Operativni sistemi

(pitanja i odgovori)

Sadržaj

Uvod

1. Koje poslove obavlja operativni sistem?

Operativni sistem objedinjuje raznorodne delove računara u skladnu celinu i sakriva od korisnika one detalje funkcionisanja ovih delova, koji nisu bitni za korišćenje računara. (upravlja sastavnim delovima računara i pruža korisniku pristupačno radno okruženje)

2. Šta obuhvata pojam datoteke?

Pojam datoteke obuhvata sadržaj (korisnički podaci) i atribute (npr. veličina datoteke i vreme njenog nastanka) datoteke.

3. Šta se nalazi u deskriptoru datoteke?

U deskriptoru datoteke nalaze se atributi datoteke.

4. Šta omogućuju datoteke?

Datoteke omogućuju trajno čuvanje podataka.

5. Šta obavezno prethodi čitanju i pisanju datoteke?

Čitanju i pisanju datoteke obavezno prethodi njeno otvaranje, radi pripreme zahtevanog pristupa podacima.

6. Šta sledi iza čitanja i pisanja datoteke?

Iza čitanja i pisanja datoteke sledi njeno zatvaranje. Time se sačuvaju atributi i sadržaj datoteke i ujedno onemogući pristupanje njenim podacima do njenog novog otvaranja.

7. Šta obuhvata pojam procesa?

Pojam procesa obuhvata aktivnost (angažovanje procesora na izvršavanju korisničkog programa), sliku (adresni prostor procesa sa naredbama izvršavanog programa, stekom i podacima koji se obrađuju u toku izvršavanja programa) i atribute (npr. stanje procesa i njegov prioritet) procesa.

8. Šta se nalazi u deskriptoru procesa?

U deskriptoru procesa nalaze se atributi procesa.

9. Koja stanja procesa postoje?

Stanja procesa su: čeka, spreman i aktivan.

10. Kada je proces aktivan?

Proces je aktivan kada procesor izvršava program.

11. Šta je kvantum?

Kvantum je unapred određeni vremenski interval za koji aktivni proces prepušta procesor spremnom procesu istog (najvišeg) prioriteta.

12. Šta se dešava nakon isticanja kvantuma?

Trenutke isticanja kvantuma označavaju prekidi sata. Obrada ovakvih prekida sata izaziva prevođenje aktivnog procesa u stanje spreman i preključivanje procesora na onaj od ostalih procesa najvišeg prioriteta, koji je najduže u stanju spreman. Pri tome aktivirani proces prelazi iz stanja spreman u stanje aktivan.

13. Po kom kriteriju se uvek bira aktivan proces?

Aktivan proces se bira po prioritetu procesa i vremenu provedenom u stanju spreman.

14. Koji prelazi su mogući između stanja procesa?

Mogući prelazi su: aktivan $\rightarrow$ čeka, čeka $\rightarrow$ spreman i spreman $\leftrightarrow$ aktivan

15. Koji prelazi nisu mogući između stanja procesa?

Nisu mogući prelazi: čeka $\rightarrow$ aktivan i spreman $\rightarrow$ čeka.

16. Šta omogućuju procesi?

Procesi omogućuju bolje iskorišćenje računara (procesora) i njegovu bržu reakciju na dešavanje vanjskih događaja.

17. Šta karakteriše sekvencijalni proces?

Proces je sekvencijalan ako je njegov trag (redosled u kome se izvršavaju naredbe programa) određen u vreme programiranja, ondosno ako zavisi samo od obrađivanih podataka. Trag sekvencijalnog procesa može da se prikaže kao nit koja povezuje izvršavane naredbe u redosledu njihovog izvršavanja. Nit sekvencijalnog procesa nasleđuje njegove atribute (stanje i prioritet). Mana sekvencijalnih procesa je da su neosetljivi na vanjske događaje.

18. Šta karakteriše konkurentni proces?

Proces sa više istovremeno postojećih niti naziva se konkurentan proces. On odgovara izvšavanju konkurentnog programa. Samo jedna nit može biti u stanju aktivna, dak su ostale u stanju čeka ili spremna. Preključivanja procesora sa jednog od ovih izvršavanja na drugo, zavisno od vanjskih događaja, uzrokuju da se odgovarajuće niti prepliću u redosledu koji nije određen u vreme programiranja.

19. Šta ima svaka nit konkurentnog procesa?

Svaka nit konkurentnog procesa ima svoj prioritet, svoje stanje, svoj stek, pa i svoj deskriptor.

20. Koju operaciju uvodi modul za rukovanje procesorom?

Modul za rukovanje procesorom uvodi operaciju preključivanja.

21. Po čemu se razlikuju preključivanja između niti istog procesa i preključivanja između niti raznih procesa?

U toku preključivanja procesora između niti istog procesa ne dolazi do izmene adresnog prostora procesa, pa je ovakvo preključivanje brže (kraće) od preključivanja između niti raznih procesa.

22. Koje operacije uvodi modul za rukovanje kontrolerima?

Modul za rukovanje kontrolerima, odnosno njegovi drajveri uvode (drajverske) operacije ulaza i izlaza.

23. Šta karakteriše drajvere?

Svaki od drajvera specijalizovan je za jednu vrstu kontrolera, opslužuje jednu klasu ulazno-izlaznih uređaja i ima zadatak da konkretan ulazno-izlazni uređaj predstavi u apstraktnom obliku sa jednoobraznim i pravilnim načinom korišćenja. Obrađivači prekida ulaze u sastav drajvera.

24. Koje operacije uvodi modul za rukovanje radnom memorijom?

Modul za rukovanje radnom memorijom uvodi operacije zauzimanja i oslobađanja.

25. Koje operacije poziva modul za rukovanje radnom memorijom kada podržava virtuelnu memoriju?

Kada podržava virtuelnu memoriju, modul za rukovanje radnom memorijom poziva operacije ulaza i izlaza (jer se tada brine o prebacivanju sadržaja stranica između radne i masovne memorije).

26. Koje operacije uvodi modul za rukovanje datotekama?

Modul za rukovanje datotekama uvodi operacije otvaranja, zatvaranja, čitanja i pisanja.

27. Koje operacije poziva modul za rukovanje datotekama?

Modul za rukovanje datotekama poziva operacije ulaza i izlaza (jer se brine o prebacivanju delova sadržaja datoteka između radne i masovne memorije) i operaciju zauzimanja (radi zauzimanja dovoljno velikog baferskog prostora za bafere, koji su potrebni za pomenuto prebacivanje)

28. Šta omogućuju *multiprocessing* i *multithreading*?

Multiprocessing i *multithreading* omogućuju bolje iskorišćenje procesora, istovremenu podršku većeg broja korisnika (višekorisnički režim rada) i bržu reakciju računara na vanjske događaje.

29. Koje operacije uvodi modul za rukovanje procesima?

Modul za rukovanje procesima uvodi operacije stvaranja i uništavanja.

30. Koje operacije poziva modul za rukovanje procesima?

Modul za rukovanje procesima poziva operacije čitanja, zauzimanja i oslobađanja.

31. Koje module sadrži slojeviti operativni sistem?

Slojeviti operativni sistem sadrži 5 modula, i to za rukovanje: procesom, datotekama, radnom memorijom, kontrolerima i procesorom. (nabrojani su moduli od najvišeg do najnižeg sloja hijerarhije)

32. Šta omogućuju sistemski pozivi?

Sistemski pozivi omogućuju prelazak iz korisničkog prostora u sistemski prostor, radi poziva operacija operativnog sistema.

33. Koje adresne prostore podržava operativni sistem?

Operativni sistem podržava korisnički i sistemski prostor (korisnički prostor je adresni prostor koji poseduje svaki od procesa, a sistemski prostor je adresni prostor koji poseduje operativni sistem).

34. Šta karakteriše interpreter komandnog jezika?

Interpreter komandnog jezika je poseban proces iz korisničkog sloja, koji je zadužen za preuzimanje i interpretiranje komandi komandnog jezika. On posreduje između korisnika i operativnog sistema. Koristi operativni sistem na programskom nivou, jer u toku svog rada poziva sistemske operacije.

35. Koji nivoi korišćenja operativnog sistema postoje?

Postoje programski (omogućuje ga sistemska biblioteka) i interaktivni nivo (omogućuju ga komande komandnog jezika) korišćenja operativnog sistema.

Konkurentno programiranje

1. Šta je preplitanje?

Preplitanje je mešanje izvršavanja naredbi raznih niti, odnosno niti i obrađivača prekida.

2. Da li preplitanje ima slučajan karakter?

Preplitanje ima slučajan karakter, jer nije unapred poznato posle izvršavanja koje naredbe će se desiti prekid i eventualno preključivanje.

3. Šta izaziva pojavu preplitanja?

Preplitanje izazivaju prekidi i eventualno preključivanje (izazvano obradom prekida, nakon čega procesor izvrši naredbe potprograma preključivanja i zatim nastavi da izvršava naredbe druge niti).

4. Da li preplitanje može uticati na rezultat izvršavanja programa?

Preplitanje može da utiče na rezultat izvršavanja programa. Rezultati mogu da budu stohastični.

5. Šta su deljene promenljive?

Kada istoj promenljivoj pristupa više niti ili niti i obrade prekida, tu promenjivu nazivamo deljena promenljiva.

6. Šta je preduslov očuvanja konzistentnosti deljenih promenljivih?

Preduslov očuvanja konzistentnosti deljenih promenljivih je obezbeđenje međusobne isključivosti izvršavanja operacija deljenih promenljivih (omogućavanje da novo izvršavanje bilo koje od operacija deljene promenljive može početi tek nakon završetka prethodno započetog izvršavanja neke od ovih operacija).

7. Šta su kritične sekcije?

Tela operacija deljenih klasa ili delovi ovih tela, čije izvršavanje je kritično za konzistentnost deljenih promenljivih, nazivaju se kritične sekcije.

8. Šta je sinhronizacija?

Sprovođenje vremenskog usklađivanja izvršavanja kritičnih sekcija, kojim se postiže njihova međusobna isključivost, naziva se sinhronizacija. Vrsta sinhronizacije, koja ostvaruje neki dodatan uslov u pogledu izvršavanja kritičnih sekcija, naziva se uslovna sinhronizacija.

9. Koje vrste sinhronizacije postoje?

Postoje sinhronizacija zadužena za ostvarenje međusobne isključivosti i uslovna sinhronizacija.

10. Šta je atomski region?

Atomski region je kritična sekcija u kojoj su onemogućeni prekidi, samim tim i obrada prekida, pa ni štetna preplitanja.

11. Šta sužava primenu atomskih regiona?

Primenu atomskih regiona sužava zahtev da oni budu što kraći, jer je činjenica da oni odlažu obradu novih prekida i usporavaju reakciju procesora na vanjske događaje.

12. Čemu služi propusnica?

Propusnice i isključivi regioni se zasnivaju na zaustavljanju aktivnosti niti, kada ona pokuša da uđe u kritičnu sekciju deljene promenljive kojoj već pristupa neka druga nit. Svaka deljena promenljiva

poseduje jednu propusnicu za ulazak u njene kritične sekcije. Propusnica može biti slobodna ili zauzeta i bez nje nije moguć ulazak u kritičnu sekciju deljene promenljive.

13. Šta se dešava sa niti koja zatraži, a ne dobije propusnicu?

Nit, koja zatraži, a ne dobije propusnicu, zaustavlja svoju aktivnost i prelazi u stanje čeka.

14. Šta se dešava kada nit vrati propusnicu?

Kada nit vrati propusnicu, ona se dodeljuje nekoj drugoj niti, koja tada iz stanja čeka, prelazi u stanje spreman. U kritičnu sekciju ulazi tek kada postane aktivna, tj. kada se procesor preključi na nju.

15. Kako se štiti konzistentnost propusnica?

Konzistentnost propusnica štiti se pomoću atomskih regiona.

16. Šta je isključivi region?

Isključivi region je ona kritična sekcija koja međusobnu isključivost ostvaruje korišćenjem propusnica.

17. Šta uvode poželjne osobine konkurentnih programa?

Poželjne osobine konkurentnih programa uvode tvrdnju isključivanja nepoželjnog i tvrdnju usključivanja poželjnog.

18. Po čemu se konkurentno programiranje razlikuje od sekvencijalnog?

Konkurentno programiranje se razlikuje od sekvencijalnog po rukovanju nitima i deljenim promenljivama.

19. Koje prednosti ima konkurentna biblioteka u odnosu na konkurentni programski jezik?

Korišćenjem konkurentne biblioteke izbegavaju se aktivnosti vezane za definisanje sintakse i semantike programskog jezika, kao i aktivnosti vezane za zahvate na kompajleru. Njena dodatna prednost je što omogućuje da se za konkurentno programiranje koristi već postojeći, znači poznat programski jezik.

20. Kako se opisuju niti?

Niti se opisuju funkcijom, čiju adresu kasnije prosleđujemo kao argument poziva konstruktora klase thread.

21. Kako se kreiraju niti?

Niti se kreiraju pomoću konstruktora klase thread. Nakon kreiranja niti, neophodno je odrediti njen odnos prema funkciji main(). Da ne bi došlo do prevremenog kraja aktivnosti kreirane niti, koristi se operacija join(), koja zaustavlja aktivnost svog pozivaoca, sve dok se ne završi aktivnost niti na koju se odnosi ova operacija. Pomoću operacije detach() saopštava se da regularan kraj aktivnosti niti, na koju se ova operacija odnosi, može da nastupi i kao posledica kraja aktivnosti procesa, kome dotična nit pripada.

22. Kada se zauzima propusnica deljene promenljive?

Propusnica deljene promenljive se zauzima na početku kritične sekcije (pre obavljanja izlaznih i ulaznih operacija niti).

23. Kada se oslobađa propusnica deljene promenljive?

Propusnica deljene promenljive se oslobađa na kraju kritične sekcije (posle obavljanja izlaznih i ulaznih operacija niti).

24. Kakvu ulogu ima klasa mutex?

Klasa mutex omogućava međusobnu isključivost krtičnih sekcija, upotrebom propusnice. Objekat ove klase predstavlja propusnicu.

25. Kakve operacije sadrži klasa mutex?

Klasa mutex sadrži operacije lock() i unlock(). Operacija lock() omogućuje zauzimanje propusnice, a operacija unlock() oslobađanje propusnice.

26. Kakvu ulogu ima klasa unique_lock?

Klasa unique_lock ima isti efekat kao pozivanje operacija klase mutex, lock() na početku i unlock() na kraju kritične sekcije. Razlog za to je što konstruktor klase unique_lock poziva operaciju lock(), a destruktor operaciju unlock().

27. Kakve operacije sadrži klasa unique_lock?

Klasa unique_lock sadrži operaciju lock() i operaciju unlock(). Konstruktor ove klase poziva operaciju lock(), a destruktor operaciju unlock().

28. Kakvu ulogu ima klasa condition_variable?

Klasa conditiona_variable služi za ostvarenje uslovne sinronizacije.

29. Kakve operacije sadrži klasa condition_variable?

Klasa condition_variable sadrži operacije wait() i notify_one(). Obe se koriste u isključivom regionu. Operacija wait() omogućuje zaustavljanje aktivnosti niti, koja pozove ovu operaciju, dok se ne ispuni dati uslov. Operacija notify_one() omogućuje objavu ispunjenja datog uslova, radi nastavka aktivnosti jedne od niti koje očekuju ispunjenje dotičnog uslova.

30. U pozivu koje od operacija klase condition_variable se vraća propusnica?

Propusnica se vraća u pozivu operacije wait().

31. Koje vrste razmene poruka postoje?

Postoje sinhrona i asinhrona razmena poruka. Sinhrona razmena poruka se još naziva i randevu.

32. U čemu se razlikuju sinhrona i asinhrona razmena poruka?

Asinhrona razmena poruka je razmena u kojoj se aktivnost pošiljaoca zaustavlja pri slanju poruka samo kada je komunikacioni kanal pun, dok se aktivnost primaoca zaustavlja pri prijemu poruka samo kada je ovaj kanal prazan. Kod sihrone razmene poruka se uvek zaustavlja aktivnost niti koja prva započne razmenu poruka, bez obzira da li se radi o pošiljaocu ili primaocu. Aktivnost ove niti ostaje zaustavljena dok i druga nit ne započne razmenu poruka. Sinhrona razmena poruka dozvoljava da se u komunikacionom kanalu ne čuva poruka, već samo njena adresa. Tada primalac može direktno preuzeti poruku od pošiljaoca, čime se izbegava potreba da se poruka prepisuje u komunikacioni kanal, što je neizbežno kod asinhrone razmene poruka.

33. Šta omogućuje funkcija sleep_for()?

Funkcija sleep_for() omogućuje uspavljivanje niti.

34. Po kojim ciljevima se konkurentno programiranje razlikuje od sekvencijalnog?

Opisivanje obrada podataka je jedini cilj sekvencijalnog, a osnovni cilj konkurentnog programiranja. Bolje iskorišćenje računara i njegovo čvršće sprezanje sa okolinom su dodatni ciljevi konkurentnog programiranja, po kojima se ono razlikuje od sekvencijalnog programiranja.

35. Zašto operacija condition_variable::wait() predstavlja prikriveni kraj isključivog regiona?

Operacija condition_variable::wait() predstavlja prikriveni kraj isključivog regiona, jer ona vraća propusnicu i izaziva preključivanje na nivou niti. Ova nit može da dobije vraćenu propusnicu i da uđe u neki od isključivih regiona iste deljene promenljive.

36. Šta je mrtva petlja?

Mrtva petlja je pojava međuzavisnosti niti, koja dovodi do trajnog zaustavljanja aktivnosti niti, a to ima za posledicu da izvršavanje konkurentnog programa nema kraja.

Sinhronizacija pomoću semafora

1. Šta karakteriše semafor?

Semafor je sredstvo za sinhronizaciju niti (reguliše prolaz niti kroz kritičnu sekciju) i zasniva se ideji saobraćajnog semafora, koji reguliše ulazak vozova u stanicu sa jednim kolosekom.

2. Koje operacije su vezane za semafor?

Za semafor su vezane operacije stop() i resume().

3. Kako semafor obezbeđuje sinhronizaciju međusobne isključivosti?

Sinhronizacija niti, koju omogućuje semafor, zasniva se na zaustavljanju aktivnosti niti, kao i na omogućavanju nastavljanja njihove aktivnosti. Ulazak niti u kritičnu sekciju zavisi od stanja semafora. Kada stanje semafora dozvoli ulazak niti u kritičnu sekciju, pri ulasku se semafor prevodi u stanje koje onemogućuje ulazak druge niti u kritičnu sekciju. Ako se takva nit pojavi, njena aktivnost se zaustavlja pred kritičnom sekcijom. Pri izasku niti iz kritične sekcije, semafor se prevodi u stanje koje dozvoljava novi ulazak u kritičnu sekciju i ujedno omogućuje nastavak aktivnosti niti koja najduže čeka na ulaz u kritičnu sekciju (ako takva nit postoji).

4. Kako se obično implementira semafor?

Semafori se obično implementiraju u okviru operativnog sistema i tada se njihova implementacija obično zasniva na (kratkotrajnom) onemogućenju prekida.

5. U čemu se semafori razlikuju od isključivih regiona?

Semafori i isključivi regioni predstavljaju dva različita pristupa sinhronizaciji. Isključivi regioni su prilagođeni objektno orijentisanom programiranju, dok su semafori prilagođeni procedurnom programiranju.

6. Koji semafori postoje?

Postoje binarni i generalni semafori. Postoji i posebna vrsta binarnog semafora, raspodeljeni binarni semafor.

7. Šta karakteriše binarni semafor?

Binarni semafor je semafor čije stanje ne može preći vrednost 1.

8. Šta karakteriše raspodeljeni binarni semafor?

Raspodeljeni binarni semafor se realizuje pomoću više binarnih semafora, za koje važi ograničenje da suma njihovih stanja ne može preći vrednost 1.

9. Šta karakteriše generalni semafor?

Genearlni semafor je semafor čije stanje može sadržati vrednost veću od 1.

10. Šta omogućuje raspodeljeni binarni semafor?

Pomoću raspodeljenog binarnog semafora se ostvaruje uslovna sinhronizacija, tako što se na ulazu u svaku kritičnu sekciju poziva operacija stop() jednog od njegovih binarnih semafora, a na izlazu iz nje operacija resume() tog ili nekog od preostalih binarnih semafora.

11. Šta omogućuje binarni semafor?

On omogućuje ostavarenje sinhronizacije međusobne isključivosti, ako se njegovo stanje inicijalizuje na vrednost 1. Tada su njegove operacije stop() i resume() slične operacijama lock() i unlock() klase

mutex. Ako se stanje binarnog semafora inicijalizuje na 0, tada su njegove operacija stop () i resume() slične operacijama wait() i notify_one() klase condition_variable.

12. Šta omogućuje generalni semafor?

Generalni semafor omogućuje ostvarenje uslovne sinhronizacije prilikom rukovanja resursima. Pozitivno stanje generalnog semafora može predstavljati broj slobodnih primeraka nekog resursa. Zahvaljujući tome, zauzimanje primeraka resursa se može opisati pomoću operacije stop(), a njegovo oslobađanje pomoću operacije resume() generalnog semafora.

13. Koje su prednosti i mane semafora?

Prednosti semafora su u tome što su jednostavni i efikasni, a mana je što raspodeljeni binarni semafori nisu baš najpodesnije sredstvo za opisivanje uslovne sinhronizacije.

Izvedba CppTss-a

1. Šta obuhvata izvedba CppTss-a?

Izvedba CppTss-a uključuje neophodne delove operativnog sistema, koji su potrebni za samostalno izvršavanje konkurentnog programa. Ona obuhvata ulazno-izlazne module, izvršioca (kernel) i virtuelnu mašinu.

2. Koja klasa omogućuje stvaranje atomskih regiona?

Stvaranje atomskih region omogućuje klasa Atomic_region. Njen konstruktor onemogućuje prekide, a destruktor poništava akciju konstruktora.

3. Koju klasu nasleđuju klase koje opisuju ponašanje drajvera?

Klase koje opisuju ponašanje drajvera nasleđuju klasu Driver.

4. Šta omogućuje operacija start_interrupt_handling() klase Driver?

Operacija start_interrupt_handling() klase Driver omogućuje smeštanje adrese obrađivača prekida u tabelu prekida. Prvi argument poziva ove operacije predstavlja broj vektora prekida, a drugi adresu obrađivača prekida.

5. Koje operacije sadrži klasa Event?

Klasa Event sadrži operacije expect() i signal().

6. Šta omogućuje operacija expect() klase Event?

Opeacija expect() klase Event omogućuje zaustavljanje aktivnosti niti, koja pozove ovu opraciju, dok se ne desi vanjski događaj koga reprezentuje objekat klase Event. Tome prethode prevođenje ove niti u stanje “čeka” i preključivanje procesora na spremnu nit. Operacija except() poziva se samo iz atomskog regiona.

7. Šta omogućuje operacija signal() klase Event?

Operacija signal() klase Event omogućuje objavu dešavanja nekog vanjskog događaja, radi nastavka aktivnosti niti koja (najduže) očekuje dešavanje dotičnog događaja. Operaciju signal() ima smisla pozivati samo iz obrađivača prekida, jer jedino oni opisuju reakciju na dešavanje vanjskih događaja. Izvršavanje operacije signal() prevodi nit, koja je dočekala dešavanje vanjskog događaja, u stanje “spremna”. Operacija signal() poziva se samo jednom i to na kraju obrade prekida.

1. Na koje drajvere se oslanjaju ulazno-izlazni moduli CppTss-a?

U ulazno-izlazne module CppTss-a spadaju znakovni i blokovski ulazno-izlazni moduli. Znakovni ulazno-izlazni modul se oslanja na drajver ekrana i drajver tastature. Blokovski ulazno-izlazni modul se oslanja na drajver diska.

2. Šta se dešava u obradi prekida ekrana?

U obradi prekida ekrana objavljuje se da je prikaz prethodnog znaka završen.

3. Do čega dovodi pokušaj niti da prikaže novi znak dok kontroler ekrana prikazuje prethodni znak?

Pokušaj niti da prikaže novi znak dok kontroler ekrana prikazuje prethodni znak, zaustavlja aktivnost niti.

4. Šta se dešava u obradi prekida tastature?

U obradi prekida tastature kod znaka se preuzima iz registra podataka i smešta u cirkularni bafer, ako on nije pun i zatim se objavljuje da je kod znaka preuzet.

5. Do čega dovodi pokušaj niti da preuzme znak kada je cirkularni bafer drajvera tastature prazan?

Pokušaj niti da preuzme znak kada je cirkularni bafer drajvera tastature prazan zaustavlja njenu aktivnost.

6. Šta se desi kada se napuni cirkularni bafer drajvera tastature?

Kada se napuni cirkularni bafer drajvera tastature, nit preuzima znak. Kada se cirkularni bafer popuni do kraja, polje first_empty klase Keyboard_driver (koje sadrži indekse cirkularnog bafera) se anulira.

7. Koje klase nasleđuje klasa Terminal_out?

Klasa Terminal_out nasleđuje klasu mutex.

8. Koje klase nasleđuje klasa Terminal_in?

Klasa Terminal_in nasleđuje klasu mutex.

9. Šta rade operacije klase Terminal_out za prikaz brojeva?

Operacije klase Terminal_out za prikaz brojeva omogućuju formatiranje prikazivanog podatka (njegovo pretvaranje u niz znakova). One koriste operaciju string_put() klase Terminal_out za prikaz niza znakova.

10. Šta rade operacije klase Terminal_in za preuzimanje brojeva?

Operacija klase Terminal_in za preuzimanje brojeva, string_get(), poziva opereaciju edit() klase Terminal_in, koja je zadužena za eho znakova na ekranu i njihovo primitivno editiranje. Preostale operacije klase Terminal_in koriste operaciju string_get() za preuzimanje niza znakova, koji odgovaraju raznim tipovima podataka. One zatim konvertuju te znakove u odgovarajući tip podataka.

11. Šta zaključava operacija Terminal_in::string_get()?

Operacija Terminal_in::string_get() zaključava tastaturu do kraja izvršavanja, i ekran, kada počne eho znakova, preuzetih u ovoj operaciji.

12. Šta otključava operacija Terminal_in::string_get()?

Operacija Terminal_in::string_get() otključava tastaturu na kraju izvršavanja, i ekran, nakon eha znakova i njihovog primitivnog editiranja.

13. Šta zaključava operacija Terminal_out::string_put()?

Operacija Terminal_out::string_put() zaključava ekran.

14. Šta otključava operacija Terminal_out::string_put()?

Operacija Terminal_out::string_put() otključava ekran.

15. Šta obuhvata primitivno editiranje koje podržava klasa Terminal_in?

Primitivno editiranje, koje podržava klasa Terminal_in, obuhvata pozivanje operacije edit() klase Terminal_in, koja je zadužena za eho znakova na ekranu.

16. Šta se desi u obradi prekida diska?

U obradi prekida diska objavljuje se kraj prenosa bloka.

17. Operacije koje klase ulazno-izlaznih modula CppTss-a vraćaju poruku greške?

Operacije klasa Disk_driver i Disk vraćaju poruku greške (vraćaju -1 kao error, ako je broj bloka veći nego što je moguće).

Virtuelna mašina CppTss-a

1. Koji zadatak ima virtuelna mašina CppTss-a?

Virtuelna mašina za potrebe ostatka biblioteke CppTss:

- emulira kontrolere tastature, ekrana i diska,
- emulira mehanizam prekida,
- podržava okončanje izvršavanja konkurentnog programa,
- podržava rukovanje pojedinim bitima memorijskih lokacija,
- podržava rukovanje numeričkim koprocesorom
- podržava rukovanje stekom

2. Šta omogućuje emulacija hardvera biblioteci CppTss?

Emulacija hardvera omogućuje da konkurentni program ne pristupa stvarnom, nego emuliranom hardveru, pa se izvršava kao običan (neprivilegovan) Linux proces. Zato, u slučaju grešaka, on ne može ugroziti funkcionisanje operativnog sistema u okviru koga se izvršava.

3. Na čemu se zasniva emulacija mehanizma prekida?

Emulacija mehanizma prekida se zasniva na:

- uvođenju (emulirane) tabele prekida
- uvođenju (emuliranog) bita prekida i
- obezbeđenju nezavisnosti (asinhronosti) između prekida i izvršavanja konkurentnog programa

4. Šta važi za mehanizam Linux signala?

Na mehanizmu Linux signala zasniva se nezavisnost (emuliranih) prekida od izvršavanja konkurentnog programa. On omogućuje da se na pojavu signala reaguje izvršavanjem odabrane funkcije, koja se naziva obrađivač signala. Signali su unapred određeni, a svaki od njih je pridružen jednoj vrsti događaja. Kada se takav događaj desi, mehanizam signala zaustavi izvršavanje programa, radi pokretanja izvršavanja odgovarajućeg obrađivača prekida. Nakon obrade dotičnog signala, moguć je nastavak zaustavljenog izvršavanja programa.

5. Šta omogućuje klasa Linux_signals?

Klasa Linux_signals opisuje reakciju na Linux signale.

6. Šta omogućuje klasa Linux_timer?

Klasa Linux_timer određuje učestanost dešavanja signala SIGVTALRM (pridružen događaju isticanja zadatog vremenskog intervala).

7. Šta podržava klasa Interrupt?

Klasa Interrupt:

- uvodi (emuliranu) klasu prekida,
- omogućuje registrovanje odlaganja obrade (emuliranog) prekida.
- reguliše redosled pozivanja obrađivača pojedinih (emuliranih) prekida.

8. Koja funkcija pokreće odloženu obradu (emuliranih) prekida?

Funkcija ad__restore_interrupts() pokreće odloženu obradu (emuliranih) prekida.

9. Šta omogućuje klasa Keyboard_controller?

Klasa Keyboard_controller opisuje kontroler tastature. Njeno polje data_reg predstavlja registar podataka kontrolera, a njena operacija input() opisuje ponašanje kontrolera tastature. Ako postoji znak za preuzimanje, on se preuzima u okviru operacije input() posredstvom odgovarajućeg Linux sistemskog poziva. Ujedno se poziva obrađivač prekida tastature, posredstvom operacije handler(), klase Interrupt, koja emulira tabelu prekida.

10. Šta omogućuje klasa Display_controller?

Klasa Display_controller opisuje kontroler ekrana. Njena polja data_reg i status_reg predstavljaju registre podataka i stanja kontrolera, a njena operacija output() opisuje ponašanje kontrolera ekrana. Ako postoji znak za prikazivanje, on se prikazuje u okviru operacije output(), posredstvom odgovarajućeg Linux sistemskog poziva. Ujedno se poziva obrađivač prekida ekrana posredstvom operacije handler(), klase Interrupt, koja emulira tabelu prekida.

11. Šta omogućuje klasa Disk_controller?

Klasa Disk_controller opisuje ponašanje kontrolera diska. Njena polja operation_reg, buffer_reg, block_reg i status_reg odgovaraju registrima smera prenosa, bafera, bloka i stanja kontrolera, a njena operacija transfer() opisuje ponašanje kontrolera diska. Konstruktor klase Disk_controller koristi sistemski poziv calloc() radi zauzimanja radne memorije, u kojoj se čuvaju blokovi emuliranog diska. Inercija diska se emulira brojanjem poziva operacije transfer(). Kada broj poziva ove operacija postane jednak procenjenom broju vremenskih jedinica potrebnom za prenos blooka, tada se obavi prenos bloka u okviru ove operacije i ujedno se pozove obrađivač prekida diska posredstvom operacije handler(), klase Interrupt, koja emulira tabelu prekida.

12. Šta omogućuje klasa Linux_terminal?

Klasa Linux_terminal omogućuje da se (za potrebe emulacija) isključi eho Linux terminala, prevede Linux terminal u režim rada bez linijskog editiranja i obezbedi da poziv čitanja znaka sa terminala bude neblokirajući.

13. Koja klasa omogućuje prevođenje Linux terminala u režim rada bez linijskog editiranja?

Prevođenje Linux terminala u režim rada bez linijskog editiranja omogućuje klasa Linux_terminal.

14. Koja klasa omogućuje isključivanje eha Linux terminala?

Isključivanje eha Linux terminala omogućuje klasa Linux_terminal.

15. Koja klasa obezbeđuje da čitanje znakova sa Linux terminala bude neblokirajuće?

Klasa Linux_terminal omogućuje da čitanje znakova sa Linux terminala bude neblokirajuće.

16. Kojim Linux sistemskim pozivom se okončava izvršavanje konkurentnog programa?

Izvršavanje konkurentnog programa okončava se sistemskim pozivom exit().

17. Šta omogućuje rukovanje pojedinim bitima memorijskih lokacija?

Rukovanje pojedinim bitima memorijskih lokacija omogućuju asemblerske naredbe za dobijanje indeksa najznačajnijeg postavljenog bita u reči (bsr), za postavljanje datog bita reči (bts) i za njegovo čišćenje (btr). Funkcije ad__get_index_of_most_significiant_set_bit(), ad_set_bit() i ad_clear_bit() posreduju u korišćenju ovih asemblerskih naredbi.

18. Šta omogućuje rukovanje numeričkim koprocesorom?

Rukovanje koprocesorom omogućuje klasa I387_npx. Ona omogućuje da se pripremi i preuzme inicijalni sadržaj registara numeričkog koprocesora. Njen konstruktor inicijalizuje registre numeričkog

koprocesora i smešta njihov inicijalni sadržaj u polje initial_contex ove klase. To omogućuju asemblerske naredbe fninit i fnsave.

19. Šta omogućuje rukovanje stekom?

Rukovanje stekom omogućuje funkcija ad__stack_init(). Ona na stek smesti povratnu adresu funkcije koja opisuje stvaranu nit (destroy()) i povratnu adresu funkcije preključivanja (thread_function()), kao i kontekst niti na koju se procesor prvi put preključuje. Funkcija ad__stack_swap() ima ulogu funkcije preključivanja. Funkcija preljučivanja prvo na stek aktivne niti smesti njen kontekst, zatim ona kao važeći stek postavi stek novoaktivirane niti i na kraju sa ovog steka preuzme kontekst novoaktivirane niti.

20. Šta obuhvata kontekst niti?

Kontekst niti obuhvata:

- pokazivač prethodnog frejma,
- početno stanje (emuliranog) bita prekida,
- početno stanje status (flag) registra procesora i
- početni sadržaj registara procesora.

1. Koje izuzetke podržava klasa Failure?

Klasa Failure podržava izuzetke nastale u toku izvršavanja konkurentnog programa.

2. Šta omogućuje klasa List_link?

Klasa List_link omogućuje obrazovanje dvosmernih listi pomoću polja left i right. Operacije ove klase omogućuju preuzimanje vrednosti ovih polja, uvezivanje novog elementa na kraj liste, izvezivanje elementa sa početka liste, proveru da li u listi ima uvezanih elemenata, odnosno da li je lista prazna.

3. Šta omogućuje klasa Permit?

Klasa Permit opisuje rukovanje propusnicama. Njene operacije omogućuju:

- proveru da li je propusnica isključive promenljive zauzeta,
- zauzimanje ove propusnice,
- njeno oslobađanje i
- rukovanje listama deskriptora niti.

4. Šta sadrže objekti klase Permit?

Objektni klase Permit sadrže:

- polje free, koje čuva stanje propusnice,
- pokazivačko polje previous, koje omogućuje obrazovanje liste propusnica,
- polje admission_list, oko kog se obrazuje liste deskriptora niti koje čekaju na propusniicu da bi ušle u isključivi region i
- polje fulfiled_list, oko kog se obrazuje lista deskriptora niti koje čekaju na propusnicu nakon ispunjenja uslova.

5. Šta je uslov konzistentnosti propusnice?

Uslov konzistentnosti propusnice je da poziv operacije provere da li je propusnica zauzeta i poziv operacije zauzimanja propusnice moraju biti u istom atomskom regionu.

6. Šta sadrže objekti klase Descriptor?

Objekti klase Descriptor sadrže:

- polje stack_top, koje sadrži pokazivač (adresu) vrha steka niti,
- polje priority, koje sadrži prioritet niti,
- polje last, koje sadrži adresu poslednje dobijene propusnice,
- polje tag, namenjeno za smeštanje priveska deskriptora niti.

7. Koju klasu nasleđuje klasa Descriptor?

Klasa Descriptor nasleđuje klasu List_link, da bi bilo moguće deskriptore niti uvezivati u liste.

8. Šta karakteriše nultu nit?

Nulta nit nalazi se u nultoj listi spremnih niti i ima prioritet 0 (kao i nulta lista). Ona angažuje procesor (aktivna je) kada nema drugih spremnih niti. Kada je nulta nit u stanju spremna, tada je njen deskriptor uvezan u nultu listu spremnih niti. Nulta nit ne može biti u stanju čeka.

9. Šta karakteriše klasu Ready_list?

Klasa Ready_list omogućuje rukovanje spremnim nitima. Svakom od prioriteta niti se dodeljuje posebna lista spremnih niti i podrazumeva se da se deskriptor spremne niti uvek uvezuje na kraj liste

spremnih niti, koja odgovara prioritetu dotične niti. Takođe se podrazumeva da se deskriptor spremne niti uvek izvezuje sa početka odabranje liste spremnih niti.

10. Koje operacije omogućuju rukovanje multi-listom?

Rukovanje multi-listom omogućuju operacije klase Ready:

- highest() - dobijanje prioriteta najprioritetnije neprazne liste spremnih niti
- insert() - uvezivanje deskriptora niti na kraj odgovarajuće liste spremnih niti,
- extract() - izvezivanje deskriptora niti sa početka najprioritetnije neprazne liste spremnih niti
- higher_than() - poređenje prioriteta najprioritetnije neprazne liste spremnih niti, sa prioritetom zadate niti.

11. Koju klasu nasleđuje klasa Kernel?

Klasa Kernel nasleđuje klasu Descriptor, da bi jedini objekat klase Kernel reprezentovao deskriptor main niti.

12. Šta reprezentuje jedini objekat klase Kernel?

Jedini objekat klase Kernel reprezentuje deskriptor main() niti. Konstruktor ove klase proglašava aktivnom main() nit.

13. Šta omogućuje klasa Kernel?

Klasa Kernel omogućuje rukovanje procesorom, a ono se svodi na preključivanje procesora sa jedne niti na drugu. U nadležnosti klase Kernel je i podrška viših slojeva iz hijerarhijske strukture CppTss izvršioca.

14. Kada dolazi do raspoređivanja u okviru klase Kernel?

Do raspoređivanja dolazi:

- kada se pojavi spremna niti sa višim prioritetom od aktivne (operacija schedule()) i
- na kraju kvantuma (operacija periodic_schedule())

15. Koju klasu nasleđuje klasa mutex?

Klasa mutex nasleđuje klasu Permit.

16. Šta omogućuje klasa Driver?

Klasa Driver omogućuje smeštanje adrese obrađivača prekida u tabelu prekida. Pored toga, ona uvodi definiciju klase Event, koja omogućuje zaustavljanje aktivnosti niti do dešavanja događaja i objavu dešavanja događaja.

17. Šta registruje Timer_driver::interrupt_handler()?

Timer_driver::interrupt_handler() registruje otkucaje sata, tj.:

- brojanje otkucaja sata, radi praćenja sistemskog vremena,
- odbrojavanje otkucaja sata preostalih do kraja kvantuma aktivne niti kao i
- odbrojavanje otkucaja sata preostalih do buđenja uspavane niti.

18. Šta omogućuje Timer_driver::interrupt_handler()?

Kada broj otkucaja, preostalih do isticanja kvantuma aktivne niti, padne na nulu, potrebno je pokretnuti periodično raspoređivanje. Takođe, kada broj otkucaja, preostalih do buđenja uspavane niti, padne na nulu, potrebno je probuditi sve niti za koje je nastupio trenutak buđenja. Sve ovo omogućuje operacija Timer_driver::interrupt_handler().

19. Šta karakteriše odsečke slobodne radne memorije?

Slobodnu radnu memoriju obrazuje celi broj jedinica sastavljenih od UNIT bajta. Rukovanje slobodnom radnom memorijom podrazumeva da se uvek zauzima (odnosno oslobađa) celi broj ovih jedinica. Zauzimanja (oslobađanja) ovakvih zona slobodne radne memorije uzrokuju iscepkanost slobodne radne memorije u odsečke, zbog čega se odsečci uvezuju u jednosmernu listu.

20. U kom redosledu su uvezani odsečci slobodne radne memorije?

Odsečci radne memorije su uvezani u jednosmernu listu, uređenu u rastućem redosledu njihovih početnih adresa. Početak svakog odsečka sadrži svoju veličinu (u jedinicama od po UNIT bajta) i pokazivač narednog odsečka.

21. Koju klasu nasleđuje klasa Memory_fragment?

Klasa Memory_fragment nasleđuje klasu mutex.

22. Šta važi za konzistentnost operacija Memory_fragment::take() i Memory_fragment::free()?

Konzistentnost operacija Memory_fragment::take() i Memory_fragment::free() je očuvana. Ove operacije pozivaju globalni operatori new() i delete(), u kojima dolazi do zaključavanja i otključavanja jedinog objekta klase Memory_fragment.

23. Gde se sve štiti konzistentnost zaključavanjem i otključavanjem jedinog objekta klase Memory_fragment?

Konzistentnost zaključavanjem i otključavanjem jedinog objekta klase Memory_fragment štiti se u definicijama funkcija operator new() i operator delete().

24. Koje operacije sadrži klasa thread?

Klasa thread sadrži operacije join() i detach().

25. Šta koristi klasa thread za ostvarenje konzistentnosti?

Za ostvarenje konzistentnosti, klasa thread koristi objekat klase mutex.

26. Šta karakteriše deskriptore uspavanih niti?

Deskriptori uspavanih niti se uvezuju u listu u hronološkom redosledu buđenja niti. Svakom od ovih deskriptora je dodeljen privezak, koji pokazuje relativno vreme buđenja (relativni broj otkucaja do buđenja) u odnosu na prethodnika u listi. Ovakava lista zove se delta lista.

27. Ko budi uspavane niti?

Uspavane niti budi sistemska nit Wake_up_daemon, čije vreme buđenja je uvek jednako najranijem vremenu buđenja korisničkih niti. Ova sistemska nit poziva operaciju awake().

Svojstva datoteka

1. Na šta ukazuje ime datoteke?

Poželjno je da ime datoteke ukazuje na njen konkretan sadržaj i na vrstu njenog sadržaja (radi klasifikacije datoteka po njihovom sadržaju).

2. Od koliko delova se sastoji ime datoteke?

Imena datoteka su dvodelna. Prvi deo imena datoteke označava njen sadržaj, a drugi deo označava vrstu njenog sadržaja, odnosno njen tip. Ova dva dela imena datoteke obično razdvaja tačka. (npr. godina1.txt)

3. Od koliko delova se sastoji ime imenika?

Imena imenika su jednodelna, jer nema potrebe za klasifikacijom imenika.

4. Šta obuhvata rukovanje datotekom?

Rukovanje datotekom obuhvata rukovanje njenim sadržajem i njenim imenom.

5. Šta karakteriše hijerarhijsku organizaciju datoteka?

Razvrstavanjem datoteka u imenike nastaje hijerarhijska organizacija datoteka, u kojoj su na višem nivou hijerarhije imenici, a na nižem nivou se nalaze datoteke koje pripadaju navedenim imenicima. Ovakva hijerarhijska organizacija povlači za sobom i hijerarhijsko označavanje datoteka. Hijerarhijsku oznaku ili putanju (*path name*) datoteke obrazuju ime imenika kome datoteka pripada i ime datoteke. Ime imenika se završava znakom /. Hijerarhijska organizacija datoteka ima više nivoa, kada imenik pored datoteka obuhvata i druge imenike. Obuhvaćeni imenici se nalaze na nižem nivou hijerarhije. Na vrhu hijerarhijske organizacije datoteka se nalazi korenski imenik (root) koji obično nema ime. Hijerarhijska organizacija datoteka dozvoljava da postoje datoteke (imenici) sa istim imenima, pod uslovom da pripadaju raznim imenicima. Zahvaljujući hijerarhijskoj organizaciji datoteka, moguće je rukovanje skupovima datoteka. Npr. moguće je kopiranje celog imenika, odnosno kopiranje svih datoteka i imenika, koji mu pripadaju.

6. Šta važi za apsolutnu putanju?

Navođenje apsolutne putanje datoteke, sa svim prethodećim imenicima, potrebno je kad god je moguć nesporazum, zbog datoteka (imenika) sa istim imenima. Ali, ako postoji mogućnost određivanja nekog imenika kao radnog, tada se njegova putanja može podrazumevati i ne mora se navoditi.

7. Šta važi za relativnu putanju?

Radni imenik omogućuje korišćenje relativnih putanja. (ne piše se apsolutna putanja, već relativna putanja od radnog imenika)

8. Koje datoteke obrazuju sistem datoteka?

Datoteke koje pripadaju istoj hijerarhijskoj organizaciji obrazuju sistem datoteka.

9. Koja su prava pristupa datotekama?

Prava pristupa datotekama su pravo čitanja, pravo pisanja i pravo izvršavanja (za izvršne datoteke).

10. Koje kolone ima matrica zaštite?

Matrica zaštite ima tri kolone, po jednu za svaku kategoriju korisnika (vlasnik datoteke, saradnici, ostali korisnici).

11. Čemu je jednak broj redova matrice zaštite?

Broj redova matrice zaštite jednak je broju datoteka.

12. Gde se mogu čuvati prava pristupa matrice zaštite?

Prava pristupa matrice zaštite mogu se čuvati u deskriptorima datoteke (redovi matrice zaštite raspoređeni po deskriptorima raznih datoteka), ako su vezana za datoteke, a ako su prava pristupa matrice zaštite vezana za korisnike, elemente njenih kolona čuvaju pojedini korisnici.

13. Šta je potrebno za sprečavanje neovlašćenog menjanja matrice zaštite?

Za sprečavanje neovlašćenog menjanja matrice zaštite potrebno je znati za svaku datoteku ko je njen vlasnik, jer jedino on sme da zadaje i menja prava pristupa sebi, svojim saradnicima i ostalim korisnicima. Takođe, potrebno je i razlikovanje korisnika, da bi se među njima mogao prepoznati vlasnik datoteke. To se postiže predstavljanjem (*login*) korisnika.

14. Kada korisnici mogu posredno pristupiti spisku lozinki?

Korisnici mogu posredno pristupiti spisku lozinki jedino u dva slučaja, radi svog predstavljanja i radi izmene svoje lozinke.

15. Koju dužnost imaju administratori?

Sva druga rukovanja spiskovima imena i lozinki registrovanih korisnika (sem predstavljanja korisnika i promene lozinke), kao što su ubacivanja u ove spiskove novih parova (ime, lozinka) ili njihovo izbacivanje iz ovih spiskova, nalaze se u nadležnosti poverljive osobe, koja se naziva administrator. Zaštita datoteka zavisi od odgovornosti i poverljivosti administratora. Zbog prirode njegovog posla, ima smisla administratora izuzeti iz zaštite datoteka, s tim da on tada svoje nadležnosti mora vrlo oprezno koristiti.

16. Šta sadrži numerička oznaka korisnika?

Sadrži dva redna broja, prvi označava korisnika, a drugi grupu kojoj korisnik pripada.

17. Kakvu numeričku oznaku imaju saradnici vlasnika datoteke?

Saradnici vlasnika datoteke imaju isti redni broj grupe kao i vlasnik.

18. Kakvu numeričku oznaku imaju ostali korisnici?

Ostali korisnici imaju redni broj grupe različit od rednog broja grupe vlasnika.

19. Kada se obavlja provera prava pristupa datoteci?

Provera prava pristupa datoteci obavlja se samo pre prvog pristupa.

20. Čime se bavi sigurnost?

Sigurnost se odnosi na uspešnost zaštite od neovlašćenog korišćenja ne samo datoteka, nego i ostalih delova računara kojima upravlja operativni sistem. Sigurnost se bavi načinima prepoznavanja ili indentifikacije korisnika, kao i načinima provere njihovih prava pristupa.

Sloj operativnog sistema za rukovanje procesima

1. Šta omogućuju sistemske operacije za rukovanje procesima?

Sistemske operacije za rukovanje procesima omogućuju stvaranje procesa, uništavanje procesa, izmenu atributa procesa (npr. izmena radnog imenika procesa).

2. Šta obuhvata stvaranje procesa?

Stvaranje procesa obuhvata stvaranje njegove slike i deskriptora, kao i pokretanje njegove aktivnosti.

3. Šta obuhvata uništavanje procesa?

Uništavanje procesa obuhvata zaustavljanje njegove aktivnosti, kao i uništenje njegove slike i deskriptora.

4. Šta sadrži slika procesa?

Slika procesa sadrži izvršavane mašinske naredbe, promenljive i stek. Ona obuhvata niz lokacija radne memorije sa uzastopnim (logičkim) adresama.

5. Za šta se koristi slobodna radna memorija procesa?

Slobodna radna memorija procesa je na raspolaganju procesu za širenje (punjenje) steka, ali i za stvaranje dinamičkih promenljivih (taj deo slobodne radne memorije se naziva *heap*).

6. Koji atributi procesa postoje?

Atributi procesa nalaze se u deskriptoru procesa i karakterišu aktivnost procesa. Oni obuhvataju:

- stanje procesa (spreman, aktivan, čeka)
- sadržaje procesorskih registara (zatečene u njima pre poslednjeg preključivanja procesora sa procesa)
- numeričku oznaku vlasnika procesa,
- oznaku procesa stvaraoca,
- trenutak pokretanja aktivnosti procesa,
- ukupno trajanje aktivnosti procesa (ukupno vreme angažovanja procesora),
- podatke o slici procesa (njenoj veličini i položaju u radnoj i masovnoj memoriji),
- podatke o datotekama koje proces koristi,
- podatak o radnom imeniku procesa i
- razne podatke neophodne za rukovanje aktivnošću procesa (poput prioriteta procesa ili položaja sistemskog steka procesa, koga koristi operativni sistem u toku obavljanja sistemskih operacija)

7. Koje sistemske operacije za rukovanje procesima postoje?

Postoje operacije stvaranja procesa (*fork()* i *exec()*), operacije uništenja procesa (*exit()*), posebna operacija *wait()* (omogućuje zaustavljanje aktivnosti procesa stvaraoca i preključivanje procesora na stvarani proces).

8. Koji atributi se nasleđuju od procesa stvaraoca prilikom stvaranja procesa?

Prilikom stvaranja procesa od procesa stvaraoca nasleđuju se npr. numerička oznaka vlasnika procesa, podatak o radnom imeniku procesa ili njegov prioritet.

9. Koji atributi procesa nastanu prilikom njegovog stvaranja?

Prilikom stvaranja procesa nastaju npr. podaci o slici procesa.

10. U kojim stanjima može biti proces stvaralac nakon stvaranja novog procesa? ???

Kada se u okviru stvaranja procesa, stigne do pokretanja njegove aktivnosti, moguće je prekljućivanje procesora sa procesa stvaraoca na stvarani proces. To se desi ako je prioritet stvaranog procesa viši od prioriteta procesa stvaraoca. U tom slučaju proces stvaralac dospeva među spremne procese. Inače tamo dospeva stvarani proces. Znači za sad aktivan i spreman. Može biti i u stanju čeka?..

11. Šta je stepen multiprogramiranja?

Stepen multiprogramiranja je najveći mogući broj slika procesa, koje mogu istovremeno da postoje u radnoj memoriji.

12. Šta karakteriše kopiju slike procesa?

Kopija slike procesa se nalazi u masovnoj memoriji. U toku aktivnosti procesa, menja se samo deo njegove slike u radnoj memoriji, jer se menjaju samo vrednosti njegovih promenjivih i njegov stek, pa je prilikom izbacivanja slike procesa, potrebno samo njen izmenjeni deo prebaciti u kopiju slike u masovnoj memoriji. Ali, pri vraćanju slike u radnu memoriju prebacuje se cela njena kopija, da bi se u radnoj memoriji obnovila cela slika procesa. Vraćanje slike jednog procesa vezano je za uništavanje drugog procesa, jer se tada oslobađa prostor u radnoj memoriji. Podaci o kopiji slike procesa (koja nastaje istovremeno sa slikom procesa ili prilikom njenog prvog izbacivanja, a nestaje istovremeno sa slikom procesa) čuvaju se u deskriptoru procesa, zajedno sa podacima o slici procesa.

13. Koje raspoređivanje je vezano za zamenu slike procesa?

Za zamenu slike procesa vezano je dugoročno raspoređivanje, u okviru koga se odabira proces čija se slika izbacuje, kao i proces čija se slika ubacuje.

14. Šta karakteriše rukovanje nitima unutar operativnog sistema?

Rukovanje nitima može, ali i ne mora, biti u nadležnosti sloja za rukovanje procesima. Kada je rukovanje nitima povereno sloju za rukovanje procesima, tada operativni sistem nudi sistemske operacije za rukovanje nitima, koje omogućuju stvaranje, uništavanje i sinhronizaciju niti. U ovom slučaju, deskriptori i sistemski stek niti se nalaze u sistemskom prostoru, dok se sopstveni stek niti nalazi u korisničkom prostoru (unutar slike procesa).

15. Šta karakteriše rukovanje nitima van operativnog sistema?

Kada rukovanje nitima nije u nadležnosti operativnog sistema, brigu o nitima potpuno preuzima konkurentna biblioteka. Rukovanje nitima se odvija u korisničkom prostoru, u kome se nalaze i deskriptor niti, kao i stekovi niti. Osnovna prednost je efikasnost, jer su pozivi potprograma konkurentne biblioteke brži od poziva sistemskih operacija. Mana je što poziv blokirajuće sistemske operacije iz jedne niti dovodi do zaustavljanja aktivnosti procesa kome ta nit pripada. Na taj način se sprečava konkurentnost unutar procesa, jer zaustavljanje aktivnosti procesa sprečava aktivnost njegovih spremnih niti. Ovo značajno umanjuje praktičnu vrednost ovog pristupa.

Sistemski procesi

1. Šta karakteriše nulti proces?

Tipičan primer sistemskog procesa je nulti proces ili beskonačni proces, na koga se procesor prekljućuje, kada ne postoji drugi spreman proces. U toku aktivnosti nultog procesa izvršava se beskonačna petlja, što znači da je beskonačan proces uvek aktivan ili spreman (ne prelazi u stanje čeka). Njegov prioritet je niži od prioriteta svih ostalih procesa, a on postoji za sve vreme aktivnosti operativnog sistema.

2. Šta je karakteristično za proces dugoročni raspoređivač?

Dugoročni raspoređivač je sistemski proces koji se brine o zameni slika procesa (kada za to ima potrebe). On se periodično aktivira, radi pozivanja operacije za zamenu slika procesa. Nakon toga on se uspava (pozivom odgovarajuće sistemske operacije iz sloja za rukovanje kontrolerima), tj. njegova aktivnost se zaustavlja, dok ne nastupi trenutak za njegovo ponovno aktiviranje. Dugoročni raspoređivač postoji za sve vreme aktivnosti operativnog sistema.

3. Šta radi proces indentifikator?

Proces identifikator je sistemski proces koji podržava predstavljanje korisnika. Proces indentifikator opslužuje terminal, da bi posredstvom njega stupio u interakciju sa korisnikom u toku predstavljanja, radi preuzimanja njegovog imena i lozinke. Po preuzimanju imena i lozinke, on proverava njihovu ispravnost i, ako je prepoznao korisnika, tada stvara proces komunikator, koji nastavlja interakciju sa korisnikom. Pri tome, proces identifikator svoj terminal prepušta stvorenom procesu komunikatoru i zaustavlja svoju aktivnost, koja se nastavlja tek nakon završetka aktivnosti procesa komunikatora. Proces identifikator postoji za sve vreme aktivnosti operativnog sistema.

4. Ko stvara proces komunikator?

Proces identifikator stvara proces komunikator, kome prepušta svoj terminal i zaustavlja svoju aktivnost.

5. Šta sadrži slog datoteke lozinki?

Svaki slog datoteke lozinki sadrži:

- ime i lozinku korisnika,
- numeričku oznaku korisnika,
- putanju radnog imenika korisnika i
- putanju izvršne datoteke, sa inicijalnom slikom korisničkog procesa komunikatora.

6. Šta označava SUID (“switch user identification”)?

Pošto korisnici nemaju načina da pristupe datoteci lozinki, postavlja se pitanje kako omogućiti korisniku da sam izmeni svoju lozinku. Ako je administrator vlasnik izvršne datoteke sa inicijalnom slikom procesa za izmenu lozinki, dovoljno je naznačiti da on treba da bude i vlasnik procesa nastalog na osnovu ove izvršne datoteke (SUID). Zahvaljujući tome, vlasnik ovakvog procesa je administrator bez obzira koji je korisnik pokrenuo proces. U tom slučaju, nema smetnje da korisnik, koji izazove stvaranje procesa za izmenu lozinke, pristupi datoteci lozinki. Pri tome, proces za izmenu lozinki dozvoljava korisniku samo da izmeni sopstvenu lozinku, tako što od korisnika dobije ime, važeću i novu lozinku, pa ako su ime i važeća lozinka ispravni, on važeću lozinku zameni novom.

7. Šta je neophodno za podmetanje trojanskog konja?

Za podmetanje trojanskog konja (procesa koji opslužuje terminal, oponašajući proces indentifikator) narednom korisniku neophodno je da se korisnik (koji ga podmeće) ne odjavi sa sistema.

8. Šta karakteriše simetričnu kriptografiju?

Ako algoritam dekriptovanja direktno i jednoznačno sledi iz algoritma kriptovanja, a ključ dekiptovanja iz ključa kriptovanja, reč je o simetričnoj kriptografiji. Poznavanje ključa kriptovanja omogućuje i dekriptovanje, tako da ključ kriptovanja predstavlja tajnu kao i ključ dekriptovanja. Algoritmi kriptovanja i dekriptovanja ne predstavljaju tajnu, jer je praksa pokazala da se takva tajna ne može sačuvati. Simetrična kriptografija nije podesna za kriptovanje poruka. U praksi, simetrična kriptografija se zasniva na DES i IDEA pristupima.

9. Šta karakteriše asimetričnu kriptografiju?

U asimetričnoj kriptografiji, iz ključa kriptovanja ne može se odrediti ključ dekriptovanja, pa poznavanje ključa kriptovanja ne omogućuje dekriptovanje. Ključ kriptovanja se naziva javni ključ, jer je dostupan svima, a ključ dekriptovanja privatni ključ, jer je dostupan samo osobama ovlašćenim za dekriptovanje. Algoritmi kriptovanja su jednostavni, a njima odgovaraju komplikovani algoritmi dekriptovanja, pa je asimetrična kriptografija mnogo sporija od simetrične. U praksi, asimetrična kriptografija se zasniva na RSA algoritmu.

10. Na čemu se temelji tajnost kriptovanja?

Tajnost kriptovanja se zasniva na činjenici da (1) komplikovanost algoritma kriptovanja i dekriptovanja, (2) velika dužina ključeva kriptovanja i dekriptovanja, kao i (3) veliki broj ovih ključeva čine praktično neizvodljivim pokušaj da se dekriptuje kriptovani tekst probanjem, jednog po jednog, svih mogućih ključeva dekriptovanja.

Sloj operativnog sistema za rukovanje datotekama

1. Kako se predstavlja sadržaj datoteke?

Punu slobodu rukovanja podacima nudi predstava sadržaja datoteke kao niz bajta, pod uslovom da se ovaj niz može, po potrebi, produživati ili skraćivati, kao i da se bajtima iz niza može pristupati u proizvoljnom redosledu, korišćenjemn rednog broja bajta, za njegovu identifikaciju. Sadržaji datoteka nalaze se u blokovima masovne memorije.

2. Gde se javlja interna fragmentacija?

Pojava da poslednji blok datoteke nije popunjen do kraja naziva se interna fragmentacija. (znači, javlja se unutar bloka) Ona je važna, jer nepopunjeni poslednji blok datoteke predstavlja neupotrebljen deo masovne memorije.

3. Šta karakteriše kontinualne datoteke?

Kontinualne datoteke karakteriše to da se njihov sadržaj nalazi u susednim blokovima (čiji redni brojevi se uzastopni). Kod njih redni broj bloka sa traženim bajtom se određuje kao količnik rednog broja bajta i veličine bloka, izražene brojem bajta koji sadrži svaki blok. Ostatak deljenja ukazuje na relativni položaj bajta u bloku.

4. Koji oblik evidencije slobodnih blokova masovne memorije je podesan za kontinualne datoteke?

Evidencija slobodnih blokova masovne memorije treba da olakša pronalaženje dovoljno dugačkih nizova susednih blokova. Zato je podesna evidencija u obliku niza bita, u kome svaki bit odgovara jednom bloku i pokazuje da li je on zauzet (0) ili slobodan (1).

5. Šta je eksterna fragmentacija?

Eksterna fragmentacija je pojava iscepkanosti slobodnih blokova masovne memorije u kratke nizove susednih blokova. Ona otežava rukovanje kontinualnim datotekama. Nastaje kao rezultat višestrukog stvaranja i uništenja datoteka u slučajnom redosledu. Eksterna fragmentacija izaziva neupotrebljivost slobodnih blokova masovne memorije. Problem eksterne fragmentacije se može rešiti sabijanjem datoteka, tako da svi slobodni blokovi budu potisnuti iza datoteka i da tako obrazuju niz susednih blokova.

6. Šta karakteriše rasute datoteke?

Pošto upotrebnu vrednost kontinualnih datoteka smanjuju problem eksterne fragmentacije, potreba da se unapred zna veličina kontinualnih datoteka i teškoće sa njihovim produžavanjem, umesto njih se koriste rasute datoteke. Sadržaj rasutih datoteka smešten (rasut) je u nesusednim blokovima masovne memorije. Kod rasutih datoteka, redni brojevi bajta se preslikavaju u redne brojeve blokova pomoću posebne tabele pristupa.

7. Šta karakteriše tabelu pristupa?

Pomoću tabele pristupa se kod rasutih datoteka redni brojevi bajta preslikavaju u redne brojeve blokova. Veličina tabele pristupa ograničava dužinu rasutih datoteka. Tabele pristupa se čuvaju u blokovima masovne memorije.

8. Šta ulazi u sastav tabele pristupa?

Elementi tabele pristupa sadrže redne brojeve blokova. Indekse ovih elemenata određuje količnik rednog broja bajta i veličine bloka. Tabela pristupa se deli na odsečke. Početni odsečak nije veći od

bloka masovne memorije, uvek je prisutan i ima p elemenata. Dodatni odsečci jednaki su bloku masovne memorije, prisutni su samo kad su neophodni. Svaki dodatni odsečak može da sadrži b elemenata tabele pristupa (b > p). Dodatnih odsečaka ima ukupno $1 + b + b^2$.

Tabela pristupa zauzima:

- jedan blok (ili njegov deo) sa početnim odsečkom
- jedan blok sa dodatnim odsečkom
- blok prvog stepena indirekcije - poseban blok, zauzima se kada zatreba još isečaka, sadrži do b rednih brojeva blokova sa dodatnim odsečcima i u svakom od njih se nalazi b novih elementa.
- blok drugog stepena indirekcije - zazima se po potrebi, sadrži do b rednih brojeva blokova prvog stepena indirekcije, sa do b blokova sa dodatnim odsečcima, a u svakom od njih se nalazi b novih elemenata tabele pristupa

9. Kada rasuta datoteka ne zauzima više prostora na disku od kontinualne datoteke?

Rasuta datoteka ne zazima više prostora od kontinualne kada sadrži samo početni odsečak. (kontinualna datoteka zauzima minimalno jedan blok, a početni odsečak rasute nije veći od jednog bloka)

10. Koji oblik evidencije slobodnih blokova masovne memorije je podesan za rasute datoteke?

Za evidenciju slobodnih blokova masovne memorije kod rasute datoteke podesan je oblik liste slobodnih blokova. Slobodni blokovi, uvezani u ovu listu, sadže redne brojeve ostalih slobodnih blokova.

11. Kada dolazi do gubitka blokova prilikom produženja rasute datoteke?

Kada samo promenjena kopija bloka evidencije slobodnih blokova dospe u masovnu memoriju, blok isključen iz ove evidencije postaje izgubljen, jer njegov redni broj nije prisutan niti u ovoj evidenciji, a niti u tabeli pristupa neke od rasutih datoteka.

12. Kada dolazi do mogućnosti višestrukog nezavisnog korišćenja istog bloka prilikom produženja rasute datoteke?

Ako samo promenjena kopija bloka tabele pristupa dospe u masovnu memoriju, blok pridružen ovoj tabeli ostaje i dalje uključen u evidenciji slobodnih blokova, pa je moguće istovremeno uključivanje istog bloka u više različitih datoteka, čime se narušava njihova konzistentnost. Zato je potrebno uvek prebaciti u masovnu memoriju prvo promenjenu kopiju bloka evidencije slobodnih blokova, pa tek iza toga i promenjenu kopiju bloka tabele pristupa.

13. Kada pregled izmena ukazuje da je sistem datoteka u konzistentnom stanju?

Ako u pregledu izmena nije registrovan potpun opis nameravane izmene, tada izmena nije ni započeta, pa je sistem datoteka u konzistentnom stanju. Ako su u pregledu izmena registrovani potpuni opis nameravane izmene i njeno uspešno obavljanje, tada je sistem datoteka u konzistentnom stanju.

Kada je u pregledu izmena registrovan potpun opis nameravane izmene, ali nije registrovano njeno uspešno obavljanje, tada je sistem datoteka moguće vratiti u konzistentno stanje.

14. Kako se ubrzava pristup datoteci?

Pristup datoteci se ubrzava tako što se u radnoj memoriji zauzme prostor za više bafera, namenjenih za čuvanje kopija korišćenih blokova (koji će se koristiti u skorijoj budućnosti).

15. Od čega zavisi veličina bloka?

Na brzinu prebacivanja podataka između radne i masovne memorije važan uticaj ima i veličina bloka, jer što je blok veći , u proseku se potroši manje vremena na prebacivanje jednog bajta između radne i masovne memorije. Međutim što je blok veći, veća je i interna fragmentacija. Ta dva oprečna zahteva utiču na izbor veličine bloka, koja se kreće od 512 bajta do 8192 bajta.

16. Šta sadrži deskriptor kontinualne datoteke?

Kontinualne datoteke zahtevaju od sloja za rukovanje datotekama da za svaku datoteku vodi evidenciju ne samo o njenom imenu, nego i o rednom broju početnog bloka datoteke, kao i o dužini datoteke. Podatke o početnom bloku i dužini datoteke (izražena brojem bajta ili brojem blokova) čuva njen descriptor. Deskriptor kontinualne datoteke sadrži, pored atributa koji omogućuju preslikavanje rednih brojeva bajta u redne brojeve blokova, sadrži i:

- numeričku oznaku vlasnika datoteke,
- prava pristupa datoteci za njenog vlasnika,
- njegove saradnike i za ostale korisnike,
- podatak da li je datoteka zaključana ili ne,
- SUID podatak da li numerička oznaka vlasnika datoteke postaje numerička oznaka vlasnika procesa stvorenog na osnovu sadržaja datoteke (važi samo za izvršne datoteke),
- datum poslednje izmene datoteke.

17. Kako se rešava problem eksterne fragmentacije?

Problem eksterne fragmentacije se može rešiti sabijanjem datoteka, tako da svi slobodni blokovi budu potisnuti iza datoteka i da tako obrazuju niz susednih blokova. Mana ovog postupka je dugotrajnost.

18. Kako se ublažava problem produženja kontinualne datoteke?

Problem produženja kontinualne datoteke se ublažava ako se dozvoli da se kontinualna datoteka sastoji od više kontinualnih delova. Pri tome se za svaki od ovih delova u deskriptoru datoteke čuvaju podaci o rednom broju početnog bloka dotičnog dela i o njegovoj dužini. Ovakav pristup je pogodan za veoma dugačke datoteke (čuvanje zvučnog ili video zapisa).

19. Šta sadrži deskriptor rasute datoteke?

Deskriptor rasute datoteke sadrži početni odsečak tabele pristupa, redni broj njenog prvog dodatnog odsečka, redni broj bloka prvog stepena indirekcije i redni broj bloka drugog stepena indirekcije. On takođe sadrži i dužinu rasute datoteke. Deskriptor rasute datoteke sadrži i sve nabrojano kod deskriptora kontinualne datoteke.

20. Šta je imenik?

Datoteke se grupišu u skupove datoteka. Oni se nazivaju imenici. Za imenike su dovoljna jednodelna imena, jer nema potrebe za njihovom klasifikacijom.

21. Šta sadrže elementi tabela otvorenih datoteka?

Svaki element tabele otvorenih datoteka sadrži adresu kopije deskriptora odgovarajuće datoteke iz tabele deskriptora datoteka. Indeks elementa tabele otvorenih datoteka (u kome je adresa kopije deskriptora otvorene datoteke) predstavlja povratnu vrednost poziva sistemske operacije otvaranja datoteke. On se koristi kao argument u pozivima drugih sistemskih operacija sloja za rukovanje datotekama.

22. Koje sistemske operacije za rukovanje datotekama postoje?

Sistemske operacije za rukovanje datotekama su:

- operacije otvaranja datoteke (open()),
- operacije stvaranja datoteke (create()),
- operacija zatvaranja datoteke (close()),
- operacije zaključavanja/otključavanja datoteke (flock()),
- operacije čitanja datoteke (read()),
- operacije pisanja datoteke (write()),
- operacija izmene pozicije datoteke (seek()),
- operacije za izmenu atributa datoteke (u nju spada operacija za izmenu imena datoteke),
- operacija za stvaranje linka (dodatnog imena) datoteke (link()),
- operacija uništenja datoteka (unlink())

Postoje i sistemske operacije za rukovanje imenicama, to su mount() koja omogućuje spajanje dva sistema datoteka, i unmount(), koja razdvaja dva prethodno spojena sistema datoteka.

23. Koji su argumenti sistemskih operacija za rukovanje datotekama?

- open() - argument je putanja datoteke (kao dodatni argument može se javiti oznaka koja govori da li se otvara samo za čitanje ili pisanje/čitanje),
- create() - argument je putanja stvarane datoteke,
- close() - argument je indeks otvorene datoteke,
- flock() - argument je indeks otvorene datoteke,
- read() i write() - argument je indeks otvorene datoteke i broj bajta koji se čitaju ili pišu (pored toga, write() još sadrži adresu zone u koju se smeštaju pročitani bajti, a read() adresu zone u koju se smeštaju bajti za pisanje),
- seek() - argument je indeks otvorene datoteke i podatak o novoj poziciji,
- operacije za izmenu atributa imaju obavezne argumente putanju datoteke i novu vrednost menjanog atributa datoteke,
- link() - argument je putanja sa imenom datoteke i putanja sa linkom,
- unlink() - argument je putanja uništavane datoteke.

24. Šta karakteriše specijalne datoteke?

Specijalne datoteke predstavljaju pojedine ulazne ili izlazne uređaje. Služe za opisivanje tih ulaznih ili izlaznih uređaja. Dele se na znakovne (podržavaju sekvencijalno čitanje ili pisanje znakova) i blokovske (podržavaju čitanje ili pisanje blokova) specijalne datoteke.

25. Šta sadrži deskriptor specijalne datoteke?

Deskriptor specijalne datoteke obuhvata atribute, kao numerička oznaka vlasnika datoteke, prava pristupa datoteci za njenog vlasnika, za njegove saradnike i ostale korisnike ili podatak da li je datoteka zaključana ili nije. U deskriptorima se nalaze podaci o odgovarajućem drajveru i primerku uređaja, koga on opslužuje. Deskriptor sadrži i redni broj drajvera i redni broj uređaja.

26. Šta omogućuju blokovske specijalne datoteke?

Blokovske specijalne datoteke omogućuju direktne pristupe blokovima diska, što je važno, na primer, kod pronalaženja izgubljenih blokova, kod sabijanja datoteka, ili kod pripremanja disk jedinica za korišćenje.

27. Šta omogućuje rukovanje particijama?

Blokovska specijalna datoteka ne mora bude vezana za jedan celi disk, već se može odnositi na deo diska, koji se naziva prticija. Rukovanje particijama omogućava i formiranje logičkih disk jedinica, koje obuhvataju više particija na raznim fizičkim diskovima. To omogućava: (1) brži pristup blokovima logičke disk jedinice, (2) veću pouzdanost logičke disk jedinice ili (3) oboje.

Sloj operativnog sistema za rukovanje radnom memorijom

1. Kakav može biti logički adresni prostor?

Logički adresni prostor može biti:

- kontinualan,
- sastavljen od segmenata raznih veličina,
- sastavljen od stranica iste veličine i
- sastavljen od segmenata raznih veličina, koji se sastoje od stranica razne veličine.

2. Šta karakteriše kontinualni logički adresni prostor?

Kontinualni logički adresni prostor se sastoji od jednog niza uzastopnih logičkih adresa, koji počinje od logičke adrese 0. Veličina kontinualog adresnog prostora nadmašuje potrebe prosečnog procesa. Neophodno je poznavati najvišu ispravnu logičku adresu, koja se zova granična adresa. Logičke adrese procesa ne smeju biti veće od njegove granične adrese. Za kontinualni logički prostor se podrazumeva da se niz uzastopnih logičkih adresa preslikava u niz uzastopnih fizičkih adresa. Ova dva niza se razlikuju po tome što niz uzastopnih fizičkih adresa ne počinje od 0 nego od neke adrese koja se zove bazna adresa. Dodavanjem bazne adrese logičkoj adresi nastaje fizička adresa.

3. Šta karakteriše segmentirani logički adresni prostor?

Segmentirani logički adresni prostor se sastoji od više nizova uzastopnih logickih adresa (za svaki segment po jedan niz). Da bi se znalo kom segmentu pripada logička adresa, njeni značajniji biti sadrže adresu njenog segmenta, a manje značajni biti unutrašnju adresu u njenom segmentu. Svaki segment je kontinualan, pa ga karakterišu njegova granična i bazna adresa. Za translaciju logičke adrese u fizičku potrebna je tabela segmenata. Broj njenih elemenata određuje najveći broj segmenata, a svaki element sadrži graničnu i baznu adresu odgovarajućeg segmenta. Adresa segmenta indeksira element tabele segmenata.

4. Šta karakteriše stranični logički adresni prostor?

Stranični logički adresni prostor sastoji se od jednog niza uzastopnih logičkih adresa (podeljenog u stranice iste veličine). Da bi se znalo kojoj stranici pripada logička adresa, njeni značajniji biti sadrže adresu njene stranice, a manje značajni biti unutrašnju adresu u njenoj stranici. Svaka stranica je kontinualna, pa ima svojstva kontinualnog logičkog adresnog prostora. Veličina stranice je unapred poznata i jednaka stepenu broja dva. Svaku stranicu karakteriše samo njena bazna adresa. Za translaciju je potrebna tabela stranica. Broj njenih elemenata određuje najveći broj stranica, a svaki element sadrži baznu adresu odgovarajuće stranice. Adresa stranice indeksira element tabele stranica.

5. Šta karakteriše straničnо segmentirani logički prostor adresni prostor?

Straničnо segmentirani logički prostor adresni prostor se sastoji od više nizova uzastopnih logičkih adresa (za svaki segment po jedan niz podeljen u stranice iste veličine). Da bi se znalo kom segmentu pripada logička adresa, njeni najznačajniji biti sadrže adresu njenog segmenta, njeni srednji biti adresu njene stranice, a najmanje značajni biti unutrašnju adresu u njenoj stranici. Svaki segment je kontinualan, kao i stranica, s tim da je njena veličina unapred poznata i jednaka stepenu broja dva. Za translaciju je potrebna jedna tabela segmenata i više tabela stranica (za svaki segment po jedna). Elementi tabele segmenta sadrže graničnu adresu stranice u segmentu i baznu adresu tabele stranica segmenata. Adresa segmenata indeksira element tabele segmenata.

6. Šta karakteriše translacione podatke?

Svaki proces mora da ima neophodne translacione podatke koji obuhvataju ili graničnu i baznu adresu, ili tabelu segmenata, ili tabelu stranica, ili tabelu segmenata sa pripadnim tabelama stranica. Translacione podatke pripremi operativni sistem, prilikom stvaranja procesa. Oni se menjaju prilikom preključivanja, pa utiču na trajanje preključivanja.

7. Šta karakteriše translaciju logičkih adresa kontinualnog logičkog adresnog prostora u fizičke adrese?

Da bi translacija bila moguća, neophodno je da se u lokacije fizičke radne memorije smesti slika procesa. U ovom slučaju, translacija logičkih adresa u fizičke adrese odgovara dinamičkoj relokaciji i olakšava zamenu slika procesa.

8. Koji logički adresni prostor se koristi kada veličina fizičke radne memorije prevazilazi potrebe svakog procesa?

Kada veličina fizičke radne memorije prevazilazi potrebe svakog procesa koriste se kontinualni i segmentirani logički adresni prostor.

9. Šta karakteriše segmentaciju?

Translaciji prethodi smeštanje slike procesa u lokacije fizičke radne memorije. Segmenti logičkog adresnog prostora procesa sadrže delove njegove slike. Prednost segmentacije je što ubrzava zamenu slika procesa, jer se segment naredbi ne mora izbacivati, ako je ranije već izbačen u masovnu memoriju, niti ubacivati, ako već postoji u fizičkoj radnoj memoriji.

Ako se dozvoli da svakom potprogramu ili promenljivoj odgovara poseban segment, radi se o punoj segmentaciji. U tom slučaju, stvaraju se uslovi za dinamičko linkovanje potprograma za program, što doprinosi racionalnom korišćenju masovne memorije. Puna segmentacija dozvoljava i deljenje promenljivih između raznih procesa, ako se segment iste promenljive uključi u slike više procesa. Elementi tabele segmenata se proširuju oznakom prava pristupa segmentu.

Rukovanje segmentima se obavlja posredstvom sistemskih programa, kao što su kompajler i linker.

10. Koji logički adresni prostor se koristi kada je važno racionalno korišćenje fizičke radne memorije?

Kada je važno racionalno korišćenje fizičke radne memorije koriste se segmentirani i stranično segmentiran logički adresni prostor.

11. Koji logički adresni prostor se koristi kada je veličina fizičke radne memorije nedovoljna za pokrivanje potreba tipičnog procesa?

Kada je veličina fizičke radne memorije nedovoljna za pokrivanje potreba tipičnog procesa, koriste se stranični (zove se i virtuelni adresni prostor) i stranično segmentiran logički adresni prostor.

12. Šta sadrže elementi tabele stranica?

Elementi tabele stranica sadrže baznu adresu odgovarajuće stranice.

13. Šta karakteriše virtuelni adresni prostor?

Virtuelni adresni pripada virtuelnoj memoriji i sadrži virtuelne adrese. Stranice virtuelnog adresnog prostora sa slikom procesa se nalaze na masovnoj memoriji. Kada zatreba, kopije neophodnih virtuelnih stranica se automatski prebacuju iz masovne u fizičku radnu memoriju i obrnuto. Praktična upotrebljivost virtuelne memorije se temelji na svojstvu lokalnosti izvršavanja programa (za izvršavanje programa je dovoljno da u fizičkoj radnoj memoriji uvek bude samo deo programa).

14. Po kom principu se prebacuju kopije virtuelnih stranica?

Kopije neophodnih virtuelnih stranica, kada zatreba, automatski se prebacuju iz masovne memorije u fizičku radnu memoriju i obrnuto. Do prebacivanja u obrnutom smeru dolazi, kada je potrebno osloboditi lokacije fizičke radne memorije sa kopijama, u međuvremenu izmenjenih virtuelnih stranica. Da bi se znalo koja kopija virtuelne stranice je izmenjena, neophodno je automatski registrovati svaku izmenu svake od kopija virtuelnih stranica.

15. Šta karakteriše straničnu segmentaciju?

Svaki segment uvodi sopstveni virtuelni adresni prostor. Stranice virtuelnog adresnog prostora segmenata se nalaze u masovnoj memoriji, a u fizičkim stranicama se nalaze samo kopije neophodnih virtuelnih stranica segmenata. Prednost stranične segmentacije je da ona omogućava dinamičko proširenje segmenata. Stanična segmentacija može otvorenim datotekama dodeljivati posebne segmente i na taj način ponuditi koncept memorijski preslikane datoteke. Pristup ovakvoj datoteci ne zahteva sistemske operacije za čitanje, pisanje ili pozicioniranje. Mana ovog koncepta je da se veličina datoteke izražava celim brojem stranica i što virtuelni adresni prostor segmenata može biti suviše mali za pojedine datoteke.

16. Koji logički adresni prostor se koristi kada je važno racionalno korišćenje fizičke radne memorije, a ona ima nedovoljnu veličinu?

Kada je važno racionalno korišćenje fizičke radne memorije, a ona ima nedovoljnu veličinu, koristi se stranično segmentiran logički adresni prostor

17. Kako se deli fizička radna memorija?

Fizička radna memorija se deli na lokacije koje stalno zauzima operativni sistem i na ostale slobodne lokacije koje su na raspolaganju za stvaranje procesa i druge potrebe.

18. Kako se deli virtuelni adresni prostor?

Viruelni adresni prostor se deli tako što mu je gornja polovina rezervisana za operativni sistem, a donja polovina je na raspolaganju svakom procesu. Gornja polovina se zato naziva sistemski virtuelni adresni prostor, a donja korisnički virtuelni adresni prostor.

19. U kom obliku može biti evidencija slobodne fizičke memorije?

Za potrebe kontinualnog ili segmentiranog logičkog adresnog prostora, evidencija slobodne fizičke memorije može da bude u obliku niza bita u kome svaki bit odgovara grupi susednih lokacija. Češće se pravi u obliku liste slobodnih odsečaka fizičke radne memorije. Na početku rada operativnog sistema, ovakva lista sadrži jedan odsečak, koji obuhvata celu fizičku slobodnu radnu memoriju. Ovakav odsečak se drobi u više kraćih odsečaka, kao rezultat višestrukog stvaranja i uništavanja procesa u slučajnom redosledu. Za potrebe virtuelnog adresnog procesa evidencija može biti u obliku niza bita, u kome svaki bit odgovara slobodnoj fizičkoj stranici. Alternativa je da se slobodne fizičke stranice vežu u listu. Atraktivna je i evidencija o slobodnim odsečcima veličine jednake multiplu fizičke stranice, jer se u virtuelnom adresnom prostoru uvek zauzima celi broj stranica.

20. Kod kog adresnog prostora se javlja eksterna fragmentacija?

Eksterna fragmentacija se javlja kod kontinualnog i segmentiranog adresnog prostora.

21. Kako se nazivaju skupovi fizičkih stranica, koji se dodeljuju procesima?

Skupovi fizičkih stranica, koji se dodeljuju procesima (za čuvanje kopija svih virtuelnih stranica koje su neophodne procesoru za izvršavanje pojedinih mašinskih naredbi), nazivaju se minimalni skupovi.

22. Kada treba proširiti skup fizičkih stranica procesa?

Uvećanje skupa fizičkih stranica ima smisla samo ako to dovodi do smanjivanja učestanosti straničnih prekida. Skup fizičkih stranica procesa treba proširiti kada je, u toku aktivnosti, učestanost straničnih prekida iznad neke (iskustveno određene) gornje granice.

23. Kada treba smanjiti skup fizičkih stranica procesa?

Ako je učestanost straničnih prekida ispod neke (iskustveno određene) donje granice, tada treba smanjiti skup fizičkih stranica.

24. Kada ne treba menjati veličinu skupa fizičkih stranica procesa?

U slučaju da je učestanost straničnih prekida između donje i gornje granice, tada nema potrebe menjati veličinu skupa fizičkih stranica procesa. U tom slučaju, skup fizičkih stranica (odnosno, njima odgovarajući skup virtuelnih stranica) obrazuje radni skup.

25. Koji pristupi oslobađanja fizičkih stranica obezbeđuju smanjenje učestalnosti straničnih prekida nakon povećavanja broja fizičkih stranica procesa? ???

Sledeći pristupi oslobađanja fizičkih stranica obezbeđuju smanjenje učestanosti straničnih prekida nakon povećavanja broja fizičkih stranica:

- pronalaženje fizičke stranice koja nije korišćena u nekom prethodnom periodu (*Not Recently Used - NRU*),
- pronalaženja najmanje korišćene fizičke stranice (*Least Recently Used-LRU*),
- softverska simulacija pronalaženja najmanje korišćene fizičke stranice (*Not Frequently Used - NFU/aging*)

26. Koji pristupi oslobađanja fizičkih stranica koriste bit referenciranja? ???

Sledeći pristupi oslobađanja fizičkih stranica koriste bit referenciranja:

- pronalaženje fizičke stranice koja nije korišćena u nekom prethodnom periodu (*Not Recently Used - NRU*),
- pronalaženja najmanje korišćene fizičke stranice (*Least Recently Used-LRU*),
- softverska simulacija pronalaženja najmanje korišćene fizičke stranice (*Not Frequently Used - NFU/aging*),
- *second chance*,
- *clock*,
- *wclock*

27. Koji pristupi oslobađanja fizičkih stranica koriste bit izmene? ???

Sledeći pristupi oslobađanja fizičkih stranica koriste bit izmene:

- *Not Recently Used - NRU*
- oslobađanje fizičke stranice sa najstarijim sadržajem (*First In First Out - FIFO*) (koristi i bit izmene),
- *second chance*,
- *clock*,
- *wclock*

28. Na šta se oslanja rukovanje virtuelnom memorijom?

Rukovanje virtuelnom memorijom oslanja se na operacije sloja za rukovanje kontrolerima.

Sloj operativnog sistema za rukovanje kontrolerima

1. Šta karakteriše ulazno-izlazne uređaje?

Ulazni i izlazni uređaji se dele na blokovske i znakovne uređaje. Razlikuju se u pogledu jedinice pristupa (za blokovske uređaje jedinica pristupa je blok, za znakovne uređaje jedinica pristupa je znak), u pogledu načina pristupa (dok mnogi blokovski uređaji dozvoljavaju direktan pristup, znakovni uređaji podržavaju samo sekvencijalni pristup), i u pogledu upravljanja (za razliku od blokovskih, znakovni uređaji dozvoljavaju dinamičko podešavanje njihovih pojedinih funkcionalnih karakteristika). Svaki ulazno-izlazni uredjaj ima svoj drajver.

2. Koja svojstva imaju drajveri?

Svaki od drajvera namenjen je za rukovanje određenom klasom uređaja. Obično, jedan drajver može da opsluži više primeraka uređaja iste klase. Driveri su u tesnoj saradnji sa kontrolerima ulaznih i izlaznih uređaja i kriju sve posebnosti funkcionisanja ovih kontrolera.

3. Šta karakteriše tabela drajvera?

Tabela drajvera sadrži:

- adrese operacija inicijalizacije
- adrese operacija ulaza i izlaza
- adrese upravljačkih operacija

Redni broj drajvera indeksira element ove tabele, koji sadrži polja sa adresama pojedinih operacija datog drajvera. Polja, namenjena za adrese operacija, koje dotični drajver ne podržava, sadrže adresu posebne lažne operacije, čije obavljanje nema efekta.

4. Šta podrazumeva podela drajvera na gornji i donji deo?

Gornji deo drajvera obrazuju operacije drajvera, a obrađivači prekida obrazuju donji deo drajvera. Opearcije drajvera se pozivaju iz slojeva iznad sloja za rukovanje kontrolerima, a obrađivače prekida poziva mehanizam prekida, znači hardver ispod operativnog sistema.

5. Kada se pozivaju operacije drajvera blokovskih uređaja?

Aktivnost drajvera blokovskih uređaja započinje nakon inicijalizacije njihovih kontrolera (pozivanjem drajverske operacije inicijalizacije). Za prenos blokova ka kontrolerima i od njih se pozivaju drajverske operacije ulaza i izlaza blokova. Drajverska operacija izlaza bloka se poziva kada se zahteva da izmenjeni sadrzaj bafera sloja za rukovanje datotekama bude sačuvan na disku.

6. Šta sadrži lista zahteva?

U listi zahteva se nalaze zahtevi za prenos blokova. Svaki zahtev u ovakvoj listi zahteva mora da sadrži:

- smer zahtevanog prenosa bloka
- redni broj ovog bloka
- adresu bafera koji učestvuje u prenosu, kao i
- adresu deskriptora procesa, čija aktivnost se zaustavlja do obavljanja zahtevanog prenosa bloka.

7. Šta spada u nadležnosti drajvera blokovskih uređaja ali i kontrolera?

U nadležnosti drajvera blokovskih uređaja, ali i kontrolera spada određivanje načina preslikavanja blokova u sektore. Takođe, u iste nadležnosti spada i optimizacija kretanja glave diska.

8. Kada se uzastopni blokovi preslikavaju u prostorno uzastopne sektore?

Ako kontroler automatski prebacuje sve sektore staze, iznad koje se kreće glava diska, u svoju lokalnu radnu memoriju, tada nema smetnje da se uzastopni blokovi preslikaju u prostorno uzastopne sektore.

9. Na koji drajver se odnosi elevator algoritam?

Elevator algoritam se odnosi na drajver blokovskih uređaja. Elevator algoritmom se optimizuje kretanje glave diska i obezbeđuje pravedno usluživanje svih zahteva.

10. Koju ulogu imaju sistemski procesi posrednici?

U slučaju znakovnih uređaja, kao što su štampači ili mrežni kontroleri, uvode se posebni sistemski procesi posrednici koji posreduju u korišćenju pomenutih uređaja. Svaki od ovih procesa pristupa svom znakovnom uređaju kao specijalnoj datoteci, koju zaključava, da bi obezbedio međusobnu isključivost u toku njenog korišćenja.

11. Kada se specijalna datoteka tipično zaključava?

Kada procesi pristupaju znakovnim uređajima, pristupaju im kao specijalnim datotekama, i tada ih zaključavaju, da bi obezbedili međusobnu isključivost u toku njenog korišćenja.

12. Šta sadrži drajver terminal?

Drajver terminal sadrži ulazni bafer (služi za smeštanje znakova, prispelih sa tastature) i eho bafer (služi za smeštanje znakova, upućenih ka ekranu).

13. U kom slučaju nisu potrebni eho bafer i obrađivač prekida ekrana?

Za grafičke (memorijski preslikane) terminale nije potreban eho bafer, niti obrađivač prekida ekrana, jer ovakvi terminali poseduju video memoriju čiji sadržaj se periodično prikazuje prilikom osvežavanja ekrana.

14. Šta omogućuje upravljačka operacija drajvera terminala?

Argumenti njenog poziva utiču ne samo na funkcionisanje, npr. teminala, nego i na funkcionisanje drajvera terminala. Upravljačka operacija omogućuje da se drajveru terminala saopšti da interpretira znakove koji dolaze sa tastature, ili da ih ne interpretira. U prvom slučaju, u nadležnosti drajvera terminala se nalazi editiranje znakova prispelih sa tastature. U sklopu toga drajver terminal mora, npr. da omogući brisanje poslednje prispelog znaka. Takođe drajver terminala se brine o interpretaciji upravljačkih znakova "prelazak na novu liniju", "prelazak na početak linije", kao i drugih upravljačkih znakova.

15. Koje operacije sadrži gornji deo drajvera sata?

Gornji deo ovog drajvera čine sistemske operacije za preuzimanje ili izmenu sistemskog vremena i za uspavljivanje procesa, odnosno za odlaganje njegove aktivnosti, dok ne istekne zadati vremenski interval.

16. Šta ne može da meri drajver sata?

Drajver sata ne može da meri dužine vremenskih intervala, koji su kraći od perioda prekida sata, a kojih ima u toku aktivnosti procesa. To izaziva da merenje trajanja aktivnosti procesa ne mora biti precizno.

17. Šta omogućuje obrađivač prekida iz donjeg dela drajvera sata?

U nadležnosti obrađivača prekida sata nalazi se više poslova, kao što su :

- održavanje sistemskog vremena,
- praćenje isticanja kvantuma aktivnog procesa,
- praćenje ukupnog korišćenja procesorskog vremena aktivnog procesa,
- provera da li je nastupilo vreme buđenja uspavanog procesa
- skupljanje statistika o aktivnosti procesa.

Obrađivač prekida sata broji prekide sata, a njihov zbir predstavlja sistemsko vreme (lokalno vreme u računaru).

Sloj operativnog sistema za rukovanje procesorom

1. Šta karakteriše tipične ciljeve raspoređivanja?

Raspoređivanje brine o izboru spremnog procesa na koga će se prekljućiti procesor sa aktivnog. Tipični ciljevi su, na primer, poboljšanje iskorišćenja procesorskog vremena, ravnomerna raspodela procesorskog vremena, sto kraći odziv na korisničku akciju ili neki drugi oblik postizanja potrebnog kvaliteta usluge. Ovi ciljevi nisu uvek saglasni, pa se ne mogu istovremeno ostvariti.

2. Šta je cilj raspoređivanja za neinteraktivno korišćenje računara?

Za neinteraktivno korišćenje računara, cilj raspoređivanja je poboljšanje iskorišćenja procesorskog vremena. Ovakav cilj se ostvaruje minimiziranjem preključivanja na neophodan broj (samo nakon pozivanja blokirajućih sistemskih operacija ili nakon kraja aktivnosti procesa)

3. Šta je cilj raspoređivanja za interaktivno korišćenje računara?

Za interaktivno korišćenje računara ciljevi raspoređivanja su ravnomerna raspodela procesorskog vremena između istovremeno postojećih procesa, odnosno između njihovih vlasnika (korisnika, koji istovremeno koriste računar) i što kraći odziv na korisničku akciju.

4. Zašto je uvedeno kružno raspoređivanje?

Kružno raspoređivanje je uvedeno radi ravnomernog raspoređivanja procesorskog vremena između istovremeno postojećih procesa i što kraćeg odziva na korisničku akciju. Tada ono svakom od istovremeno postojećih procesa dodeljuje isti vremenski interval, nazvan kvantum. Kružno raspoređivanje se uspešno primenjuje i u situaciji kada hitnost svih procesa nije ista, pa se zbog toga procesima dodeljuju razni prioriteti.

5. Šta doprinosi ravnomernoj raspodeli procesorskog vremena?

Ravnomernoj raspodeli procesorskog vremena doprinosi uvođenje kvantuma. Po isticanju kvantuma, aktivni proces prepušta procesor spremnom procesu, koji najduže čeka na svoj kvantum.

Dinamička izmena prioriteta procesa doprinosi ravnomernosti raspodele procesorskog vremena:

- između procesa, ako se uspostavi obrnuta proporcionalnost između prioriteta procesa i obima u kome je on iskoristio poslednji kvantum.
- između korisnika, ako se uspostavi obrnuta proporcionalnost između prioriteta procesa, koji pripadaju nekom korisniku, i ukupnog dela u procesorskom vremenu tog korisnika u toku njegove interakcije sa računarom.

Ravnomerna raspodela procesorskog vremena može se postići i bez izmena prioriteta, lutrijskim raspoređivanjem.

6. Šta je cilj raspoređivanja za multimedijalne aplikacije?

Za multimedijalne aplikacije, koje zahtevaju visoku propusnost podataka i njihovu isporuku sa pravilnim periodom, cilj raspoređivanja je garantovanje procesima potrebnog broja kvantuma u pravilnim vremenskim razmacima.

7. Do čega dovodi skraćenje kvantuma?

Skraćenje kvantuma dovodi do bržeg odziva, ali i do smanjenja iskorišćenja procesora zbog trošenja vremena na preključivanje.

8. Šta se postiže uticanjem na nivo prioriteta i na dužinu kvantuma?

Uticanjem na nivo prioriteta i na dužinu kvantuma obezbeđujemo dobar odziv procesora na korisničke procese, kao i kvalitetno iskorišćenje procesorskog vremena za pozadinkse procese. Što je proces duže aktivan duži mu je kvantum i niži prioritet.

Mrtva petlja

1. Šta je mrtva petlja?

Mrtva petlja je problematična pojava trajnog zaustavljanja aktivnosti međusobno zavisnih procesa.

2. Po čemu se živa petlja razlikuje od mrtve petlje?

Za mrtvu petlju je karakteristična blokirajuća sistemska operacija zaključavanja, dok je za živu petlju karakteristična neblokirajuća sistemska operacija zaključavanja. U slučaju neblokirajuće sistemske operacije zaključavanja, procesi upadaju u beskonačnu petlju, pokušavajući da zaključaju datoteku, koju je zaključao drugi proces. Živa petlja se po svom ishodu, suštinski, ne razlikuje od mrtve petlje.

3. Koji uslovi su potrebni za pojavu mrtve petlje?

Pojava mrtve petlje je vezana za zauzimanje resursa. Za pojavu mrtve petlje potrebno je da budu ispunjena četiri uslova:

- zauzimani resursi se koriste u režimu međusobne isključivosti
- resursi se zauzimaju jedan za drugim, tako da proces, nakon zauzimanja izvesnog broja resursa, mora da čeka da zauzme preostale resurse,
- resurse oslobađaju samo procesi koji su ih zauzeli,
- postoji cirkularna međuzavisnost procesa

4. Kako se u praksi tretira problem mrtve petlje?

Postoje četiri pristupa tretiranja problema mrtve petlje:

- sprečavanje pojave mrtve petlje (onemogućavanjem važenja nekog od četiri neophodna uslova za njenu pojavu)
- izbegavanje pojave mrtve petlje
- otkrivanje pojave mrtve petlje i oporavak od nje
- ignorisanje pojave mrtve petlje

5. Na čemu se temelji sprečavanje mrtve petlje?

Kod sprečavanja pojave mrtve petlje, važenje prvog uslova obično nije moguće sprečiti, jer se resursi najčešće koriste u režimu međusobne isključivosti. Važenje drugog uslova se može sprečiti, ako se unapred zna koliko je potrebno resursa i ako se oni svi zauzmu pre korišćenja. Važenje trećeg uslova se obično ne može sprečiti, jer najčešće ne postoji način da se zauzeti resurs privremeno oduzme procesu. Važenje četvrtog uslova se može sprečiti, ako se resursi uvek zauzimaju u unapred određenom redosledu, koji isključuje mogućnost cirkularne međuzavisnosti procesa.

6. Šta karakteriše izbegavanje mrtve petlje?

Izbegavanje pojave mrtve petlje zahteva poznavanje podataka (1) o maksimalno mogućim zahtevima za resursima, (2) o ukupno postavljenim zahtevima za resursima i (3) o stanju resursa. Udovoljava se samo onim zahtevima za koje se proverom ustanovi da, nakon njihovog ispunjavanja, postoji redosled zauzimanja i oslobađanja resursa u kome se mogu zadovoljiti maksimalno mogući zahtevi svih procesa. Ovakva provera je komplikovana, a samim tim i neefikasna.

7. Šta karakteriše otkrivanje i oporavak od mrtve petlje?

Otkrivanje mrtve petlje se zasniva na sličnom pristupu kao i izbegavanje pojave mrtve petlje. Proverava se da li postoji proces, čijim zahtevima se ne može udovoljiti ni za jedan redosled zauzimanja i oslobađanja resursa. Pored komplikovanosti ovakve provere, problem je i šta učiniti kada se i otkrije pojava mrtve petlje. Ako se resursi ne mogu privremeno oduzeti od procesa, preostaje

jedino uništavanje procesa, radi oslobađanja resursa koje oni drže. To nije uvek prihvatljiv zahtev. Zbog toga ovaj pristup nema veliki praktični značaj.

8. Šta karakteriše ignorisanje mrtve petlje?

Ignorisanje mrtve petlje je pristup koji je najčešće primenjen u praksi, jer se ovaj problem ne javlja tako često da bi se isplatilo da ga rešava operativni sistem. Prema tome, kada se mrtva petlja javi, na korisniku je da se suoči sa ovim problemom, i da ga reši na način, koji je primeren datim okolnostima.

1. Od čega se sastoje komande znakovnog komandnog jezika?

Komanda znakovnog jezika započinje operatorom (npr. u obliku putanje izvršne datoteke, koja opisuje rukovanje), a završava operandom ili operandima (npr. u obliku putanja datoteka, kojima se rukuje).

2. Kako se zadaju komande grafičkih komandnih jezika?

Kod grafičkih komandnih jezika, zadavanje komandi vrši se pomoću grafičke predstave komande (icon) ili spiska operatora komandi (menu). Nakon izbora operatora sledi, po potrebi, dijalog u kome korisnik navodi (ili opet bira) operand ili operande komande. Grafički komandni jezici još više pojednostavljuju zadavanje komandi, ako korisniku omogućuju da ne bira operator, nego samo operande komandi.

3. Šta su ciljevi znakovnih komandnih jezika?

Ciljevi znakovnih komandnih jezika obuhvataju:

- omogućavanje izvršavanja pojedinih (korisničkih) programa
- omogućavanje kombinovanja izvršavanja više (korisničkih) programa
- omogućavanje pravljenja komandnih datoteka

4. Šta omogućuju znakovni komandni jezici?

Znakovni komandni jezici omogućuju izvršavanje pojedinih programa, kombinovanje izvršavanja više korisničkih programa i pravljenje komandnih datoteka.

5. Šta omogućuju čarobni znakovi?

U okviru operanada komandi se mogu javiti čarobni (specijalni) znakovi, kao što je npr. znak *. Njegova upotreba je vezana, pre svega, za imena datoteka, i namenjena je za skraćeno označavanje grupa datoteka. Npr. *.obj označava sve objektne datoteke u radnom imeniku. Zahvaljujući magičnim znakovima moguće je, na primer, jednom komandom uništiti sve objektne datoteke iz radnog imenika (uništi *.obj) ili odštampati sve tekst datoteke iz radnog imenika koje počinju znakom d, a završavaju znakom 1 (odštampaj d*1.txt).

6. Šta omogućuje preusmeravanje?

Zahvaljujući obradi znakova komande, moguće je interpreteru znakovnog komandnog jezika saopštiti i da preusmeri standardni ulaz i standardni izlaz sa tastature i ekrana na proizvoljno odabrane datoteke. Ovo je važno za pozadinske procese, koji nisu u interakciji sa korisnikom. Zahvaljujući preusmeravanju, pozadinski proces ne ometa interaktivni rad korisnika, jer umesto tastature i ekrana, koristi odabrane datoteke. Preusmeravanje standardnog ulaza najavljuje znak <, a preusmeravanje standardnog izlaza znak >. Preusmeravanje standardnog ulaza i izlaza predstavlja osnovu za kombinovanje izvršavanja više korisničkih programa.

7. Čemu služe pipe?

Umesto preusmeravanja standardnog ulaza i standardnog izlaza, moguće je nadovezati standardni izlaz jednog procesa na standardni ulaz drugog procesa i tako obrazovati tok procesa (pipe). Nadovezivanje u tok se označava pomoću znaka |.

8. Čemu služi baferovana specijalna datoteka?

Razmena podataka između dva procesa, koji su povezani u tok, ostvaruje se posredstvom posebne specijalne datoteke. Njoj odgovara bafer u radnoj memoriji. Ova baferovana specijalna datoteka služi

prvom od ovih procesa, kao standardni izlaz, a drugom od njih kao standardni ulaz. Prvi proces samo piše u ovu datoteku, a drugi samo čita iz nje.

9. Šta karakteriše pozadinske procese?

Pozadinski procesi se razlikuju od običnih (interaktivnih) procesa po tome što interpreter znakovnog komandnog jezika, nakon stvaranja pozadinskog procesa, ne čeka kraj njegove aktivnosti, nego nastavlja interakciju sa korisnikom. Zato su pozadinski procesi u principu neinteraktivni.

10. Šta karakteriše komandne datoteke?

Komandne datoteke opisuju okolnosti pod kojima se izvršavaju korisnički programi, a sadržaj komandne datoteke preuzima na interpretiranje interpreter znakovnog komandnog jezika. Komandne datoteke imaju poseban tip, da bi ih interpreter znakovnog komandnog jezika mogao prepoznati. Ime svake komandne datoteke predstavlja ispravnu komandu znakovnog komandnog jezika. Imamo korisničke i administratorske komande.

11. Šta omogućuju korisničke komande?

Korisničke komande omogućuju:

- rukovanje datotekama,
- rukovanje imenicima,
- rukovanje procesima,
- razmenu poruka između korisnika.

12. Šta omogućuju administratorske komande?

Administratorske komande omogućuju:

- pokretanje i zaustavljanje rada računara,
- spašavanje i vraćanje datoteka,
- rukovanje vremenom,
- kompresovanje datoteka,
- ažuriranje podataka o korisnicima i njihovim pravima,
- generisanje izveštaja o korišćenju računara,
- rukovanje konfiguracijom računara,
- proveru ispravnosti rada računara,
- pripremu diskova za korišćenje

Klasifikacija operativnih sistema

1. Šta karakteriše operativne sisteme realnog vremena?

Operativne sistemi realnog vremena su namenjeni za primene računara u kojima je neophodno obezbediti reakciju na vanjski događaj u unapred zadatom vremenu. Zbog toga su ovi operativni sistemi podređeni ostvarenju što veće brzine korisničkih programa. Operativni sistemi realnog vremena su zajedno sa računarom, ugrađeni u sistem, čije ponašanje se ili samo prati, ili čijim ponašanjem se upravlja.

Zadatak operativnih sistema realnog vremena je da samo stvore okruženje za korisničke programe, jer komunikaciju sa krajnjim korisnikom obavljaju korisnički programi. Zato se obično koriste samo na programskom nivou.

2. Šta karakteriše multiprocesorske operativne sisteme?

Multiprocesorski operativni sistemi upravljaju računarskim sistemima sa više procesora opšte namene, koji pristupaju zajedničkoj radnoj memoriji. Podrazumeva se da ove procesore i operativnu memoriju povezuje sabirnica. Kod modula za rukovanje procesorom prisutne su istovremene aktivnosti više procesa na raznim procesorima, pa se sinhronizacija procesa ne ostvaruje onemogućenjem prekida, nego zauzimanjem sabirnice. Mogućnost istovremene aktivnosti više procesa na raznim procesorima usložnjava raspoređivanje.

3. Koje module sadrži mikrokernel?

Mikrokernel sadrži:

- modul za rukovanje procesima,
- modul za razmenu poruka,
- modul za rukovanje radnom memorijom,
- modul za rukovanje kontrolerima,
- modul za rukovanje procesorom.

Mikrokernel ne sadrži modul za rukovanje datotekama.

4. Šta karakteriše poziv udaljene operacije “RPC”?

Ako pozivana operacija ne odgovara potprogramu koji se lokalno izvršava u okviru aktivnosti procesa pozivaoca, nego odgovara potprogramu koji se izvršava u okviru aktivnosti drugog, udaljenog procesa, aktivnog na udaljenom računaru, reč je o pozivu udaljene operacije (Remote Procedure Call - RPC). Proces koji poziva udaljenu operaciju se nalazi u ulozi klijenta (primaoca usluge), a proces koji obavlja udaljenu operaciju se nalazi u ulozi servera (davaoca usluge). Poziv udaljene operacije ima oblik poziva potprograma, u kome se navode oznaka (ime) operacije i njeni argumenti. Ovakav potprogram se naziva klijentski potprogram, jer je klijent njegov jedini pozivalac.

5. Šta radi klijentski potprogram?

Koraci klijentskih programa:

1. Pronalaženje procesa servera koji pruža zahtevanu uslugu
2. Pakovanje argumenata u poruku zahteva
3. Slanje serveru poruke zahteva
4. Prijem poruke od servera sa rezultatom zahtevane usluge
5. Raspakivanje prispele poruke odgovora
6. Isporuka rezultata pružanja zahtevane usluge pozivaocu potprograma

6. Za šta su zaduženi serverski potprogrami?

Serverske potprograme poziva jedino server, postoje dva serverska potprograma i imaju zadatak da pružaju zahtevane usluge, što se obavlja u više koraka.

Prvi serverski potprogram obuhvata :

- prijem poruke zahteva i
- raspakivanje argumenata iz ove poruke

Drugi serverski potprogram obuhvata :

- pakovanje rezultata usluge (koju je pružio server) u poruku odgovora i
- slanje klijentu ove poruke odgovora

Između poziva ova dva potprograma nalazi se operacija koja odgovara izvršavanju zadate usluge.

7. Koji problemi su vezni za poziv udaljene operacije?

U toku poziva udaljene operacije mogu da se pojave problemi, čija pojava nije moguća kod poziva lokalne operacije, pa je moguće:

1. nepronalaženje servera koji pruža zahtevanu uslugu
2. da se izgube ili poruka zahteva ili poruka odgovora
3. da dođe do otkaza ili servera ili klijenta u toku rada

8. Šta podrazumeva dinamičko linkovanje klijenta i servera?

Dinamičko linkovanje klijenta i servera podrazumeva trenje i povezivanje servera koji vrši zahtevanu uslugu sa klijentom, koji zahteva uslugu. Ovaj proces se odvija pomoću servera imena (*name server*), koji sadrži podatke o svim ostalim serverima.

9. Koje operacije podrazumeva protokol razmene poruka između klijenta i servera?

Protokol razmene poruka između klijenta i servera podrazumeva:

- sistemsku operaciju zahtevanja usluge
- sistemsku operaciju prijema zahteva
- sistemsku operacija slanja odgovora

10. Za šta su zadužene sistemske operacije koje ostvaruju protokol razmene poruka?

Sistemska operacija zahtevanja usluge je namenjena klijentu i poziva se iz njegovog potprograma. Ona omogućuje slanje poruke zahteva i prijem poruke odgovora. Sistemske operacije prijema zahteva i slanja odgovora su namenjene serveru i omogućuju prijem poruke zahteva i slanje poruke odgovora. Sistemska operacija prijema zahteva se poziva iz prvog serverskog potprograma, a sistemska operacija slanja odgovora se poziva iz drugog serverskog potprograma. Ove tri sistemske operacije su blokirajuće. Sistemska operacija zahtevanja usluge zaustavlja aktivnost klijenta do stizanja odgovora ili do isticanja zadanog vremenskog perioda. Sistemska operacija prijema zahteva zaustavlja aktivnost servera do stizanja zahteva, a sistemska operacija slanja odgovora zaustavlja aktivnost servera do isporuke odgovora ili do isticanja zadanog vremenskog intervala. Ove tri sistemske operacije su zadužene za prenos poruka. Pored slanja i prijema poruka, one (1) potvrđuju prijem poruka, (2) retransmituju poruke, čiji prijem nije potvrđen, (3) šalju upravljačke poruke, kojim se proverava i potvrđuje aktivnost servera. U nadležnosti ovih operacija je i rastavljanje poruka u pakete, sastavljanje poruka od paketa, potvrda prijema paketa i retransmisija paketa čiji prijem nije potvrđen, kao i prilagođenje brzine slanja paketa brzini kojom oni mogu biti primani. Koriste usluge drajvera sata. One pozivaju i (neblokirajuće) operacije gornjeg dela drajvera mrežnog kontrolera. Takođe, brinu se i o baferima, najmjenjenim za (privremeno) smeštanje poruka.

11. Šta sadrže poruke koje razmenjuju klijent i server?

Svaka od poruka, koje se razmenjuju između procesa, se sastoji od upravljačkog dela poruke i sadržaja poruke.

Upravljački deo poruke obuhvata:

- adresu odredišnog procesa (kome se poruka upućuje)
- adresu izvorišnog procesa (od koga poruka krece)
- opis poruke (vrstu, redni broj...)

12. Šta je potrebno za sigurnu razmenu poruka između klijenta i servera?

Komunikacione linije su pristupačne svim korisnicima, pa je svaki od njih u poziciji da preuzima tuđe poruke i da šalje poruke u tuđe ime. Sprečavanje preuzimanja tuđih poruka zasniva se na kriptovanju poruka, a sprečavanje slanja poruka u tuđe ime se zasniva na nedvosmislenoj međusobnoj identifikaciji procesa.

Bitnu ulogu u sigurnosti razmene poruka igra i poseban server, koji se naziva poverenik. On poseduje unapred dogovoren poseban ključ kriptovanja za komunikaciju sa svakim procesom. Zahvaljujući tome, klijent može da pošalje poruku povereniku, koja sadrži ime klijenta i ime servera sa kojim klijent želi da ostvari sigurnu komunikaciju. Poverenik šalje samo klijentu i serveru interni ključ kriptovanja za sigurnu komunikaciju i ujedno se obavi njihova međusobna identifikacija, tako da se drugi procesi ne mogu umešati u njihovu komunikaciju. Kod asimetrične kriptografije, uloga poverenika je da čuva javne ključeve i tako osigura međusobnu identifikaciju procesa.

13. Šta karakteriše digitalni potpis?

Asimetrična kriptografija omogućuje i digitalno potpisivanje poruka, radi neopozivog pripisivanja poruke njenom pošiljaocu. Digitalni potpis se šalje uz poruku. Digitalni potpis jednoznačno reprezentuje poruku, tj. predstavlja otisak prsta poruke. Otisak prsta formiraju jednosmerne funkcije na osnovu sadržaja poruke. Digitalni potpis nastaje kada se otisak prsta poruke dekriptuje, primenom algoritma kriptovanja i javnog ključa.

14. Od čega zavisi propusnost servera?

Manja propusnost servera uslovljena je strogom sekvencijalnošću njegove aktivnosti, koja takođe utiče i na sporije pružanje usluga.

15. Šta sadrže dozvole na kojima se zasniva zaštita datoteka u distribuiranom sistemu?

Da bi klijent dobio neku uslugu, on mora da poseduje odgovarajuću dozvolu, koju prosleđuje serveru u okviru zahteva za uslugom. Dozvola sadrži :

- redni broj servera,
- redni broj deskriptora datoteke (odnosno, deskriptora imenika)
- oznaku vrste usluge i
- oznaku ispravnosti dozvole.

16. Šta karakteriše distribuiranu sinhronizaciju?

Saradnja procesa, aktivnih na raznim računarima, zahteva njihovu sinhronizaciju, što se ostvaruje razmenom poruka. Najjednostavniji način za ostvarenje sinhronizacije se zasniva na uvođenju procesa koordinatora. Njemu se obraćaju svi procesi, zainteresovani za sinhronizaciju, a koordinator donosi odluke o njihovoj sinhronizaciji. Proces koordinator spada u centralizovane algoritme sinhronizacije. Postoje i distribuirani algoritmi sinhronizacije koji se zasnivaju na međusobnom dogovaranju procesa, zainteresovanih za sinhronizaciju. Ovaj algoritam ima veću razmenu poruka i komplikovaniji je od

centralizovanih algoritama, ali nema veći praktični značaj. Za distribuirane operativne sisteme nije samo bitno da omoguće efikasnu sinhronizaciju procesa, nego i da podrže poseban oblik sinhronizacije procesa, koji obezbeđuje da se obave ili sve operacije iz nekog niza pojedinačnih akcija ili nijedna od njih. Ovakav niz operacija se naziva transakcija, a transakcije koje imaju svojstvo da se obave u celosti ili nikako, nazivaju se atomske transakcije. Distribuirani računarski sistemi su neefikasniji kod pojave mrtve petlje, nego centralizovani (jednoprocesorski) računar.

17. Šta karakteriše distribuirani računarski sistem?

Distribuirani računarski sistem je zamišljen da integriše mnoštvo računara u moćan multiračunarski sistem. Moguće je od više jeftinih i malih računara napraviti moćan multiračunarski sistem povoljne cene. Ovakav sistem ima i veću pouzdanost i mogućnost proširenja. Ovakav sistem omogućuje i deljenje skupih resursa ovakvog multiračunarskog sistema između više korisnika, a nudi i prilagodljivost zahtevima korisnika, željenu raspoloživost i predvidivost odziva, čak i veću sigurnost.

18. Šta karakteriše distribuiranu softversku platformu?

Uniforman način pružanja usluga podrazumeva skrivanje različitosti računara od kojih je obrazovan distribuirani računarski sistem. To se može postići, ako se iznad raznorodnih operativnih sistema, pojedinih računara distribuiranog računskog sistema napravi posebna softverska platforma (*middleware*) za razvoj distribuiranih softverskih sistema. Ona ima ulogu distribuiranog operativnog sistema. Distribuirana softverska platforma je obično specijalizovana tako da nudi konzistentan skup operacija, koje omogućuju razvoj željene vrste distribuiranih softverskih sistema. Imamo distribuirane softverske platforme za podršku distribuiranih dokumenata (WWW) ili podršku distribuiranom objektno orjentisanom programiranju (CORBA,.NET). U osnovi ovakvih sistema se krije klijent-server model. Server se nalazi na strani rukovaoca distribuiranim dokumentima (*web site*) ili rukovaoca objektima, a klijent se nalazi na strani korisnika distribuiranih dokumenata (*web browser*) ili pozivaoca operacija udaljenih objekata.