potrebu menjanja drugog.

zato se ne treba menjati OS kada se dodaje novi uredjaj vec samo napisati drajver za taj uredjaj

**- Ogranizacija slojeva u UI podsistemu**

Softverski sloj:

* kernel
* kernel IO podsistem - rad sa uredjajem kao logickim resursom, ne brine o detaljima stvarnih uredjaja, genericke komande- citaj, pisi
* drajveri uredjaja 1..N - stvarno cekanje u redovima i rasporedjivanje UI operacija, nivo softvera koji stvarno komunicira sa hardverom uredjaja

hardverski sloj:

* kontroleri uredjaja 1..N
* uredjaji 1..N

**\* Baferi?**

Prostor u memoriji u koji se skladiste podaci koji se prenose od/ka UI uredjaja

Proces ne mora da ceka na spore ui operacije:

Pri upisu, proces postavi odjednom sve svoje podatke u bafer a kontroler ih salje ka uredjaju

Pri citanju, najpre se svi podaci prihvate u bafer, a proces im onda tek pristupa

**15. Objasniti korisne karakteristike U/I baferovanja:**

- Bafer je prostor u memoriji sa u koji se pise ili sa kojeg se cita.

Proces podatke koje upisuje u U/I smesta u bafer i ostavlja na kontroleru zadatak da ih on dalje upise te ne mora cekati na to izvrsenje. Takodje ne mora cekati na ucitavanje podataka, vec proces potrebnim podacima moze pristupiti tek nakon sto su one ucitane u bafer.

Posto ui oper koristi podatke iz bafera -> proces moze nakon izdavanje ui oper da promeni podatke svoje

!Stranica u toku ui oper cak ne mora biti ni u glavnoj mem, moze biti zamenjena na disk.

**16. Zasto se uvodi dvostruki bafer?**

-Proces ucitava podatke iz jednog bafera dok OS ucitava sledeci blok u drugi bafer

-Onda proces predje na drugi bafer a os se vrati na prvom itd

-Slicno ali obrnuto za upis

**17. Sta je kruzni bafer?**

- Prosirenje dvostrukog bafera u kojem se uvodi N bafera.

- U/I redom cita i pise u te bafere dok os pise/cita.

**18. Koji su parametri performanse diska?**

disk - puno sporiji od radne memorije, pristupa mu se za pristup fajlovima i stranicama u VM

Parametri performanse diska:

* cekanje na uredjaj - da postane raspoloziv procesu
* cekanje na u/i kanal - ako se kanal deli sa drugim UI uredjajima
* vreme pozicioniranja - vreme potrebno glavi diska da se pomeri na potrebnu stazu
* rotaciono kasnjenje - vreme potrebno disku da se rotira tako da trazeni sektor bude ispod glave
* vreme prenosa - vreme potrebno da s podaci ucitaju i upisu tako sto se sektor krece ispod glave rotiranjem diska

**\*Kako OS utice na vreme pozicioniranja ?**

- Organizacijom zahteva pristupa stazama - rasporedjivanje diska

**19. Koje su politike rasporedjivanja diska?**

* FIFO
* SSTF - bira se zahtev koji se najbrze izvrsi – zahteva najmanje pomeranje rucice -> minimizuje se vreme pozicioniranja
  + Ove dve politike mogu dovesti do gladovanja
* SCAN - pomera rucicu samo u jednom smeru, kada dodje do kraja menja smer
  + dovodi do dugog cekanja staze na periferiji
* C-SCAN - ogranicava skeniranje na jedan smer, kada se dodje do kraja vraca se na pocetak

**20. Koji su tipovi formatiranja diska?**

* Fizicko formatiranje - deljenje diska u sektore (zaglavlje, podaci, kod za ispravku greske)
* Particionisanje - disk se deli na particije, svaka particija je logicki disk
* Logicko formatiranje - OS inicijalizuje fajl sistem na disku
* Raw disk - particija bez fajl sistema, podaci kao niz blokova - omogucava kontrolu niskog nivoa i moze biti efikasnije u spec slucajevima jer nema fajl sistema kao sloja izmedju

**21. Sta je boot particija I KAKO SE POKRECE SISTEM?**

- Boot parrticija je ona particija koja mora da sadrzi kod OSa da bi se sistem mogao startovati

- Racunar iz ROM memorije pokrece jednostavan program koji identifikuje boot particiju

- Boot particija startuje kod za ucitavanje OS sa predefinisane lokacije - prvi sektor diska - MASTER BOOT RECORD

**23. Sta je prostor za razmenu - SWAP SPACE?**

- Prostor na disku namenjen za VM - vm koristi disk kao dodatak radnoj memoriji

- Izdeljen je na slotove za menjanje stranica.

**24. Koje su 2 varijante implementacije prostora za razmenu?**

1. Nalazi se kompletna slika procesa - kod i podaci
2. Samo podaci koji se menjaju prilikom izvrsavanja - stack i heap, a kod i static podaci su svakako na disku, nema potrebe i u swap spaceu

**25. Gde moze biti smesten prostor za razmenu?**

1. Moze biti u fajlu - koristi funkcije file sistema, jednostavno je ali manje efikasno
2. U posebnoj Raw particiji - ne koristi funckije file sistema, tezi je za upravljanje i nefleksibilan za promenu velicine - ali efikasniji

**26. Karakteristike ssd:**

* Brzi i pouzdaniji od HDD jer nema pokretnih delova
* Manjeg kapaciteta i skuplji od HDD
* Nema kasnjenja jer nema rotacije i pozicioniranja
* Isto je vreme pristupa za sve delove pa se najcesce rasporedjuje politikom FCFS

**27. Sta je RAID ?**

- Redundant Array of Independent Disks

- To je skup odvojenih i nezavisnih fizickih diskova koje OS vidi kao jednu celinu, odnosno jedan uredjaj.

- podaci su rasporedjeni po svim diskovima

- redundantni diskovi se koriste za skladiste dodatnih info

-- one sluze za obnovljivost podataka u slucaju otkaza diska

**28. Sta je kes diska?**

- Bafer u glavnoj memoriji za kopije nekih sektora diska

- Kada se pojavio IO zahtev za pristup sektoru, prvo se proverava da l je taj sektor u kešu

**29. Koje su politike za popunjavanje kesa diska prisutne?**

* Zamena najmanje skoro koriscenog - novi blok se ubacuje na mesto onog koji je najranije koriscen (koristi mehanizam steka - izbacuje se blok s dna steka)
* Zamena najredje koriscenog - novi blok ide na mesto onog koji je najmanje puta koriscen (uvodi se potreba za brojacem koriscenja svakog bloka) - problem je sto vrednost brojaca moze da zavara ako se blok ref malo puta ali se tada zbog lokalnosti njemu cesto pristupa
* Zamena na bazi ucestalosti koriscenja - uvode se dve sekcije, brojac se uvecava samo ako se blok pomera iz jedne sekcije u drugu - brojac ostaje isti ako se blok u kratkom periodu cesto ref sto prevazilazi prethodno navedeni problem sa lokalnoscu

**12. FAJL SISTEM.PPT**

**1. Sta je fajl?**

- Resurs koji sadrzi grupe podataka.

\*- Skup logicki povezanih zapisa koji se tretira kao jedan entitet. Referencira se po imenu.

**2. Sta je sistem za upravljanje fajlovima?**

- Sistemski softver zaduzen za rad sa fajlovima.

- Obezbedjuje apstrakciju resursa uskladistenih u sekundarnoj memoriji.

**3. Sta je fajl sistem?**

- Skup metoda i struktura podataka koje OS koristi za upravljanje fajlovima.

**4. Navedi neke od tipova fajl sistema: (ekstenzija??)**

- ext2,3,4, NFS, FAT, NTFS

**5. Koje su pozeljne osobine fajlova?**

* Dugorocno postojanje
* Deljivost izmedju procesa
* Struktura - interna struktura takva da je aplikacijama pogodna za koriscenje

**6. Navedi operacije nad fajlovima:**

* Kreiranje/Brisanje
* Otvaranje/Zatvaranje
* Citanje/Upisivanje

**7. Sta je polje a sta zapis podatka?**

- Polje je osnovni element podatka koji sadrzi pojedinacnu vrednost definisanu duzinom i vrstom podataka.

- Zapis je skup logicki povezanih polja koje aplikacija vidi kao celinu.

**8. Navedi komponente sistema za upravljanje fajlovima:**

* Drajveri - direktno upravljanje sa najnizim nivoom
* Osnovni FS - bavi se blokovima podataka koji se razmenjuju sa uredjajem
* Osnovni U/I Supervizor - generalna organizacija U/I zahteva za sve uredjaje
* Logicki U/I - pruza standardni interfejs izmedju aplikacija preko FS do uredjaja

**9. Koji su kriterijumi izbora organizacije fajlova?**

* Vreme pristupa
* Jednostavnost azuriranja
* Ekonomicnost skladista
* Jednostavnost odrzavanja
* Pouzdanost

**10. Navedi tipove organizacije fajlova:**

1. Gomila
2. Sekvencijalni fajl
3. Index sekvencijalni fajl
4. Indeksni fajl - razliciti indeksi za razlicita polja
5. Direktni/Hesirani fajl

**11. Sta sve sadrzi upravljacki blok fajla?**

* Osnovne info - ime i tip
* Adresne informacije - volumen, pocetna adresa i velicina
* Upravljanje pristupom - vlasnik i prava pristupa
* Info o upotrebi - vreme kreiranja, koji korisnik je kreirao, vreme poslednjeg citanja, korisnik koji je poslednji izvrsio citanje, vreme posl modif, korisnik koji je izvr posl modif, info o trenutnoj upotrebi(koji procesi ga koriste, da li je zakljucan, da li je izmenjen u mem)

**12. Sta je direktorijum?**

- Fajl koji obezbedjuje mapiranje imena fajlova na fajl resurse na disku.

**13. Navedi operacije nad direktorijumom:**

* Pretrazivanje fajla
* Kreiranje/Brisanje fajla
* Listanje direkt
* Azuriranje direkt

**14. Koje strukture direktorijuma postoje?**

1. Lista stavki - najjednostavnija, svaki fajl mora imati unique name
2. U dva nivoa - svaki korisnik dobija svoj direktorijum, fajlovi moraju biti unikatni na nivou korisnika
3. Hijerarhijska struktura - opste prihvacena - svaki direktorijum moze sadrzati poddirektorijume i fajlove

**15. Koja su prava pristupa fajlu moguca?**

* Nikavo - korisnik nije svestan ni postojanja fajla
* Znanje - korisnik moze da utvrdi da fajl postoji i ko je vlasnik
* Izvrsavanje - korisnik moze da ucita i izvrsi program, ne moze kopirati fajl
* Citanje - cesto ukljucuje i pravo kopiranja
* Dodavanje podataka - korisnik moze samo da dodaje pod u fajl, ali ne da menja prethodne i brise
* Azuriranje - ima pravo potpune izmene/brisanja/dodavanja pod u fajl
* Promena prava pristupa - moze menjati pravo pristupa nad fajlom
* Brisanje - moze obrisati fajl

**16. Kome se mogu dodeliti prava pristupa fajlu?**

* Individualnom korisniku
* Grupi korisnika
* Svima

**17. Koja su dva pristupa u dodeljivanju prostora fajlu.**

1. Dodeljivanje unapred - potrebno je unapred znati maks velicinu fajla sto je tesko
2. Dinamicko dodeljivanje - delovi se dodeljuju prema potrebi

**18. Navedi prednosti i mane velikih i malih velicina delova fajla:**

Mali

(+) rasipanje prostora je minimalno i lakse je rasporedjivanje prostora

(-) komplikovanija evidencija dodeljivanja (potrebna je tabela)

Veliki

(+) performansa pri pristupu je dobra jer su podaci u susednim lokacijama

(-) teska relokacija pri dodeljivanju novih fajlova

**19. Koje su tri najcesce koriscene metode dodeljivanja fajlova?**

1. Susedno dodeljivanje
2. Ulancano dodeljivanje
3. Indeksno dodeljivanje

**20. Karakteristike susednog dodeljivanja:**

- Jedan skup susednih blokova se dodeljuje fajlu u trenutku stvaranja - dodeljivanje unapred delova promenljive velicine

- Samo jedna stavka po fajlu ide u tabelu dodeljivanja - poc blok i duzina

- Jednostavan pristup - poc blok + i

- Problem je eksterna fragmentacija

**21. Karakteristike ulancanog dodeljivanja:**

- Dodeljivanje se vrsi na bazi pojedinacnog bloka fiksne velicine

- Svaki blok ima referencu na sledeci u lancu

- Nema eksterne fragmentacije

- Samo jedna stavka po fajlu u tabeli dodeljivanja - poc blok i duzina

- Problem je sto ne omogucava direktan pristup fajlu i nije u saglasnosti sa principom lokalnosti

**22. Karakteristike indeksnog dodeljivanja:**

- Dodeljeni delovi ne moraju biti susedni i ne pokazuju na sledeci deo

- Tabela Dodeljivanja za svaki fajl sadrzi indeks delova.

- Indeks sadrzi jednu stavku za svaki dodeljeni deo i cuva se u posebnom bloku.

- Stavka u tab dod sadrzi adresu bloka u kojem je indeks.

- Najpopularniji oblik dodeljivanja fajlova.

- Dve varijante - blokovi fiksne velicine(nema spoljne fragmentacije) i blokovi prom velicine(poboljsana lokalnost)

**23. Koje su varijante realizacije tabele dodeljivanja diska?**

- Tabela bitova - za svaki blok ima bit koji ukazuje na to da li je blok slobodan ili zauzet. Problem je velicina tabele za velike diskove i sporo pronalazenje slobodnog bloka.

- Ulancani slobodni delovi - svaki slob deo pokazuje na sledeci i sadrzi info o duzini. Problem je spoljna fragm.

- Indeksiranje - kao i kod indeksnog dodeljivanja fajla

- Lista slobodnih delova - svaki blok ima svoj broj, odrzava se lista brojeva slob blokova, posto je velika vecinski se skladisti na disku a mali deo u glavnoj mem zbog brzeg pristupa

**24. Navedi varijante evidencije prava pristupa:**

* Matrica pristupa - za svakog subjekta se definise skup operacija nad odredj objektom
* Lista kontrole pristupa - dekompozicija matrice po kolonama - za svaki fajl se navode korisnici i njihova prava pristupa
* Lista sposobnosti - dekomp matrice po vrstama - za svakog korisnika se navode prava pristupa nad fajlovima

**25. Kako je kontrola pristupa realizovana u UNIX-based sistemima?**

- Prava pristupa se definisu za 3 subjekta - vlasnik fajla, korisnici koji pripadaju grupi kojoj fajl pripada i ostali korisnici

- Moguca prava pristupa su citanje pisanje i izvrsavanje

**26. Koje su strukture za upravljanje fajlovima na disku?**

- Boot control block - info potrebne za start OS

- Volume control block - generalne info o volumenu - broj i velicina blokova, lista slob..

- Struktura direktorijuma - definise spisak i organizaciju fajlova

- Upravljacki blok fajla - info o fajlu

**27. Koje su strukture za upravljanje fajlovima u mem?**

- Info o svakom volumenu u FS

- Globalna tabela otvorenih fajlova - sadrzi kopije upr blokova otvorenih fajlova

- Lokala tabela otvorenih fajlova - po jedna za svaki proces, info o tekucoj poziciji unutar fajla, mod pristupa itd..

- Baferi - blokovi fajla koji se citaju/upisuju na disk

- Kes direktorijuma - info o direktorijumima kojima je nedavno pristupano

**28. Kako OS implementira FS?**

- Sadrzi genericke funkcije za rad sa fajl sistemom

- Konkretan FS implementira te funkcije