Borislav Đorđević
Dragan Pleskonjić
Nemanja Maček

Operativni sistemi

Zbirka rešenih zadataka

Viša elektrotehnička škola
Beograd, 2004.

Autori: dr Borislav Đorđević
mr Dragan Pleskonjić
Nemanja Maček

Recenzenti: prof. dr Borivoj Lazić
mr Slobodan Obradović

Izdavač: Viša elektrotehnička škola u Beogradu

Za izdavača: mr Dragoljub Martinović

Lektor: Milena Dorić

Tehnička obrada: Nemanja Maček, Borislav Đorđević, Dragan Pleskonjić

Dizajn: Jovana Lazović (MASSVision)

Štampa: MST Gajić
štampano u 200 primeraka

CIP – Каталогизација у публикацији
Народна библиотека Србије, Београд

004.451(075.8)(076)

ЂОРЂЕВИЋ, Борислав
Operativni sistemi: zbirka rešenih zadataka / Borislav Đorđević, Dragan Pleskonjić, Nemanja Maček. – Beograd : Viša elektrotehnička škola, 2005 (Beograd : MST Gajić). – VI, 205 str. ; 24 cm

Tiraž 200. – Bibliografija: str. 205.

ISBN 86-85081-15-7
1. Плескоњић, Драган 2. Мачек, Немања

а) Оперативни системи – Задаци
COBISS.SR-ID 120862988

Predgovor

Zbirka rešenih zadataka iz operativnih sistema je prvenstveno namenjena osnovnom kursu iz nastavnog predmeta Operativni sistemi, koji autori izvode na Višoj elektrotehničkoj školi u Beogradu, a mogu je koristiti i studenti koji nastavu iz istog ili srodnih predmeta pohađaju na drugim fakultetima ili višim školama, odnosno i svi koji žele da provere svoje znanje iz ove oblasti. Zbirku, takođe, mogu koristiti i nastavnici kao pomoćnu nastavnu literaturu potrebnu kao stručni podsticaj pri izvođenju nastave iz predmeta Operativni sistemi. Zbirka je nastala kao rezultat višegodišnjeg iskustva koje su autori stekli prilikom praktičnog i teorijskog rada i izvođenja nastave na Višoj elektrotehničkoj školi u Beogradu.

Zbirka zadataka omogućava čitaocima da potpuno i pouzdano shvate i bolje razumeju koncepte i algoritme korišćene pri projektovanju pojedinih delova savremenih operativnih sistema. Koncepti i značajniji algoritmi, obrađeni u ovoj knjizi, uglavnom su zasnovani na implementacijama u postojećim besplatnim i komercijalnim operativnim sistemima.

Pretpostavljamo da čitaoci koji žele da se bave izučavanjem ove materije poznaju osnove arhitekture računara, operativnih sistema i računarskih mreža sa aspekta prosečnog korisnika, kao i osnovne strukture podataka i osnove programiranja na jeziku C.

U zbirci su kroz pitanja i rešene zadatke najpre obrađeni koncepti savremenih operativnih sistema, a potom administracija *Linux* sistema. Na kraju zbirke priloženi su i primeri testova sa prvog i drugog dela ispita iz navedenog predmeta.

Autori se zahvaljuju svima koji su na bilo kakav način pomogli u realizaciji ove knjige.

Autori

Sadržaj

1. Uvod u operativne sisteme

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove, funkcije i karakteristike operativnih sistema, kao i na njihovu klasifikaciju prema različitim kriterijumima. Razmotreni su pojmovi monolitnog i hijerarhijskog modela operativnog sistema, mikrokernel arhitektura, razdvajanje korisničkog i sistemskog režima rada, privilegovani instrukcijski set i mehanizam sistemskih poziva.

1.1. Koji su osnovni ciljevi koje je neophodno da operativni sistem postigne pri posredovanju između korisnika i računarskog sistema?

Operativni sistem treba da:

- izvršava korisničke programe i olakša rešavanje korisničkih problema,
- korišćenje računarskog sistema učini podesnijim za korisnika,
- omogući efikasnije korišćenje hardvera računarskog sistema.

1.2. Navedite osnovne funkcije operativnog sistema.

- upravljanje poslovima (raspoređivanje i sekvenciranje poslova) i interpretacija komandnog jezika,
- upravljanje resursima,
- rukovanje ulazno/ izlaznim operacijama,
- rukovanje greškama i prekidima,
- višestruki pristup,
- zaštita resursa od zlonamernih napada, slučajnih grešaka korisnika i grešaka u korisničkim programima i samom operativnom sistemu,
- obezbeđivanje dobrog interfejsa za operatora i korisnika,
- obračun korišćenja računarskih resursa.

1.3. Navedite osnovne karakteristike operativnih sistema.

- konkurentnost, odnosno postojanje više simultanih, paralelnih aktivnosti, kao što je koegzistencija više programa u memoriji,
- deoba resursa,
- postojanje dugotrajne memorije,
- determinizam po pitanju izvršavanja programa i nedeterminizam po pitanju opterećenja.

1.4. a. Definišite efikasnost i premašenje operativnog sistema.

b. Na osnovu kojih kriterijuma se određuje efikasnost operativnog sistema?

(a) Efikasnost e je odnos vremena u kom procesor radi za korisnika i ukupnog vremena potrebnog da se određeni posao ili grupa poslova obavi, tj: $e=t_{korisno}/t_{ukupno}$ $(0<e<1)$. Premašenje o (*overhead*) je odnos vremena u kome procesor radi na održavanju samog sistema i ukupnog vremena: $o=t_{održavanje}/t_{ukupno}$ $(0<o<1)$.

(b) Kriterijumi za određivanje efikasnosti operativnog sistema su srednje vreme između dva uzastopna posla, vreme neiskorišćenosti operativnog sistema, vreme prolaska (turn-around time) za *batch* poslove, vreme odziva za interaktivne sisteme, iskorišćenost resursa i propusna moć, odnosno ukupna veličina posla koji može biti obavljen interaktivno u nekom vremenu.

1.5. Navedite razloge zbog kojih se operativni sistemi najčešće pišu u programskom jeziku C.

- Izvorni kod se mnogo brže piše i kompaktniji je.
- Kod je čitljiviji, tako da se greške lakše mogu otkloniti.
- Operativni sistem se lakše može preneti na drugu računarsku arhitekturu.

1.6. Jedna od poželjnih osobina operativnog sistema je visoka efikasnost. U kom slučaju se može dozvoliti da operativni sistem "pregazi" ovaj princip i nepotrebno "pojede" resurse ?

Na jednokorisnički sistemima iskorišćenje sistema uvećava se uvođenjem grafičkog korisničkog interfejsa (GUI - *graphical user interface*). GUI dodatno

opterećuje procesor i memoriju, ali optimizuje interakciju između korisnika i sistema i kao takav smatra se prihvatljivim gubitkom.

1.7. Zašto ne postoje višekorisnički jednoprocesni operativni sistemi?

Višekorisnički operativni sistemi obezbeđuju više virtuelnih mašina, tj. omogućavaju većem broju korisnika da koriste sistem istovremeno. Svaki korisnik u sistem donosi nove procese i zahteva njihovo izvršenje. U opštem slučaju više korisnika može zahtevati izvršenje svojih procesa počev od istog trenutka. Takvi procesi se moraju izvršavati paralelno ili kvaziparalelno, što na jednoprocesnim operativnim sistemima nije moguće.

1.8. Navedite primere (a) jednokorisničkih jednoprocesnih, (b) jednokorisničkih višeprocesnih i (c) višekorisničkih višeprocesnih operativnih sistema.

(a) MS-DOS

(b) OS/2, Microsoft Windows 3.x/9x/ME

(c) UNIX, Microsoft Windows 2000/XP/2003 sa terminal servisima (*terminal services*)

1.9. Navedite najznačajnije karakteristike (a) operativnih sistema za paketnu obradu, (b) interaktivnih operativnih sistema, (c) operativnih sistema sa deljenjem vremena, (d) *real-time* i (e) distribuiranih operativnih sistema.

(a) Korisnici predaju svoje poslove na izvršenje posredstvom ulaznih jedinica. Poslovi se zatim odvijaju jedan za drugim u nizu, pri čemu korisnik nema mogućnost komuniciranja sa svojim poslom.

(b) Sistem izvršava veliki broj manjih transakcija, takvih da je rezultat sledeće transakcije u opštem slučaju nepoznat.

(c) Svakom korisniku, odnosno korisničkom programu dodeljuje se jedan kvantum vremena centralnog procesora. Po isteku dodeljenog vremenskog kvantuma, proces se prekida, a procesor se dodeljuje sledećem procesu u redu čekanja.

(d) Sistem prima informacije sa senzora i daje odziv u propisanom vremenskom intervalu.

(e) Sistem deli zadatke obrade (izračunavanja) većem broju procesora koji ne dele memoriju niti sistemski časovnik. Procesori međusobno komuniciraju preko mreže.

1.10. U čemu je osnovna razlika između upitnih i transakcionih sistema?

Krajnji korisnici upitnih sistema ne menjaju sadržaj baze podataka. Krajnji korisnici transakcionih sistema menjaju sadržaj baze posle svake obavljene transakcije. Modifikacije baze podataka su česte, i uobičajeno se obavljaju nekoliko puta u sekundi.

1.11. U kom slučaju se korišćenje sistema sa deljenjem vremena smatra boljim rešenjem od korišćenja personalnog računara, odnosno radne stanice za jednog korisnika?

Korićenje sistema sa deljenjem vremena ima smisla:

- ukoliko je potrebno obezbediti deljenje istih resursa između većeg broja korisnika,
- ukoliko su zadaci obrade veliki, tako da se ne mogu obaviti na personalnom računaru u zadovoljavajućem roku.

1.12. U čemu je razlika između simetričnog (SMP) i asimetričnog multiprocesiranja u višeprocesorskim sistemima?

SMP - svi procesori su ravnopravni (nema odnosna gospodar-rob) i svaki procesor izvršava istu kopiju operativnog sistema, pri čemu te kopije međusobno komuniciraju kad god je to potrebno. U idealnom slučaju, svakom procesoru se dodeljuje jedan proces koji se izvršava nezavisno od procesa na ostalim procesorima.

Asimetrično multiprocesiranje - svakom procesoru je dodeljen specifičan posao. Glavni procesor (*master*) kontroliše ceo sistem i dodeljuje poslove ostalim procesorima (*slaves*). Ovaj koncept je karakterističan za ektremno brze i velike sisteme.

1.13. Koji je osnovni problem karakterističan za projektovanje *real-time* operativnih sistema?

Projektant operativnog sistema mora obezbediti algoritme za raspoređivanje tako da odziv operativnog sistema ne premaši propisane vremenske limite. Nepoštovanje vremenskih limita može izazvati krah *real-time* sistema.

1.14. Bazu podataka sa pratećom aplikacijom je potrebno postaviti na četiri servera. Kako se mogu udružiti ovi serveri ukoliko je potrebno obezbediti (a) visoku pouzdanost, (b) visoke performanse?

(a) Asimetrično - jedan čvor (server) opslužuje zahteve, dok su ostali u budnom, ali neaktivnom stanju. U slučaju otkaza glavnog servera, jedan od pratećih servera preuzima ulogu glavnog servera.

(b) Simetrično - svi serveri su aktivni i izvršavaju aplikaciju (bazu).

1.15. Koje funkcionalne grupe programa čine operativni sistem?

- upravljanje procesorom,
- upravljanje memorijom,
- upravljanje ulazom i izlazom,
- upravljanje podacima (datotekama),
- upravljanje sekundarnom memorijom,
- umrežavanje,
- zaštita,
- korisnički interfejs.

1.16. Zašto se u teorijskom modelu operativnog sistema (*paper-model OS*) sloj upravljanja datotekama nalazi iznad slojeva upravljanja ulazom i izlazom i upravljanja memorijom?

1.17. U čemu je razlika između monolitnih i hijerarhijskih operativnih sistema?

Operativni sistem monolitne strukture se sastoji od skupa procedura bez ikakvog grupisanja ili hijerarhije.

Operativni sistem slojevite strukture se deli na više slojeva, od kojih svaki ima tačno određenu funkciju (upravlja tačno određenim resursima) i oslanja se isključivo na funkcije nižih slojeva, kojima pristupa pomoću poziva koji je sličan sistemskim pozivima. Slojeviti operativni sistem se deli u određeni broj nivoa (N), od koji se svaki gradi na vrhu prethodnog nivoa. Najniži nivo (*layer 0*) je hardver, a najviši nivo (*layer N*) je korisnički interfejs. Sa ovakvim modularnim konceptom, svi nivoi koriste isključivo servise nižih nivoa.

1.18. a. Koja je osnovna karakteristika operativnih sistema sa mikrokernel arhitekturom ?

b. Zašto su ovakvi sistemi pouzdaniji od monolitnih?

(a) Osnovna karakteristika operativnih sistema sa mikrokernel arhitekturom je postojanje minimalnog, pouzdanog jezgra visokih performansi. Sve druge funkcije jezgra potiskuju se u takozvani korisnički prostor i realizuju kao zasebni korisnički moduli koji se jednostavno mogu dodati bez uticaja na osnovno jezgro. Moduli između sebe komuniciraju slanjem poruka (*message passing*).

(b) Ovakav sistem radi pouzdanije jer se mnogo manje koda se izvršava u sistemskom režimu.

1.19. Objasnite kako razdvajanje korisničkog i sistemskog režima funkcioniše kao poseban oblik zaštite sistema.

Programi se u korisničkom režimu (*user mode*) izvravaju na zahtev korisnika, a u sistemskom režimu (*monitor mode*, *kernel mode*, *system mode*) na zahtev operativnog sistema. Bit režima (*mode bit*) dodat računarskom hardveru određuje trenutni režim rada: *monitor* (0) ili *user* (1).

Privilegovane instrukcije (kao što su instrukcije za rad sa ulazno-izlaznim uređajima) mogu se izvršiti jedino u sistemskom režimu. Operativni sistem ima pristup privilegovanom setu instrukcija i pomoću njega uspostavlja kontrolu nad celim sistemom u svakom trenutku. Projektant operativnog sistema ne sme dozvoliti korisničkim programima da uspostave kontrolu nad računarom u korisničkom režimu - time se ostvaruje poseban oblik zaštite sistema.

1.20. Koja od sledećih instrukcija ne mora biti privilegovana: (a) postavljanje vrednosti tajmera, (b) očitavanje sistemskog časovnika, (c) brisanje memorije, (d) isključivanje prekida ili, (e) prelazak iz korisničkog u sistemski režim ?

Instrukcija pomoću koje se očitava sistemski časovnik ne mora biti privilegovana.

1.21. Sledeće instrukcije su privilegovane: (a) prelazak u korisnički režim, (b) prelazak u supervizorski režim, (c) čitanje iz zaštićene memorije, (d) upis u zatićenu memoriju, (e) prihvatanje instrukcije iz zatićene memorije, (f) sprečavanje prekida koji izaziva tajmer, (g) omogućavanje prekida koji izaziva tajmer. Odaberite minimalni skup

instrukcija koje moraju biti privilegovane, tako da se ne naruši neophodan stepen zaštite sistema.

- prelazak u supervizorski režim,
- čitanje iz zaštičene memorije,
- upis u zatićenu memoriju,
- sprečavanje prekida koji izaziva tajmer.

1.22. Koja je uloga sistemskih poziva?

Sistemski pozivi omogućavaju procesima koji rade u korisničkom režimu da zatraže uslugu operativnog sistema. Na taj način korisnički program može da zatraži izvršenje privilegovanih instrukcija, kao što je rad sa ulazno-izlaznim uređajima. Sistemski pozivi se realizuju pomoću sistema prekida: korisnički program postavlja parametre sistemskog poziva na određene memorijske lokacije ili registre procesora i inicira prekid. Operativni sistem posle prekida preuzima kontrolu, uzima parametre, izvrši tražene radnje, rezultat stavlja u određene memorijske lokacije ili registre i vraća kontrolu programu.

1.23. Koja je uloga sistemskih programa?

Sistemski programi su programi koji pomoću sistemskih poziva obezbeđuju neku osnovnu funkcionalnost korisnicima (kopiranje datoteka, rad sa ulazno-izlaznim uređajima itd.) omogućavajući im da obave neke učestale zadatke.

1.24. Kako virtuelna mašina odgovara na sistemske pozive?

Sistemske pozive korisničkih programa primaju odgovarajući operativni sistemi, a hardverske operacije koje šalju ti operativni sistemi prema svojim virtuelnim mašinama prihvata monitor virtuelnih mašina i realizuje ih u skladu sa hardverom ispod sebe.

2. Kernel, procesi i niti

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za jezgro operativnog sistema i procese. Razmotrene su strukture neophodne za upravljanje procesima, stanja kroz koja proces prolazi u toku svog "života" i odgovarajuće tranzicije u dijagramu stanja, funkcije planera

poslova i dispečera sistema i osnovni mehanizmi za komunikaciju između procesa. Takođe su obrađeni i osnovni pojmovi vezane za lake procese, odnosno niti, osnovni višenitni modeli i prednosti korišćenja višenitnih programa.

2.1. Koji su hardverski preduslovi neophodni za realizaciju jezgra operativnog sistema, odnosno za nadogradju hardvera jezgrom u hijerarhijskom modelu?

- mehanizam prekida (omogućava izvršavanje kontrolnog programa, odnosno prebacivanje kontrole sa korisničkog programa na rutinu operativnog sistema),
- zaštitni mehanizam adresiranja memorije (sprečava pogrešno adresiranje, odnosno mogućnost da jedan proces svoje podatke upiše deo memorije dodeljen drugom procesu),
- privilegovani set instrukcija (omogućavaju operativnom sistemu da maskira prekide, dodeli procesor drugom procesu, pristupi zaštićenim memorijskim adresama ili obavi ulazno-izlaznu operaciju),
- *real-time* časovnik (omogućava zakazivanje i raspoređivanje poslova).

2.2. Navedite osnovne delove jezgra prema modelu koji je definisao A.Lister.

- prvi nivo obrade prekida (rutine za određivanje uzroka prekida i iniciranje odgovarajuće prekidne rutine),
- dispečer sistema, odnosno planer poslova niskog nivoa (dodeljuje procesor procesima),
- rutine za ostvarivanje komunikacije između procesa.

2.3. Šta je proces i šta sve obuhvata?

Najjednostavnije rečeno, proces je program ili deo programa u stanju izvršavanja. Proces je aktivna dinamička celina, koji obuhvata:

- tri fundamentalne memorijske sekcije: programsku ili tekst sekciju, u kojoj se nalazi kod, srek sekciju i sekciju podataka, koja sadrži globalne promenljive,
- vrednost programskog brojača i vrednosti ostalih registara procesora koji su od interesa,

- ulazno-izlazne resurse koje eventualno koristi, kao što su, na primer, datoteke.

2.4. Šta je kontrolni blok procesa i šta je njime omogućeno?

Kontrolni blok procesa je deo operativne memorije, odnosno memorijska struktura koja sadrži kontrolne informacije neophodne za upravljanje tim procesom (jedinstveni identifikator procesa - PID, kontekst, prioritet, trenutno stanje i informacije o resursima koje proces koristi). Pomoću kontrolnog bloka omogućeno je multiprogramiranje, odnosno višestruko prekidanje i nastavak izvršenja procesa.

2.5. Ukratko opišite stanja u osnovnom dijagramu stanja procesa.

- START: nastanak procesa,
- READY: proces je spreman za rad, dobio je sve potrebne resurse osim procesora, i čeka da mu dispečer dodeli procesor,
- RUN: procesor izvršava instrukcije tekućeg procesa,
- WAIT: proces čeka na neki događaj jer su mu za dalje izvršenje potrebni neki resursi koji mu trenutno nisu na raspolaganju,
- STOP: kraj izvršenja procesa.

2.6. a. Definišite tranzicije RUN-WAIT i WAIT-READY.

b. Da li je na jednoprocesnim sistemima moguća tranzicija RUN-WAIT?

(a) RUN-WAIT: oduzimanje procesora procesu ukoliko je neki resurs neophodan za izvršenje procesa u međuvremenu postao zauzet ili nedostupan.

WAIT-READY: proces se vraća na kraj procesorskog reda posle oslobađanja resursa koji je neophodan za njegovo izvršenje.

(b) Tranzicija RUN-WAIT je moguća i u višeprocesnim i u jednoprocesnim operativnim sistemima. Na primer, u oba slučaja štampanje ne može da se nastavi dok se u štampač ne doda papir, s tim što se u slučaju višeprocesnog sistema u stanje WAIT dovodi *print spooler*, a u slučaju jednoprocesnog proces koji je inicirao štampanje.

2.7. a. Kada proces prelazi iz stanja RUN u stanje READY?

b. Kada je ova tranzicija moguća?

(a) Proces prelazi iz stanja RUN u stanje READY posle isteka vremenskog kvantuma.

(b) Ova tranzicija je moguća samo ako operativni sistem podržava pretpražnjenje.

2.8. Koji su mogući uzroci za suspendovanje procesa u proširenom dijagramu stanja procesa ?

- korisnik privremeno suspenduje proces da bi oslobodio resurse za izvršenje drugih procesa,
- operativni sistem suspenduje neke procese da bi sprečio pojavu zastoja i efekat zasićenja usled prevelike količine keširanih podataka.

2.9. Navedite situacije u kojima proces može ostati bez procesora?

- isticanje vremenskog kvantuma,
- kreiranje novog procesa (dete proces), pri čemu proces roditelj mora da sačeka da se proces dete izvrši,
- prekid,
- postavljanje zahteva za ulazno-izlaznom operacijom, posle čega se prebacuje u red čekanja na ulazno-izlazne uređaje.

2.10. Koje su osnovne funkcije (a) planera poslova i (b) dispečera sistama.

(a) Planer poslova deli poslove na procese, procesima dodeljuje prioritet na osnovu određenih algoritama i dovodi procese u red čekanja na procesor.

(b) Dispečer sistema odličuje koji će proces, kada i koliko dugo dobiti procesor. Prilikom zamene konteksta procesa, odnosno dodele procesora drugom procesu iz reda, dispečer pamti stanje procesa koji se prekida (kako bi se kasnije mogao nastaviti) i puni memoriju stanjem novog procesa kome se dodeljuje procesor.

2.11. Šta obavljaju sistemski pozivi fork i exec na UNIX sistemima?

Proces roditelj kreira novi proces pomoću *fork* sistemskog poziva. Sistemski poziv fork duplira kreira kopiju adresnog prostora roditelja i dodeljuje je detetu. Adresni prostor procesa deteta puni se programom koji treba da se izvrši pomoću sistemskog poziva *exec*.

2.12. Koji su resursi unikatni za svaku nit, a koji su zajednički sa ostalim nitima istog procesa.

Unikatni resursi: identifikator niti (*thread ID*), vrednost programskog brojača, vrednosti registra procesora, stek.

Zajednički resursi: kod sekcija, sekcija podataka, otvorene datoteke, signali.

2.13. Navedite primere u kojima višenitnih procesi postižu bolje performanse od jednonitnih procesa.

- Web server kod kog svaka nit opslužuje različit zahtev (drugog klijenta),
- aplikacija koja obavlja aktivnosti koje se mogu izdeliti na manje podzadatke koje se mogu izvršavati paralelno, kao što je množenje matrica,
- interaktivni programi koji rade u grafičkom radnom okruženju.

2.14. Navedite primere u kojima višenitnih procesi *ne* postižu bolje performanse od jednonitnih procesa.

- programi čiji se kod izvršava strogo sekvencijalno, odnosno ne može se izdeliti na manje podzadatke koje se mogu izvršavati paralelno,
- komandni ioterpreter, kao što je *Bourne-again shell*.

2.15. Koje su prednosti upotrebe višenitnih programa?

- Smanjenje vremena odziva - višenitna tehnika omogućava interaktivnim aplikacijama da nastave sa radom, čak i u slučaju da je deo programa blokiran ili da izvršava neku dugotrajnu operaciju,
- efikasnije deljenje resursa i ekonomičnost - na primer, deljenje kod segmenta omogućava da se veliki broj niti jedne aplikacije izvršava koristeći isti adresni prostor. Takođe, kreiranje procesa i prebacivanje konteksta je sporije od kreiranja niti, odnosno prebacivanja konteksta niti,

- bolje iskorišćenje višeprocesorske arhitekture.

2.16. Koje su osnovne razlike između korisničkih i kernelskih niti?

- Za razliku od korisničkih, kernelske niti ne moraju pripadati procesu.
- Raspoređivanje procesora korisničkim nitima obavlja biblioteka za rad sa korisničkim nitima, dok raspoređivanje za kernelske niti obavlja procesor.
- Kernel nije svestan korisničkih niti - potpuna podršku za rad sa korisničkim nitima obezbeđuje posebna biblioteka.

2.17. Objasnite koje akcije izvršava kernel prilikom prebacivanja konteksta između kernelskih niti ?

Prebacivanje konteksta između kernelskih niti obično obuhvata pamćenje vrednosti registara procesa tekuće niti i postavljanje vrednosti registara za nit kojoj će biti dodeljen procesor.

2.18. Koji višenitni modeli postoje?

- model više-u-jednu (više korisničkih niti se mapira u jednu kernelsku nit),
- model jedna-u-jednu (svaka korisnička nit se mapira u jednu kernelsku nit),
- model više-u-jednu (više korisničkih niti se mapira u manji ili isti broj kernelskih niti).

2.19. Koji se resursi obično troše prilikom kreiranja niti?

Kreiranje korisničke, odnosno kernelske niti, obično zahteva malu strukturu podataka u koju će biti smeštene vrednosti registara, stek i prioritet niti.

2.20. Napišite Win32 program koji generiše niz prostih brojeva na sledeći način: korisnik najpre unosi broj, a zatim posebna nit generiše rastući niz uzastopnih prostih brojeva manjih ili jednakih unetom broju..

2.21. Napišite Win32 program koji obavlja višenitnu kriptoanalizu šifrata "brutal-force" metodom. Svaka od osam niti treba da obavlja pretragu zasebne oblasti polja ključeva (ključevi su 64-bitni).

2.22. a. Koja je funkcija sistemskog poziva *clone* na Linux sistemu?

b. U čemu je razlika između *fork* i *clone* sistemskih poziva?

(a) Sistemski poziv *clone* kreira Linux niti.

(b) Oba poziva kreiraće procese decu. Sistemski poziv *fork* kreira kopiju adresnog prostora procesa roditelja i dodeljuje je detetu na korišćenje. Posle sistemskog poziva *clone*, roditelj i dete postaju niti koji dele isti adresni prostor.

2.23. Koje se metode koriste za ostvarivanje komunikacije između procesa?

- slanje poruka,
- semaforske tehnike,
- monitori,
- pozivi udaljenih procedura.

2.24. U čemu je razlika između sinhronog i asinhronog slanja i primanja poruka?

Ukoliko je slanje poruka blokirajuće, proces koji šalje poruke se blokira, dok drugi proces kom je ta poruka namanjena ne primi poruku. Ukoliko je slanje ne-blokirajuće, proces koji šalje poruku nastavlja svoje aktivnosti ne čekajući potvrdu o prijemu. Ukoliko je primanje poruke blokirajuće, proces koji prima poruku se blokira sve dok poruku ne dobije. Ukoliko je primanje poruke neblokirajuće, proces će pokušati da primi poruku; ako je poruka stigla, prima je, a ako nije, kao rezultat se prihvata prazan niz, a proces nastavlja da radi dalje bez blokade

2.25. U čemu je razlika između direktne i indirektne komunikacije?

Direktna komunikacija:

- između para procesa koji žele da komuniciraju automatski se uspostavlja link,
- link se uspostavlja samo za dva procesa.

Indirektna komunikacija:

- link se uspostavlja između para procesa koji dele sanduče,
- linku se mogu pridružiti više od dva procesa,

- između para procesa može postojati više različitih linkova; svaki link odgovara jednom sandučetu.

2.26. Šta je priključak (*socket*) i pomoću kojih se parametara definiše?

Socket se definiše kao krajnja tačka komunikacije. Par procesa koji komuniciraju preko mreže formiraju par priključaka, po jedan za svaki proces, na svakoj strani mreže. *Socket* se definiše pomoću IP adrese računara na kome se formira i broja porta. Mehanizam priključaka funkicioniše na principu klijent-server arhitekture: server osluškuje portove analizirajući da li je stigao neki zathev od klijenata na tom portu. Kada zahtev stigne, server pruža odgovarajući servis.

2.27. Šta su pozivi udaljenih procedura i šta je njima omogućeno?

Pomoću poziva udaljenih procedura procesu se omogućava da pozove proceduru na drugom računaru. Komunikacija između procesa ostvaruje se slanjem poruka, parametara i rezultata obrade između *stub* struktura formiranih na klijentskoj i serverskoj strani.

3. Raspoređivanje procesa i dodela procesora

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za raspoređivanje procesa i dodelu procesora. Razmotreni su značajniji algoritmi prema kojima planer poslova niskog nivoa (dispečer) bira proces iz reda čekanja na procesor. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

3.1. Algoritmom za raspoređivanje procesa određen je red izvršenja procesa koji čekaju na dodelu procesora. Ukoliko se na sistemu nalazi n procesa, koliko postoji različitih rasporeda za izvršenje procesa?

Postoji $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$ mogućih rasporeda.

3.2. Šta je svrha multiprogramiranja?

Svrha multiprogramiranja je u tome da na sistemu uvek postoji proces koji se izvršava, odnosno da se procesor maksimalno iskoristi.

3.3. U čemu je razlika između raspoređivanja sa pretpražnjenjem i raspoređivanja bez pretpražnjenja ?

U slučaju raspoređivanja sa pretpražnjenjem, procesor se može oduzeti procesu koji nije završio svoje aktivnosti (na primer, ukoliko istekne vremenski kvantum za dodelu procesora). U slučaju raspoređivanja bez pretpražnjenja, to nije moguće.

3.4. a. Pod kojim okolnostima se procesor dodeljuje drugom procesu?

b. Koje od tih okolnosti zahtevaju pretpražnjenje?

(a) Procesor se dodeljuje drugom procesu ukoliko:

- tekući proces pređe u stanje čekanja resurs (na primer, čeka kraj ulazno-izlazne operacije koju je inicirao),
- proces roditelj čeka da proces dete završi svoje aktivnosti,
- tekući proces završi svoje aktivnosti,
- tekućem procesu istekne vremenski kvantum,

(b) Dodela procesora nakon isticanja vremenskog kvantuma.

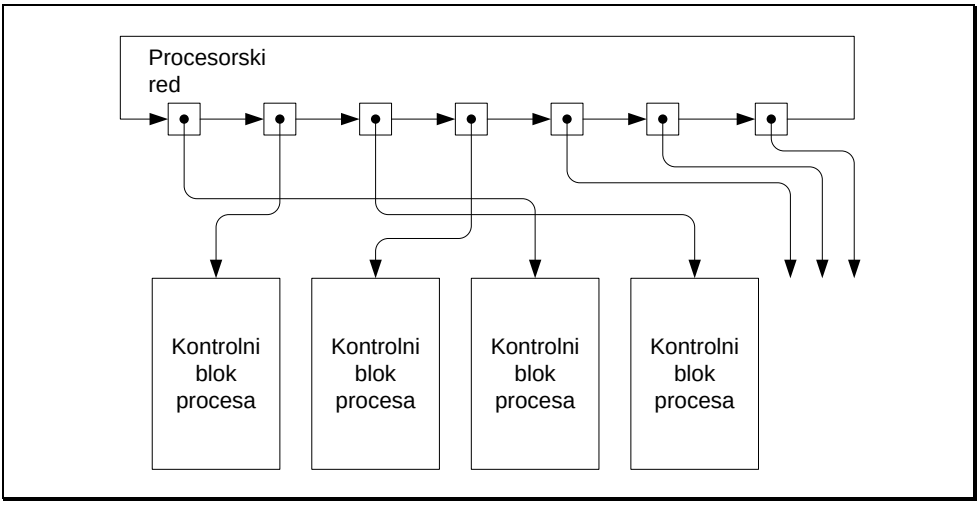
3.5. Koji kriterijumi utiču na izbor algoritama za raspoređivanje procesa?

- iskorišćenje procesora,
- propusna moć sistema,
- vreme potrebno za kompletiranje procesa (*turnaround time*),
- vreme čekanja u redu za dodelu procesora (*waiting time*),
- vreme odziva (*response time*).

3.6. Pretpostavite da je *Round Robin* (RR) algoritam za raspoređivanje implementiran tako da su članovi reda za dodelu procesora pokazivači na kontrolne blokove procesa (slika 3.1).

a. Kakav se efekat postiže stavljanjem dva pokazivača na isti proces u red čekanja na procesor?

b. Kako se može modifikovati osnovni RR algoritam tako da se isti efekat postigne bez uvođenja dvostrukih pokazivača?



Slika 3.1 *Procesorski red koji čine pokazivači na PCB*

(a) Postiže se efekat sličan povećanju prioriteta procesa, jer proces češće dobija procesor na korićenje.

(b) Uvođenjem dva kvantuma različita vremenska kvantuma, pri čemu se veći kvantum dodeljuje procesima sa višim prioritetom, a manji procesima sa nižim prioritetom.

3.7. Pretpostavite da su na sistem u kom se raspoređivanje vrši u više procesorskih redova uvedene različite veličine vremenskih kvantuma za svaki red. Kako u tom slučaju treba rasporediti procese po redovima? Kakav je uticaj na efikasnost operativnog sistema?

Procese koji zahtevaju češće opsluživanje od strane korisnika treba smestiti u red sa manjim kvantumom. Procese koji ne zahtevaju često opsluživanje treba smestiti u red sa većim kvantumom, tako da se ukupan broj prebacivanja konteksta smanji.

Smanjivanjem ukupnog broja prebacivanja konteksta povećava se efikasnost operativnog sistema.

3.8. Kako se rešava problem zakucavanja procesa niskog prioriteta ukoliko se raspoređivanje procesa obavlja na osnovu prioriteta procesa?

Procesima se povećava prioritet sa vremenom provedenim u redu čekanja na procesor. Rezultujući prioritet se formira na osnovu početnog prioriteta koji proces dobija kada uđe u red čekanja na procesor i vremena provedenog u redu.

3.9. Posmatrajte algortiram sa pretpražnjenjem zasnovan na dinamički izmenljivim prioritetima, takav da se viši prioritet označava većim brojevima prioriteta. Svim procesima se dodeljuje inicijalni prioritet 0 kada uđu u red čekanja na procesor. Ukoliko proces čeka na procesor (stanje READY), prioritet mu se menja stepenom d1, a ako se izvršava (stanje RUN), prioritet se menja stepenom d2. Algoritam se prilagođava izmenom parametara d1 i d2.

a. Koji se algoritam dobija ukoliko je d2>d1>0?

b. Koji se algoritam dobija ukoliko je d1<d2<0?

(a) FCFS (*First come, first served*)

(b) LIFO (*Last in, first out*)

3.10. Navesti parametre kojima se definišu vienivovsko raspoređivanje sa povratnom spregom između redova čekanja na procesor (MLFQ - *multilevel feedback queue*)?

- broj redova,
- algoritam za raspoređivanje za svaki red,
- kriterijumi za određivanje reda u koji će proces ući nakon kreiranja,
- kriterijumi za određivanje kada proces može preći u red višeg ili nižeg prioriteta.

3.11. U kakvoj se vezi (ukoliko veza uopšte postoji) nalaze algoritmi za raspoređivanje na osnovu prioriteta i SJF (*shortest job first*)?

Najkraći posao ima najviši prioritet.

3.12. Zašto je poželjno da algoritam za raspoređivanje procesa češće dodeljuje procesor procesima koji dominantno koriste U/I ulazno-izlazne uređaje (*I/O-bound*)? Da li se na takav način stvara mogućnost zakucavanja procesa koji dominantno koriste procesor (*CPU-bound*)?

Procesi koji dominantno koriste U/I koriste procesor u relativno kratkim intervalima. Ukoliko im se procesor odmah dodeli, ovi procesi će brzo preći u

stanje čekanja na resurs i oslobodiće procesor, tako da mogućnost zakucavanja *CPU-bound* procesa praktično ne postoji.

3.13. Kako veličina vremenskog kvantuma utiče na performanse RR algoritma?

U slučaju velikih kvantuma RR prelazi u FCFS algoritam. U slučaju ultra malih vremenskih kvantuma, svaki proces se izvršava brziom od manjom od 1/n brzine realnog procesora. Što su kvantumi manji, broj prebacivanja konteksta je veći, pa će samim tim i efektivna brzina izvršavanja procesa biti manja.

3.14. Dat je sledeći skup procesa čija su vremena izvršavanja na procesoru (*CPU-burst time*, izražena u milisekundama) i prioriteti dati u sledećoj tabeli:

Proces	Vreme izvršavanja	Prioritet
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Procesi su u sistem naišli u poretku P1, P2, P3, P4, P5, svi približno u trenutku t=0.

a. Nacrtati *Gannt*-ove karte dodele procesora ukoliko se raspoređivanje vrši na osnovu sledećih algoritama: FCFS, SJF bez pretpražnjenja, raspoređivanje na osnovu prioriteta bez pretpražnjenja (manji broj znači veći prioritet) i RR sa kvantumom Q=1.

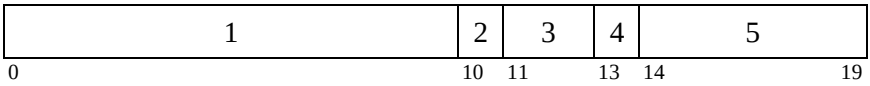
b. Odrediti vreme potrebno za kompletiranje procesa (*tournaround time*) za svaki proces (za sve gore pomenute algoritme).

c. Odrediti vreme čekanja za svaki proces i srednje vreme čekanja (za sve gore pomenute algoritme). Za koji je algoritam srednje vreme čekanja najmanje ?

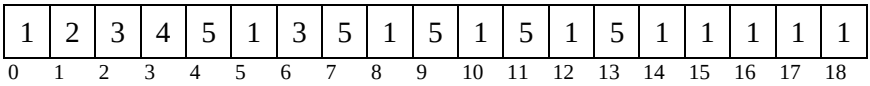
Kašnjenje dispečera zanemariti.

(a) *Gannt*ove karte dodele procesora:

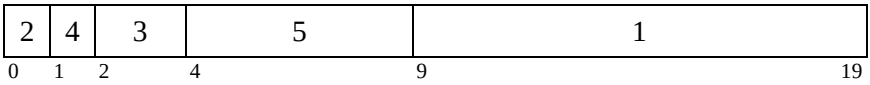
FCFS:



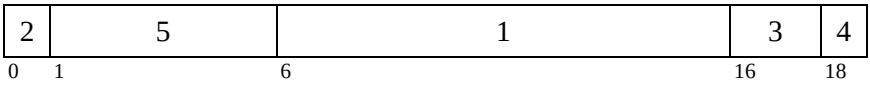
RR (Q = 1):



SJF (bez pretpražnjenja):



Rapoređivanje na osnovu prioriteta (bez pretpražnjenja):



(b) Vreme potrebno za kompletiranje procesa:

	FCFS	**RR**	**SJF**	**Prioritetno**
P1	10	19	19	16
P2	11	2	1	1
P3	13	7	4	18
P4	14	4	2	19
P5	19	14	9	6

(c) Vreme čekanja

	FCFS	**RR**	**SJF**	**Prioritetno**
P1	0	9	9	6
P2	10	1	0	0
P3	11	5	2	16
P4	13	3	1	18
P5	14	9	4	1
sr.vreme[1]:	9.6	5.4	3.2	8.2

Srednje vreme čekanja je najmanje u slučaju SJF bez pretpražnjenja.

3.15. Dat je sledeći skup procesa čija su vremena nailaska u sistem (*arrival time*) i vremena izvršavanja na procesoru, izražena u milisekundama data u sledećoj tabeli:

Process	Vreme nailaska	Vreme izvršavanja
P1	0	8
P2	0.4	4
P3	1	1

Nacrtati *Gannt*-ovu kartu i odrediti srednje vreme potrebno za kompletiranje procesa i srednje vreme čekanja[2] ukoliko se raspoređivanje procesa obavlja po:

a. FCFS algoritmu,

b. SJF algoritmu (bez pretpražnjenja),

c. SJF algoritmu sa vremenom čekanja na procese (*idle time*) t_{idle}=1,

d. SRTF (*shortest remaining time first*) algoritmu, odnosno SJF algoritmu sa pretpražnjenjem.

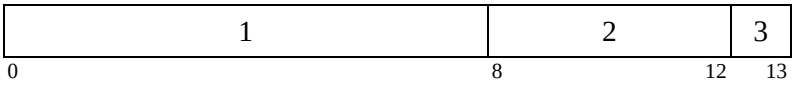
Kašnjenje dispečera zanemariti.

---

[1] Srednje vreme čekanja se računa kao srednja vrednost vremena čekanja za sve procese.

[2] Obratite pažnju: vreme potrebno za kompletiranje procesa se u ovom slučaju računa kao razlika vremena u kom je proces završio sve aktivnosti i vremena nailaska u sistem. Slično važi i za vreme čekanja, koje se računa kao razlika vremena u kom je proces dobio procesor i vrmena nailaska procesa u sistem

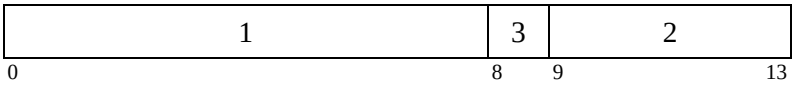
(a) *Gannt*-ova karta dodele procesora - FCFS algoritam:



Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	8	0
P2	12 - 0.4 = 11.6	8 - 0.4 = 7.6
P3	13 - 1 = 12	12 - 1 = 11
srednje vreme:	$\frac{8+11.6+12}{3} = 10.53$	$\frac{0+7.6+11}{3} = 6.2$

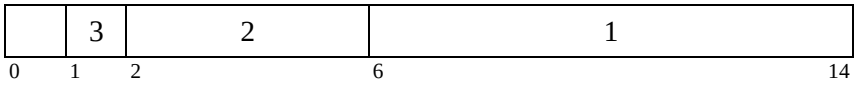
(b) *Gannt*-ova karta dodele procesora - SJF algoritam:



Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	8	0
P2	13 - 0.4 = 12.6	9 - 0.4 = 8.6
P3	9 - 1 = 8	8 - 1 = 7
srednje vreme:	$\frac{8 + 12.6 + 8}{3} = 9.53$	$\frac{0 + 8.6 + 7}{3} = 5.2$

(c) *Gannt*-ova karta dodele procesora - SJF algoritam sa vremenom čekanja na procese [3] (*idle time*) t_{idle}=1.

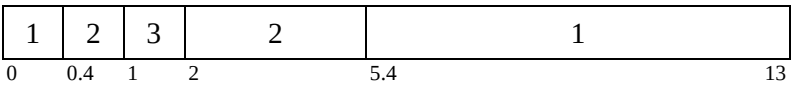


[3] Vreme čekanja na procese (*idle time*) je vremenski interval u kom planer poslova niskog nivoa čeka da u sistem stigne još nekoliko procesa. Procesima koji u sistem uđu u tom intevalu povećaće se vreme čekanja, ali će se u isto vreme povećati i broj procesa u redu, a samim tim i perfromanse raspoređivanja.

Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	14	6
P2	6 - 0.4 = 5.6	2 - 0.4 = 1.6
P3	2 - 1 = 1	0
srednje vreme:	$\frac{14 + 5.6 + 1}{3} = 6.86$	$\frac{6 + 1.6 + 0}{3} = 2.53$

(d) *Gannt*-ova karta dodele procesora - SRTF algoritam



Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	13	5.4 - 0.4 = 5
P2	5.4 - 0.4 = 5	2 - 1 = 1
P3	2 - 1 = 1	0
srednje vreme:	$\frac{13 + 5 + 1}{3} = 6.33$	$\frac{5 + 1 + 0}{3} = 2$

3.16. Dat je sledeći skup procesa čija su vremena nailaska u sistem (*arrival time*) i vremena izvršavanja na procesoru, izražena u milisekundama data u sledećoj tabeli:

Process	Vreme nailaska	Vreme izvršavanja
P1	0	7
P2	2	4
P3	4	1
P4	5	4

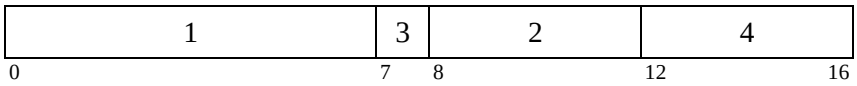
Nacrtati *Gannt*-ovu kartu i odrediti srednje vreme potrebno za kompletiranje procesa i srednje vreme čekanja ukoliko se raspoređivanje procesa obavlja po:

a. SJF algoritmu.

b. SRTF algoritmu.

Kašnjenje dispečera zanemariti.

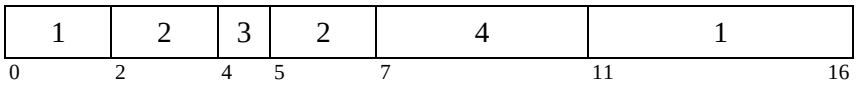
(a) *Gannt*-ova karta dodele procesora - SJF algoritam:



Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	7	0
P2	12 - 2 = 10	8 - 2 = 6
P3	8 - 4 = 4	7 - 4 = 3
P4	16 - 5 = 7	12 - 5 = 7
srednje vreme:	7	4

(b) *Gannt*-ova karta dodele procesora - SRTF algoritam:



Vremena potrebna za kompletiranje procesa i vremena čekanja su:

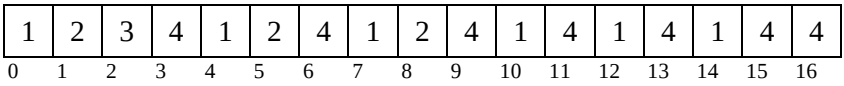
	vreme kompletiranja	**vreme čekanja**
P1	16	11-2 = 9
P2	7 - 2 = 5	5 - 4 = 1
P3	5 - 4 = 1	0
P4	11 - 5 = 6	7 - 5 = 2
srednje vreme:	7	3

3.17. Četiri procesa su u trenutku t=0 ušli u red čekanja na procesor u sledećem redosledu: P1, P2, P3, P4. Vremena izvršavanja na procesoru za ova četiri procesa su 6, 3, 1 i 7 vremenskih jedinica, respektivno. Ukoliko se raspoređivanje procesa vrši prema Round Robin algoritmu sa kvantumom (a) Q=1, (b) Q=2, (c) Q=3, (d) Q=4, (e) Q=5, (f) Q=6, (g) Q=7

- nacrtati *Gannt*-ovu kartu i odrediti srednje vreme izvršavanja na procesoru i srednje vreme čekanja (kašnjenje dispečera zanemariti),

- odrediti koliko puta je obavljena zamena konteksta i koliko je ukupno vremena potrebno da sva četiri procesa završe aktivnosti (kašnjenje dispečera je dl=0.01 vremenskih jedinica).

(a) *Gannt*-ova karta dodele procesora za slučaj RR, Q=1:

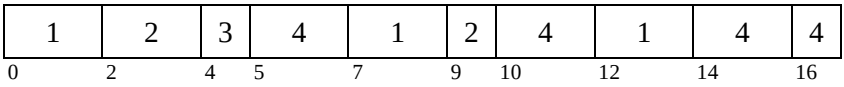


Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	15	3+2+2+1+1 = 9
P2	9	1+3+2 = 6
P3	3	2
P4	17	3+2+2+1+1+1=10
srednje vreme:	11	6.75

Obavljeno je 16 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 16 \cdot 0.01 = 17.16$.

(b) *Gannt*-ova karta dodele procesora za slučaj RR, Q=2:

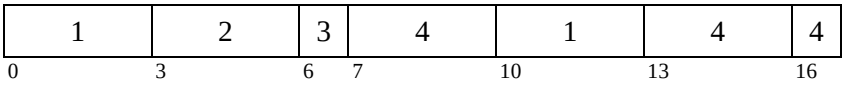


Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	14	5+3 = 8
P2	10	2+5 = 7
P3	5	4
P4	17	5+3+2=10
srednje vreme:	11.5	7.25

Obavljeno je 9 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 9 \cdot 0.01 = 17.09$.

(c) *Gannt*-ova karta dodele procesora za slučaj RR, Q=3:

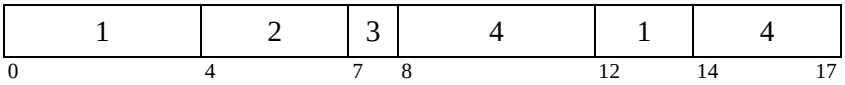


Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	13	7
P2	6	3
P3	7	6
P4	17	7+3=10
srednje vreme:	10.75	6.5

Obavljeno je 6 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 6 \cdot 0.01 = 17.06$.

(d) *Gannt*-ova karta dodele procesora za slučaj RR, Q=4:

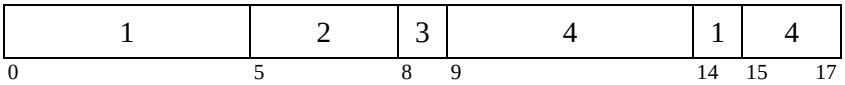


Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	14	8
P2	7	4
P3	8	7
P4	17	8+2=10
srednje vreme:	11.50	7.25

Obavljeno je 5 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 5 \cdot 0.01 = 17.05$.

(e) *Gannt*-ova karta dodele procesora za slučaj RR, Q=5:

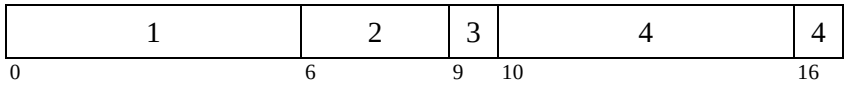


Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	15	9
P2	8	5
P3	9	8
P4	17	9+1=10
srednje vreme:	12.25	8

Obavljeno je 6 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 5 \cdot 0.01 = 17.05$.

(f) *Gannt*-ova karta dodele procesora za slučaj RR, Q=6:

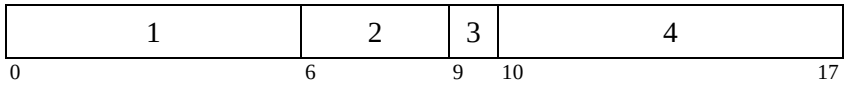


Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	6	0
P2	9	6
P3	10	9
P4	17	10
srednje vreme:	10.5	6.25

Obavljene su je 4 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 4 \cdot 0.01 = 17.04$.

(g) *Gannt*-ova karta dodele procesora za slučaj RR, Q=7:



Vremena potrebna za kompletiranje procesa i vremena čekanja su:

	vreme kompletiranja	**vreme čekanja**
P1	6	0
P2	9	6
P3	10	9
P4	17	10
srednje vreme:	10.5	6.25

Obavljeno je 3 zamena konteksta, tako da je ukupno vreme potrebno da sva četiri procesa završe aktivnosti: $t = 17 + 4 \cdot 0.01 = 17.03$.

4. Sinhronizacija procesa

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za sinhronizaciju procesa. Razmotreni su različiti načini realizacije kritične sekcije, kao i značajniji mehanizmi sinhronizacije procesa: semafori i monitori. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

4.1. a. Šta je to stanje trke (*race condition*)?

b. Od čega zavise vrednosti deljenih podataka nakon završene trke?

c. Da li je krajnji rezultat trke izvestan ili neizvestan?

d. Kako se može sprečiti pojava trke?

(a) Stanje trke je situacija u kojoj veći broj procesa konkurentno pristupa zajedničkim podacima.

(b) Krajnje vrednosti zajedničkih promenljivih zavise od toga koji će proces poslednji završiti sa radom, odnonso od sekvence naredbi kojima se ti podaci modifikuju, što dalje zavisi od redosleda prekidnih signala i načina raspređivanja procesa.

(c) U trci se pobednik obično ne zna, tako da je krajnji rezultat neizvestan.

(d) Sinhronizacijom procesa.

4.2. a. Koja je posledica moguća ukoliko procesi nesinhronizovano pristupaju zajedničkim podacima?

b. Šta je kritična sekcija?

c. Ko razrešava problem kritične sekcije?

(a) Nekonzistentnost zajedničkih podataka.

(b) Kritična sekcija je deo programskog koda u kome proces pristupa zajedničkim podacima ili modifikuje zajedničke podatke (kao što su vrednosti memorijskih lokacija, datoteke, itd.)

(c) Programer.

4.3. Kako se može realizovati kritična sekcija?

- softverski,
- hardverski,
- pomoću sistemskih poziva operativnog sistema,
- višim programskim strukturama za sinhronizaciju, kao što su monitori.

4.4. Koje su osnovne pretpostavke softverske realizacije kritične sekcije?

- međusobno isključenje,
- proces van kritične sekcije ne sme sprečiti druge procese da uđu u kritičnu sekciju,
- proces ne sme neograničeno dugo da čeka na ulazak u kritičnu sekciju,
- proces ne sme neograničeno dugo da ostane u svojoj kritičnoj sekciji,

4.5. a. Objasnite funkcije ulazne, kritične i izlazne sekcije.

b. Za sledeći kod odrediti naredbe koje pripadaju ulaznoj, izlaznoj i kritičnoj sekciji. Deljiva promenljiva je *share*.

```
while (busy); busy = 1; share++; busy=0;
```

(a) ulazna sekcija - deo koda u kome proces zahteva od sistema da uđe u kritičnu sekciju,

kritična sekcija - deo koda u kome proces pristupa ili modifikuje vrednosti zajedničkih podataka,

izlazna sekcija - deo koda u kome proces obaveštava ostale procese da je napustio svoju kritičnu sekciju.

(b) ulazna sekcija: `while (busy);`

kritična sekcija: `busy=1; share++;`

izlazna sekcija: `busy=0;`

4.6. Objasnite ukratko ideje na kojima su zasnovana značajnija softverska rešenja kritične sekcije: (a) algoritam striktne alternacije, (b) *Dekker-Petersonov* algoritam, (c) pekarski algoritam. Na koliko su procesa ovi algoritmi realno primenljivi ?

(a) Samo onaj proces čija je vrednost indeksa jednaka vrednosti zajedničke promenjive za sinhronizaciju (*turn*) može ući u svoju kritičnu sekciju. Algoritam je primenljiv za sinhronizaciju dva procesa.

(b) Binarna promenljiva *flag* se deklariše za svaki proces pojedinačno i ukazuje na to da li proces želi ili ne želi da uđe u svoju kritičnu sekciju. Zajednička promenljiva *turn* ukazuje na proces koji ima prednost ulaska u kritičnu sekciju i obezbeđuje mehanizam međusobnog isključenja. Algoritam je primenljiv za sinhronizaciju dva procesa.

(c) Svakom procesu se dodeljuje broj (procesima se redom dodeljuju sve veći brojevi). Proces ostaje u ulaznoj sekciji sve dok ne postane proces sa najmanjim brojem, odnosno najviim prioritetom, a zatim ulazi u kritičnu sekciju.

4.7. Da li je sledeći algoritam za rešenje problema kritične sekcije za dva procesa ispravan ili ne? Obrazložite rešenje.

```
int flag[2]; flag[0]=flag[1]=0; int share,

int P0 {

   while (1) {flag[0]=1; while (flag[1]); share++;
   flag[0]=false;}

}

int P1 {

   while (1) {flag[1]=1; while (flag[0]); share++;
   flag[1]=false;}

}
```

Napomena: kod za procese P1 i P2 sadrži samo ulazne, kritične i izlazne sekcije, koje su od značaja za analizu. Zajednička promenljiva je share.

Rešenje je, na prvi pogled, dobro, jer je zadovoljeno međusobno isključenje, a procesi van kritičnih sekcija ne utiču jedan na drugog. Sledeća sekvenca pokazuje da je ovaj algoritam pogrešan:

- u trenutku t_0 proces P0 postavlja svoj fleg: flag[0]=1,
- dispečer zatim dodeljuje procesor procesu P1,
- u trenutku t_0 proces P1 postavlja svoj fleg:, flag[1]=1.

Posle toga, oba procesa upadaju u beskonačnu petlju, jer su oba flega postavljena na 1, tako da nijedan proces ne može da uđe u svoju kritičnu sekciju.

4.8. Koja je instrukcija mikroprocesora neophodna za realizaciju kritičnih sekcija na nivou hardvera ?

Potrebna je nedeljiva instrukcija pomoću koje se može:

- pročitati i izmeniti memorijska reč ili
- razmeniti sadržaj dve memorijske reči.

4.9. U čemu se ogleda nedeljivost operacija *signal* i *wait* ?

- Operacije *signal* i *wait* se ne mogu podeliti na više ciklusa.
- Dva procesa ne mogu istovremeno izvršavati ove operacije nad istim semaforom.

4.10. Pokazati na primeru da, ukoliko operacije *signal* i *wait* nisu nedeljive, međusobno isključenje može biti narušeno.

4.11. a. Šta je binarni semafor? Navedite primer binarnog semafora.

b. Šta je brojački semafor? Navedite primer brojačkog semafora.

(a) Binarni semafori su specijalna klasa semafora, čija vrednost može biti samo 0 ili 1. Primer binarnog semafora je semafor *mutex* koji obezbeđuje mehanizam međusobnog isključenja.

(b) Semafori čije vrednosti mogu biti 0, 1, 2, ... su brojački semafori. Primer brojačkog semafora je semafor koji dopušta određeni broj konekcija na mrežni server.

4.12. Procesi P1 i P2 se izvršavaju kvaziparalelno. Obezbediti pomoću semafora da se instrukcija Ins2() iz procesa P2 izvrši posle istrukcije Ins1() iz procesa P1.

Problem se može rešiti pomoću binarnog semafora čija je inicijalna vrednost 0. Neka se dati semafor zove *proceed*. Kod za procese P1 i P2 je:

```
int P1 { ... ; Ins1(); signal(proceed); ... }
int P2 { ... ; wait(proceed); Ins2(); ... }
```

4.13. a. Šta znači kada se kaže da je proces "zauzet čekanjem" (*busy waiting*) ?

b. Kako se ovaj nedostatak može otkloniti ?

(a) Pojava koja se zove *busy waiting* (zaposleno čekanje) nedostatak je softverske i hardverske realizacije kritične oblasti i semafora. Proces ne radi ništa korisno, već u *while* petlji proverava vrednost neke promenljive kako bi saznao da li može ući u svoju kritičnu oblast ili ne (i tako troši procesorsko vreme).

(b) Pojava "zaposlen čekanjem" može se ukloniti uvođenjem proširene definicije semafora. Semafor se definiše kao struktura koju čine vrednost semafora (koja može biti i negativna) i semaforski red (lista pokazivača na procese koji čekaju na semaforu). Proces koji izvršava *wait* operaciju nad semaforom, čija je vrednost 0, blokira se i prevodi u semaforski red. Proces oslobađa procesor koji se predaje nekom drugom procesu koji nije blokiran.

4.14. a. Kada nepravilno raspoređene operacije signal i *wait* mogu da dovedu do zastoja?

b. Navesti primer za to.

(a) Nepravilno raspoređene operacije signal i *wait* mogu da dovedu do zastoja ukoliko jedan ili više procesa beskonačno dugo čekaju na događaje koji se nikada neće dogoditi.

(b) Neka su S i Q dva binarna semafora čija je inicijalna vrednost 1. Ukoliko se procesi P1 i P2, čiji je kod dole naveden, izvršavaju kvaziparalelno, doći će do zastoja:

```
int P1() {wait(S); wait(Q); .. ; signal(S); signal(Q)}
int P2() {wait(Q); wait(S); .. ; signal(Q); signal(S)}
```

4.15. Sledeći kod sadrži tipičnu programersku grešku u radu sa semaforima.

```
signal(mutex); kritična sekcija; wait(mutex);
```

Šta će se dogoditi u ovom slučaju?

Više procesa će moći da uđe u svoje krtitične sekcije istovremeno. Dalji razvoj situacije i krajnji rezultat su nepredvidivi.

4.16. Sledeći kod sadrži tipičnu programersku grešku u radu sa semaforima.

```
wait(mutex); kritična sekcija; wait(mutex);
```

Šta će se dogoditi u ovom slučaju?

Umesto oslobađanja semafora, nastupiće zastoj.

4.17. Semaforskim tehnikama rešite problem ograničenog bafera (problem je opisan u knjizi "Operativni sistemi: koncepti", strana 83).

Za sinhronizaciju se koriste tri semafora:

- brojački semafor item_available, kojim se obezbeđuje da potrošač ne može uzeti ništa iz bafera dok proizvođač to ne stavi u bafer. Vrednosti semafora pripadaju intervalu [0,N-1], a inicijalna vrednost je 0;
- brojački semafor space_available, kojim se obezbeđuje da potrošač ne može uzeti ništa iz bafera dok proizvođač to ne stavi u bafer. Vrednosti semafora pripadaju intervalu [0,N-1], a inicijalna vrednost je 1;
- binarni semafor buffer_ready kojim se bafer štiti kao nedeljivi resurs. Inicijalna vrednost semafora je 1.

Radi jednostavnijeg rešenja, operacije kojima se jedan element stavlja u bafer i uzima iz bafera nazvaćemo *deposit* i *extract*. Kod za proizvođački (*producer*) i potrošački (*consumer*) proces je:

```
int producer() {
     do {
          item next_produced;
```

```
          wait (space_available);
          wait (buffer_ready):
               deposit (next_produced);
          signal (buffer_ready);
          signal (item_available);
     }    while (1);
}

int consumer() {
     do {
          wait (item_available);
          wait (buffer_ready):
               item next_consumed = extract();
          signal (buffer_ready);
          signal (space_available);
     }    while (1);
}
```

4.18. Semaforskim tehnikama rešite problem čitalaca i pisaca (problem je opisan u knjizi "Operativni sistemi: koncepti", str. 84).

Za sinhronizaciju se koriste dva binarna semafora:

- *write*, koji obezbeđuje međusobno isključenje procesa pisaca i čitalaca (inicijalna vrednost je 1),
- *counter_ready*, koji štiti promenljivu *read_count*, koja ukazuje na broj aktivnih čitalaca (inicijalna vrednost semafora je 1, a promenljive 0).

Kod procesa pisaca je:

```
int writer() {
     do {
          wait (write);
               /* proces menja sadržaj objekta */
          signal (write);
     }    while (1);
}

int reader() {
     do {
          wait (counter_ready);
               readcount++;
               if (readcount==1) wait (write);
          signal (counter_ready);
          /* proces čita sadržaj objekta */
          wait (counter_ready);
               readcount--;
               if (readcount==0) signal (write);
          signal (counter_ready)
     }    while (1);
```

```
}
```

4.19. Semaforskim tehnikama rešite problem večere filozofa (problem je opisan u knjizi "Operativni sistemi: koncepti", strana 86). Obezbedite rešenje koje ne izaziva zastoj ni u jednoj situaciji.

Svaki filozof predstavlja proces, a svaka viljuška binarni semafor *fork*[i] čija je inicijalna vrednost 1. Zastoj je moguć ukoliko svi filozofi odjednom uzmu levu viljušku. Zato ćemo dozvoliti da filozof uzme viljuške samo ako su obe slobodne - uzimanje viljušaka proglasićemo kritičnom sekcijom. Kritičnu sekciju štitimo binarnim semaforom *taking_forks* (inicijalna vrednost 0).

Kod za sve filozofe je:

```
int philosopher_i() {
     do {
          wait (taking_forks);
               wait (fork[i]);
               wait (fork[(i+1)%5]);
          signal (taking_forks);
          /* filozof jede */
          wait (taking_forks);
               signal (fork[i]);
               signal (fork[(i+1)%5]);
          signal (taking_forks);
          /* filozof misli */
     }    while (1);
}
```

4.20. [Problem uspavanog berberina]. Berbernica se sastoji od čekaonice sa n stolica i radne sobe sa berberskom stolicom. Ukoliko nema mušterija koje zahtevaju uslugu, berberin će zaspati. Ukoliko mušterija uđe u berbernicu i vidi da su sve stolice zauzete, izaći će napolje. Ukoliko je berberin zauzet ali u čekaonici ima slobodnih stolica, mušterija će sesti na jednu od njih i sačekati. Ukoliko berberin spava, mušterija će ga probuditi. Napisati program kojim će aktivnosti berberina i mušterija biti sinhronizovane.

4.21. [Problem pušača]. Na sistemu postoje tri procesa koja predstavljaju pušače i jedan proces koji predstavlja dobavljača. Svaki pušač u beskonačnoj petlji najpre mota cigaretu, a zatim je pali. Da bi pušač smotao cigaretu, potrebni su mu sledeći sastojci: duvan, rizla i šibice. U početnom stanju jedan od pušača ima rizlu, drugi duvan, a treći šibice. Dobavljač ima neograničene zalihe ovih sastojaka, ali im

donosi samo po dva sastojka. Pušač koji ima preostali sastojak smotaće cigaretu i signalizirati dobavljaču da je završio cigaretu. Dobavljač će zatim ponovo doneti dva sastojka, čime se.ciklus ponavlja. Napisati program kojim će aktivnosti pušača i dobavljača biti sinhronizovane.

4.22. Neka se na sistemu nalazi n procesa kojima su dodeljeni brojevi koji označavaju prioritete. Ni jedan od dva procesa nemaju isti prioritet. Napišite monitor za dodelu tri štampača ovim procesima. Štampači se dodeljuju na osnovu prioriteta procesa.

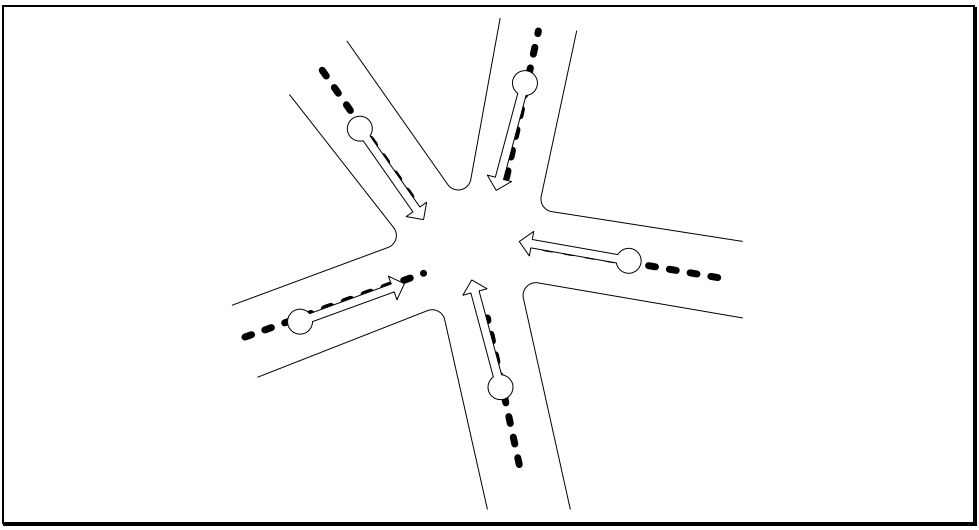
4.23. Datoteku deli veći broj procesa kojima su dodeljeni različiti numerički identifikatori. Datoteci može istovremeno pristupiti veći broj procesa, pod uslovom da je suma njihovih identifikatora manja od n. Napišite monitor za sinhronizaciju pristupa datoteci.

5. Zastoj

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za zastoj (*deadlock*) i uslove pod kojima se zastoj može pojaviti. Razmotrene su osnovne metode upravljanja zastojem i neki značajniji algoritmi, kao što je bankarski algoritam. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

5.1. Navedite nekoliko primera zastoja koji nijsu vezani za računarski sistem.

- Dve osobe, od kojih se jedna penje uz merdevine, a druga spušta niz merdevine (problem dva jarca na brvnu),
- dva voza koja se kreću jedan prema drugom istom šinom,
- zvezdasta raskrsnica (slika 5.1), u kojoj lako dolazi do zastoja.



Slika 5.1. *Primer zastoja nevezan za računarski sistem*

5.2. Pod kojim uslovima nastupa zastoj?

Zastoj nastupa ako su istovremeno ispunjena sledeća četiri uslova

(1) pri korišćenju resursa poštuje se princip međusobnog isključenja,

(2) dodela resursa obavlja se bez pretpražnjenja - procesu se ne može oduzeti resurs,

(3) proces zadržava jedan resurs jer mu je potreban i čeka na resurs koji koristi neki drugi proces,

(4) kružno čekanje (proces P0 čeka na resurs koji drži P1, proces P1 čeka na resurs koji drži P1, ..., proces Pn-1 čeka na resurs koji drži Pn, proces Pn čeka na resurs koji drži P0).

5.3. Da li je moguće da se sistem nađe u stanju zastoja ukoliko na sistemu postoji samo jedan proces? Objasnite.

Nije moguće. Ovo je direktna posledica uslova zadržavanja resursa i čekanja na drugi.

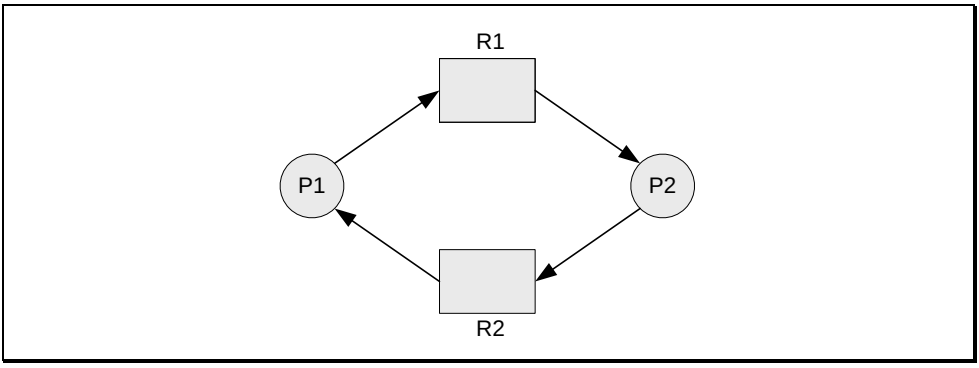
5.4. Sistem čine četiri resursa istog tipa (svaki resurs ima jednu instancu) i tri procesa. Svaki proces zahteva najviše dva resursa. Da li se sistem nalazi u stanju zastoja ?

Pretpostavimo da se se sistem nalazi u stanju zastoja. U tom slučaju, svaki proces bi morao da drži jedan resurs i zahteva još jedan koji nije dostupan. Međutim, kako na sistemu ima tri procesa i četiri resursa, jedan proces može da koristi dva resursa istovremeno. Takav proces će završiti posao i osloboditi resurse, a posle toga ostalim procesima mogu biti dodeljena po dva resursa. Sistem nije u stanju zastoja.

5.5. Kako se na osnovu grafa dodeljenih resursa određuje da li se sistem nalazi u stanju zastoja?

- ako graf ne sadrži kružne putanje, zastoja sigurno nema,
- ako graf sadrži kružne putanje a resursi u tim putanjama imaju samo po jednu instancu, svi procesi koji pripadaju kružnoj putanji su u zastoju,
- ako graf sadrži kružne putanje, a resursi u tim putanjama imaju više instanci, može se dogoditi da zastoja i nema.

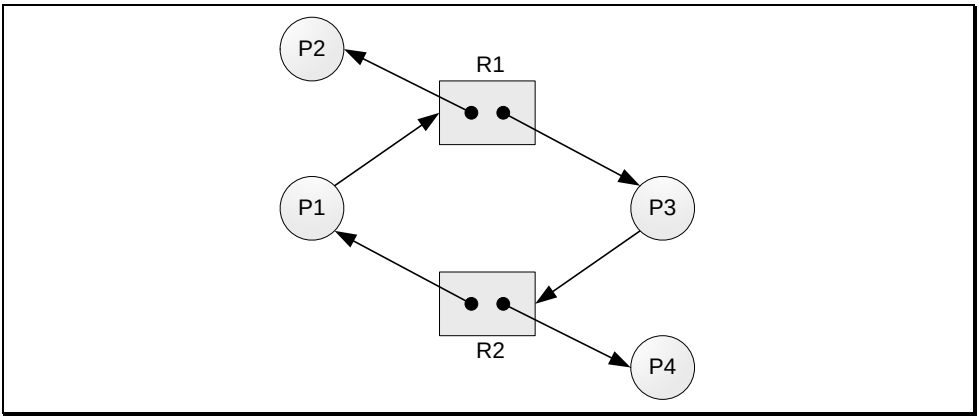
5.6. Da li se sistem opisan grafom dodele resursa sa slike 5.2. nalazi u stanju zastoja? Obrazložite odgovor.



Slika 5.2. *Graf dodele resursa*

Na grafu sa slike 5.2. postoji kružna putanja: P1-R1-P2-R2. Proces P1 drži resurs R2, a zahteva resurs R1. Proces P2 drži resurs R1, a zahteva resurs R2. Može se zaključiti da se sistem nalazi u stanju zastoja.

5.7. Da li se sistem opisan grafom dodele resursa sa slike 5.3 nalazi u stanju zastoja? Obrazložite odgovor.



Slika 5.3. *Graf dodele resursa*

Na grafu sa slike 5.3 postoji kružna putanja: P1-R1-P3-R2-P1, ali zastoja nema jer resursi R1 i R2 imaju po dve instance. Na primer, proces P4 može završiti svoje aktivnosti i osloboditi jednu instancu resursa R2, koja se zatim može dodeliti procesu P3, čime se prekida krug i eliminiše zastoj.

5.8. a. Da li do zastoja može doći ukoliko su resursi od značaja za procese periferni uređaji poput štampača, plotera i skenera?

b. Ukoliko do zastoja može doći, obrazložite odgovor navoženjem situacije u kojoj je zastoj moguć. Ukoliko do zastoja ne može doći, obrazložite zašto ne može.

c. Da li se ispravnim korišćenjem spool tehnike može izbeći zastoj?

5.9. Pretpostavite da se sistem nalazi u stanju koje nije bezbedno. Objasnite kako procesi mogu da završe svoje aktivnosti bez ulaska u stanje zastoja?

5.10. U realnom računarskom sistemu broj raspoloživih resursa, broj procesa i njihove potrebe nisu konstantne. Procesi napuštaju sistem kada završe svoje aktivnosti, novi resursi se pojavljuju, a stari bivaju istrošeni ili zamenjeni. Ukoliko je zastoj pod kontrolom bankarskog algoritma, koje se od sledećih promena mogu obaviti bezbedno, bez dovođenje sistema u stanje mogućeg zastoja, i pod kojim okolnostima?

a. dodavanje novih resursa u sistem (povećanje vrednosti elementa vektora *available*).

b. uklanjanje resursa iz sistema (smanjenje vrednosti elementa vektora *available*),

c. povećanje maksimalnih zahteva jednog procesa (povećanje vrednosti elemenata matrice *max*),

d. smanjenje maksimalnih zahteva jednog procesa (smanjenje vrednosti elemenata matrice *max*),

e. povećanje broja procesa u sistemu,

f. smanjenje broja procesa u sistemu.

5.11. Sistem se sastoji od m resursa istog tipa koje deli n procesa. Procesi mogu zahtevati i otpustiti samo po jedan resurs u jednom vremenskom trenutku. Maksimalna potreba svakog procesa je između 1 i m resursa, a suma svih maksimalnih potreba manja od m+n. Dokažite da se sistem ne nalazi u zastoju.

5.12. Sistemu obavlja 5.000 poslova mesečno bez mehanizama za detekciju ili izbegavanje zastoja. Do zastoja dolazi u proseku dva puta mesečno. posle pojave zastoja, operator mora da prekine i ponovo pokrene u proseku 10 poslova. Svaki posao košta 2e procesorskog vremena, a poslovi koji se prekidaju posle zastoja su u proseku 50% izvršeni. Trenutno, 30% procesorskog vremena je neiskorišćeno (*idle time*). Programer procenjuje da će implementiranje algoritma za izbegavanje zastoja uvećati vreme izvršavanja poslova za 10% i vreme potrebno za kompletiranje poslova za 20%.

a. Navedite argumente za i protiv implementiranja algoritma za izbegavanje zastoja.

b. Da li se porpusna moć sistema posle implementiranja pomenutog algoritma nalazi u prihvatljivim granicama?

5.13. Pretpostavite da se resursi procesima dodeljuju na sledeći način:

- procesi zahtevaju i otpuštaju po jedan resurs u jednom trenutku,

- ukoliko se zahtev za dodelom resursa ne može zadovoljiti jer resursi nisu raspoloživi, resursi se oduzimaju od blokiranih procesa i dodeljuju procesu koji ih zahteva.

a. Da li može doći do zastoja. Ako može, navedite primer. Ako ne može, navedite razlog.

b. Da li može doći do zakucavanja (beskonačnog blokiranja, *starvation*) procesa?

(a) Do zastoja ne može doći, jer postoji pretpražnjenje, tj. resurs se može oduzeti jednom procesu i dodeliti drugom.

(b) Do zakucavanja procesa može doći - proces može biti doveden u stanje da nikad ne dobije sve potrebne resurse, ukoliko mu se resursi kontinualno oduzimaju i dodeljuju drugim procesima.

5.14. Posmatrajte sistem u kome se nalaze tri procesa (P0, P1 i P2) i resurs A sa 12 instanci. Stanje sistema je dato sledećom tabelom:

Proces	***Allocation***	***Max***	***Need***
P0	5	10	5
P1	2	4	2
P2	2	9	7

Resurs A ima 3 slobodne instance Odredite da li je sistem u bezbednom stanju.

Sistem je u bezbednom stanju - sekvenca P1, P0, P2 doveće do zadovoljenja potreba svih procesa.

Proces P1 najpre uzima još dve instance resursa, a zatim ih vraća:

Proces	***Allocation***	***Max***	***Need***	***Available***
P0	5	10	5	
P1	4	4	0	1
P2	2	9	7	

Proces	***Allocation***	***Max***	***Need***	***Available***
P0	5	10	5	
P1	završio aktivnosti			5
P2	2	9	7	

Proces P0, zatim, uzima još pet instanci resursa, a potom ih vraća:

Proces	***Allocation***	***Max***	***Need***	***Available***
P0	10	10	0	0
P1	završio aktivnosti			
P2	2	9	7	

Proces	***Allocation***	***Max***	***Need***	***Available***
P0	završio aktivnosti			10
P1	završio aktivnosti			
P2	2	9	7	

Na kraju, proces P2 uzima još sedam instanci resursa:

Proces	***Allocation***	***Max***	***Need***	***Available***
P0	završio aktivnosti			3
P1	završio aktivnosti			
P2	9	9	7	

Proces	***Allocation***	***Max***	***Need***	***Available***
P0	završio aktivnosti			12
P1	završio aktivnosti			
P2	završio aktivnosti			

5.14. Posmatrajte sistem u kome se nalaze tri procesa (P0, P1 i P2) i resurs A sa 12 instanci. Stanje sistema je dato sledećom tabelom:

Proces	***Allocation***	***Max***	***Need***
P0	5	10	5
P1	2	4	2
P2	2	9	7

Resurs A ima 2 slobodne instance Odredite da li je sistem u bezbednom stanju.

Sistem nije u bezbednom stanju. Na primer, za sekvencu P1, P0, P2, proces P1 bi zadovoljio svoje potrebe i vratio resurse, ali se sistem može dovesti u zastoj ukoliko P0 zatraži maksimalan broj instanci resursa.

5.15. Posmatrajte sistem u kome se nalazi pet procesa (P0, P1, P2, P3 i P4) i tri resursa sa sledećim karakteristikama: resurs A (10 instanci), resurs B (5 instanci), resurs C (7 instanci). Stanje sistema u trenutku t_0 dato je sledećom tabelom:

	Available			***Max***		
Proces	**A**	**B**	**C**	**A**	**B**	**C**
P0	0	1	0	7	5	3
P1	2	0	0	3	2	2
P2	3	0	2	9	0	2
P3	2	1	1	2	2	2
P4	0	0	2	4	3	3

a. Da li je sistem u stabilnom stanju?

b. Da li će sistem da odobri zathev P1 [4] (1, 0, 2)?

c. Da li će sistem da odobri zathev P4 (3, 3, 0)?

d. Da li će sistem da odobri zathev P0 (1, 2, 2)?

e. Da li će sistem da odobri zathev P3 (1, 1, 0)?

(a) Najpre se određuje matrica potreba - *need* (svaki element matrice *need* se računa kao razlika odgovarajućih elemenata matrica *max* i *allocation*):

	Allocation			***Max***			***Need***			***Available***		
Proces	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	0	1	0	7	5	3	7	4	3			
P1	2	0	0	3	2	2	1	2	2			
P2	3	0	2	9	0	2	6	0	0	3	3	2
P3	2	1	1	2	2	2	0	1	1			
P4	0	0	2	4	3	3	4	3	1			

Zatim se primenom bankarskog algoritma pronalazi sekvenca P1, P3, P4, P2, P0 koja dokazuje da je sistem u stabilnom stanju [5].

[4] Proces P1 zahteva dodelu jedne instance resursa A i dve instance resursa C.

[5] Uvek se polazi od procesa sa manjim zahtevima, a kasnije se rešavaju problemi najzahtevnijih procesa..

(b) Proces P1 izdaje zahtev za dodelom reursa *request*=(1,0,2). Primenićemo algortiam koji razrešava zahteve za dodelom resursa.

Najpre se proverava da li je *request* ≤ *available*. Kako je (1,0,2) ≤ (3,3,2), uslov je ispunjen, pa se zatim obavlja kvazi-dodela resursa na osnovu zahteva.

	Allocation			***Max***			***Need***			***Available***		
Proces	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	0	1	0	7	5	3	7	4	3			
P1	3	0	2	3	2	2	0	2	0			
P2	3	0	2	9	0	2	6	0	0	2	3	0
P3	2	1	1	2	2	2	0	1	1			
P4	0	0	2	4	3	3	4	3	1			

Zatim se primenom bankarskog algoritma pronalazi sekvenca P1, P3, P4, P0, P2 koja zadovoljava uslove stabilnosti. To znači da će sistem ispuniti zahtev P1(1,0,2)

(c) Sistem neće odobriti zahtev procesa P4 za dodelu resursa: *request*=(3,3,0), jer je zahtev veći od raspoloživih resursa.

(d) Sistem neće odobriti zahtev procesa P0 za dodelu resursa: *request*=(1,2,2), jer se posle obavljene kvazi-dodele, sistem ne nalazi u stabilnom stanju.

	Allocation			***Max***			***Need***			***Available***		
Proces	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	1	3	2	7	5	3	6	2	1			
P1	2	0	0	3	2	2	1	2	2			
P2	3	0	2	9	0	2	6	0	0	2	1	0
P3	2	1	1	2	2	2	0	1	1			
P4	0	0	2	4	3	3	4	3	1			

(e) Sistem će odobriti zahtev procesa P4 za dodelu resursa: *request*=(1,1,0), jer je *request* ≤ *available*, tj. (1,1,0) ≤ (3,3,2), a sistem se posle obavljene kvazi-dodele, nalazi u stabilnom stanju (sekvenca P3, P4, P1, P2, P0).

	Allocation			*Max*			*Need*			*Available*		
Proces	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
P0	0	1	0	7	5	3	7	4	3			
P1	2	0	0	3	2	2	1	2	2			
P2	3	0	2	9	0	2	6	0	0	2	2	2
P3	2	1	1	2	2	2	0	1	1			
P4	1	1	2	4	3	3	3	2	1			

5.16. Posmatrajte sistem u kome se nalaze pet procesa (P0, P1, P2, P3 i P4) i četiri resursa. Stanje sistema u trenutku t_0 dato je sledećom tabelom:

	Allocation				*Max*			
Proces	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	0	0	1	2	0	0	1	2
P1	1	0	0	0	1	7	5	0
P2	1	3	5	4	2	3	5	6
P3	0	6	3	2	0	6	5	2
P4	0	0	1	4	0	6	5	6

U trenutku t_0 slobodna je 1 instanca resursa A, 5 instanci resursa B i 2 instance resursa C, odnosno *available*=(1,5,2,0). Koristeći bankarski algoritam, odredite:

a. Kako izgleda matrica potreba *need*?

b. Da li je sistem u bezbednom stanju?

c. Da li će sistem da odobri zathev P1 (0, 4, 2, 0 ?

(a) Svaki element matrice *need* se računa kao razlika odgovarajućih elemenata matrica *max* i *allocation*:

	Allocation				*Max*				*Need*			
Proces	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	0	0	1	2	0	0	1	2	0	0	0	0
P1	1	0	0	0	1	7	5	0	0	7	5	0
P2	1	3	5	4	2	3	5	6	1	0	0	2
P3	0	6	3	2	0	6	5	2	0	0	2	0
P4	0	0	1	4	0	6	5	6	0	6	4	2

b) Sistem je u bezbednom stanju, jer sledeća sekvenca procesa zadovoljava uslove stabilnosti:

- P0 uzima (0,0,0,0) a zatim vraća (0,0,1,2) ⇒ *available*=(1,5,3,2),
- P3 uzima (0,0,2,0) a zatim vraća (0,6,5,2) ⇒ *available*=(1,11,6,4),
- P2 uzima (1,0,0,2) a zatim vraća (2,3,5,6) ⇒ *available*=(2,14,11,8),
- P1 uzima (0,7,5,0) a zatim vraća (1,7,5,0) ⇒ *available*=(3,14,11,8),
- P4 uzima (0,6,4,2) a zatim vraća (0,6,5,6) ⇒ *available*=(3,14,12,12).

c) Najpre se proverava da li je *request* ≤ *available*. Kako je (0,4,2,0) ≤ (1,5,2,0), uslov je ispunjen, pa se zatim obavlja kvazi-dodela resursa na osnovu zahteva. Procesu P1 dodeljujemo (0,4,2,0):

	Allocation				***Max***				***Need***			
Proces	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
P0	0	0	1	2	0	0	1	2	0	0	0	0
P1	1	4	2	0	1	7	5	0	0	3	3	0
P2	1	3	5	4	2	3	5	6	1	0	0	2
P3	0	6	3	2	0	6	5	2	0	0	2	0
P4	0	0	1	4	0	6	5	6	0	6	4	2

Posle dodele resursa, slobodna je 1 instanca resursa A i 1 instanca resursa B, odnosno *available*=(1,1,0,0).

Sistem je u bezbednom stanju, jer sledeća sekvenca procesa zadovoljava uslove stabilnosti:

- P0 uzima (0,0,0,0) a zatim vraća (0,0,1,2) ⇒ *available*=(1,1,1,2),
- P2 uzima (1,0,0,2) a zatim vraća (2,3,5,6) ⇒ *available*=(2,4,6,6),
- P3 uzima (0,0,2,0) a zatim vraća (0,6,5,2) ⇒ *available*=(2,10,9,8),
- P1 uzima (0,3,3,0) a zatim vraća (1,7,5,0) ⇒ *available*=(3,14,11,8),
- P4 uzima (0,6,4,2) a zatim vraća (0,6,5,6) ⇒ *available*=(3,14,12,12).

Sistem odobrava zahtev za dodelom resursa.

5.17. Posmatrajte sistem u kome se nalazi pet procesa (P0, P1, P2, P3 i P4) i tri resursa sa sledećim karakteristikama: resurs A (7 instanci), resurs B (2 instance), resurs C (6 instanci). Stanje sistema u trenutku t_0 dato je sledećom tabelom:

	Allocation			*Request*		
Proces	**A**	**B**	**C**	**A**	**B**	**C**
P0	0	1	0	0	0	0
P1	2	0	0	2	0	2
P2	3	0	3	-	-	-
P3	2	1	1	1	0	0
P4	0	0	2	0	0	2

U trenutku t_0 ni jedan resurs nema slobodnih instanci, odnosno *available*=(0,0,0). Odredite da li je sistem u stanju zastoja, ako:

a. proces P2 zahteva ne zahteva resurse, tj. *request*(P3)=(0,0,0).

b. proces P2 zahteva jednu instancu resursa C, tj. *request*(P3)=(0,0,0).

(a) Sistem nije u stanju zastoja. Primenom algoritma za detekciju zastoja može se dokazati da postoji sekvenca P0, P2, P3, P1, P4 nakon koje bi svi procesi završili svoje aktivnosti. Interesantno je da sistem u trenutku t_0 nema raspoloživih resursa, međutim u sistemu postoje dva procesa koji ne traže ni jedan dodatni resurs (procesi P0 i P2). Oni mogu da obave svoje aktivnosti i vrate resurse, koji se zatim mogu dodeliti drugim procesima.

- Kada P0 završi svoje aktivnosti, oslobodiće 1 instancu resursa B. Posle toga je *availible*=(0,1,0).
- Kada P2 završi svoje aktivnosti, oslobodiće po 3 instance resursa A i C. Nakon toga je *availible*=(3,1,3).
- Proces P3 je najmanje zahtevan, tako da mu sistem može dodeliti jednu instancu resursa A. Kada P3 završi svoje aktivnosti, on vraća (3,1,1) instanci resursa sistemu, pa je *availible*=(5,2,4).
- Zatim P1 uzima (2,0,2) i vraća (4,0,2) $\Rightarrow$ *available*=(7,2,4)
- Na kraju, P4 uzima (0,0,2) i vraća (0,0,4) $\Rightarrow$ *available*=(7,2,6)

Kada svi procesi završe svoje aktivnosti, na sistemu ostaje 7 instanci resursa A, dve instance resursa B i 6 instanci resursa C.

(b) Sistem je u stanju zastoja. U trenutku t_0, na sistemu nema raspoloživih resursa, međutim proces P0 ne traži ni jedan dodatni resurs. On može da obavi svoje aktivnosti i oslobodi jednu instancu resursa B. Posle toga je

availible=(0,1,0). Međutim, dalje ni jedan proces ne može da zadovolji svoje potrebe za resursima.

5.18. Posmatrajte sistem u kome se nalazi pet procesa (P0, P1, P2, P3 i P4) i tri resursa sa sledećim karakteristikama: resurs A (7 instanci), resurs B (2 instance), resurs C (6 instanci). Stanje sistema u trenutku t_0 dato je sledećom tabelom:

	Allocation			***Request***		
Proces	**A**	**B**	**C**	**A**	**B**	**C**
P0	1	1	0	1	0	0
P1	2	0	0	2	0	2
P2	2	0	2	0	1	0
P3	1	1	1	1	0	0
P4	0	0	2	0	0	2

U trenutku t_0 slobodne su po jedna instanca resursa A i C, odnosno *available*=(1,0,1). Odredite da li je sistem u stanju zastoja.

Sistem nije u stanju zastoja, jer postoji sekvenca P0, P2, P3, P1, P4 posle koje će svi procesi završiti svoje aktivnosti.

- P0 uzima (1,0,0) a zatim vraća (2,1,0) $\Rightarrow$ *available*=(2,1,1)
- P2 uzima (0,1,0) a zatim vraća (2,1,2) $\Rightarrow$ *available*=(4,1,3)
- P3 uzima (1,0,0) a zatim vraća (2,1,1) $\Rightarrow$ *available*=(5,2,4)
- P1 uzima (2,0,0) a zatim vraća (4,0,2) $\Rightarrow$ *available*=(7,2,6)
- P4 uzima (0,0,2) a zatim vraća (0,0,4) $\Rightarrow$ *available*=(7,2,8)

6. Upravljanje memorijom

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za upravljanje memorijom. Razmotrene su razlike između fizičkog i logičkog adresnog prostora i osnovne metode kontinualne i diskontinualne alokacije memorije, kao što su straničenje i segmentacija. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

6.1. Navedite osnovne funkcije koje operativni sistem treba da postigne vezane za upravljanje memorijom.

Operativni sistem treba da:

- vodi evidenciju o korišćenju memorije,
- dodeljuje memoriju procesima kad je zahtevaju,
- oduzme memoriju procesima kad im je više nepotrebna,
- obavlja *swap* ukoliko operativna memorija nije dovoljnog kapaciteta za smeštaj svih procesa.

6.2. Kroz koje faze program prolazi, počev od izvornog koda do izvršavanja u okvirima nekog procesa?

1. Izvorni kod: adrese su simboličke.
2. Vreme prevođenja: generišu se relativne adrese jer obično nije poznato gde će proces koji će izvršavati generisani kod biti smešten u memoriji.
3. Vreme punjenja u memoriju: povezivač i punilac na bazi relokatibilnog koda generišu apsolutne adrese i pune program u memoriju.
4. Vreme izvršavanja.

6.3. a. Koje su prednosti dinamičkog punjenja memorije programom ?

b. U kojim slučajevima se može koristiti tehnika preklapanja (*overlay technique*)?

c. Šta je dinamičko povezivanje (*linking*)?

(a) Prednosti dinamičkog punjenja su sledeće:

- bolje iskorišćenje memorije - memorija se ne puni rutinama koje se ne koriste
- pogodno je u slučajevima u kojima velika rutina opslužuje zahtev koji se retko pojavljuje
- ne zahteva specijalnu podršku operativnog sistema (dinamičko punjenje je odgovornost programera).

(b) Tehnika preklapanje se koristi ukoliko je proces veći od memorije koja mu se može dodeliti. Preklapanje implementira programer.

(c) Dinamičko vezivanje funkcioniše na sledeći način:

- Povezivanje se odlaže do faze izvršenja.
- Malo parče koda (*stub*) se koristi da locira odgovarajuću rezidentnu bibliotečku rutinu.
- *Stub* se menja adresom rutine i izvršava rutinu.
- Podrška operativnog sistema je neophodna.

6.4. Glavni program (GP, veličine 60KB) poziva u datom redosledu sledeće segmete programa: A (20KB), B (50K) i C (40KB). Segment B u jednom trenutku poziva segment D (40KB), koji opet poziva segment E (70KB). Segment C u jednom trenutku poziva segmet F (120KB).

a. Skicirati kako se za dati primer može primeniti tehnika preklapanja (*overlay*)?

b. Koliko je najmanje radne memorije potrebno za izvršavanje ovog programa?

c. Koliko bi memorije bilo potrebno bez primene tehnike preklapanja?

(a) overlay 1: GP+A = 60K+20K = 80K

overlay 2: GP+B+D+E= 60+50+40+70 = 220K

overlay 3: GP+C+F = 60+40+70 = 170K

(b) max (80, 220, 170) = 220K

(c) GP+A+B+C+D+E+F = 60+20+50+40+70+40+70 = 350K

6.5. U čemu je razlika između logičkih i fizičkih adresa?

Adresa koju generiše procesorka instrukcija je logička adresa, a adresa same memorijske jedinice je fizička adresa. Logičke i fizičke adrese su iste u fazi prevođenja i u fazi punjenja, ali se razlikuju u fazi izvršavaja programa. Skup svih logičkih adresa koje generie program je logički (virtuelni) adresni prostor, a skup svih fizičkih adresa koje njima odgovaraju je fizički adresni prostor.

6.6. a. Gde se obavlja mapiranje virtuelnog adresnog prostora u fizički?

b. Objasnite najprostiju šemu MMU sa relokacionim registrom.

(a) Mapiranje virtuelnog adresnog prostora u fizički obavlja hardverski uređaj koji se naziva MMU (Memory-Management Unit).

(b) Relokacioni registar definiše adresu fizičkog početka programa. Svaka logička adresa koju generiše program se sabira sa vrednošću relokacionog registra, i tako se dobija fižička adresa. Logički adresni prostor koji se nalazi u opsegu [0, max] mapira se u opseg [R+0, R+max] fizičkog adresnog prostora.

6.7. a. Šta je interna fragmentacija?

b. Šta je eksterna fragmentacija?

c. Da li je eksterna fragmentacija karakteristična za kontinualnu ili diskontinualnu alokaciju memorije?

d. Kako se eksterna fragmentacija može smanjiti?

(a) Unutrašnja (interna) fragmentacija se javlja ukoliko se procesu dodeli deo memorije (na primer, stranica) veća od memorije koju proces zahteva. Preostala memorija ostaje neiskorišćena sve dok proces ne oslobodi svu memoriju koju je zauzeo.

(b) Spoljašnja (eksterna) fragmentacija je karakteristična za sisteme sa kontinualnom dodelom memorije. Proces se ne može dodeliti određena količinu memoriju, jer na sistemu ne postoji dovoljno veliki kontinualni memorijski blok (bez obzira što ukupna količina slobodne memorije premašuje zahteve procesa).

(c) Karakteristična je za kontinualnu alokaciju.

(d) Eksterna fragmentacija se može smanjiti kompakcijom memorije (povremenim premeštanjem sadržaja memorije u cilju dobijanja većih šuplljina). Kompakcija je moguća ako je relokacija programa dinamička i ako se radi u vreme izvršavanja.

6.8. Opišite sledeće algoritme za dodelu memorije: (a) *First fit*, (b) *Best fit*, (c) *Worst fit*.

d. Opravdajte korišćenje *Worst fit* algoritma za dodelu memorije. U čemz je njegova potencijalna prednost u odnosu na *Best fit* ?

(a) *First fit*: pretražuje se slobodna memorija i procesu dodeljuje prvi slobodni blok dovoljne veličine.

(b) *Best fit*: pretražuje se slobodna memorija i procesu se dodeljuje najmanji slobodni blok koji zadovoljava zahtev.

(c) *Worst fit*: pretražuje se slobodna memorija i procesu se dodeljuje najveći slobodni blok.

(d) Opravdanje za korišćenje *Worst fit* algoritma je u tome što se posle dodele memorije generiše veći (upotrebljiviji) slobodni blok, nego u slučaju *best fit* algoritma.

6.9. Odredite kako će (a) *First fit*, (b) *Best fit* i (c) *Worst fit* algoritmi za dodelu memorije dodeliti memorijske particije od 100K, 500K, 200K, 300K i 600K (navedene redom kojim se nalaze na sistemu) procesima koji zahtevaju 212K, 417K, 112K i 426K memorije.

d. Koji algoritam obezbeđuje najefikasnije iskorišćenje memorije?

(a) *First fit*:

- 212K se smešta u particiju veličine 500K, pri čemu ostaje prazna particija veličine 500K-212K=288K. Posle dodele, sledeće particije su prazne (redom): 100K, 288K, 200K, 300K, 600K. Algoritam je obavio dva poređenja.
- 417K se smešta u particiju veličine 600K, pri čemu ostaje prazna particija veličine 600K-417K=183K. Posle dodele, sledeće particije su prazne (redom): 100K, 288K, 200K, 300K, 183K. Algoritam je obavio 5 poređenja.
- 112K se smešta u particiju veličine 288K, pri čemu ostaje prazna particija veličine 288K-112K=176K. Posle dodele, sledeće particije su prazne (redom): 100K, 176K, 200K, 300K, 600K. Algoritam je obavio dva poređenja.
- Procesu koji zahteva 426K ne može se dodeliti memorija zbog eksterne fragmentacije. Algoritam je obavio 5 poređenja.

  Proces koji zahteva 426K ne može odmah da se izvrši i mora da sačeka da neki drugi proces oslobodi memoriju. Obavljeno je ukupno 14 poređenja između veličine memorije koju proces zahteva i slobodnih particija.

(b) *Best fit*:

- 212K se smešta u particiju veličine 300K, pri čemu ostaje prazna particija veličine 300K-212K=88K. Posle dodele, sledeće particije su prazne (redom): 100K, 500K, 200K, 88K, 600K.
- 417K se smešta u particiju veličine 500K, pri čemu ostaje prazna particija veličine 500K-417K=83K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 200K, 88K, 600K. Algoritam je obavio 5 poređenja.
- 112K se smešta u particiju veličine 200K, pri čemu ostaje prazna particija veličine 200K-112K=88K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 88K, 88K, 600K. Algoritam je obavio 5 poređenja.
- 426K se smešta u particiju veličine 600K, pri čemu ostaje prazna particija veličine 600K-426K=174K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 88K, 88K, 174K. Algoritam je obavio 5 poređenja.

  Svaki proces dobija zahtevanu memoriju. Obavljeno je ukupno 20 poređenja između veličine memorije koju proces zahteva i slobodnih particija.

(c) *Worst fit*

- 212K se smešta u particiju veličine 600K, pri čemu ostaje prazna particija veličine 600K-212K=388K. Posle dodele, sledeće particije su prazne (redom): 100K, 500K, 200K, 300K, 388K. Algoritam je obavio 5 poređenja.
- 417K se smešta u particiju veličine 500K, pri čemu ostaje prazna particija veličine 500K-417K=83K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 200K, 300K, 388K. Algoritam je obavio 5 poređenja.
- 112K se smešta u particiju veličine 388K, pri čemu ostaje prazna particija veličine 388K-112K=278K. Posle dodele, sledeće particije su prazne (redom): 100K, 83K, 200K, 300K, 278K. Algoritam je obavio 5 poređenja.
- Procesu koji zahteva 426K ne može se dodeliti memorija zbog eksterne fragmentacije. Algoritam je obavio 5 poređenja.

  Proces koji zahteva 426K ne može odmah da se izvrši i mora da sačeka da neki drugi proces oslobodi memoriju. Obavljeno je ukupno 20

poređenja između veličine memorije koju proces zahteva i slobodnih particija.

(d) U ovom slučaju *Best fit* algoritam obezbeđuje najbolje iskorišćenje memorije.

6.10. Zašto je veličina stranice uvek jednaka 2^n ?

Straničenje se implementira razbijanjem adrese na X bitova za broj stranice i Y bitova za ofset. Kako je svaka pozicija u adresi stepen broja 2, razbijanje adrese na broj stranice i ofset rezultuje veličinom stranice 2^n.

6.11. Posmatrajte logički adresni prostor koji čini 8 stranica veličine 1024 memorijske reči, mapiran u 32 okvira fizičke memorije. Koliko bita ima u logičkim, a koliko u fizičkoj adresama?

Jedan okvir (odnosno jedna stranica) veličine 1024B može se alocirati sa 10 bitova. Logički adresni prostor čini 16 stranica, koje se mogu alocirati sa 4 bita, a fizički 128 okvira, koji se mogu alocirati sa 7 bita. Znači, u logičkim adresama ima 14 bitova, a u fizičkim 17.

6.12. Kako se na sistemu sa straničenjem može omogućiti jednom procesu da pristupa podacima koji pripadaju drugom procesu?

Da bi proces mogao da pristupi okviru sa tužom stranicom, operativni sistem mora da dozvoli ažuriranje tabele stranica zapisima koji sadrže stranice drugih procesa. Na taj način se može realizovati efikasno deljenje podataka između procesa, jer procesi pristupaju istoj fizičkoj adresi, koja se nalazi na različitim logičkim adresama.

6.13. Posmatrajte sistem na kome se tabela stranica čuva u memoriji.

a. Ako pristup memorijskoj lokaciji traje 0.2μsec, koliko traje referenciranje adrese kroz tabelu stranica?

b. Ako se koristi asocijativna memorija i ako se 25% memorijskih referenci kroz tabelu stranica mogu naći pomoću asocijativne memorije, koje je efektivno vreme pristupa nekoj memorijskoj lokaciji? Pristup asocijativnoj memoriji traje 1nsec.

(a) 0.4μsec = 0.2μsec za pristup tabeli stranica + 0.2μsec za pristup memorijskoj lokaciji.

(b) teff = 0.25 · (0.2μsec + 1nsec) + 0.75 · (0.4μsec + 1nsec) = 351nsec.

6.14. Data je tabela stranica jednog procesa na sitemu sistemu sa veličinom stranice 2KB.

stranica	okvir
0	1
1	4
2	3
3	7
4	12

Koje fizičke adrese odgovaraju sledećim logičkim adresama: (a) 251, (b) 3129, (c) 10000, (d) 6066 ?

broj stranice = ceo broj količnika logičke adrese i veličine stranice

ofset = ostatak pri deljenju logičke adrese i veličine stranice

(a) Logička adresa 251

broj stranice=[251/2048]=0 $\Rightarrow$ okvir=1, ofset=251%2048=251, fizička adresa=1x2048+251=2299

(b) Logička adresa 3129

broj stranice=[3129/2048]=1 $\Rightarrow$ okvir=4, ofset=3129%2048=1081, fizička adresa=4x2048+1081=9273

(c) Logička adresa 10000

broj stranice=[10000/2048]=4 $\Rightarrow$ okvir=12, ofset=10000%2048=1808, fizička adresa=12x2048+1808=26384

(d) Logička adresa 6066

broj stranice=[6066/2048]=2 $\Rightarrow$ okvir=3, ofset=6066%2048=1970, fizička adresa=3x2048+1970=8114

6.15. Zašto je jednostavnije obezbediti deljenje *re-entrant* procedura pomoću segmenata nego pomoću stranica ?

Zato što je segmentacija zasnovana na logičkoj podeli memorije, a ne na fizičkoj kao čisto straničenje. To znači da *re-entrant* procedure bez obzira na veličinu zahtevaju samo jedan ulaz u tabeli segmenata svih programa koje tu proceduru koriste. U slučaju straničenja, *re-entrant* procedure zahtevaju više ulaza u tabelama stranica, zavisno od veličine.

6.16. Data je sledeća tabela segmenata:

Segment	***Base***	***Length***
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Koje fizičke adrese odgovaraju sledećim logičkim adresama: (a) 0,430, (b) 1,10, (c) 2,500, (d) 3,400, (e) 4,112?

(a) 219 + 430 = 649

(b) 2300 + 10 = 2310

(c) illegalna referenca (premašena vrednost limit registra)

(d) 1327 + 400 = 1727

(e) illegalna referenca (premašena vrednost limit registra)

6.17. a. Koji je mehanizam prevođenja adresa karakterističan za Intel 80386 mikroprocesore?

b. Koje su prednosti ukoliko translacioni mehanizam obezbeđuje hardver, a ne operativni sistem?

c. Koje su mane ovakvog načina prevođenja adresa?

(a) Intel 80386 koristi segmentaciju sa dvonivovskim straničenjem. Logička adresa se sastoji iz identifikatora segmenta (*selector*) i pomeraja u segmentu (*offset*). Iz tabele segmenata se čita adresa direktorijuma stranica, tabele stranica i pomeraja u tabeli stranica, a iz tabele stranica adresa okvira u fizičkoj memoriji.

(b) Efikasnost prevođenja i jednostavnost kernela.

(c) Višestruki pristup tabelama. Može se u nekoj meri ubrzati keširanjem.

6.18. Četiri procesa izvršavaju isti editor teksta, realizovan kao *re-entrant* procedura veličine 20KB. Za privatne podatke potrebno je 3KB. Odrediti koliko je memorije potrebno i koje su osobine dodeljene memorije po pitanju kontinulanosti na sitemu sa:

a. multiprogramiranjem sa particijama promenljive veličine,

b. straničenjem (veličina stranice je 2KB),

c. segmentacijom.

(a) Za svaki proces je potrebno po 23KB kontinuilanog prostora (20KB za kod i 3KB za podatke). Ukupno je potrebno 4·23KB=92KB memorije.

(b) Potrebno je 10 stranica za kod (10·2KB=20KB) koji će procesi deliti i po dve stranice za podatke za svaki proces (4·2·2KB=4·4KB). Svaki proces koristi 3KB stranica dodeljenih za podatke, tako da ostaje 1KB neiskorišćen. Znači, ukupno je potrebno 18 stranica (36KB) koje se mogu nalaziti bilo gde u memoriji. Pri tome, imamo 4·1KB=4KB interne fragmentacije.

(c) Svaki proces zahteva dva kontinualna segmenta: deljivi kod segment veličine 20KB i segment veličine 3KB za podatke. Znači, ukupno je potrebno pet kontinualnih segmenata: jedan veličine 20KB i četiri segmenta veličine 3KB. Pri tom nema interne fragmentacije.

7. Virtuelna memorija

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za virtuelnu memoriju. Razmotrena je tehnika učitavanja stranica na zahtev i osnovni algoritmi za izbor žrtve pri zameni stranica, kao što su FIFO, optimalni algoritam, LRU, LFU i MFU. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

7.1. Šta je virtuelna memorija i zašto se koristi?

Virtuelna memorija dozvoljava izvršenje procesa koji se ne nalaze celi u memoriji. Deo procesa koji se trenutno koristi nalazi se u memoriji, dok se ostatak programa nalazi na disku (*swap*). Iz tog razloga, logički adresni prostor može biti znatno veći od fizičkog.

7.2. U čemu je razlika između *swap* i *demand paging* tehnika (učitavanje stranica na zahtev)?

- *Swap*: u fizičku memoriju se sa diska prebacuje ceo proces kada je potreban.

- *Demand paging*: u fizičku memoriju se ne prebacuje ceo proces, već samo stranice koje se trenutno traže, i to najčešće samo ona koja je neophodna. *Demand paging* zahteva manje ulazno-izlaznih operacija i fizičke memorije i obezbeđuje brži odziv u odnosu na klasičan *swap*..

7.3. Koji je hardver neophodan za realizaciju tehnike učitavanja stranica na zahtev?

1. Tabela stranica sa bitom validnosti koji opisuje trenutni položaj stranica.
2. Sekundarna memorija (disk) za smeštaj stranica koje nisu prisutne u memoriji.

7.4. a. Pod kojim okolnostima dolazi do PF prekida (*page fault*)?

b. Šta radi operativni sistem kada dođe do PF prekida?

(a) Do PF prekida dolazi ukoliko proces želi da pristupi stranici koja se trenutno ne nalazi u fizičkoj memoriji.

(b) Operativni sistem proverava da li je pristup memoriji validan i prekida proces ukoliko nije. Ukoliko je pristup validan, operativni sistem traži slobodni okvir i inicira ulazno-izlaznu operaciju koja učitava stranicu sa diska u slobodni okvir. Posle toga se ažurira tabela stranica i ponovo izvršava instrukcija koja je izazvala prekid.

7.5. Na izbor veličine stranice utiču mnogi faktori. Navedite faktore koji idu u prilog:

a. smanjenju veličine stranice i

b. povećanju veličine stranice.

(a) Bolje iskorišćenje memorije i manja interna fragmentacija.

(b) Smanjenje tabele stranica, smanjenje broja ulazno-izlaznih stranica prilikom učitavanja ili zamene stranica, TLB efikasnost.

7.6. Dat je proces kome je dodeljeno m inicijalno praznih okvira? Proces želi da pristupi određenim stranicama koje se nalaze na disku. U nizu referenci dužine p nalazi se n različitih stranica (m<n<p).

a. Koliko će se najmanje PF prekida dogoditi?

b. Koliko će se najviše PF prekida dogoditi?

(a) n

(b) p-m

7.7. Sistem sa 2^{18}B fizičke memorije obezbeđuje korisnike virtuelnom memorijom kapaciteta 2^{32}B, realizovanom pomoću stranica veličine 4KB. Korisnički proces generiše virtuelnu adresu 11123456. Objasnite kako sistem određuje adekvatnu fizičku adresu. Koje operacije obavlja hardver a koje operativni sistem ?

Virtuelna adresa je: 0001 0001 0001 0010 0011 0100 0101 0110.

Hardver: kako je veličina stranice 4KB=2^{12}B, veličina tabele stranica je 2^{20}. Viših 20 bita (0001 0001 0001 0010 0011) određuje pomeraj u tabeli stranica, a nižih 12 (0100 0101 0110) pomeraj u stranici. Sistem u tabeli stranica dalje proverava bit validnosti i određuje da li se stranica nalazi u fizičkoj memoriji.

Operativni sistem: ukoliko se stranica ne nalazi u fizičkoj memoriji, dolazi do PF prekida. Operativni sistem na osnovu određenog algoritma bira žrtvu i učitava stranicu sa diska u novi okvir. Nakon toga se ažurira bit validnosti i broj okvira u koji je stranica smeštena u tabeli stranica.

7.8. Koje su od sledećih programerskih tehnika pogodne za korišćenje na sistemu na kom se primenjuje DP tehnika: (a) stek, (b) sekvencijalna pretraga, (c) pretraga pomoću binarnog stabla, (d) operacije sa vektorima, (e) indirekcija? Obratite pažnju na broj PF prekida koji date tehnike generišu.

Za korišćenje su pogodne sledeće tehnike: (a) stek, (b) sekvencija pretraga i (c) operacije sa vektorima.

7.9. Pretpostavite da sistem na kom je implementirana tehnika učitavanja stranica na zahtev čuva tabelu stranica u registrima. Ukoliko ima praznih okvira ili ukoliko stranica koja se žrtvuje nije modifikovana, PF prekid se može opslužiti za 8msec. Ukoliko je stranica koja se žrtvuje modifikovana, opsluživanje PF prekida traje 20msec. Pristup lokaciji u fizičkoj memoriji traje 0.1μsec. Pretpostavite da su stranice koje se žrtvuju u 70% slučajeva modifikovane. Odredite gornju granicu verovatnoće pojave PF prekida koja će efektivno vreme pristupa održati manjim od 0.2μsec.

$t_{EA} = 0.2\mu sec = (1-P)\cdot 0.1\mu sec + (0.3p)\cdot 8msec + (0.7p)\cdot 20msec$

$p \cong 0.000006$

7.10. Posmatrajte sistem na kom je implementirana tehnika učitavanja stranica na zahtev. Adrese se prevode pomoću tabele stranica u operativnoj memoriji. Radi ubrzavanja pristupa memoriji, uvedena asocijativna memorija za smeštaj pojedinih zapisa iz tabele stranica. 80% memorijskih referenci prisutno je u asocijativnoj memoriji i ne izaziva PF prekide. 10% memorijskih nije prisutno u asocijativnoj memoriji i ne izaziva PF prekide, a preostalih 10% izaziva PF prekide. Pristup operativnoj memoriji zahteva 1μsec vremena, pristup disku na kome se obavlja straničenje 50msec, dok se vreme pristupa asocijativnoj memoriji može zanemariti. Odredite efektivno vreme pristupa memoriji.

$t_{EA} = 0.8 \cdot 1\mu sec + 0.1 \cdot (2\mu sec) + 0.1 \cdot (5002\mu sec) = 501.2\mu sec \cong 0.5msec.$

7.11. a. Rangirajte algoritme prema broju PF prekida koji izazivaju: LRU, FIFO, optimalni algoritam, algoritam "druga šansa".

b. Za koje je od gore pomenutih algoritama karakteristična *Belady*-jeva anomalija?

	Algoritam	**Broj PF prekida**	***Belady*-jeva anomalija**
1.	Optimalni	najviše	ne
2.	LRU		ne
3	Druga šansa		da
4.	FIFO	najmanje	da

7.12. Pretpostavite da je na sistemu na kom je implementirana tehnika učitavanja stranica na zahtev izmereno sledeće iskorišćenje resursa:

- procesor 20%,
- disk na kome se čuvaju stranice (*paging disk*) 97,7%,
- ostali ulazno-izlazni uređaji 5%.

Objasnite koji će od sledećih postupaka povećati iskorišćenje procesora (najverovatnije): (a) ugradnja bržeg procesora, (b) ugradnja većeg diska za straničenje, (c) povećanje stepena multiprogramiranja, (d) smanjenje stepena multiprogramiranja, (e) povećanje količine

RAM memorije, (f) instalacija bržeg hard diska ili RAID sistema, (g) implementacija prepaging mehanizma u algoritme za straničenje.

Očigledno je da sistem provodi najviše vremena na straničenju, odnosno da je izražena pre-alokacija memorije. Ukoliko se stepen multiprogramiranja smanji ili se poveća količina RAM memorije, veći broj stranica koje procesi zahtevaju mogu ostati rezidentne u memoriju. To znači da će procesi izazivati manji broj PF prekida i iskorišćenje procesora će se povećati.

(a) ugradnja bržeg procesora – ne,

(b) ugradnja većeg diska za straničenje – ne,

(c) povećanje stepena multiprogramiranja – ne,

(d) smanjenje stepena multiprogramiranja – da,

(e) procesi će izazivati manji broj PF prekida i iskorišćenje procesora će se povećati,

(f) instalacija bržeg hard diska ili RAID sistema - da, jer će procesor dobiti veću količinu podataka za kraće vreme, tako da će biti bolje iskorišćen,

(g) implementacija prepaging mehanizma u algoritme za straničenje - da, jer će procesor dobiti veću količinu podataka za kraće vreme, tako da će biti bolje iskorišćen.

7.13. Data je dvodimenzionalna matrica definisana na sledeći način:

```
int A[ ][ ] = new int[2048][2048];
```

Proces koji upravlja matricom ima na raspolaganju tri okvira (veličina stranice je 4KB). Promenljiva tipa int zauzima jednu memorijsku reč veličine 2B. Pod pretpostavkom da se kod nalazi u prvom okviru, da su ostala dva okvira inicijalno prazna i da se zamena stranica obavlja po LRU algoritmu, odredite koliko PF prekida generišu sledeće petlje za inicijalizaciju matrice:

a.
```
for (int i = 0; i < 2048; i++)
   for (int j = 0; j < 2048; j++)
   A[i][j] = 0;
```

b.
```
for (int j = 0; j < 2048; j++)
   for (int i = 0; i < 2048; i++)
   A[i][j] = 0;
```

(a) 1024 PF prekida

(b) 1024 x 1024 PF prekida.

7.14. Projektant je razvio novi algoritam za zamenu stranica. U nekim situacijama u fazi testiranja, prisutan je efekat *Belady*-jeve anomalije. Da li se algoritam može smatrati optimalnim ili ne? Obrazložite odgovor.

7.15. Posmatrajte segmentaciju kao straničenje sa promenljivom veličinom stranice. Definišite dva algoritma za zamenu segmenata zasnovana na (a) FIFO i (b) LRU algoritmima za zamenu stranica. Obratite pažnju na to da su segmenti različite veličine i da segment odabran za žrtvovanje može biti manji i ostaviti particiju nedovoljne veličine. Razmotrite strategije koje se mogu implementirati na sistemima na kojima su segmenti relokatibilni i strategije koje se mogu implementirati na sistemima na kojima relokacija segmenata nije dozvoljena.

(a) FIFO: odaberite prvi segment koji je dovoljno veliki za smeštaj novog segmenta. Veličina slobodnog bloka, određena veličinom tog segmenta i kontinualnog neiskorišćenog prostora koji se nalazi pre i posle segmenta mora biti veća ili jednaka segmentu koji se smešta u memoriju. Ukoliko relokacija nije dozvoljena, odaberite niz segmenata smeštenih na kontinualnim lokacijama u memoriji. Ukoliko je relokacija dozvoljena, preuredite memoriju tako da prvih N segmenata budu kontinualni i zamenite ih novim segmentom.

(b) LRU: odaberite dovoljno veliki segment koji najduže nije korišćen. Veličina slobodnog bloka, određena veličinom tog segmenta i kontinualnog neiskorišćenog prostora koji se nalazi pre i posle segmenta mora biti veća ili jednaka segmentu koji se smešta u memoriju. Ukoliko ni jedan segment nije dovoljno veliki, a relokacija nije dozvoljena, odaberite kombinaciju kontinualnih "najstarijih" segmenata u memoriji. Ukoliko je relokacija dozvoljena, preuredite memoriju tako da najstarijih N segmenata budu kontinualni i zamenite ih novim segmentom.

7.16. a. Definišite algoritam za zamenu stranica koji žrtvuje stranicu smeštenu u okviru asociranim s najmanjim brojem stranica.

b. Šta se postiže takvim algoritmom?

c. Koliko će PF prekida biti izazvano u slučaju sledećeg niza memorijskih referenci, ako je procesu dodeljeno četiri okvira: 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2?

(a) Svakom okviru se pridružuje brojač stranica asociranih sa tim okvirom. Inicijalna vrednost brojača je 0. Vrednost brojača se inkrementira svaki put kada se okvirom napuni nekom stranicom, a dekrementira svaki put kada neka stranica, koja je asocirana sa okvirom, postane nepotrebna. Prilikom zamene, žrtvuje se stranica koja se nalazi u okviru sa najmanjom vrednošću brojača. U slučaju da dva ili više okvira imaju identične vrednosti brojača. primenjuje se FIFO algoritam.

(b) Algoritam ravnomerno distribuira najšešće korišćene stranice svuda po memoriji: svaki okvir je asociran sa približno jednakim brojem stranica.

(c) 14 PF prekida.

7.17. a. Šta je uzrok efekta zasićenja (*trashing*)?

b. Kako se može detektovati kad je sistem ušao u zasićenje?

c. Kako se problem zasićenja može otkloniti nakon detekcije?

(a) Alokacija nedovoljno velikog broja okvira za proces, tako da proces veoma često izaziva PF prekide.

(b) Proverom vrednosti stepena multiprogramiranja, iskorišćenja procesora i diska na kom se čuvaju stranice.

(c) Smanjenjem nivoa multiprogramiranja.

7.18. Analizirajte sistem sa tehnikom učitavanja stranica na zahtev za koji su izmerene sledeće vrednosti iskorišćenja resursa:

a. procesor 11%, disk 97%,

b. procesor 85%, disk 6%,

c. procesor 14%, disk 3%.

Interpretirajte rezultate za svaki sistem (probajte da odredite šta se dešava) i predložite eventualne da li ima smisla povećati ili smanjiti stepen multiprogramiranja.

(a) Pojavilo se zasićenje (trashing). Razmotrite mogućnost smanjenja stepena multiprogramiranja.

(b) Iskorišćenje procesora je dovoljno visoko. Ne treba ništa dirati.

(c) Potrebno je povećati stepen multiprogramiranja.

7.19. Posmatrajte sistem virtuelne memorije sa n okvira koji su u početnom trenutku prazni. Dat je sledeći niz od 20 uzastopnih memorijskih referenci u vremenu: 1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6. Skicirajte stanje u okvirima i odredite koliko će se PF prekida dogoditi ukoliko se zamena stranica obavlja po: LRU, FIFO i optimalnom algoritmu, za sledeće slučajeve: (a) n=2, (b) n=3, (c) n=4, (d) n=5 i (e) n=6.

(a) n=2

LRU: PF=18

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	PF	PF	PF	PF	PF	PF	h	PF	PF	PF	PF	PF	PF	h	PF	PF
1	1	3	3	2	2	5	5	2	2	2	2	7	7	3	3	1	1	3	3
	2	2	4	4	1	1	6	6	1	1	3	3	6	6	2	2	2	2	6

FIFO: PF=18

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	PF	PF	PF	PF	PF	PF	h	PF	PF	PF	PF	PF	PF	h	PF	PF
1	1	3	3	2	2	5	5	2	2	2	2	7	7	3	3	1	1	1	6
	2	2	4	4	1	1	6	6	6	6	3	3	6	6	2	2	2	3	3

Optimalni algoritam: PF=15

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	PF	PF	PF	h	PF	h	PF	PF	PF	h	PF	PF	H	PF	PF
1	1	3	4	4	1	5	6	6	1	1	3	3	3	3	3	1	1	3	3
	2	2	2	2	2	2	2	2	2	2	2	7	6	6	2	2	2	2	6

(b) n=3

LRU: PF=15

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	PF	PF	PF	PF	PF	h	PF	PF	PF	h	PF	PF	h	h	PF
1	1	1	4	4	4	5	5	5	1	1	1	7	7	7	2	2	2	2	2
	2	2	2	2	2	2	6	6	6	6	3	3	3	3	3	3	3	3	3
		3	3	3	1	1	1	2	2	2	2	2	6	6	6	1	1	1	6

FIFO: PF=16

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	PF	PF	PF	PF	PF	h	PF	PF	PF	h	PF	PF	h	PF	PF
1	1	1	4	4	4	4	6	6	6	6	3	3	3	3	2	2	2	2	6
	2	2	2	2	1	1	1	2	2	2	2	7	7	7	7	1	1	1	1
		3	3	3	3	5	5	5	1	1	1	1	6	6	6	6	6	3	3

Optimalni algoritam: PF=11

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	PF	PF	h	h	PF	PF	h	h	PF
1	1	1	1	1	1	1	1	1	1	1	3	3	3	3	3	3	3	3	6
	2	2	2	2	2	2	2	2	2	2	2	7	7	7	2	2	2	2	2
		3	4	4	4	5	6	6	6	6	6	6	6	6	6	1	1	1	1

(c) n=4

LRU: PF=10

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	PF	PF	PF	h	h	PF	H	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	5	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	7	7	7	7	1	1	1	1

FIFO: PF=14

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	PF	PF	h	PF	PF	PF	h	PF	PF	h	PF	h
1	1	1	1	1	1	5	5	5	5	5	3	3	3	3	3	1	1	1	1
	2	2	2	2	2	2	6	6	6	6	6	7	7	7	7	7	7	3	3
		3	3	3	3	3	3	2	2	2	2	2	6	6	6	6	6	6	6
			4	4	4	4	4	4	1	1	1	1	1	1	2	2	2	2	2

Optimal: PF=8

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	PF	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	5	6	6	6	6	6	6	6	6	6	6	6	6	6

(d) n=5

LRU: PF=8

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	PF	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7

FIFO: PF=10

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	PF	PF	PF	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
		3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
						5	5	5	5	5	5	7	7	7	7	7	7	7	7

Optimalni algoritam: PF=7

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	4	6	6	6	6	6	6	6	6	6	6	6	6	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7

(e) n=6

LRU: PF=7

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	4	4	4	4	7	7	7	7	7	7	7	7
						5	5	5	5	5	5	5	5	5	5	5	5	5	5
							6	6	6	6	6	6	6	6	6	6	6	6	6

FIFO: PF=10

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	PF	PF	PF	h
1	1	1	1	1	1	1	1	1	1	1	1	7	7	7	7	7	7	7	7
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2
			4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3
						5	5	5	5	5	5	5	5	5	5	5	5	5	5
							6	6	6	6	6	6	6	6	6	6	6	6	6

Optimalni algoritam: PF=7

1	**2**	**3**	**4**	**2**	**1**	**5**	**6**	**2**	**1**	**2**	**3**	**7**	**6**	**3**	**2**	**1**	**2**	**3**	**6**
PF	PF	PF	PF	h	h	PF	PF	h	h	h	h	PF	h	h	h	h	h	h	h
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
			4	4	4	4	4	4	4	4	4	7	7	7	7	7	7	7	7
						5	5	5	5	5	5	5	5	5	5	5	5	5	5
							6	6	6	6	6	6	6	6	6	6	6	6	6

8. Ulazno-izlazni podsistem

Ulazno-izlazni podsistem obezbeđuje mehanizme za komunikaciju sa ulazno-izlaznim uređajima. Bez ulazno-izlaznog podsistema, korisnik procesima ne može zadati ulazne podatke, niti može od procesa preuzeti rezultate obrade. Takođe, proces nije u mogućnosti da rezultate obrade sačuva na disku ili nekom drugom medijumu niti da ostvari komunikaciju sa drugim računarom. Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za ulazno-izlazni podsistem: funkcije podsistema, podelu uređaja prema različitim kriterijuma, hardver od značaja za ulazno-izlazni podsistem, servise koje pruža ulazno-izlazni podsistem, proces prevođenja zahteva u ulazno-izlazne operacije i UNIX System V mehanizam tokova.

8.1. Navedite osnovne funkcije ulazno-izlaznog podsistema.

- upravljanje i kontrola ulazno-izlaznih uređaja i operacija koje ti uređaji izvršavaju,
- obezbeđivanje što jednostavnijeg interfejsa prema korisniku i prema ostatku sistema.

8.2. a. Objasnite dva aspekta nezavisnosti uređaja.

b. Koja je posledica nezavisnosti uređaja?

(a) 1. Nezavisnost programa od konkretnog modela uređaja koji se koristi.

2. Nezavisnost programa od konkretne vrste uređaja.

Različiti modeli uređaja istog tipa moraju se sa aspekta programa jednako posmatrati. Na primer, program pristupa CD-ROM uređaju, bez obzira na to da li je kao uređaj korišćen određen model TEAC ili LG CD-ROM uređaja. Takođe, mora se obezbediti da program na što istovetniji način može pročitati podatke sa diska ili CD-ROM uređaja.

(b) Programi ne rade sa realnim, konkretnim uređajima, već sve ulazno-izlazne operacije obavljaju pomoću virtuelnih uređaja. Primeri virtuelnih uređaja su tokovi (streams) i datoteke (karakteristično za UNIX i Linux sisteme).

8.3. Svaki uređaj je znatno sporiji od operativne memorije i samog procesora, što znači da su ulazno-izlazne operacije usko grlo računarskog sistema i da je poželjno obaviti ih što efikasnije. Šta se može učiniti na nivou operativnog sistema u pogledu povećanja efikasnosti obavljanja ulazno-izlaznih operacija ?

Ulazno-izlazni podsistem mora obezbediti konkurentnost u radu sa uređajima i različite mehanizme keširanja, baferovanja i korišćenja *spool* tehnike.

8.4. Navedite osnovne prednosti smeštanja funkcionalnosti upravljanja ulazno-izlaznim uređajima u odvojeni sloj koji se nalazi van kernela.

Prednosti: kernel je pojednostavljen (rasterećen algoritama za upravljanje konkretnim uređajima), a kako se mnogo manje koda izvršava u kernelskom režimu, manje je verovatno da će greške u kodu izazvati krah operativnog sistema.

8.5. Klasifikujte uređaje u opšte kategorije prema (a) nameni, (b) jediničnoj količini transfera, (c) metodi pristupa i (d) deljivosti. Navedite po jedan uređaj za svaku kategoriju.

(a) klasifikacija uređaja prema nameni,

- uređaji za dugotrajno skladištenje podataka - disk, traka,
- uređaji za prenos podataka - mrežna kartica, modem,
- uređaji koji obezbeđuju interfejs ka korisniku - monitori, tastatura.

(b) klasifikacija uređaja prema jediničnoj količini transfera,

- blok uređaji - disk, traka,
- karakter uređaji - štampač, terminal,

- mrežni uređaji.

(c) klasifikacija uređaja prema metodi pristupa,

- uređaji sa sekvencijalnim pristupom - modem, magnetna traka,
- uređaji sa direktnim prisitupom - disk, CD-ROM.

(d) klasifikacija uređaja prema deljivosti,

- deljivi uređaji – disk,
- nedeljivi uređaji - tastatura, miš.

8.6. Objasnite kako se časovnik i tajmer koriste za realizaciju funkcije pretpražnjenja (*preemption*).

Tajmeri koji mere proteklo vreme ili posle određenog vremena generišu prekidni signal koriste se za realizaciju tehnike pretpražnjena (preemption) pri raspoređivanju procesora. Pretpražnjenje funkcioniše na sledeći način: tajmer generiše prekidni signal, izvršenje tekućeg procesa se prekida, a zatim se poziva dispečer da na osnovu algoritma za raspoređivanje procesora odabere proces iz procesorskog reda i preda mu procesor na korišćenje.

8.7. Objasnite dva načina za pristup registrima kontrolera, zavisno od toga da li su memorijski i ulazno-izlazni adresni prostor razdvojeni.

- Ukoliko su memorijski i ulazno-izlazni adresni prostor razdvojeni, procesor ima posebne instrukcije za upis i čitanje podataka sa određene adrese na kontroleru.
- Ukoliko se ulazno-izlazni adresni prostor može tretirati kao memorijski adresni prostor, transferi se mogu obaviti običnim instrukcijama za rad sa memorijom (memorijski mapirane ulazno-izlazne operacije). Tipičan primer je memorija na grafičkim karticama koja služi za smeštaj slike.

8.8. a. Objasnite kako funkiconiše tehnika prozivanja uređaja (polling).

b. Koji je osnovni nedostatak tehnike prozivanja?

c. Kako se taj nedostatak može ukloniti?

(a) Prozivanje uređaja se obavlja u petlji: računar ponavlja čitanje statusnog registra i analizira *busy* bit sve dok vrednost bita ne postane 0.

(b) Osnovni nedostatak tehnike prozivanja je trošenje procesorskog vremena na petlju prozivanja. U slučaju sporih uređaja procesor ostaje "zaposlen čekanjem" u dugačkim petljama čekajući da vredost busy bita postane 0.

(c) Nedostatak se može ukloniti uvođenjem hardverskog mehanizma koji omogućava ureađaju da označi procesoru kada je komanda izvršena (mehanizam prekida). Kada uređaj završi operaciju, kontroler šalje prekidni signal procesoru preko prekidne linije. Procesor će završti tekuću instrukciju, posle čega se prekida sekvencijalno izvršavanje programa, kako bi se obradio prekid. Posle obrade prekida, procesor nastavlja izvršavanje programa.

8.9. U čemu je razlika između prekidnih signala poslatih po maskirajućim i nemaskirajućim prekidnim linijama?

Prekidni signal poslat po nemaskirajućoj liniji uvek može da prekine izvršenje tekućeg procesa. Nemaskirajuća linija se koristi za slanje prekidnih signala usled kritičnih hardverskih grešaka, kao što su greške u memoriji. Prekidni signal poslat po maskirajućoj liniji ne prekida izvršenje procesa sve dok se proces na primer nalazi u kritičnoj sekciju. Maskirajuća linija se koristi za slanje prekidnih signala koji potiču od normalnih operacija i standardnih grešaka koje se javljaju na ulazno-izlaznim uređajima.

8.10. Kako se na nivou operativnog sistema rešava problem uniformnosti uređaja?

Srodni uređaji, kao što su, na primer, diskovi, grupišu se u opšte klase uređaja. Svakoj klasi uređaja pristupa se preko standardnog skupa funkcija, koji se naziva interfejs. Razlike koje postoje među uređajima iste klase sakrivene su u specijalnim modulima jezgra operativnog sistema koji se nazivaju veznici ili drajveri (*device drivers*). Iznad svih drajvera nalazi se sloj ulazno-izlaznog podsistema jezgra koji je, praktično, nezavistan od hardvera - svi detalji vezani za konkretan harver prepušteni su drajverima. Proizvođači uređaja, po pravilu, za svoje uređaje pišu drajvere za razne opratívne sisteme, tako da se njihovi uređaji mogu priključiti bez intervencije projektanata samog operativnog sistema.

8.11. U čemu je razlika između (a) blokirajućih, (b) neblokirajućih i (c) asinhronih sistemskih poziva?

(a) Proces koji izdaje blokirajući sistemski poziv za izvršavanje operacije se blokira i čeka da se operacija izvrši.

(b) Proces se posle izdavanja neblokirajućeg sistemskog poziva ne blokira, nego nastavlja da radi. Ne-blokirajući sistemski poziv vraća kontrolu procesu vrlo brzo, obično sa pojavom prvih podataka.

(c) Asinhroni sistemski pozivi posle iniciranja ulazno-izlazne operacije odmah vraćaju kontrolu procesu koji ih je pozvao. Asinhrona ulazno-izlazna operacija će se završiti u nekom budućem trenutku vremena, što će se označiti upisom odgovarajućeg podatka u neku memorijsku strukturu ili putem prekidnog signala.

8.12. a. Šta je međumemorija (bafer)?

b. Navedite osnovne razloge za korišćenje tehnike baferovanja.

(a) Bafer (buffer) je deo memorije koji funkcioniše na principu proizvođač-potrošač i služi za čuvanje privremenih podataka prilikom transfera između dva uređaja ili između uređaja i aplikacije.

(b) Osnovni razlozi za korišćenje tehnike baferovanja su:

- usklađivanje različitih brzina između potrošača i proizvođača,
- prilagođavanje različitih veličina trasfera podataka,
- održavanje semantike kopriranja.

8.13. Šta je keširanje? U čemu je razlika između keša i bafera?

Keš (*cache*) je region brze sistemske memorije koji čuva kopiju podataka, obično sa diska. Keširanje (*caching*) je tehnika kopiranja delova diska u keš memoriju, čime se osetno povećavaju performanse disk I/O sistema. Razlika između keša i bafera je u tome što bafer čuva trenutno aktuelne podatke, a keš čuva bilo koju kopiju sa diska.

8.14. a. Šta je *spool*?

b. Koje su prednosti upotrebe spool tehnike?

(a) *Spool* je bafer koji privremeno čuva izlazne podatke namenjene nekom nedeljivom uređaju (kao što je štampač). Procesi upisuju podatke namenjene uređaju na disk, a operativni sistem upravlja *spool* baferom na disku tako što opslužuje jedan po jedan zahtev.

(b) Proces relativno brzo obavlja postavljanje svog zahteva u spool bafer, a nakon toga je slobodan da dalje obavlja svoje aktivnosti. Nedeljivi

uređaji se koriste kao prividno deljivi, što omogućava većem broju procesa da istovremeno koriste uređaj.

8.15. a. Iz čega se sastoji jedan tok (*stream*)? Koje su funkcije tih delova?

b. Pomoću kog mehanizma se obavlja transfer podataka između redova čekanja u jednom toku?

(a) Tok se sastoji iz glave (*stream head*), koja komunicira sa korisničkom procesom, drajverskim krajem, koji kontroliše uređaj i proizvoljnim brojem modula između njih. Svaki od ovih delova sadrži par redova čekanja: red čekanja za čitanje (*read queue*) i red čekanja za upis (*write queue*).

(b) Pomoću sistema prosleđivanja poruka.

8.16. a. Kako se mogu uvećati performanse ulazno-izlaznog podsistema?

Navedite prednosti i mane ukoliko se povećanje performansi obavlja na nivou (b) aplikacije, odnosno korisničkog softvera, (c) operativnog sistema, odnosno drajvera, (d) hardvera.

(a) Balansiranim korišćenjem procesora, memorije i ulazno-izlaznog podsistema, korišćenjem DMA kontrolera i keš mehanizama, smanjivanjem broja prekida povećanjem jedinične količine transfera, smanjivanjem broja prebacivanja konteksta procesora.

(b) Prednosti: greške u razvoju algoritama za povećanje performansi ne mogu dovesti do havarije sistema, a razvoj algoritama ne zahteva ponovo punjenje memorije drajverima niti obaranje sistema. Mane: ne može se direktno koristiti funkcionalnost jezgra, prisutan je veliki broj prebacivanja konteksta procesa.

(c) Prednosti: mogu koristiti sve strukture podataka jezgra, tako da se mogu postići bolje performanse. Mane: narušavanje strogih pravila u jezgru može dovesti do ozbiljnih grešaka u sistemu.

(d) Prednosti: performanse se mogu najviše uvećati specijalnim hardverskim implementacijama, kao što je, na primer, čist hardveski RAID kontroler. Mane: obično se sve radi u asembleru, razvoj traje dugo i zahteva specijalnu opremu za testiranje.

8.17. Posmatrajte sledeće scenarije ulazno-izlaznih operacija na jednokorisničkom PC računaru:

a. miš sa grafičkim korisničkim interfejsom,

b. magnetna traka na višeprocesnom operativnom sistemu,

c. disk sa korisničkim podacima,

d. grafička kartica sa direktnom vezom sa magistralom, dostupnost preko memorijski mapiranih ulazno-izlaznih operacija (ulazno-izlazni adresni prostor može se tretirati kao memorijski adresni prostor).

Za svaki scenario odredite da li biste projektovali operativni sistem tako da se koristi baferovanje, keširanje, spool tehnika ili kombinacija ovih metoda. Da li bi ste koristili prozivanje uređaja ili mehanizam prekida? Obrazložite svoje odgovore.

(a) Baferovanje se može koristiti za snimanje pokreta miša u periodu kada se odvijaju aktivnosti višeg prioriteta. *Spool* i keširanje nisu prikladni za ovaj scenario. U ovom slučaju je pogodno koristiti mehanizam prekida.

(b) Baferovanje se može koristiti za usklađivanje brzine između trake i izvorišta/odredišta ulazno-izlazne operacije, a keširanje za čuvanje podataka koji se nalaze na traci radi bržeg pristupa. *Spool* tehnikom se može obezbediti deljivost trake. U ovom slučaju je pogodno koristiti mehanizam prekida (mehanizam prekida je pogodan za spore uređaje, kao što su trake).

(c) Baferovanje se može korisiti za privremeno skladištenje podataka koji se prenose sa diska ka korisničkom procesu i obratno, a keširanje za čuvanje podataka koji se nalaze na disku radi bržeg pristupa. *Spool* tehnika nije potrebna zato što je disk deljiv uređaj (posebni algoritmi za raspoređivanje disk zahteva omogućavaju višestruki pristup disku). I u ovom slučaju je pogodno koristiti mehanizam prekida.

(d) Dvostruko baferovanje se može koristiti za skladištenje sledećeg ekrana dok se iscrtava trenutni. Keširanje i *spool* nisu prikladni za korišćenje zato što je grafička kartica brz uređaj. Tehnika prozivanja i mehanizam prekida su su nepotrebni ukoliko se radi o uređaju čiji se ulazno-izlazni adresni prostor može se tretirati kao memorijski adresni prostor.

8.18 Kako DMA povećava konkurentnost sistema?

Dok DMA obavlja transfer, procesor može obavljati druge aktivnosti.

8.19. Zašto je potrebno povećati brzinu magistrale i ulazno-izlaznih uređaja ukoliko se brzina procesora poveća?

Ukoliko se poveća samo brzina procesora, ulazno-izlazni podsistem će postati usko grlo računarskog sistema. S druge strane, ukupne performanse sistema koji 50% vremena obavlja ulazno-izlazne operacije, a 50% vremena koristi procesor za neka izračunavanja povećaće se za samo 50 % ukoliko se brzina procesora udvostruči. Znači, potrebno je povećati i brzinu magistrale i ulazno-izlaznih uređaja, kako bi se ukinulo usko grlo računarskog sistema i povećale performanse sistema u celini (što je bolje rešenje od povećanja performansi pojedinačnih komponenti).

8.20. Navedite primer, odnosno situaciju u kojoj je značajno i poželjno da operativni sistem ne bude fer prema zahtevima za izvođenjem ulazno-izlaznih operacija.

Poželjno je da ulazno-izlazne operacije koje inicira kernel imaju veći prioritet od korisničkih zahteva. U ove operacije, na primer, spadaju:

- upis meta-podataka sistema datoteka,
- swap i razmena stranica između operativne memorije i diskova.

9. Sekundarne memorije

Za razliku od operativne memorije, sadržaj sekundarnih memorija se ne gubi nakon isključivanja napajanja. Na sekundarnim memorijama smešten je operativni sistem, programi i podaci koji se obrađuju. Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za strukturu diskova i postupke kojima se diskovi pripremaju za rad. Takođe, razmotreni su nivoi keširanja diskova, algoritmi za raspoređivanje zahteva za rad sa diskovima (*First Come First Served*, *Shortest Seek Time First*, *SCAN*, *C-SCAN*, *LOOK* i *C-LOOK*) i RAID strukture koje se mogu iskoristiti za formiranje stabilnih podsistema sekundarne memorije. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

9.1. Razvoj savremenih disk uređaja usmeren je ka povećavaju kapaciteta i performansi. Objasnite da li je u tom smislu potrebno smanjiti ili povećati: (a) rotacionu brzinu, (b) gustinu magnetnog medijuma?

(a) Povećanjem rotacione brzine smanjuje se rotaciono kašnjenje, a samim tim i povećava brzinu čitanja/upisa sa magnetnog mediuma.

(b) Povećanje gustine magnetnog medijuma ima za posledicu povećanje kapaciteta i povećanje brzina čitanja/upisa sa magnetnog mediuma.

9.2. Koje su dve osnovne funkcije RAM memorije na savremenim disk uređajima?

- Usklađivanje brzine transfera disk mediuma i disk interfejsa (*speed-matching buffer*).
- Keširanje na nivou disk uređaja.

9.3. Kako se obavlja mapiranje logički-fizički blok na diskovima?

Mapiranje jednodimenzionih logičkih adresa (LBN - *Logical Block Number*) u trodimenzione fizičke adrese (PBN - *cylinder/head/sector*) obavlja se počevši od LBN=0 koje se pridružuje prvom bloku na početnom cilindru. Zatim se LBN blokovi dodaju sukcesivno, sve do punog kapaciteta diska.

9.4. Klasifikujte savremene disk kontrolere namenjene za PC arhitekturu, ako su kriterijumi složenost, funkcionalnost i performanse.

- Kompaktni kontroleri - nemaju poseban lokalni procesor za upravljanje kontrolerom i ne poseduju veliku bafersku memoriju na kontrolerskoj kartici.
- Baferski kontroleri - optimalna klasa za većinu operativnih sistema i većinu aplikativnih primena kada je u pitanju kompromis između performansi i cene. Sadrže relativno veliku bafersku memoriju koja je organizovana na FIFO principu.
- Keš kontroleri - najsloženiji i najkvalitetniji kontroleri, visokih performansi. Keš arhitektura uključuje kvalitetnije lokalne procesore, 16 bitne ili 32 bitne, veliku keš memoriju i mali FIFO bafer za spregu sa sistemskom magistralom. Algoritam za upravljanje keš memorijom je relativno složen. Veličina keš memorije može biti relativno velika i reda je više megabajta.

9.5. Koji se administrativni postupci ubrajaju u pripremu diskova za rad?

- Formatiranje diskova (*low level formatting*), odnosno upisivanje oznaka koje predstavljaju granice staza i sektora na magnetni medijum.
- Kreiranje particija, odnosno okvira za smeštanje sistemi datoteka.

- Kreiranje sistema datoteka, odnosno formiranje meta struktura neophodnih za rad sa sistemima datoteka.

9.6. a. Kako se obavlja remapiranje defektnih blokova na diskovima pomoću rezervnih sektora?

b. Zašto se na svakom cilindru diska ostavlja rezervni neiskorišćen prosto ?

c. Šta je klizanje sektora (*sector slipping*)?

(a) Za svaki defektivan sektor nađe se alternativni sektor i tom alternativnom se dodeli ista adresa kao defektivnom, koji zauvek ispada iz igre. Kada se sistem obrati defektivnoj adresi, unutrašnji kontroler diska automatski translira adresu na nivu poziciju.

(b) Korišćenje rezervnih sektora obara performanse jer algoritmi koji raspoređuju zahteve za rad sa diskovima gube pravu informaciju o cilindru. Zbog toga, većina diskova na svakom cilindru ostavlja altrenativni prostor koji se može iskoristiti za remapiranje, a tehnika remapiranja se uvek trudi da remapira sektor sa alternativnim na istom cilindru.

(c) Tehnika klizanja sektora (*sector slipping*) - prilikom zamene defektivnog bloka alternativom, svi ostali sektori između njih klizaju na gore. Na primer, neka je sektor 17 defektivan, sektor 203 rezervni, a prvi sektor pre alternative 202. Prilikom oporavka, sektor 202 se pomera u rezervni (203), 201 u 202, 200 u 201, a na kraju defektivni sektor 17 u 18. Na ovaj način relativni položaj sektora ostaje isti.

9.7. a. Šta je Master Boot Record?

b. Objasnite kako teče *boot-strap* rutina?

c. Da li se programi neophodni u ranoj fazi podizanja operativnog sistema nalaze u sistemima datoteka ili na fiksnim područjima diska? Obrazložite svoje odgovore.

(a) Master Boot Record (MBR) je prvi logički sektor, odnosno prvi sektor prve staze sa prve površine diska u kome se nalaze informacije o particijama (*partition table*) i mali program koji očitava particionu tabelu, proverava koja je particija aktivna, i očitava prvi sektor aktivne particije (*boot sector*).

(b) U boot sektoru se nalazi mali program čijim pokretanjem započinje *boot-strap* rutina, odnosno punjenje RAM memorije operativnim sistemom. Proces podizanja operativnih sistema sa diska zahteva da na sistemu postoji program inicijalni program koji će otpočeti podizanje operativnog sistema (*boot loader*). Podizanje operativnog sistema započinje u ROM memoriji računara, a nastavlja se sa diska, sa kog se najpre pročita prvi sektor. Kod upisan u MBR najpre identifikuje aktivnu particiju u particionoj tabeli, a zatim izvršava kod upisan u boot sektoru aktivne particije, koji dalje puni memoriju kernelom.

(c) Delovi koda kojim se memorija puni u toj ranoj fazi nalaze se na fiksnim područjima diska, a ne u sistemima datoteka, zato što u toj fazi nema kernela, pa nemamo podršku za sistem datoteka. Rana faza podizanja operativnog sistema se završava učitavanjem jezgra.

9.8. Ukratko objasnite i uporedite keširanje na nivou: (a) operativnog sistema (*file caching*), (b) disk kontrolera (*HBA level caching*), (c) disk uređaja (*disk drive level caching*), (d) aplikacije.

(a) Svaki operativni sistem poseduje sopstveno keširanje - keširanje na ovom nivou naziva se file-caching zato što keš sadrži kopije datoteka.

(b) Baferski kontroleri u kombinaciji sa disk keširanjem na nivou operativnog sistema daju mnogo bolje rezultate za mnogo nižu cenu. Koncept keširanja na nivou disk kontrolera se, praktično, ne koristi za klasične disk kontrolere.

(c) Disk uređaj, preciznije disk keš memorija, optimalno je mesto za tehniku predikovanog čitanja jer niko ne poznanje bolje svoj servo-sistem i raspored podataka na mediumu, kao sam disk uređaj.

(d) Svaka kvalitetnija aplikacija, bilo da JE reč o bazama podataka ili o određenim vrstama video servera ili Internet servera ima veoma specifičan rad sa diskom, ima svoje datoteke i strukture od izuzetne vaznosti za sopstvene performanse. Zahvaljujući dobrom poznavanju sopstvenih potreba u radu sa diskom, kvalitena aplikacija rezerviše svoj memorijski prostor i kešira disk saglasno svojim potrebama, po pravilu osetno bolje nego generalni disk keš na nivou operativnog sistema (*OS level cache*).

9.9. Šta se postiže pravilnim raspoređivanjem zahteva za rad sa diskom?

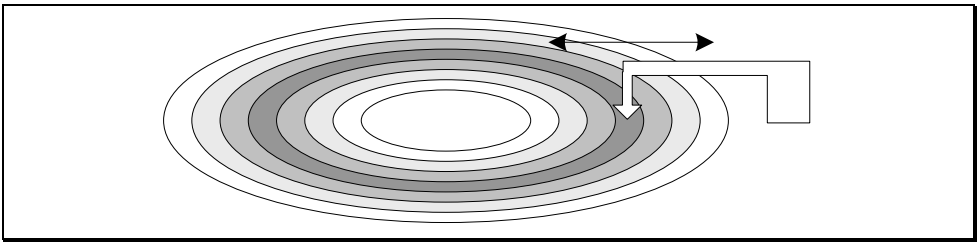
Pravilnim raspređivanjem ovih zahteva (*disk scheduling*), ukupno vreme pozicioniranja ili rotacionog kašnjenja se može smanjiti.

9.10. Napisati C++ program za simulaciju raspoređivača zahteva za rad sa diskom koji koristi SCAN i C-SCAN algoritme. Disk zahteve generisati pomoću pseudoslučajnih brojeva.

9.11. Koji su algoritmi za raspoređivanje disk zahteva osim FCFS upotrebljivi u jednokorisničkom okruženju ? Obrazložite svoj odgovor.

U jednokorisničkom okruženju, red za čekanje na uređaje je obično prazan. Ukoliko zahtevi za rad sa diskom stižu od jednog procesa i to tako da proces zahteva pristup jednom bloku ili sekvenci uzastopnih blokova, FCFS je logično rešenje za raspoređivanje disk zahteva. Međutim, programski kod LOOK algoritma je, takođe, jako jednostavan, a sam algoritam postiže mnogo veće performanse u slučaju da veći broj procesa konkurentno pristupa disku (na primer, *Web browser* prima podatke u pozadini dok operativni sistem prikazuje stranicu, ili dok je druga aplikacija aktivna u prvom planu).

9.12. Objasnite zašto SSTF favorizuje cilindre diska koji se nalaze bliže polovini radijusa ploče (slika 9.1) u odnosu na periferne i centralne cilindre.



Slika 9.1 *Cilindri diska*

Cilindri diska koji se nalaze bliže polovini radijusa ploče imaju najmanju srednju udaljenost od ostalih cilindara. Iz tih razloga, glave za čitanje i pisanje imaju tendenciju udaljavanja od centralnih i perifernih cilindara.

9.13. a. Zašto se rotaciono kašnjenje zanemaruje u algoritmima za raspoređivanje disk zahteva?

b. Objasnite kako treba modifikovati algoritme SSFT, SCAN i C-SCAN da bi pri odlučivanju uzimali u obzir i rotaciono kašnjenje.

(a) Većina diskova ne snabdeva host računar informacijama o trenutnom uglu, odnosno trenutnoj poziciji ploče. Ukoliko bi se ove informacije slale računaru, vreme potrebno da informacija dođe do od diska do raspoređivača nije fiksno, tako da bi zbog velike ugaone brzine informacija o trenutnom uglu, odnosno o trenutnoj poziciji ploča, bila neispravna. Dodatno, zahtevi za rad sa diskom obično se šalju preko brojeva logičkih blokova, a mapiranje između logičkih blokova i fizičkih trodimenzionalnih adresa je složeno.

9.14. Da li su algoritmi za raspoređivanje disk zahteva potrebni ukoliko se podaci kojima se pristupa nalaze na RAM diskovima? Objasnite odgovor.

Osnovna funkcija raspoređivača disk zahteva je smanjenje premašenja koje nastaje usled pomeranja glava diska. Kako RAM ima uniformno vreme pristupa, raspoređivanje zahteva je nepotrebno.

9.15. Posmatrajte disk sa 5000 cilindara označenih brojevima od 0-4999. Glave za čitanje i pisanje se trenutno nalaze cilindru 143, a prethodno su bile pozicionirane na cilindru 125. Trenutni raspored reda za rad sa diskom u FIFO poretku je: 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. Odredite redosled opsluživanja disk zahteva i ukupan pomeraj (ukupan broj cilindara obuhvaćen *seek* sekvencom), ukoliko se za raspoređivanje zahteva koriste sledeći algoritmi: (a) FCFS, (b) SSTF, (c) SCAN, (d) LOOK, (e) C-SCAN, (f) C-LOOK. Za početni trenutak usvojite momenat u kome su glave diska na 143 cilindru.

(a) Raspored zahteva prema FCFS algoritmu je (brojevi u zagradi predstavljaju pomeraj koji glave diska prave pri prelasku na sledeći cilindar, izražen u broju cilindara):

143, 86 (57), 1470 (1384), 913 (557), 1774 (861), 948 (826), 1509 (561), 1022 (487), 1750 (728), 130 (1620).

Ukupan pomeraj glava: 7801 cilindara.

(b) Raspored zahteva prema SSTF algoritmu je: 143, 130 (13), 86 (44), 913 (827), 948 (35), 1022 (74), 1470 (448), 1509 (39), 1750 (241), 1774 (24).

Ukupan pomeraj glava: 1745 cilindara.

(c) Raspored zahteva prema SCAN algoritmu je: 143 , 913 (770), 948 (35), 1022 (74), 1470 (448), 1509 (39), 1750 (241), 1774 (24), 4999 (3225), 130 (4869), 86 (44).

Ukupan pomeraj glava: 9769 cilindara.

(d) Raspored zahteva prema LOOK algoritmu je: 143 , 913 (770), 948 (35), 1022 (74), 1470 (448), 1509 (39), 1750 (241), 1774 (24), 130 (486), 86 (44).

Ukupan pomeraj glava: 3319.

(e) Raspored zahteva prema C-SCAN algoritmu je: 143 , 913 (770), 948 (35), 1022 (74), 1470 (448), 1509 (39), 1750 (241), 1774 (24), 4999 (3225), 0 (4999), 86 (86), 130 (44).

Ukupan pomeraj glava: 9813.

(f) Raspored zahteva prema C-LOOK algoritmu je: 143 , 913 (770), 948 (35), 1022 (74), 1470 (448), 1509 (39), 1750 (241), 1774 (24), 86 (1688), 130 (44).

Ukupan pomeraj glava: 3363.

9.16. Koji karakteristike rada sa masovnom memorijom se poboljšavaju ukoliko je masovna memorija realizovana pomoću RAID sistema?

- RAID uvećava performanse paralelizmom (tehnikom deljenja podataka između različitih diskova) i konkurentnim operacijama. U slučaju malih traka, jedan veći logički blok fizički se smešta na više diskova, a u slučaju velikih traka, mali zahtevi se opslužuju sa jednog diska, što može dovesti do paralelnog izvršenja više nezavisnih disk operacija u istom trenutku.
- Verovatnoća otkaza RAID strukture koju čini N diskova uvećava se N puta. Problem pouzdanosti se rešava uvođenjem redundanse, odnosno čuvanjem dodatnih informacija koje obezbeđuju mogućnost potpunog povratka podataka u slučaju otkaza jednog diska.

9.17. a. Objasnite kako funkcionišu RAID nivoi 0 i 1.

b. Objasnite kako funkcionišu RAID 0+1 i RAID 1+0.

c. Objasnite posledice otkaza jednog diska na RAID 0+1 i RAID 1+0 sistemima.

(a) RAID 0 (*block striping*) - stripe na nivou jednog ili više blokova podataka. Poseduje paralelizam, ali ne i redundansu. U slučaju otkaza jednog diska, svi podaci se nepovratno gube.

RAID 1 (*disk mirroring*) - svaki disk ima svoje ogledalo. Redundansa je potpuna, ali nema paralelizma.

(b) U kombinaciji 0+1, skup diskova deli podatke, a potom se sve stripe jedinice u celini kopiraju u svoje ogledalo. Druga kombinacija 1+0 znači da svaki disk ima svoje ogledalo a onda se podaci dele između ogledala. 1+0 teorijski je bolji u odnosu na 0+1.

(c) Kod 0+1 u slučaju da jedan disk otkaže cela grupa stripe jednica je neraspoloživa na 2 mesta, dok je ona u ogledalu raspoloživa. U slučaju 1+0 otkaz jednog diska utiče na pouzdanost samo te stripe jedince koja ima samo jednu raspoloživu kopiju.

9.18. Data su četiri identična diska kapaciteta 10GB sa sledećim karakteristikama:

- prosečno vreme pozicioniranja (*average seek time*): 6.3msec,
- pozicioniranje s kraja na kraj (*full stroke seek*): 15msec,
- pozicioniranje na sledeći cilindar (*track to track seek*): 0.8msec,
- ugaona brzina: 7200rpm (8.33 msec),
- pristup medijumu (*sustain data rate*): 24.5MB/sec,
- propusna moć: 160MB/sec,
- srednje vreme otkaza (*mean time to failure*): 200.000 časova (23 godine),
- cena: $50.

Od ovih diskova formiran je RAID-0 sa *stripe* jedinicom veličine 4KB (8 blokova).

a. Skicirati raspored stripe jedinicama ovog RAID sistema. Koliki je kapacitet ovog RAID-a? Koliko blokova ima ovaj RAID, a koliko stripe jedinica?

b. Odredite na kom disku se nalazi logički blok 153.

c. U redu za korišćenje diska nalaze se redom zahtevi sa sledećim veličinama transfera: 1K, 8K, 12K, 2K, 4K. Koliko zahteva može opslužiti ovaj RAID istovremeno za ovaj red, a koliko uopšte?

d. Odredite cenu po gigabajtu i stepen iskorišćenja prostora.

e. Odredite optimalnu veličinu trake (*stripe*) za radno opterećenje sa stepenom konkurentnosti 4 (4 istovremenih zahteva) i prosečnom veličinim transfera 5K. [6]

f. Odredite srednje vreme otkaza za ovaj RAID

(a) Sledeća tabela predstavlja raspored stripe jedinica datog sistema:

disk0	**disk1**	**disk2**	**disk3**
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23
...	...	...	...
N-4	N-3	N-2	N-1

Kapacitet ovog RAID-0 sistema je 40GB:

broj blokova = broj diskova · broj blokova na disku,

broj blokova = 4 · (10GB/512) = 78125000 blokova,

broj traka = broj blokova / veličina trake u blokovima,

broj traka = 78125000 / 8 = 9765625,

(b) Prvo se određuje kojoj traci pripada blok.

Stripe = Integer (logički blok / veličina trake) = Integer(153/8) = 19.

Disk = (stripe % broj diskova) = moduo (19/4) = 3 (logički blok se nalazi na disku broj 3, a kako se diskovi numerišu počev od nule, to je četvrti disk u RAID-u).

(c) I/O queue = (1K, 8K, 12K, 2K, 4K)

Teorijski, RAID 0 može opslužiti onoliko zahteva koliko ima diskova u RAID-u, pod uslovom da je transfer manji od veličine trake i da se trake nalaze na različitim diskovima. U ovom slučaju RAID trenutno može da

[6] Koristite formulu $x = \sqrt{t(c-1)vs/n}$, gde je x optimalna veličina trake (*stripe*), t vreme koje se računa kao zbir srednjeg vremena pozicioniranja i srednjeg rotacionog vremena, v brzina pristupa medijumu, c stepen konkurentnosti, s prosečna količina transfera po zahtevu, i n broj diskova u RAID-u.

izvršava najviše tri zahteva (1K, 2K, 4K), pod uslovom da se zahtevi odnose na trake smeštene na različitim diskovima. Teorijski, ovaj RAID može izvršiti 4 zahteva simultano.

(d) Cena po GB = ukupna cena / ukupni kapacitet.

Cena po GB = 4 x 50$ / 4 x 10GB = 5$/GB.

Stepen iskorišćenja prostora za RAID 0 je 100%.

(e) t = 6.3 + (8.33/2) = 10.47 msec,

v = brzina pristupa medijumu = 24.5MB/sec,

c = stepen konkurentnosti = 4,

s = prosečna količina transfera po zahtevu = 5KB,

n = broj diskova koji čine RAID = 4,

$$x = \sqrt{t(c-1)vs/n} = \sqrt{10.47\text{msec} \cdot (4-1) \cdot 24.5\text{MB/sec} \cdot 5\text{KB}/4} = 31.02\text{KB}$$

Ova vrednost se zaokružuje na x=32KB (64 blokova).

(f) Srednje vreme otkaza za RAID 0 se računa kao količnik srednjeg vremena otkaza za jedan disk i broja diskova u sistemu. U ovom slučaju je:

MTTF(RAID-0) = MTTF(disk) / n

MTTF(RAID-0) = 200.000/4 = 50.000 sati = oko 6 godina.

Napomena: ova vrednost ima više teorijski značaj. Za sve praktične slučajeve treba primeniti citate iz Marfijevog zakona.

9.19. Data su četiri identična diska kapaciteta 10GB sa sledećim karakteristikama:

- prosečno vreme pozicioniranja (*average seek time*): 6.3msec,
- pozicioniranje s kraja na kraj (*full stroke seek*): 15msec,
- pozicioniranje na sledeći cilindar (*track to track seek*): 0.8msec,
- ugaona brzina: 7200rpm (8.33 msec),
- pristup medijumu (*sustain data rate*): 24.5MB/sec,
- propusna moć: 160MB/sec,
- srednje vreme između otkaza: 200.000 sati (23 godine),

- cena: $50.

Od ovih diskova formiran je RAID-1.

a. Skicirati ovaj RAID sistem. Koliki je kapacitet i koliko logičkih blokova ima ovaj RAID?

b. Odredite na kom disku se nalazi logički blok 335.

c. U redu za korišćenje diska nalaze se redom zahtevi sa sledećim veličinama transfera: 1K, 8K, 12K, 2K, 4K. Koliko zahteva može opslužiti ovaj RAID istovremeno za ovaj red, a koliko uopšte?

d. Odredite cenu po gigabajtu.

e. Odredite srednje vreme otkaza za ovaj RAID, ako je srednje vreme oporavka (*mean time to recovery*) MTTR = 1 čas.

(a) Sledeća tabela ilustruje dati RAID:

disk0	**disk1**	**disk2**	**disk3**
0	0	$^N/_2$	$^N/_2$
1	1	$^N/_2$+1	$^N/_2$+1
2	2	$^N/_2$+2	$^N/_2$+2
3	3	$^N/_2$+3	$^N/_2$+3
4	4	$^N/_2$+4	$^N/_2$+4
5	5	$^N/_2$+5	$^N/_2$+5
...	...	...	...
$^N/_2$-1	$^N/_2$-1	N-1	N-1

Kapacitet ovog RAID-1 sistema je 20GB:

broj blokova = (broj diskova / 2) · broj blokova na disku,

broj blokova = (4/2) · (10GB/512) = 39062500 blokova,

(b) Logički blok 335 nalazi se u disku 0 (na kome se nalaze logički blokovi 0-19.531.249), a njegova kopija na disku 1.

(c) I/O queue = (1K, 8K, 12K, 2K, 4K)

Teorijski, RAID 1 može opslužiti onoliko zahteva koliko parova diskova ima u RAID-u, pod uslovom da se zahtevi odnose na različite diskove. U ovom slučaju RAID trenutno može da izvršava najviše dva zahteva, pod uslovom da se zahtevi odnose na različite diskove.

(d) Cena po GB = ukupna cena / ukupni kapacitet.

Cena po GB = 4 x 50$ / 2 x 10GB = 10$/GB.

Stepen iskorišćenja prostora za RAID 1 je 50%.

(f) Srednje vreme otkaza za RAID 1 računa se na sledeći način:

$MTTF(RAID\text{-}1) = MTTF^2(disk) / 2 \cdot MTTR$

MTTF (*mean time to failure*) je srednje vreme otkaza, a MTTR (*mean time to recovery*) srednje vreme potrebno za oporavak.

$MTTF(RAID\text{-}1) = 200.000^2 / 2 \cdot 1 = 20.000.000.000$ sati = 2.283.105 godina.

9.20. Data su četiri identična diska kapaciteta 10GB, sa sledećim karakteristikama:

- prosečno vreme pozicioniranja (*average seek time*): 6.3msec,
- pozicioniranje s kraja na kraj (*full stroke seek*): 15msec,
- pozicioniranje na sledeći cilindar (*track to track seek*): 0.8msec,
- ugaona brzina: 7200rpm (8.33 msec),
- pristup medijumu (*sustain data rate*): 24.5MB/sec,
- propusna moć: 160MB/sec,
- srednje vreme između otkaza: 200.000 sati (23 godine),
- cena: $50.

Od ovih diskova formiran je RAID-3 sa *stripe* jedinicom veličine 1B.

a. Skicirati ovaj RAID sistem. Koliki je kapacitet, koliko logičkih blokova a koliko *stripe* jedinica ima ovaj RAID?

b. Odredite na kom disku se nalazi logički blok 21.411.

c. U redu za korišćenje diska nalaze se redom zahtevi sa sledećim veličinama transfera: 1K, 8K, 12K, 2K, 4K. Koliko zahteva može opslužiti ovaj RAID istovremeno za ovaj red, a koliko uopšte?

d. Odredite cenu po gigabajtu i stepen iskorišćenja prostora..

(a) Sledeća tabela ilustruje dati RAID (trake veličine 1B):

disk0	disk1	disk2	disk3
0	1	2	P
4	5	6	P
6	7	8	P
9	10	11	P
12	13	14	P
15	16	17	P
...	...	...	...
N-3	N-2	N-1	P

Kapacitet ovog RAID-1 sistema je 30GB.

broj blokova = (broj diskova - 1) · broj blokova na disku,

broj blokova = (4-1) · (10GB/512) = 58593750 blokova,

broj traka = broj blokova · broj traka u bloku,

broj traka = 58593750 · 512 = 30.000.000.000.

(b) Logički blok LA=21411 nalazi se na sva tri diska.

Početak je na prvom disku: (LA·512)%(N-1)=(21411·512)%0=0.

Početak je u LA/(N-1)=21411/3=7137 bloku prvog diska.

(c) Kako je veličina trake 1B, RAID 3 može da izvršava samo jedan zahtev u jednom trenutku vremena.

(d) Cena po GB = ukupna cena / ukupni kapacitet.

Cena po GB = 4 x 50$ / 3 x 10GB = 6.66$/GB.

Stepen iskorišćenja prostora za RAID 3 je 75%.

9.21. Data su četiri identična diska kapaciteta 10GB sa sledećim karakteristikama:

- prosečno vreme pozicioniranja (*average seek time*): 6.3msec,
- pozicioniranje s kraja na kraj (*full stroke seek*): 15msec,
- pozicioniranje na sledeći cilindar (*track to track seek*): 0.8msec,
- ugaona brzina: 7200rpm (8.33 msec),
- pristup medijumu (*sustain data rate*): 24.5MB/sec,

- propusna moć: 160MB/sec,
- srednje vreme između otkaza: 200.000 časova (23 godine),
- cena: $50.

Od ovih diskova formiran je RAID-4 sa *stripe* jedinicom veličine 2KB.

a. Skicirati ovaj RAID sistem. Koliki je kapacitet, koliko logičkih blokova, a koliko *stripe* jedinica ima ovaj RAID?

b. Odredite na kom se disku se nalazi logički blok 3.945.

c. U redu za korišćenje diska nalaze se redom zahtevi sa sledećim veličinama transfera: 1K, 8K, 12K, 2K, 4K. Koliko zahteva može opslužiti ovaj RAID istovremeno za ovaj red, a koliko uopšte?

d. Odredite cenu po gigabajtu i stepen iskorišćenja prostora..

e. Odredite srednje vreme otkaza za ovaj RAID, ako je srednje vreme oporavka (*mean time to recovery*) MTTR = 1 čas.

(a) Sledeća tabela ilustruje dati RAID (trake veličine 2KB):

disk0	**disk1**	**disk2**	**disk3**
0	1	2	P
3	4	5	P
6	7	8	P
9	10	11	P
12	13	14	P
15	16	17	P
...	...	...	...
N-3	N-2	N-1	P

Kapacitet ovog RAID-1 sistema je 30GB:

broj blokova = (broj diskova - 1) · broj blokova na disku,

broj blokova = (4-1) · (10GB/512) = 58593750 blokova,

broj traka = broj blokova / veličina trake u blokovima,

broj traka = 58593750 / 4 = 146.484.375.

(b) Prvo se određuje kojoj traci pripada blok.

Stripe = Integer (logički blok / veličina trake) = Integer(3945/4) = 986.

Disk = (stripe % (broj diskova-1)) = moduo (986/3) = 2 (logički blok se nalazi na disku broj 2, a kako se diskovi numerišu počev od nule to je treći disk u RAID-u).

(c) I/O queue = (1K, 8K, 12K, 2K, 4K).

Teorijski, RAID 4 može opslužiti onoliko zahteva koliko ima diskova u RAID-u, umanjeno za 1, pod uslovom da je transfer manji od veličine trake i da se trake nalaze na različitim diskovima. U ovom slučaju RAID trenutno može da izvršava najviše dva zahteva (1K, 2K), pod uslovom da se zahtevi odnose na trake smeštene na različitim diskovima. Teorijski, ovaj RAID može izvršiti 3 zahteva simultano.

(d) Cena po GB = ukupna cena / ukupni kapacitet.

Cena po GB = 4 x 50$ / 3 x 10GB = 6.66$/GB.

Stepen iskorišćenja prostora za RAID 4 je 75%.

(e) Srednje vreme otkaza za RAID 4 se računa na sledeći način:

$MTTF(RAID\text{-}4) = MTTF^{2}(disk) / (N \cdot (G-1) \cdot MTTR)$,

$MTTF(RAID\text{-}4) = 200.000^2/4 \cdot (4-1) \cdot 1 = 3.333.333.333$časova$ = 380.517$ god.

9.22. Data su četiri identična diska kapaciteta 10GB, sa sledećim karakteristikama:

- prosečno vreme pozicioniranja (*average seek time*): 6.3msec,
- pozicioniranje s kraja na kraj (*full stroke seek*): 15msec,
- pozicioniranje na sledeći cilindar (*track to track seek*): 0.8msec,
- ugaona brzina: 7200rpm (8.33 msec),
- pristup medijumu (*sustain data rate*): 24.5MB/sec,
- propusna moć: 160MB/sec,
- srednje vreme između otkaza: 200.000 sati (23 godine),
- cena: $50.

Od navedenih diskova formiran je RAID-5 sa *stripe* jedinicom veličine 4KB (8 blokova).

a. Skicirati ovaj RAID sistem. Koliki je kapacitet, koliko logičkih blokova, a koliko *stripe* jedinica ima ovaj RAID?

b. Odredite na kom disku se nalazi logički blok 111.

c. U redu za korišćenje diska nalaze se redom zahtevi koji imaju sledeće veličine transfera: 1K, 8K, 12K, 2K, 4K. Koliko zahteva može opslužiti ovaj RAID istovremeno za ovaj red, a koliko uopšte?

d. Odredite cenu po gigabajtu i stepen iskorišćenja prostora..

e. Odredite optimalnu veličinu trake (*stripe*) za radno opterećenje sa stepenom konkurentnosti 5. [7]

f. Odredite srednje vreme otkaza za ovaj RAID, ako je srednje vreme oporavka (*mean time to recovery*) MTTR = 1 čas.

(a) Sledeća tabela ilustruje dati RAID (trake veličine 2KB):

disk0	**disk1**	**disk2**	**disk3**
0	1	2	P
3	4	P	5
6	P	7	8
P	9	10	11
12	13	14	P
15	16	P	17
...	...	...	...
N-3	N-2	N-1	P

Kapacitet ovog RAID-1 sistema je 30GB:

broj blokova = (broj diskova - 1) · broj blokova na disku,

broj blokova = (4-1) · (10GB/512) = 58.593.750 blokova

broj traka = broj blokova / veličina trake u blokovima,

broj traka = 58.593.750 / 8= 7.324.218.

(b) Prvo se određuje kojoj traci pripada blok.

Stripe = Integer (logički blok / veličina trake) = Integer(111/8) = 13.

Iz prethodne tabele se vidi da se traka br.13 nalazi na disku 1.

(c) I/O queue = (1K, 8K, 12K, 2K, 4K).

---

[7] Koristite formulu x = 0.5KB + ¼ · tv(c-1), gde je x optimalna veličina trake (*stripe*), t vreme koje se računa kao zbir srednjeg vremena pozicioniranja i srednjeg rotacionog vremena, v brzina pristupa medijumu, c stepen konkurentnosti, n broj diskova u RAID-u.

Teorijski, RAID 5 može opslužiti onoliko zahteva koliko ima diskova u RAID-u, umanjeno za 1, pod uslovom da je transfer manji od veličine trake i da se trake nalaze na različitim diskovima. U ovom slučaju RAID trenutno može da izvršavaju najviše tri zahteva (1K, 2K, 4K), pod uslovom da se zahtevi odnose na trake smeštene na različitim diskovima. Teorijski, ovaj RAID može izvršiti 3 zahteva simultano.

(d) Cena po GB = ukupna cena / ukupni kapacitet.

Cena po GB = 4 x 50$ / 3 x 10GB = 6.66$/GB.

Stepen iskorišćenja prostora za RAID 5 je 75%.

(e) t = 6.3 + (8.33/2) = 10.47 msec,

v = brzina pristupa medijumu = 24.5MB/sec,

c = stepen konkurentnosti = 5,

x = 0.5KB + ¼ · tv(c-1) = 257.01KB.

Ova vrednost se zaokružuje na x=256KB (512 blokova).

(f) Srednje vreme otkaza za RAID 5 se računa na sledeći način:

MTTF(RAID-5) = MTTF ^{2}(disk) / (N·(G-1)·MTTR)

MTTF(RAID-5) = 200.000 ^{2} / 4·(4-1)·1 = 3.333.333.333 sati = 380.517 godina.

9.23. U čemu su prednosti i mane realizacije *swap* prostora (a) u formi posebne particije, (b) u formi datoteke u postojećem sistemu datoteka ?

(a) Prednosti: upravljanje *swap* prostorom zaobilazi rutine za rad sa datotekama, tako da se *swap* prostoru brže pristupa.

Mane: zahteva particionisanje diskova.

(b) Prednosti: ne zahteva particionisanje diskova.

Mane: upravljanje *swap* prostorom obavlja se preko rutina za rad sa datotekama što unosi kašnjenje.

10. Sistemi datoteka

Za većinu korisnika sistem datoteka je nevidljiviji aspekt operativnog sistema koji obezbeđuje mehanizam za čuvanje i pristup datotekama. Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove - datoteka, direktorijum i link i mehanizme za zaštitu i deljenje datoteka. Zatim su razmotrene metode dodele prostora datotekama i upravljanja slobodnim prostorom u sistemima datoteka, problematika pouzdanosti i efikasnosti sistema datoteka. Konkretni primeri odnose se na sisteme datoteka koji pripadaju UNIX/Linux i Windows operativnim sistemima. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

10.1. Šta je datoteka?

Datoteka je organizovani skup podataka koji ima svoje ime i koji se prema određenom prostornom rasporedu čuva na sekundarnoj memoriji.

10.2. a. Šta je kontrolni blok datoteke?

b. Koji atributi opisuju datoteku?

(a) Datoteku, osim sadržaja i imena, opisuju i dodatni atributi, koji se čuvaju u kontrolnom bloku datoteke. Kontrolni blok datoteke je najčešće direktorijumska struktura, ali se implementirati i kao zasebna tabela (na primer, indeksni čvor u second extended sistemu datoteka).

(b) U značajnije atribute datoteka spadaju: tip datoteke, opis prostornog rasporeda blokova koji čine datoteku ili ukazivač na lokaciju prvog bloka datoteke, tekuća veličina datoteke, informacije o vlasništvu i pravima pristupa, odnosno atributi koji regulišu kontrolu pristupa datoteci, vreme i datum.

10.3. Koje se operacije mogu izvršavati nad datotekama?

- kreiranje datoteke,
- čitanje podataka iz datoteke u memorijski bafer,
- upis podataka u datoteku,
- pozicioniranje unutar datoteke (*file seek*),
- brisanje datoteke,

- odsecanje datoteke (*truncate*).

Rad sa datotekama se može ubrzati uvođenjem operacija otvaranja (*open*) i zatvaranja (*close*) datoteka, koje uz pomoć odgovarajućih memorijskih struktura ubrzavaju pristup datotekama.

10.4. Šta je (a) logička a šta (b) fizička struktura datoteke?

(a) Logička struktura datoteke je način na koji je datoteka predstavljena korisniku. U najjednostavnijem slučaju, datoteka nema svoju logičku strukturu i predstavljena je kao kolekcija reči, odnosno bajtova. U jednostavnije logičke strukture spadaju strukture zapisa, pri čemu jedan zapis u datoteci može biti fiksne ili promenljive dužine. Primer zapisa je linija u tekstualnoj datoteci. Složenije strukture predstavljaju formatirani dokumenti.

(b) Pod fizičkom strukturom datoteke podrazumeva se način smeštanja datoteke na masovnim memorijskim medijumima, odnosno prostorni raspored datoteke u sistemu datoteka.

10.5. Šta je tip datoteke, a šta ekstenzija?

Pomoću tipa, operativni sistem može preliminarno da odredi vrstu datoteke i da je poveže sa nekom aplikacijom. Tip datoteke se može realizovati produženjem (ekstenzijom) imena datoteke, što je slučaj za DOS/Windows operativne sisteme.

10.6. Nabrojte značajnije tipove datoteka, zavisno od namene, na DOS i Windows operativnim sistemima i podrazumevane ekstenzije dodeljene tim tipovima datoteka.

- binarna izvršna datoteka (.exe, .com),
- prevedeni izvorni kod (.obj),
- statičke i dinamičke biblioteke (.lib, .dll),
- izvorni kod (c, cpp, asm, pas),
- batch izvršna datoteka (.bat),
- tekstualna datoteka (.txt),
- tekstualni dokument kreiran tekst procesorom (.rtf, .doc, sxw, .odt),
- arhive (.zip, .rar, .arj),

- rasterske slike (.jpeg, .bmp, .gif),
- multimedijalni sadržaji (.mp3, .wav, .wma, .avi, .mpeg).

10.7. Koje su prednosti i mane čuvanja imena programa kojim je datoteka kreirana sa ostalim atributima datoteke (kao što se radi na Mac OS operativnom sistemu)?

Čuvanjem imena programa kojim je datoteka kreirana, operativni sistem može implementirati funkcionalnost automatskog pozivanja programa prilikom pristupanja datoteci. Međutim, ime programa predstavlja premašenje (zahteva mesto u *file descriptoru*).

10.8. Objasnite sledeće metode pristupa datotekama: (a) direktan (b) sekvencijalni (c) pristup pomoću indeksne datoteke. Navedite primere kada aplikacija datoteci pristupa (d) sekvencijalno, (e) direktno.

(a) Informacije se prosleđuju u tačnom redosledu, jedna iza druge, uvek u odnosu na vrednost tekućeg ukazivača. Posle svakog pristupa datoteci, vrednost tekućeg ukazivača se ažurira. Sekvencijalni pristup zahteva da postoji mogućnost da se datoteka premota na početak, tako da vrednost tekućeg pokazivača bude nula.

(b) Omogućava pristup bilo kom delu datoteke, tako što se najpre odredi njegova pozicija na disku, a zatim pristupi podacima. Direktan pristup omogućava korisniku da pristupi krajnjem bloku datoteke bez čitanja prethodnog sadržaja.

(c) Svakoj datoteci (tabeli) pridružena je indeksna datoteka, uređena po nekom kriterijumu, pomoću koje se prilikom čitanja brzo može naći odgovarajući zapis. Prilikom upisa novog sloga u datoteku, ažurira se i indeksna datoteka.

(d) Štampanje sadržaja datoteke.

(e) Izmena sadržaja n-tog zapisa.

10.9. a. Šta je direktorijum?

b. Koje se operacije mogu izvršiti nad direktorijumom?

(a) Direktorijum je struktura koja sadrži kontrolne blokove svih datoteka logički smeštene u njemu.

(b) Operacije koje se mogu izvesti nad direktorijumima su: prikazivanje sadržaja direktorijuma ("listanje" direktorijuma), pretraživanje direktorijuma, promena imena datoteke, kreiranje i brisanje datoteka i poddirektorijuma u okviru tekućeg direktorijuma.

10.10. Opišite: (a) jednonivovsku strukturu direktorijuma, (b) dvonivovsku strukturu direktorijuma, (c) direktorijumsko stablo.

d. Kako se na jednonivovskom sistemu koji podržava duga imena datoteka može simulirati direktorijumsko stablo?

(a) Na sistemu datoteka postoji jedan direktorijum u kome se nalaze sve datoteke.

(b) Na prvom nivou se nalazi glavni direktorijum (MFD - master file directory), a na drugom poseban diretorijum za svakog korisnika (UFD - user file directory).

(c) Direktorijum najvišeg nivoa je početni ili korenski direktorijum (root). U svakom direktorijumu se mogu kreirati datoteke i poddirektorijumi (subdirectory) koji predstavljaju grane tog stabla.

(d) Korišćenjem specijalnih karaktera, kao što je tačka. Na primer, notacija temp.images.pic1 bi ukazivala na datoteku pic1 u poddirektorijumu images direktorijuma temp. Napomena: ovaj način simulacije nije ostvarljiv na sistemima datoteka na kojima je dužina imena datoteke ograničena na npr. 7 ili 8 karaktera.

10.11. Šta je apsolutna a šta relativna putanja datoteke? Navedite primer apsolutne i relativne putanje na UNIX sistemima.

Putanja datoteke je apsolutna, ako je izražena u odnosu na početni direktorijum (npr. /home/jonhsmith), a relativna ako je izražena u odnosu na bilo koji drugi direktorijum, osim početnog (npr. backup/disk1).

10.12. a. Šta je hard a šta simbolički link na UNIX sistemima ?

b. Kako su na Windows 2000 operativnom sistemu realizovani linkovi (*shortcuts*) ?

c. Koji od prethodno pomenutih linkova mogu ukazivati na nepostojeće objekte sistema datoteka i direktorijume ?

(a) Hard link je alternativno ime datoteke, odnosno alternativni ukazivač na i-node datoteke. Simbolički link je prečica ka objektu u sistemu datoteka,

odnosno zaseban objekat sistema datoteka koji koristi jedan i-node i jedan blok podataka u kom je zapisana lokacija originalnog objekta.

(b) Prečice (*shortcuts*) su veoma slične simboličkim linkovima na UNIX sistemu. Simbolički link na NTFS 5 sistemima datoteka je specijalna vrsta *reparse pointa* koji se upotrebljava da bi se izvršila redirekcija NTFS procesa na određenu datoteku ili direktorijum.

(c) Simbolički linkovi i prečice.

10.13. Koja je najmanja količina podataka koja se može deliti na mreži preko Windows protokola za deljenje datoteka ili preko UNIX NFS-a?

Direktorijum sa svim poddirektorijumima i datotekama.

10.14. Interpretirajte vlasničke odnose i prava pristupa za sledeću datoteku na UNIX operativnom sistemu:

```
$ ls -l file1
-rw-r--r--  1  u1  g1  509  Mar 10 17:21  file1
```

Reč o običnoj, regularnoj datoteci. Vlasnik datoteke dat1 je korisnik u1, a prava korisnika u1 u odnosu na datoteku su rw-, što znači da je može čitati i modifikovati, ali je ne može izvršavati. Datoteka dat1, koja pripada grupi g1; grupno pravo je r--, što znači da svi korisnici koji pripadaju grupi g1 mogu da pročitaju datoteku, ali je ne mogu modifikovati ni izvršavati. Pravo za ostatak sveta je r--, što žnači da svi ostali mogu čitati datoteku, ali je ne mogu modifikovati niti izvršavati.

10.15. Kako se na NTFS sistemima datoteka reguliše kontrola pristupa?

Svakom objektu, odnosno datoteci i direktorijumu (fascikla, odnosno *folder*) na NTFS sistemu datoteka se prilikom kreiranja dodeljuje *security descriptor*, odnosno grupa informacija koja se tiče kontrole pristupa. *Security decriptor* sadrži informacije o dozvolama za pristup resursu (NTFS dozvole) i o vlasniku resursa. NTFS dozvole se dodeljuju ili oduzimaju korisnicima i grupama, pri čemu svaki korisnik ili grupa predstavlja jedan *Access Control Entry* (ACE), a skup svih dozvola tabelu *Access Contol List* (ACL). Grupe i korisnici su u ovoj tabeli predstavljeni numeričkim vrednostima SID (Security Identifiers) koje ih jednoznačno identifikuju. NTFS dozvolama se kontroliše pristup objektima isključivo na NTFS volumenima (NTFS dozvole se ne mogu dodeljivati objektima na FAT sistemu datoteka), a dozvole važe bez obzira da li korisnik

pristupa resursu preko mreže ili lokalno. Prava pristupa data grupi prenose se na grupe i korisnike koji su u tu grupu učlanjeni.

10.16. Kako se formiraju efektivne NTFS dozvole za nekog korisnika?

- Različite dozvole dodeljene grupama kojima korisnik pripada se sabiraju.
- Različite nasleđene i eksplicitno dodeljene dozvole se sabiraju.
- Zabrana dozvole nadjačava dodelu.

11.17. Posmatrajte Windows NT sistem sa 5.000 korisnika. Pretpostavite da 4.950 korisnika zahteva pristup istoj datoteci.

a. Kako se ovim korisnicima mogu dodeliti prava?

b. Preporučite alternativnu metodu koja će ovim korisnicima dodeliti prava a istovremeno zadržati restrikcije nad svim ostalim korisnicima.

(a) Korisnicima se prava mogu dodeliti na dva načina:

i. Kreiranjem ACL sa imenima svih 4.950 korisnika kojima će biti dato pravo.

ii. Učlanjivanjem 4.950 korisnika u jednu grupu kojoj će biti dodeljena prava.

(b) Davanjem prava za pristup datoteci svim korisnicima (*everyone*, odnosno *authenticated users*), učlanjivanje preostalih 50 korisnika u korisničku grupu kojoj će biti data zabrana pristupa toj datoteci.

10.18. Šta je sistem datoteka i od kojih delova je sastavljen? Koji delovi predstavljaju premašenje sistema?

Sistem datoteka je skup metoda i struktura podataka koje operativni sistem koristi za čuvanje datoteka. Sistem datoteka čine: zaglavlje, meta podaci (strukture za organizaciju podataka na medijumu) i sami podaci, odnosno datoteke i direktorijumi. Zaglavlje i meta-podaci čine premašenje sistema, ali bez njih sistem datoteka ne može da funkcioniše.

10.19. Šta se podrazumeva pod preslikavanjem logičke strukture sistema datoteka u fizičku?

Pod preslikavanjem logičke strukture sistema datoteka u fizičku podrazumeva se uspostavljanje veze između sadržaja konkretne datoteke ili direktorijuma i

struktura na disku. Na primer, preslikavanje određuje da se sadržaj datoteke c:\log\log28.txt nalazi u blokovima 12,13, 25 i 26.

10.20. a. Koje su strukture podataka definisane u operativnoj memoriji radi ubrzavanja rada sa sistemom datoteka?

b. Kako se pristupa datoteci putem ovih struktura?

(a) Tabela otvorenih datoteka na sistemskom nivou i tabela otvorenih datoteka po procesu.

(b) Prilikom otvaranja datoteke kontrolni blok se iz direktorijuma upisuje u glavnu memorijsku tabelu. Prilikom svakog pristupa datoteci, iz tabele procesa se preko glavne tabele otvorenih blokova pronalaze blokovi datoteke na disku, a zatim se njima pristupa.

10.21. a. Navedite dve osnovne metode za aktiviranje sistema datoteka? Koji operativni sistemi koriste ove metode?

b. Koji problem može nastati ukoliko je jedan sistem datoteka montiran na više od jednog *mount-point* direktorijuma (pod pretpostavkom da operativni sistem to dozvoljava)?

(a) i. Mapiranje na logičke diskove, pri čemu se koreni direktorijum sistema datoteka poklapa sa korenim direktorijumom logičkog diska (DOS, Windows 98).

ii. Montiranje na drugi direktorijum, pri čemu se koreni direktorijum sistema datoteka poklapa sa direktorijumom na koji je montiran (UNIX, Linux, Windows XP).

(b) Postojale bi višestruke putanje ka istim datotekama, što može dovesti do korisničkih grešaka (npr. ukoliko veći broj korisnika pristupi istoj datoteci preko različitih putanja, a jedan korisnik obriše datoteku).

10.22. Šta je virtuelni sistem datoteka (VFS)?

VFS je objektno-orijentisani način realizacije sistema datoteka, koji omogućava korisniku da na isti način pristupa svim datotekama, bez obzira kom sistemu datoteka pripadaju. Korisnik se, putem sistemskih poziva, odnosno API-ja, obraća virtuelnom sistemu datoteka, a VFS obrađuje zahrev i upućuje poziv za rutine odgovarajućeg sistema datoteka.

10.23. Navedite osnovne načine za realizaciju direktorijuma.

- linearne liste,
- povezane linearne liste,
- uređene linearne liste,
- binarna stabla,
- linearne liste sa *hash* tabelom.

10.24. Posmatrajte datoteku veličine 100 blokova, čiji se kontrolni blok nalazi u memoriji (u slučaju indeksne alokacije, pretpostaviti da je i indeksni čvor u memoriji računara). Odredite koliko je disk I/O operacija potrebno za izvršenje sledećih operacija na sistemima organizovanim pomoću metoda kontinualne alokacije, vezivanja blokova i indeksnih čvorova. Pretpostavite da se u slučaju kontinualne alokacije datoteka može nastaviti samo sa kraja i da se blok, koji sadrži informacije za dodavanje u datoteku, nalazi u memoriji.

a. Blok se dodaje na početak datoteke.

b. Blok se dodaje na sredinu datoteke.

c. Blok se dodaje na kraj datoteke.

d. Blok se uklanja sa početka datoteke.

e. Blok se uklanja iz sredine datoteke.

f. Blok se uklanja sa kraja datoteke.

	Kontinualna alokacija	Vezivanje blokova	Metoda indeksnih čvorova
(a)	201	1	1
(b)	101	52	1
(c)	1	3	1
(d)	198	1	0
(e)	98	52	0
(f)	0	100	0

10.25. a. Koji je osnovni nedostatak metode dodele kontinualnog prostora?

b. Kakav je pristup datoteci ukoliko se za organizaciju sekundarne memorije koristi dodela kontinualnog prostora?

(a) Najveći problem ove metode je to što se datoteci odmah mora dodeliti ukupan adresni prostor na disku. Posle toga, datoteka ne može da raste ukoliko posle nje nema dovoljno kontinualnog slobodnog prostora.

(b) Pristup datoteci je direktan.

10.26. a. Pouzdanost metode vezivanja blokova je relativno mala - defekt jednog bloka učiniće nepristupačnim ostatak datoteke. Kako se može uvećati pouzdanost ove metode?

b. Da li eksterna fragmentacija predstavlja problem karakterističan za ovu metodu?

c. Da li eksterna fragmentacija predstavlja problem karakterističan za ovu metodu?

(a) Uvođenjem redundantih inverzivnih pokazivača.

(b) Ne, zato što svaki blok može da se dodeli bilo kojoj datoteci.

(c) Da, zato što su u opštem slučaju blokovi fiksne veličine, tako da zadnji blok ne mora biti u potpunosti iskorišćen.

10.27. a. Koji način organizacije koristi FAT sistem datoteka?

b. Šta je slack?

(a) FAT sistem datoteka predstavlja kombinivanu metodu kontinulane dodele prostora na nivou extent-a i mape datoteka. Sistem datoteka se prilikom kreiranja deli na extent-e, koji se nazivaju klasteri (*cluster*). U okvirima jednog klastera datoteci je dodeljen kontinualan prostor. Datoteci se može dodeliti ceo broj klastera u proizvoljnom rasporedu, ali jedan klaster ne mogu koristiti dve datoteke. Svi klasteri jedne datoteke čine njenu povezanu listu. Povezane liste svih datoteka čuvaju se u FAT tabeli (koja predstavlja mapu datoteke).

(b) Neiskorišćen deo klastera zove se slack i predstavlja gubitak.

10.28. a. Na koji se način može razrešiti problem adresiranja velikih datoteka na sistemima koji su organizovani pomoću metode indeksnih čvorova?

b. Kako ovaj problem razrešava UNIX?

(a) Vezivanjem indeksnih blokova (za opis prostornog rasporeda datoteke koristi se veći broj indeksnih blokova) i višenivovskim adresiranjem (kontrolni blok datoteke ukazuje na jedan indeksni blok prvog nivoa, koji ukazuje na n indeksnih blokova drugog nivoa; indeksni blokovi drugog nivoa adresiraju blokove datoteke; broj nivoa se po potrebi može povećati).

(b) UNIX razrešava problem tronivovskim adresiranjem blokova podataka.

10.29. Posmatrajte sistem koji podržava metode dodele kontinualnog prostora, vezivanja blokova i indeksnih čvorova. Na osnovu kog kriterijuma treba odlučiti koja je strategija optimalna za neki sistem datoteka?

Na osnovu prosečne veličine datoteka i metode pristupa datotekama. U slučaju da su datoteke relativno male i da im se najčešće pristupa sekvencijalno, koristi se dodela kontinualnog prostora. U slučaju da su datoteke velike i da im se, najčešće, pristupa sekvencijalno, koristi se metoda vezivanja blokova. U slučaju da su datoteke velike i da im se najčešće pristupa direktno, koristi se metoda indeksnih blokova.

10.30. a. Koje se metode koriste za upravljanje slobodnim prostorom?

b. Zašto se mape bitova moraju čuvati na disku, a ne u operativnoj memoriji?

(a) i. Mape bitova - za svaki blok diska uvodi se bit koji vrednostima 0 i 1 opisuje da li je taj blok slobodan ili zauzet. Mapa bitova se formira kao struktura podataka na disku, odnosno kao niz svih prethodno navedenih bitova.

ii. Povezane liste - početak liste ukazuje na prvi slobodni blok, a svaki slobodni blok ukazuje na sledeći.

(b) Ukoliko se čuva na disku, mapa bitova neće biti izgubljena u slučaju kraha sistema (na primer, otkaz memorije ili nestanak struje).

10.31. Koja su dva osnovna načina vođenja dnevnika transakcija?

- Praćenje svih promena u sistemu datoteka,
- Praćenje promena u meta strukturama.

10.32. a. Po pravilu, keš se deli na minimalno tri funkcionalna dela. Navedite te delove?

b. U kojoj situaciji je korišćenje RAM diska efikasnije od korišćenja keša?

(a) Keš podataka, keš meta-podataka, keš direktorijumskih blokova.

(b) U slučajevima kada korisnik (ili operativni sistem) znaju tačno koji će podaci sledeći biti potrebni. Keš je zasnovan na algoritmu, dok RAM diskom upravlja korisnik.

10.33. Šta je defragmentacija datoteka?

Defragmentacija datoteka je aranžiranje datoteka u kontinualne sekvence blokova.

10.34. a. Koji su osnovni delovi jednog UNIX sistema datoteka?

b. Šta sve sadrži jedan indeksni čvor (*i-node*)?

(a) zaglavlje (*superblock*), tabela indeksnih čvorova (*i-node tabela*), blokovi sa podacima (*data blocks*), direktorijumski blokovi (*directory blocks*) i blokovi indirektnih pokazivača (*indirection block*),

(b) tip objekta i pristupna prava za tri vlasničke kategorije, broj hard linkova na dati objekat, UID vlasnika, GID, veličina objekta izraženu u bajtovima, vreme zadnjeg pristupa objektu, vreme zadnje modifikacije objekta, vreme zadnje modifikacije indeksnog čvora objekta i lista direktnih i indirektnih pokazivača na blokove sa podacima.

10.35. Na FAT32 sistemu datoteka, sa klasterima veličine 2KB, nalazi se datoteka veličine 9.123 bajtova. Datoteka je redom smeštena u sledećim klasterima: 5, 10, 30, 40, 75.

a. Koliko blokova veličine 512 bajtova datoteka zauzima, a koliko blokova koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Ako je kapacitet sistema datoteka 10 GB, koliko je velika FAT tabela?

(a) Datoteka zauzima 5 klastera, a to je $5 \cdot 2KB/512B = 20$ blokova .

Datoteka koristi 9123/512 = 17,8 = 18 blokova, pri čemu je neiskorišćeno 94 bajta 18-tog bloka.

(b) Interna fragmentacija: 5·2048–9123= 1117 bajtova.

(c) broj klastera = kapacitet sistema datoteka / veličina klastera,

broj klastera = 10GB / 2KB = 5·1024·1024 = 5.242.880,

veličina FAT tabele = broj klastera · broj bajtova za alokaciju klastera,

veličina FAT tabele = 5242880 · 4B (32bit) = 20MB.

10.36. Na FAT16 sistemu datoteka, sa klasterima veličine 8KB, nalazi se datoteka veličine 22.345 bajtova. Datoteka je redom smeštena u sledećim klasterima: 6, 12, 15.

a. Koliko blokova veličine 512 bajtova datoteka zauzima, a koliko blokova koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Ako je kapacitet sistema datoteka 1 GB, odredite veličinu FAT tabele.

d. U kom bloku područja podataka se nalazi 5.699-ti bajt datoteke? Pretpostavite da se bajtovi datoteke, klasteri i blokovi u području podataka numerišu od 0 do N-1.

(a) Datoteka zauzima 3 klastera, a to je 3·8KB/512B= 48 blokova

Datoteka koristi 22.345/512 = 43,64 = 44 blokova, pri čemu je neiskorišćeno 475 bajtova 44-tog bloka.

(b) Interna fragmentacija: 3·8192–22.345= 2.231 bajtova.

(c) broj klastera = kapacitet sistema datoteka / veličina klastera,

broj klastera = 1·GB / 8KB = 0.125·1024·1024 = 131.072,

veličina FAT tabele = broj klastera · broj bajtova za alokaciju klastera,

veličina FAT tabele = 131.072 · 2B (16bit) = 250MB.

(d) Odredimo najpre u kom elementu klasterskog niza se nalazi traženi bajt:

5.699/512= 11,8 - to je 12 blok datoteke, odnosno 11 ti kad se broji od nule.

11/16 = 0 - nulti ulaz u *chain* kada se od nule, dakle prva pozicija, što znači *cluster* br. 6

ofset = 11%16=11 - treći blok, klaster 6 (blokovi se broje od 0)

Područje sa podacima:

cluster 0	0	1	2	3	...	10	11	12	13	14	15
cluster 1	16	17	18	19	...	26	27	28	29	30	31
...											
...											
cluster 6	96	97	98	99	...	106	107				

Blok sa podacima = broj klastera · veličina klastera + ofset

Blok sa podacima = 6 · 16 +11 = 107

10.37. Dat je UNIX sistem datoteka sa veličinom sistemskog bloka 1KB i indeksnim čvorovima u kojima se nalazi 10 direktnih pokazivača, jedan indirektni, jednan dupli indirektni i jednan trostruki indirektni pokazivač. U sistemu datoteka nalazi se datoteka veličine 7.899 bajtova, čiji je niz direktinih pokazivača u indeksnom čvoru: 2, 12, 55, 60, 76, 80, 111, 321, free, free.

a. Koliko blokova diska veličine 512B datoteka zauzima, a koliko efektivno koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Odredite najveću veličinu datoteke koja se može adresirati samo direktnim pokazivačima.

d. U kom bloku područja podataka se nalazi 3.144-ti bajt datoteke? Pretpostavite da su direktni pokazivači, sistemski blokovi, blokovi u području podataka i bajtovi datoteke numerisani od 0 do N-1.

(a) Datoteka zauzima 8 sistemskih blokova, odnosno 8·1KB = 8KB, što je ukupno 16 blokova diska.

Datoteka koristi 7.899/512 = 15,4 = 16 blokova, pri čemu je neiskorišćeno 293 bajta 16-tog bloka.

(b) Interna fragmentacija: 8·1024–7899= 239 bajtova.

(c) Najveća veličina datoteke = broj direktnih pokazivača · veličina sist. bloka

Najveća veličina datoteke = 10 · 1KB = 10KB

(d) 3144/512= 6.14 = 7 - Traženi bajt se nalazi u sedmom bloku datoteke (blok broj 6 kad se broji od nule)

Najpre se određuje koji direktni pokazivač upućuje na šesti blok:

DP = blok datoteke / veličina sistemskog bloka,

sitemski blok = 2 bloka diska,

DP = 6/2 = 3 - bloku odgovara pokazivač br 3. (četvrti po redu u indeksnom čvoru). Četvrti pokazivač ukazuje na sistemski blok SB = 60,

offset = 6%2 = 0 (blok 0, odnosno prvi blok, u tom sistemskom bloku 60),

Blok (područje sa podacima) = SB · veličina SB + offset,

veličina SB = 1KB = 2 bloka na disku,

Blok (područje sa podacima) = 60 · 2 +0 = 120.

10.38. Dat je UNIX sistem datoteka sa veličinom sistemskog bloka 8KB i indeksnim čvorovima u kojima se nalazi 10 direktnih pokazivača, jedan indirektni, jednan dupli indirektni i jednan trostruki indirektni pokazivač. U sistemu datoteka nalazi se datoteka veličine 22.045 bajtova, čiji je niz direktinih pokazivača u indeksnom čvoru: 7, 22, free, free, free, free, free, free, free.

a. Koliko blokova diska veličine 512B datoteka zauzima, a koliko efektivno koristi?

b. Kolika je interna fragmenacija za ovu datoteku?

c. Odredite najveću veličinu datoteke koja se može adresirati samo direktnim pokazivačima.

d. U kom bloku područja podataka se nalazi 15.600-ti bajt datoteke? Pretpostavite da su direktni pokazivači, sistemski blokovi, blokovi u području podataka i bajtovi datoteke numerisani od 0 do N-1.

(a) Datoteka zauzima 3 sistemska bloka, odnosno 3·8KB = 24KB, što je ukupno 48 blokova diska.

Datoteka koristi 22.045/512 = 43,05 = 44 blokova, pri čemu je neiskorišćeno 483 bajta 44-tog bloka.

(b) Interna fragmentacija: 3·8192–22.045= 2.531 bajtova.

(c) Najveća veličina datoteke = broj direktnih pokazivača · veličina sist. bloka.

Najveća veličina datoteke = 10 · 8KB = 80KB.

(d) 15600/512= 30.1 = 31 - Traženi bajt se nalazi u 31. bloku datoteke (blok broj 30 kad se broji od nule).

Najpre se određuje koji direktni pokazivač upućuje na blok 30:

DP = blok datoteke / veličina sistemskog bloka,

sitemski blok = 2 bloka diska,

DP = 30/16 = 1 - bloku odgovara pokazivač br 1. (drugi po redu u indeksnom čvoru). Drugi pokazivač ukazuje na sistemski blok SB = 22.

offset = 30%16 = 14 (blok 14 u sistemskom bloku 22),

Blok (područje sa podacima) = SB · veličina SB + offset,

veličina SB = 8KB = 16 bloka na disku,

Blok (područje sa podacima) = 22 · 16 +14 = 366.

11. Distribuirani sistemi

Distribuirani sistemi predstavljaju kolekciju procesora, odnosno računara koji ne dele zajedinčku memoriju i sistemski časovnik. Svaki procesor, odnosno računar ima sopstvenu lokanu memoriju, a međusobna komunikacija se ostvaruje putem mreže LAN ili WAN tipa. Osnovna namena distribuiranog sistema je da obezbedi efikasno i pogodno deljenje resursa. Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za mrežno-komunikacioni podsistem, distribuirane sisteme i distribuirane sisteme datoteka. Posle toga je problem sihronizacije procesa i zastoja proširen na distribuiranu okolinu. Čitalac može proveriti svoje znanje samostalnom izradom zadataka koji su navedeni u ovom poglavlju.

11.1. a. U čemu je razlika između potpuno i delimično povezanih mreža?

b. Koliko veza sadrži potpuno povezana mreže sa n sajtova?

(a) U slučaju potpuno povezane mreže svaki sajt ima direktnu fizičku vezu sa svim preostalim sajtovima. U slučaju delimično povezanih mreža, postoje direktne fizičke veze između nekih čvorova, ali ne između svih.

Ukoliko ne postoji direktna veza između dva čvora, poruka se mora rutirati kroz sekvencu komunikacionih veza.

(b) $n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 1$

11.2. Uporedite pouzdanost mreža sa fizičkom topologijom stabla, prstena i zvezde.

11.3. Zašto većina WAN mreža pripada kategoriji delimično povezanih mreža?

11.4. Koje su osnovne razlike između LAN i WAN mreža?

1. Razlika u razdaljini između sajtova. Sajtovi u WAN mreži mogu biti udaljeni od 100-1.000km (i više, po potrebi), dok je razdaljina između sajtova u LAN mreži manja od 1km.

2. Razlika u brzini. Sajtovi pristupaju WAN mrežama najčešće pomoću 56Kbps modema, dok se pristup LAN mrežama ostvaruje pomoću mrežnih kartica čija je brzina do 1Gbps.

Svi sajtovi u LAN mreži su relativno blizu tako da komunikacioni linkovi, po pravilu, imaju visoke brzine transfera i manju verovatnoću greške u komunikaciji. Sajtovi se povezuju kvalitetnim kablovima, kao što su UTP i optički kablovi, a mrežne strukture koje se najčeće koriste su prstenovi ili zvezde. Brzina komunikacije varira između 1Mb/sec do 1Gb/sec. WAN mreže čini znatno veći broj računara i mrežnih uređaja u odnosu na LAN. WAN mreže su distribuirane na širem geografskom prostoru, što izaziva pojavu pri kojoj su neke veze spore i nepouzdane. Tipični linkovi su telefonske linije, mikrotalasni linkovi i satelitski kanali. Oni se kontrolišu putem specijalnog komunikacionog procesora. Komunikacioni procesor je odgovoran za definisanje interfejsa pomoću koga će priključeni sajtovi komunicirati preko mreže i za obavljanje transfera podataka između sajtova.

11.5. Koji su mehanizmi neophodni da bi se u mreži ostvarila komunikacija?

Komunikacija u računarskim mrežama zahteva da se prethodno obezbede sledeći mehanizami:

- mehanizam za imenovanje računara na mreži i razrešavanja imena u adrese (na primer, DNS - *domain name system*, koji grupiše računare u logičke celine - domene, koji mogu biti hijerarhijski),

- strategije rutiranja,
- uvođenje mrežnih paketa,
- povezivanje procesa na mreži (komutacijom veze, poruke ili paketa),
- mehanizam za razrešavanje problema sudara na mreži.

11.6. a. Koji su slojevi definisani u ISO OSI komunikacionom modelu i koje su njihove funkcije?

b. Koji sloj u OSI modelu jedini dodaje zaglavlje (*trailer*) u procesu enkapsulacije podataka?

(a) Fizički sloj - upravlja mehaničkim i električnim detaljima fizičkog prenosa podataka. Nivo je implementiran u hardveru mrežnih kartica.

Sloj veze - upravlja okvirima, odnosno delovima paketa fiksne dužine, uključujući detekciju i korekciju grešaka koje su se desile na fizičkom nivou.

Mrežni sloj - odgovoran je za konekciju i rutiranje paketa, uključujući upravljanje adresama odlazećih paketa, dekodovanjem adresa dolaznih paketa i podržavanje informacija o rutiranju. Ruteri rade na ovom sloju.

Transportni nivo - odgovoran je za mrežne pristupe niskog nivoa i za prenose poruka između klijenata, uključujući podelu poruka na pakete, održavanje redosleda paketa i kontrolu toka.

Nivo sesije - realizuje sesije ili proces-proces komunikacione protokole.

Nivo prezentacije - rešava problem razlika u formatu između različitih računara na mreži, uključujući konverziju karaktera.

Nivo aplikacije - interaguje direktno sa korisnicima i bavi se sa ftp, procedurama udaljenog prijavljivanja na sistem, elektronskom poštom itd.

(b) Sloj veze.

11.7. Navedite značajnije protokole koji su prisutni u aplikacionom sloju TCP/IP skupa protokola.

- HTTP (HyperText Transport Protocol) - pristup Web stranicama,
- FTP (File Transport Protocol) - transfer datoteka,
- SMTP (Simple Mail Transport Protocol) - dolazeća pošta,

- POP3 (Post Office Protocol) - odlazeća pošta i
- DNS (Domain Name System) - razrešavanje imena u IP adrese.

11.8. a. Šta su distribuirani sistemi?

b. Od čega se, u najprostijem slučaju, može sastojati distribuirani sistem?

c. Koja je osnovna namena distribuiranih sistema?

(a) Distribuirani sistemi predstavljaju kolekciju procesora, odnosno računara koji ne dele zajedinčku memoriju i sistemski časovnik. Svaki procesor, odnosno računar ima sopstvenu lokanu memoriju, a međusobna komunikacija se ostvaruje putem mreže LAN ili WAN tipa.

(b) Od servera, klijenata i sekundarnih memorija rapostranjenih na raznim mestima.

(c) Efikasno i pogodno deljenje resursa.

11.9. Opišite: (a) klijent-server sisteme, (b) udružene sisteme, (c) asimetrično udruživanje, (d) simetrično udruživanje.

(a) U sistemu postoje računari koje predstavljaju servere (serveri podataka, serveri za izračunavanje i serveri za štampu) i računari koji koriste njihove usluge.

(b) Sistem se sastoje od udruženih računara, odnosno od dva ili više nezavisnih računara koji dele diskove i čvrsto su povezani LAN mrežom. Jedan nivo softvera za udruživanje se izvršava na svakom čvoru (računaru).

(c) Jedan server izvršava aplikaciju dok ostali (prateći) serveri prate rad glavnog servera u budnom ali neaktivnom stanju. U slučaju otkaza glavnog servera jedan od pratećih servera preuzeće njegovu ulogu.

(d) Svi serveri su aktivni i izvršavaju aplikaciju, čime se drastično poboljšavaju performase, ali sistem mora da izdrži otkaz jednog ili više servera kao i u perthodnom slučaju.

11.10. Navedite karakteristike distribuiranih sistema.

- transparentnost - distribuirani sistem korisniku treba da izgleda kao konvencionalni, centralizovani sistem,

- otpornost na greške - distribuirani sistem treba da nastavi funkcionisanje u slučaju bilo kog otkaza. Ako bilo koji sajt u distrubuiranom sistemu otkaže, ostali sajtovi mogu nastaviti i završiti započeti posao. Od distribuiranog sistema se očekuje da detektuje otkaz i da pronađe sajt koji će zameniti onoga ko je ispao iz igre,
- skalabilnost - sa povećanjem zahteva, sistem treba lako da prihvati dodavanje novih računara i resursa,
- deljenje resursa - distribuirani sistemi obezbeđuju mehanizme za deljenje datoteka, obradu informacija u distribuiranim bazama podataka, deljenje štampača i specijalizovanog hardvera,
- ubrzavanje izračunavanja deljenjem opterećenja. Proces se izdeli na celine koje se obrađuju na posebnim računarima u mreži. Posle toga, rezultati parcijalnih izračunavanja se spajaju.

11.12. U čemu je razlika između mrežnih i distribuiranih operativnih sistemam?

Pod mrežnim operativnim sistemima, korisnik može pristupiti udaljenim resursima na dva načina: procedurom prijavljivanja na odgovarajući udaljeni računar i transferom datoteka sa udaljene mašine na sopstvenu. U distribuiranim operativnim sistemima korisnik pristupa udaljenim resursima kao da su lokalni.

11.13. Koje su tri vrste migracija karakteristične za distribuirane operativne sisteme ?

- migracija podataka - korisnik koji želi da pristupi udaljenoj datoteci dobija: (1) kopiju cele datoteke ili (2) zahtevani deo datoteke,
- migracija izračunavanja - proces može inicirati obradu na udaljenoj strani (pomoću rpc poziva ili slanjem poruka), a zatim pokupiti rezultate,
- migracija procesa - proces kreiran na lokalnoj strani ne mora se izvršavati strogo na svom računaru, već se ceo proces ili neki njegovi delovi mogu izvršavati na drugim računarima.

11.14. a. Kako se u distribuiranim sistemima može detektovati otkaz?

b. U kom slučaju se na osnovu slanja poruka tipa <*I-am-up*> i <*Are-you-up*> može tačno utvrditi da li je otkazao link?

c. Kako se oporavlja sistem posle otkaza linka?

d. Kako se oporavlja sistem posle otkaza sajta?

e. Kako se sajt ili link posle oporavka vraćaju u sistem?

(a) Pomoću handshake procedure. Pretpostavimo da sajt A i B imaju fizički link koji funkcioniše. U fiksnim intervalima oba sajta šalju poruke tipa <*I-am-up*>. Ako npr. sajt A ne primi ovu poruku u propisnom intervalu, onda se pretpostavlja da je otkazao ili sajt ili link ili da je poruka izgubljena.

(b) Ukoliko sajt A pretpostavi mogući otkaz sajta B ili linka, poslaće poruku <*Are-you-up*> sajtu B. Ako i dalje nema odgovora, sajt A pošalje istu poruku preko druge putanje i ako mu se ovaj javi, tada je prva putanja otkazala.

(c) Ako je direktni link između A i B otkazao, to mora da se objavi svim računarima u sistemu, kako bi se različite tabele rutiranja ažurirale.

(d) Ako se pretpostavi da je host otkazao, svaki sajt u sistemu mora da dobije tu informaciju kako više ne bi pokušavao da ostvari konekciju sa tim sajtom.

(e) Kada se link ili host oporave, moraju se ponovo vratiti u sistem. Link se unosi u tabele za rutiranje, a svi sajtovi se obaveštavaju da je prethodno otkazani sajt ponovo u fukciji.

11.15. a. Objasnite tri načina za mapiranje imena datoteka u distribuiranim sistemima datoteka?

b. Šta je transparentnost a šta nezavisnost lokacije datoteke u distribuiranim sistemima datoteka?

c. Koji način mapiranja obezbeđuje transparentnost, a koji nezavisnost lokacije?

(a) 1. Kombinacija ime računara - ime datoteke <*host:filename*>. Ime datoteke se sastoji iz imena računara na kome se nalazi (*hostname*) i imena datoteke na tom računaru.

2. NFS koncept (*network filesystem*). Udaljeni direktorijumi se montiraju na lokalne direktorijume. Time se dobija stablo datoteka koje u potpunosti izgleda kao da se u celini nalazi na lokalnom računaru.

3. Totalna integracija.

(b) Transparentnost lokacije znači da ime datoteke ne otkriva ni jednu pojedinost oko njene fizičke lokacije. Nezavisnost lokacije znači da ime

datoteke ostaje isto, odnosno da ne mora da se menja ako se promeni fizička lokacija datoteke.

(c) <*host:filename*> ne obezbeđuje ni transparentnost ni nezavisnost lokacije,

NFS koncept obezbeđuje transparentnost, ali ne i nezavisnost lokacije,

Totalna integracija obezbeđujee i transparentnost i nezavisnost lokacije.

11.16. a. Na koji način DFS klijent proverava koherentnost podataka u kešu?

b. Kako se problem keš koherencije može rešiti na nivou servera?

c. Koja je metoda pouzdanija? Koja metoda manje degradira performanse, posmatrano sa klijentske strane?

(a) Klijent inicira proveru validnosti podataka, odnosno konzistentnost sa glavnom kopijom datoteke: (1) pre svakog pristupa, (2) samo na početku ili (3) u nekim periodičnim regularnim intervalima.

(b) Server vodi strogu evidenciju o svakom klijentu, prati koji je klijent uzeo koji deo datoteke na čitanje i koji je deo datoteke promenjen. Kada server detektuje konflikt, odnosno kada jedan klijent promeni deo datoteke, server obaveštava sve klijente koji koriste taj isti deo datoteke da preuzmu poslednju verziju podataka.

(c) Pouzdanija i brža metoda (s tačke gledišta klijenta) je provera validnosti podataka koju inicira server.

11.17. a. Šta je replikacija datoteka u DFS?

b. Koje su prednosti korišćenja replikacije u DFS?

c. Predložite jedan način za ažuriranje replika posle izmene?

(a) Replikacija je kreiranje više kopija iste datoteke i njihovo smeštanje na različite računare na mreži. Lokacije replika se skrivaju od korisnika, koji najčešće i nije svestan njihovog postojanja.

(b) Povećavaju se pouzdanost i performanse pri čitanju (uvek se bira replika koja je najbliža klijentu).

(c) Za klijenta se uvodi primarna replika koja se prva ažurira posle izmene datoteke. Ostale replike dobijaju poruke da nisu više validne i da se moraju ažurirati najkasnije do sledećeg korišćenja.

11.18. a. Definišite relaciju *happened before* (relaciju označite simbolom ->).

b. Predstavite relacijom događaje slanja poruka..

c. Neka je svakom fizičkom događaju dodeljena vremenska oznaka - TS (timestamp). Kako se relacija *happened before* može definisati pomoću ovih oznaka?

(a) Ako su A i B događaji istog procesa i ako je događaj A izvršen pre događaja B, tada važi A -> B.

(b) Ako je događaj A slanje poruke, a događaj B primanje te poruke, pri čemu različiti procesi šalju i primaju poruku, tada je A -> B.

(c) Relacija hapended before se za svaki par događaja A i B definiše na sledeći način: ako je A->B, tada je TS događaja A manji od TS događaja B, tj. TS(A)<TS(B).

11.19. Razmotriti sledeći slučaj konkurentnih atomskih transakcija, koje implemntiraju TS protokol: dat je zapis Q u baznoj datoteci koji ima sledeće *read* i *write* vremenske oznake:

- timestamp zadnjeg čitanja: R(Q)=100,
- timestamp zadnjeg upisa: W(Q)=150.

Na izvršenje čeka transakcija čitanja Ti, sa vremenskom oznakom TS(Ti)=50.

a. Iz koje strukture će se dogoditi čitanje: iz Q zapisa ili iz log datoteke (dnevnika)?

b. Koje su nove vrednosti R(Q) i W(Q)?

(a) TS(Ti)=50, W(Q)=150 $\Rightarrow$ TS(Ti) < W(Q)

Transakcija Ti traži vrednost Q koja pripada prošlosti, i kao takva je prepisana. Čitanje se odbacuje, a vrednost se dobija primenom *roll-back* metode, za Ti - čitanje se obavlja iz dnevnika transakcija pomoću roll-back metode.

(b) Ti nije imala uticaj na zapis Q, timestamp zadnjeg čitanja i upisa za zapis Q ostaju netaknuti: R(Q)=100, W(Q)=150

11.20. Razmotriti sledeći slučaj konkurentnih atomskih transakcija, koje implemntiraju TS protokol: dat je zapis Q u baznoj datoteci koji ima sledeće *read* i *write* vremenske oznake:

- timestamp zadnjeg čitanja: R(Q)=100,
- timestamp zadnjeg upisa: W(Q)=150.

Na izvršenje čeka transakcija čitanja Ti, sa vremenskom oznakom TS(Ti)=200.

a. Iz koje strukture će se dogoditi čitanje: iz Q zapisa ili iz log datoteke (dnevnika) ?

b. Koje su nove vrednosti R(Q) i W(Q) ?

(a) TS(Ti)=50, W(Q)=150 $\Rightarrow$ TS(Ti) $\geq$ W(Q)

Zahtev je korektan, čitanje se odvija iz zapisa Q, a posle toga se ažurira vremenska oznaka R(Q).

(b) R(Q)=200, W(Q)=150

11.21. Razmotriti sledeći slučaj konkurentnih atomskih transakcija, koje implemntiraju TS protokol: dat je zapis Q u baznoj datoteci koji ima sledeće *read* i *write* vremenske oznake:

- timestamp zadnjeg čitanja: R(Q)=100,
- timestamp zadnjeg upisa: W(Q)=150.

Na izvršenje čeka transakcija upisa Ti, sa vremenskom oznakom TS(Ti)=50.

a. Da li će doći do upisa i gde?

b. Koje su nove vrednosti R(Q) i W(Q)?

(a) TS(Ti)=50, R(Q)=100 $\Rightarrow$ TS(Ti) < R(Q)

Zahtev je nekorektan, jer transakcija pokušava upis u nešto što je već trebalo da bude pročitano. Upis u Q se odbacuje, Ti obavlja upis u dnevnik transakcija, a sve transakcije koje su čitale pogrešno (sve transakcije čitanja između TS(Ti) i sledećeg upisa u Q) moraju da se nateraju na *roll-back*.

(b) Ti nije imala uticaj na zapis Q, timestamp zadnjeg čitanja i upisa za zapis Q ostaju netaknuti: R(Q)=100, W(Q)=150

11.22. Razmotriti sledeći slučaj konkurentnih atomskih transakcija, koje implemntiraju TS protokol: dat je zapis Q u baznoj datoteci koji ima sledeće *read* i *write* vremenske oznake:

- timestamp zadnjeg čitanja: R(Q)=100,
- timestamp zadnjeg upisa: W(Q)=150.

Na izvršenje čeka transakcija upisa Ti, sa vremenskom oznakom TS(Ti)=120.

a. Da li će doći do upisa i gde?

b. Koje su nove vrednosti R(Q) i W(Q)?

(a) TS(Ti)=120, R(Q)=100 $\Rightarrow$ TS(Ti) > R(Q)

TS(Ti)=120, W(Q)=100 $\Rightarrow$ TS(Ti) < W(Q)

Zahtev je nekorektan jer transakcija Ti pokušava upis zastarele vrednosti. Upis u Q se odbacuje, Ti obavlja upis u dnevnik transakcija, ali ni jedna transakcija čitanja ne mora da radi *roll-back*.

(b) Ti nije imala uticaj na zapis Q, timestamp zadnjeg čitanja i upisa za zapis Q ostaju netaknuti: R(Q)=100, W(Q)=150

11.23. Razmotriti sledeći slučaj konkurentnih atomskih transakcija, koje implemntiraju TS protokol: dat je zapis Q u baznoj datoteci koji ima sledeće read i write vremenske oznake:

- timestamp zadnjeg čitanja: R(Q)=100,
- timestamp zadnjeg upisa: W(Q)=150.

Na izvršenje čeka transakcija upisa Ti, sa vremenskom oznakom TS(Ti)=180.

a. Da li će doći do upisa i gde?

b. Koje su nove vrednosti R(Q) i W(Q)?

(a) TS(Ti)=50, W(Q)=150 $\Rightarrow$ TS(Ti) $\geq$ W(Q)

Zahtev je korektan, transakcija obavlja upis u Q, a posle toga se ažurira vremenska oznaka W(Q).

(b) R(Q)=200, W(Q)=180

11.24. Objasnite moguće načine za realizaciju međusobnog isključenja u distribuiranim sistemima?

- centralizovan pristup: Jedan od procesa u sistemu je određen da odobrava ulaz u kritičnu sekciju i naziva se koordinator. Svaki proces koji želi da uđe u kritičnu sekciju obraća se koordinatoru sa zahtevom. Koordinator održava red čekanja za ulazak u kritičnu sekciju, po nekom algoritmu odabira procese iz reda i daje im odobrenje za ulazak u svoje kritične sekcije.
- puni distribuirani pristup: Proces Pi, koji želi da uđe u svoju kritičnu sekciju, generiše novi timestamp, TS, i šalje poruku request (Pi, TS) svim ostalim procesima u sistemu, odnosno zahteva za ulaz u kritičnu sekciju. Proces Pj posle primanja poruke odlaže slanje odgovora ukoliko se nalazi u kritičnoj sekciji, odnosno odgovara trenutno ako nije u kritičnoj sekciji i nema nameru da uđe u nju. Proces Pi može ući u svoju kritičnu sekciju tek kad primi odgovor od svih drugih procesa u sistemu.
- slanje žetona: Rešenje se zasniva na specijalnoj poruci, žetonu (*token*), koji kruži između svih procesa u sistemu. Procesi obrazuju logički krug. Proces koji je dobio žeton ima pravo da uđe u svoju kritičnu sekciju. Ako proces ne želi da uđe u svoju kritičnu sekciju, prosleđuje ga dalje.

11.25 Kako se može ostvariti sinhronizacija transakcija u centralizovanim sistemima?

- Može se ostvariti orišćenjem protokola za zaključavanje. Transakcija Ti pre pristupa zapisu Q mora da zatraži odgovarajuće pravo zaključavanja. Transakcija pristupa zapisu isključivo nakon zaključavanja.
- Može se ostvariti korišćenjem protokola zasnovanog na vremenskim oznakama.

11.26 Kako se može ostvariti izvršenje atomskih transakcija u distribuiranim uslovima?

Izvršavanje atomske transakcije omogućava se uvođenjem koordinatora transakcija na svakom računaru. Koordinaror kontroliše izvršavanje svih transakcija koje je taj računar inicirao, odnosno pokreće izvršavanje transakcije, razbija jednu transakciju u određeni broj podtransakcija i distribuira te podtransakcije ostalim sajtovima na izvršavanje, a slučaju otkaza prekida izvršenje transakcije na svim sajtovima.

11.27. a. Koje se metode za prevenciju zastoja zasnovane na vremenskim oznakama koriste u distribuiranim sistemima?

b. Koja od ovih metoda koristi pretpražnjenje (oduzimanje resursa)?

(a) 1. Metoda *wait-die* (stariji proces čeka na mlađeg da otpusti resurs). Proces Pi, koji traži resurs dodeljen procesu Pj, čekaće da proces Pj taj resurs otpusti samo ako je TS(Pi)<TS(Pj). U protivnom, proces Pi se podvrgava *rollback* operaciji.

2. Metoda *wound-wait* (stariji proces nikada ne čeka na mlađe). Proces Pi čekaće na resurs dodeljen procesu Pj samo ako je Pi mlađi, odnosno ako je TS(Pi)<TS(Pj). U protivnom, Pj se podvrgava rollback operaciji, a resurs mu se oduzima i predaje procesu Pi.

(b) Metoda *wait-die*.

11.28 Koje se obavlja (a) centralizovana, a kako (b) distribuirana detekcija zastoja u distribuiranim sistemima?

(a) Problem zastoja rešava jedan sajt, na kome se izvršava proces koordinator za detekciju zastoja. Proces koordinator traži od ostalih sajtova da mu dostave svoj *wait-for* graf, kako bi na osnovu toga konstruisao zajednički *wait-for* graf, na osnovu kog procenjuje da li je distribirani sistem u stanju zastoja ili ne.

(b) Svaki sajt konstruiše svoj modifikovani *wait-for* graf uključujući Pex čvor (eksterni čvor). Kada primeti mogućnot zastoja, sajt šalje poruku kritičnom sajtu sa kojim se može naći u stanju zastoja da oba sajta dodatno provere da li je sistem u zastoju.

11.29. Ako je red prispelih transakcija:

read TS(T1)=150,

read TS(T2)=80,

write TS(T3)=75,

write TS(T4)=160,

odredititi kakav će redosled napraviti timestamp protokol, u cilju izbegavanja *roll-back* efekata.

Da bi se izbegli kaskadne roll-back operacije, u sinhronizaciji transakcija čitanja i upisa moraju se poštovati sledeća pravila:

- transakcija Ti za čitanje podataka x mora biti odložena, ako postoji transakcija Tj koja treba da izvrši upis u x, takva da je TS(Tj)<TS(Ti) (odloži čitanje ako postoji upis pre čitanja),
- transakcija Ti za upis podataka u x mora da biti odložena ako postoji transakcija Tj koja čita ili upisuje podatke u x, takva da je TS(Tj)<TS(Ti) (odloži upis ukoliko pre upisa postoji čitanje ili drugi upis).

Zadatak se rešava jednostavno, prvo ide najstari upis, zatim čitanje i tako redom. Optimalna sekvenca je: T3, T2, T1, T4.

11.30. Ako je red prispelih transakcija:

read TS(T1)=150,

read TS(T2)=60,

write TS(T3)=75,

read TS(T4)=160,

odrediti kakav će redosled napraviti timestamp protokol, u cilju izbegavanja *roll-back* efekata.

Optimalne sekvence su: (1) T2, T3, T1, T4 i (2) T2, T3, T4, T1. Redosled transakcija T1 i T4 je nebitan jer su i T1 i T4 transakcije čitanja.

11.31. Ako je red prispelih transakcija:

read TS(T1)=150,

read TS(T2)=60,

read TS(T3)=75,

read TS(T4)=160,

odrediti kakav će redosled napraviti timestamp protokol, u cilju izbegavanja *roll-back* efekata.

Sve sekvence mogu se smatrati optimalnim jer nema transakcija upisa.

12. Zaštita i sigurnost

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za zaštitu operativnih sistema, odnosno na kontrolu pristupa resursima, i sigurnost, koja uključuje interakcije sistema sa spoljnim svetom. Razmotreni su domeni zaštite i matrice pristupa, aspekti sigurnosti i tehnike za povećanje sigurnosti sistema.

12.1. Na šta se svodi problem zaštite u operativnim sistemima?

Problem zaštite svodi se na kontrolu pristupa objektima operativnog sistema: objektima mogu pristupati samo oni korisnici koji na to imaju pravo, odnosno koji su autorizovani i nad objektom mogu izvšiti samo operacije koje pripadaju dozvoljenom skupu operacija.

12.2. Šta je domen zaštite?

Domen zaštite je kolekcija prava pristupa koja su definisana parovima (ime objekta, skup prava). Pravo pristupa je mogućnost izvršenja operacije nad objektom.

12.3. a. Kako su domeni definisani u UNIX operativnom sistemu?

b. Kako se na UNIX sistemu obavlja prebacivanje domena?

(a) Kod UNIX operativnog sistema, domeni su definisani na bazi korisnika (domain = UID).

(b) Prebacivanje domena može se realizovati putem sistema datoteka - svakoj datoteci može se dodeliti domenski bit (setuid - SUID bit). Ako se pokrene program sa postavljenim domenskim bitom, korisnik dobija identitet vlasnika datoteke; kada se program završi UID se resetuje, odnosno vraća na staru vrednost.

12.4. U čemu je razlika između liste za kontrolu pristupa objektima i liste sposobnosti?

Lista kontrole pristupa formira se za svaki objekat sistema i odgovara jednoj koloni matrice pristupa. Listu čini skup uređenih parova (domen, skup prava) - u listi su opisani svi domeni koji nad tim objektom imaju neka prava, a domeni bez prava se ne uključuju. Jednostavno rečeno, lista opisuje operacije koje procesi koji pripadaju različitim domenima mogu izvršiti nad tim objektom.

Lista sposobnosti (*capability* list) formira se za svaki domen i odgovara jednoj vrsti matrice prava pristupa. Listu čini skup uređenih parova (objekat, pravo pristupa) - u listi su opisani svi objekti nad kojima taj domen ima neka prava. Jednostavno rečeno, lista sposobnosti jednog domena opisuje operacije koje procesi tog domena mogu izvršiti nad različitim objektima.

12.5. Sigurnost obuhvata zaštitu sistema koji nisu izolovani. Od čega treba zaštititi sistem?

Sigurnost obuhvata zaštitu sistema od:

- neautorizovanog pristupa podacima i resursima,
- zlonamerne (maliciozne) modifikacije podataka,
- zlonamernog uništenja podataka i
- sprečavanja da se sistem legitimno koristi (denial of service).

12.6. Na kojim nivoima se primenjuju sigurnosne mere?

- fizički nivo,
- ljudski faktor,
- mrežni nivo,
- nivo operativnog sistema.

12.7. Na koje se sve načine može obaviti autentifikacija korisnika?

- specijalnim hardverom, kao što je ključ ili ID kartica,
- navođenjem poverljivih informacija, kao što je lozinka i
- biološkim atributima korisnika (biometrics), kao što su otisak prsta, snimak mrežnače oka (retina scan) i potpis.

12.8. Šta nije u redu sa sledećim parovima: korisničko ime - lozinka?

a. ppetar – pera,
b. root – password,
c. redneck - 1Beer$2c50,
d. admin – usetheforce,

e. ppetar – aardvark.

(a) Napadač će u prvih pet iteracija probati ovu lozinku.

(b) Ovo je verovatno prva lozinka koju će napadač da isproba.

(c) Lozinka je u redu - lako se pamti, a teško pogađa.

(d) Citat iz svima dobro poznate serije.

(e) Lozinka se lako pogađa na osnovu rečnika (dictionary-based attack).

12.9. Objasnite sledeće napade: (a) DoS, (b) spoofing i (c) sniffing.

(a) DoS (Denial of Service, odbijanje usluga) kao napad izaziva prestanak rada servisa ili programa, čime se drugima onemogućava rad sa tim servisima ili programima.

(b) Napadač prati IP adrese u IP paketima i predstavlja se kao drugi računar.

(c) Specijalnim programima presreću se TCP/IP paketi koji prolaze kroz određeni računar i po potrebi pregleda njihov sadržaj.

12.10. Šta je (a) trojanski konj, (b) klopka - trap door, (c) prepunjenje steka i bafera?

(a) Trojanski konj je ilegalni segment koda podmetnut u kod programa u cilju promene funkcija ili ponašanja originalnog programa.

(b) Klopka (*trap door*) je slučajno ili namerno ostavljeno mesto u kodu u koje uljez može da podmetne svoj kod i time ostvari neku dobit.

(c) Napadač korisiti grešku u programu, odnosno nedovoljno kontrole po pitanju razdvajnja steka, podataka i koda. Napadač šalje više ulaznih podataka, nego što program očekuje, sve dok ne dođe do steka, zatim prepisuje važeću adresu u steku adresom svog koda, koji se smešta u stek u sledećom koraku i puni deo steka svojim kodom, koji, npr. izvršava neku komandu (kopira neke podatke ili pokreće komandni interpreter).

12.11. Šta su (a) crvi, (b) virusi?

(a) Crv je proces koji koristi mehanizam umnožavanja radi degradacije sistemskih performansi.

(b) Virusi su fragmenti koda koji se ubacuju u druge legitmne programe i kao takvi predstavljaju jedan od glavnih problema personalnih računara. Virusi se mogu definisati kao hibrid crva i trojanskog konja.

12.12. Koje se metode koriste za detekciju napada na sistem?

- Detekcija zasnovana na oznaci. IDS analizira mrežni saobraćaj i traži se karakterističan uzorak koji otkriva napad, kao što je tipična sekvenca vistrukog pogrešnog logovanja.
- Detekcija anomalije. IDS otkriva anomalije u sistemu, kao što su sistemski pozivi sa velikom količinom ulaznih podataka (koji ukazuju na pokušaj napada tipa prepunjenja bafera).

13. Korisnički interfejs

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za interfejs koji operativni sistem pruža prema krajnjem korisniku. Razmotrene su dve vrste korisničkih interfejsa - alfanumerički i grafički i kontrola posla na sistemima sa paketnom obradom.

13.1. Navedite osnovne karakteristike (a) alfanumeričkih i (b) grafičkih korisničkih interfejsa.

(a) Korisnik podatke unosi i vidi u obliku alfanumeričkih karaktera. Alfanumerički korisnički interfejs implicira postojanje jezika, jednostavnog ili kompleksnog, pomoću kog se može ostvariti komunikacija, a tip i karakteristike jezika zavise od vrste operativnog sistema. Poželjne osobine svih jezika su razumljivost i jednostavnost korišćenja.

(b) Grafičko radno okruženje korisnicima najčešće nudi potpun skup alata za rad sa datotekama, korišćenje i administraciju sistema. Grafičko radno okruženje, po pravilu, opterećuje procesor i na nekim sistemima povećava rizik u smislu sigurnosti sistema.

13.2. Šta je komandni interpreter?

Komandni interpreter je sistemski proces koji prihvata komande koje korisnik zadaje, zatim ih interpretira i potom izvršava. Po potrebi, komandni interpreter pokreće odgovarajuće programe.

13.3. Koje su poželjne osobine komandnog interpretera?

- intuitivno vezana imena komandi sa akcijom koju komande obavljaju,
- prosleđivanje podrazumevanih vrednosti parametara,
- što jednostavniji format komande,
- dodela kraćeg imena komandi ili nizu komandi,
- ponavljanje prethodno zadatih komandi,
- korisniku mora biti jasno kada je sistem spreman da prihvati novu komandu, a kada je zauzet izvođenjem prethodno zadate komande,
- obaveštenje o rezultatima izvršenja komande.

13.4. Koje dodatne funkcije obavlja komandni interpreter?

- preusmeravanje ulaza i izlaza,
- povezivanje komandi u pipeline,
- zamena imena datoteka,
- rukovanje promenjivim i kontrola okruženja,
- programiranje u komandnom interpreteru.

13.5. Koje informacije spadaju u kontrolišuće informacije koje čine opis jednog posla?

- Obračunske informacije, koje se koriste radi provere validnosti korisnika i formiranja obračuna usluga na osnovu utrošenih resursa sistema.
- Informacije o raspoređivanju, koje sadrže spisak maksimalnih količina različitih resursa, koje su neophodne za izvršenje posla. Ukoliko se posao može razdvojiti na nekoliko zasebnih celina, informacije o resursima se navode za svaku celinu.
- Informacije o korišćenju ulazno-izlaznog podsistema, koje uključuju zahteve za korišćenjem ulazno-izlaznih uređaja i datoteka.
- Proceduralne informacije, koje opisuju šta korisnik očekuje od operativnog sistema.

13.6. Kako je na UNIX i Linux sistemima realizovano grafičko radno okruženje?

X Window server obezbeđuje skup alata za implementaciju grafičkog interfejsa, a programi za upravljanje prozorima (window manageri), kao što su KDE i Gnome, koriste usluge X servera i obezbeđuju grafički interfejs prema korisniku.

14. Windows familija operativnih sistema

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za principe dizajna i arhitekturu *Windows* familije operativnih sistema.

14.1. Odredite kojoj kategoriji operativnih sistema pripada MS-DOS, posmatrano sa stanovišta broja korisnika i procesa koji se mogu istovremeno izvršavati.

MS-DOS je jednokorisnički-jednoprocesni operativni sistem.

14.2. Kako se na MS-DOS sistemu obavlja kontrola pristupa?

Kontrola pristupa nije implementirana.

14.3. Korisnik želi da na MS-DOS sistemu pokrene softver sa određenim memorijskim zahtevima. Koji drajver (veznik) treba učitati u memoriju ukoliko program zahteva:

a. samo osnovnu memoriju,

b. extended memoriju,

c. expaned memoriju.

(a) Nepotrebno je učitati bilo koji drajver.

(b) HIMEM.SYS (učitava se iz datoteke CONFIG.SYS)

(c) HIMEM.SYS i EMM386.EXE (učitavaju se iz datoteke CONFIG.SYS)

14.4. Koje je drajvere potrebno učitati u memoriju na MS-DOS sistemu da bi se za korisnike obezbedila podrška za CD-ROM uređaje?

Drajver za CD-ROM uređaj konkretnog proizvođača ili univerzalni ATAPI CD drajver (učitava se iz CONFIG.SYS datoteke) i Microsoft CD ekstenziju (MSCDEX.EXE, učitava se iz AUTOEXEC.BAT datoteke).

14.5. Opišite *Windows* 3.x sa stanovišta (a) upravljanja procesorom, (b) upravljanja memorijom.

(a) *Windows* 3.x je jednokorisnički sistem sa delimičnim pretpražnjenjem. Više procesa mogu se izvršavati kvazi-paralelno, ali o trenutku kada se procesor oslobađa i dodeljuje drugom procesu, ne odlučuje operativni sistem, nego sam proces.

(b) *Windows* 3.x koristi 16-bitno adresiranje memorije i sposoban je da pokreće MS-DOS programe i Win16 programe. Win16 programi dele se na segmente, odnosno blokove memorije veličine između 16 bajtova i 64KB. Zajednički problem svih 16-bitnih verzija Windowsa je nemogućnost oslobađanja rezervisane sistemske memorije: *Windows* zahteva od aplikacija da formiraju i brišu objekte u rezervisanoj sistemskoj memoriji; ukoliko aplikacija zaboravi da obriše objekat, sistemska memorija biva izgubljena. Ukoliko sistem radi duže vreme bez prestanka, postoji mogućnost značajnog gubitka sistemske memorije. U *Windows* 3.x uveden je koncept virtuelne memorije. Virtuelna memorija realizuje se pomoću swap datoteke, čija veličina može biti fiksna ili promenljiva.

14.6. *True-type* tehnologija se smatra jednom od značajnijih novina koju je doneo Windows 3.1. Opišite ukratko *true-type* tehnologiju.

14.7. Objasnite koncept dinamičkog povezivanja.

14.8. U koju je verziju *Windows* operativnog sistema prvi put uveden koncept umrežavanja na nivou radnih grupa ?

Windows for Workgroups 3.11

14.9. Kakvo je multiprogramiranje prisutno na *Windows* 9x operativnim sistemima?

Multiprogramiranje sa pretpražnjenjem.

14.10. Jedna od osnovnih razlika između 16-bitnih i 32-bitnih Windows sistema je u broju niti koje pripadaju jednom teškom procesu. Objasnite.

32-bitni Windows ne ograničava broj niti u procesu na jednu. Proces započinje izvršenje kao jedna nit, koja može stvoriti dodatne niti.

14.11. Koji su formati izvršnih datoteka mogu pokrenuti na *Windows* 98 sistemu?

- MZ (MS-DOS program) - relokaciona izvršna datoteka.
- NE (*new executable*, Win16) - segmentno orijentisani Win16 program.
- PE (*portable executable*, Win32) - prenosive izvršne datoteke. Win32 programi se učitavaju kao memorijski mapirane datoteke. Mehanizam se zasniva na tome da upravljač virtuelnom memorijom povezuje datoteku na disku sa opsegom memorijskih adresa. Stranice se čitaju sa diska samo onda kada se pristupa odgovarajućim memorijskim adresama.
- LE (*linear executable*, VxD) - linearne izvršne datoteke, odnosno 32-bitni upravljački programi uređaja, uvedeni u Windowsu 3.0.

14.12. Koji su ciljevi razvoja Windows NT 4.0 operativnog sistema?

- portabilnost,
- proširivost, odnosno lako ostvarljiva podrška za nove uređaje,
- pouzdanost,
- kompatibilnost,
- sigurnost,
- višeprocesorska podrška,
- visoke performanse.

14.13. Koji su osnovni delovi u arhitekturi *Windows* NT sistema?

- HAL,
- Kernel,
- Egzekutiva,
- podsistemi okruženja,
- zaštitni podsistem.i

14.14. Koje funkcije obavlja kernel *Windows* NT sistema?

- dodela procesora nitima,
- rukovanje hardverskim prekidima (kernel prosleđuje prekide odgovarajućim nitima i drajverima) i hardverski i softverski generisanim izuzetcima (kao što je pristup nepostojećoj memorijskoj lokaciji),
- sinhronizacija procesora niskog nivoa ,
- oporavak posle greške ili kvara u napajanju.

14.15. Koje funkcije obavlja egzekutiva *Windows* NT sistema?

- upravljanje objektima,
- kontrola pristupa,
- upravljanje procesima,
- lokalno pozivanje procedura (Local Procedure Call - LPC),
- upravljanje virtuelnom memorijom,
- upravljanje ulazno-izlaznim podsistemom.

14.16. a. Koji podsistemi okruženja postoje na Windows NT operativnom sistemu?

b. Koliko se virtuelnih mašina podiže ukoliko se izvršavaju tri Win16 aplikacije?

c. Koliko se virtuelnih mašina podiže ukoliko se izvršavaju tri DOS aplikacije?

(a) Win32 podsistem, Win16 podsistem, OS/2 podsistem, POSIX podsistem i MS-DOS okruženje.

(b) Jedna WOW VDM, pri čemu se jedna nit WOW virtuelne mašine dodeljuje svakoj Win16 aplikaciji.

(c) Tri virtuelne DOS mašine, od kojih se svaka dodeljuje po jednoj DOS aplikaciji.

14.17. Kako međusobno komuniciraju Windows 2 000 podsistemi okruženja?

Slanjem poruka.

14.18. Koje komponente Windows 2000 operativnog sistema rade u zaštićenom režimu, a koje u korisničkom režimu rada?

HAL, kernel i egzekutiva rade u zaštićenom režimu rada (*protected mode*).

Kolekcija podsistema radi u korisničkom režimu rada (*user mode*).ž

14.19. Koje funkcije obavlja HAL Windows 2000 operativnog sistema?

Kod zavisan od procesora je izolovan u DLL biblioteke koje čine sloj apstrakcije harvera (hardware abstraction layer - HAL). Zahvaljujući tome, Windows 2 000 može biti preseljen sa jedne arhitekture na drugu sa relativno malo izmena.

14.20. Šta je vlakno (*fiber*)?

Vlakno (fiber) je kod koji radi u korisničkom režimu i koji biva raspoređen, saglasno algoritmu definisanom od strane korisnika. Samo jedno "vlakno" se može izvršavati u jednom trenutku vremena, čak i na višeprocesorskom hardveru. Windows 2000 sadrži vlakna da bi olakšalo portovanje nasleđenih Unix aplikacija koje su napisane za *fiber execution* model.

14.21. Na koji se način diskovi na Windows 2 000 operativnom sistemu mogu kombinovati u logičke volumene tolerantne na otkaze?

- *volume set* - prosta konkatenacija većeg broja diskova,
- *stripe set* - raspoređivanje logičkih klastera na više diskova po round robin algoritmu,
- *stripe set* sa proverom parnosti - varijacija *Stipe set*-a sa proverom parnosti, poznatija kao Raid nivo 5,
- *mirror Set* - identični podaci se upisuju na oba diska, odnosno particije koje čine set.

14.22. Čemu služi (a) *System Restore*, a čemu (b) *Driver Rollback* ?

(a) *System Restore* omogućava povratak sistema u prethodno stanje bez gubitka podataka. System Restore automatski kreira presečne tačke na koje se sistem može vratiti.

(b) Prilikom nadogradnje ili instalacije novih drajvera za hardver, Windows XP kreira kopiju postojećih stabilnih drajvera. Ukoliko se novi ili

nadograđeni drajveri ne pokažu kao pouzdani, administrator može vratiti stare.

14.23. Koje serverske funkcije može da obavlja Windows 2 003 server?

- domen kontroler,
- server za datoteke i DFS server (distribuirani sistema datoteka),
- server za štampu (sa podrškom za MS-DOS, Windows, UNIX, Linux, Mac OS i Novell Netware klijente),
- terminal server,
- server za udaljeni pristup i VPN,
- server koji pruža razne TCP/IP servise (kao što su DHCP i DNS),
- web server i server za elektronsku poštu.

14.24. Opišite ukratko ključne nove tehnologije Longhorn operativnog sistema: *Avalon*, *Indigo* i *WinFS*.

- *Avalon* - novi podsistem korisničkog interfejsa i API baziran na XML-u, .NET-u i vektorskoj grafici koji će omogućiti bolje korišćenje 3D kompjuterskog grafičkog hardvera i Direct3D tehnologija,
- *Indigo* - servisno orijentisan sistem poruka koji treba da omogući da programi interoperiraju kao deo .NET okruženja;
- *WinFS* (*Windows File System*) - kombinovana relaciona baza podataka i sistem datoteka, koji će omogućiti prezentovanje objekata i njihovih međusobnih relacija.

15. Linux

Pitanja navedena u ovom poglavlju odnose se na osnovne pojmove vezane za principe dizajna i arhitekturu Linux operativnog sistema. Administracija Linux sistema obrađena je u dodatku A.

15.1. Za koju procesorsku arhitekturu je namanjen Linux operativni sistem?

Linux je prvobitno namenjen 32-bitnim Intel x86 mikroprocesorima (počevši od 80386), na kojima može funkcionisati kao radna stanica ili kao server. Jezgro Linux sistema je kasnije modifikovano i prilagođeno procesorima koji ne pripadaju Intel x86 klasi, među kojima treba istaći Intel IA-64, DEC Alpha, SUN SPARC/UltraSPARC, Motorola 68000, MIPS, PowerPC i IBM mainframe S/390.

15.2. a. U kom obliku je raspoloživ Linux?

b. Da li je izvorni kod operativnog sistema javno dostupan ili ne?

c. Da li se Linux može slobodno distribuirati?

(a) Linux je raspoloživ kao besplatan operativni sistem pod GNU GPL licencom (GNU *General Public License*).

(b) Linux je softver sa otvorenim izvornim kodom (*Open Source*), što znači da mu je izvorni kod javno raspoloživ i može biti modifikovan tako da odgovara specifičnim potrebama.

(c) Linux se može slobodno distribuirati među korisnicima.

15.3. a. Šta ulazi u sastav jedne Linux distribucije?

b. Navedite neke od značajnijih distribucija Linux sistema.

(a) Kernel, sistemski softvera, instalacioni alati, softver za podizanje operatvnog sistema, razne korisničke aplikacije (kancelarijski paketi, softver za obradu bit-mapiranih slika, za prikazivanje multimedijalnog sadržaja), programski jezici i serverski paketi.

(b) SuSE, Slackware, Red Hat, Fedora, Debian, Mandrake.

15.4. Kakav je Linux operativni sistem, posmatrano sa tačke broja korisnika koji istovremeno mogu koristiti sistem i broja procesa koji se mogu istovremeno izvršavati?

Višekorisnički višeprocesni.

15.5. Kako se sistemski pozivi upućuju kernelu na Linux operativnom sistemu?

Preko sistemskih biblioteka, koje definišu standardni set funkcija preko kojih aplikacije komuniciraju sa kernelom.

15.6. a. Šta su moduli jezgra?

b. O kojoj je arhitekturi reč ukoliko Linux sistem koristi modularni kernel?

(a) Moduli kernela su delovi kernelskog koda koji može da se prevede, napuni u memoriju ili izbaci iz memorije nezavisno od ostatatka kernela. Kernelski moduli implementiraju drajvere za hardverske uređaje, novi sistem datoteka, mrežne protokole, itd.

(b) Mikro-kernel arhitektura.

15.7. Koje su osnovne komponente kernela Linux sistema?

- upravljanje procesima,
- upravljanje memorijom,
- upravljanje sistemima datoteka (VFS),
- apstrakcija mrežnih servisa,
- podrška za hardverske uređaje,
- podrška za različite sisteme datoteka,
- podrška za TCP/IP.

15.8. Pod Linux operativnim sistemom, svaki proces je u potpunosti opisan identitetom, okolinom, i kontekstom. Šta se podrazumeva pod (a) identitetom, (b) okolinom i (c) kontekstom?

(a) Identitet procesa čine identifikator (PID), akreditivi (*credentials*) koji određuju prava pristupa procesu u radu sa datotekama, i ličnost (*personality*), odnosno lični identifikator koji može imati uticaja za neke sistemske pozive.

(b) Okolina procesa se nasleđuje od procesa roditelja. U okolinu procesa spadaju vektor argumenata koje proces roditelj prosleđuje programu i vektor okoline, odnosno lista promenljivih koje definišu okolinu procesa.

(c) Kontekst procesa je stanje procesa u datom trenutku vremena.

15.9. Koje algoritme koristi Linux za dodelu procesora procesima?

- time-sharing algoritam - dodela se vrši na osnovu prioriteta procesa koji definiše korisnik i kredita (efektivni prioritet) koji raste s porastom vremena čekanja na procesor (*aging*),
- real-time algoritam - apsolutni prioriteti procesa su mnogo značajniji od ravnomerne raspodele. Linux je ipak *soft real-time* operativni sistem.

15.10. Koje metode interprocesne komunikacije koristi Linux?

- slanje signala,
- deljivu memoriju,
- *pipe* mehanizam.

15.11. a. Po kom algoritmu se bira žrtva (victim), odnosno stranica koja mora da napusti memoriju i ode na disk?

b. Koje regione obuhvata virtuelna memorija Linux sistema?

(a) LFU - *Least Frequently Used.*

(b) Fizičke stranice (okvire) i *swap* prostor na disku.

15.12. a. Linux deli uređaje u tri klase. Koje su to klase uređaja?

b. Čime su uređaji predstavljeni na Linuix sistemu?

(a) Blok uređaji (diskovi, CD-ROM uređaji), karakter uređaji (štampači) i mrežni uređaji.

(b) Svaki uređaj je predstavljen specijalnom datotekom (device node, device file) koja se nalazi u direktorijumu /dev root sistema datoteka.

15.13. Šta je definisano FHS standardom?

Filesystem Hierarchy Standard (FHS) definiše organizaciju aktivnog UNIX stabla i podelu stabla na nekoliko sistema datoteka specifične namene koje treba kreirati na odvojenim particijama ili diskovima.

15.14. Šta je *syslog*?

Kernel i mnogi sistemski programi generišu razna upozorenja i poruke o greškama, koje se upisuju u datoteke, tako da se mogu pregledati posle

izvesnog vremena. Program koji obavlja funkciju upisivanja poruka u datoteke je *syslog*.

16. Mac OS X

16.1. Šta je Darwin?

Delovi softvera koji čini Mac OS X mogu se grupisati u nekoliko logičkih celina, koje se oslanjaju na osnovu koju čini Darwin. Darwin se može posmatrati kao operativni sistem u celini ili kao skup tehnologija koje čine temelj Mac OS X.

16.2. Koji su osnovni delovi Mac OS X grafičkog podsistema?

- Quartz - skup 2D grafičkih tehnologija koje čine osnovu za rad sa rasterskom i vektorskom grafikom,
- Quartz Extreme - korišćenjem intergrisane OpenGL tehnologije operacije formiranja konačne slike na ekranu preusmeravaju se na GPU video adaptera,
- OpenGL - standard za formiranje slike na osnovu 3D objekata i tekstura,
- QuickTime - obezbeđuje podršku za rad sa multimedijalnim sadržajem.

16.3. Šta su Aliasi na HFS+ sistemu datoteka?

Aliasi su slični simboličkim linkovima u smislu da dozvoljavaju kreiranje višestrukih referenci na datoteku ili direktorijum. Međutim, ukoliko se referencirani objekat pomeri, za razliku od simboličkog linka alias ne gubi vezu sa objektom. Ukoliko je putanja neispravna, alias traži datoteku na osnovu identiteta i ažurira putanju ka datoteci. Veza se gubi u slučaju da se referencirani objekat zameni novim objektom. Korišćenje aliasa je moguće samo kroz Carbon ili Cocoa API-je.

16.4. Pod kojim uslovima HFS+ sistem datoteka obavlja *on-the-fly* defragmentaciju pri otvaranju datoteka?

- datoteka je fragmentisana,
- veličina datoteke je manja od 20MB,

- datoteka nije već otvorena,
- datoteka se može se otvoriti za čitanje i upis,
- sistem je funkcionalan duže od tri minuta.

16.5. Šta omogućava *rendezvous* protokol (*zero configuration networking*)?

Omogućava pronalaženje mrežnih uređaja na mreži, bez eksplicitnog navođenja IP adresa ili konfigurisanja DNS servera.

16.6. Da li je u podrazumevanom stanju dozvoljeno prijavljivanje *superusera* na Mac OS X operativni sistem?

U podrazumevanom stanju zabranjeno je prijavljivanje root korisnika na sistem - *root* privilegije ostvaruju se korišćenjem programa sudo.

16.7. Koje su osnovne komponente Mac OS X kernela i koje funkcije obavljaju ove komponente?

- Mach - XNU Mach komponenta zasnovana je na Mach 3.0, koja je, u ovom slučaju, implementirana kao deo kernela. Mach komponenta kernela odgovorna je za: multiprogramiranje sa pretpražnjenjem, međuprocesnu komunikaciju, zaštićenu memoriju, upravljanje virtuelnom memorijom i upravljanje prekidima.
- BSD - XNU BSD komponenta zasnovana je na FreeBSD kodu. BSD se, takođe, izvršava kao deo kernela i odgovoran je za model procesa, identifikatore korisnika, dozvole i osnovne sigurnosne polise, POSIX API, BSD sistemske pozive, TCP/IP skup protokola, BSD priključke (sockets), mrežnu barijeru, VFS i sisteme datoteka, System V IPC, kriptografski framework i sinhronizacione mehanizme,
- I/O kit - objektno orijentisani drajverski framework,
- Platform Expert - identifikuje platformu na kojoj se sistem izvršava i formira stablo priključenih uređaja.

16.8. Koje komponente pripadaju sloju osnovnih servisa?

- framework koji uključuje sistemske API-je,
- API za umrežavanje na korisničkom nivou,
- kritični delovi Carbon aplikativnog okruženja,

- API-ji za korišćenje Web servisa preko SOAP i XML-RPC.

16.9. U kojim se okruženjima izvršavaju aplikacije na Mac OS X operativnom sistemu?

- BSD
- *Carbon* - skup proceduralnih API-ja zasnovanih na Mac OS 9 API-jima,
- *Classic* - emulator kojim se ostvaruje kompatibilnost sa Mac OS 9 sistemom,
- *Cocoa* - objektno orijentisani API za razvoj aplikacija u objektnom C jeziku i Javi; Cocoa je nasleđe NEXTSTEP sistema,
- *Java* - JDK, izvršni deo (Hotspot VM, JIT) i kolekcija Java klasa (AWT, Swing, ...).

A. Administracija Linux sistema

Blok uređaji i administracija sistema datoteka

A.1. Koliko se uređaja može vezati na IDE, a koliko na SCSI kontroler?

Na IDE disk mogu se vezati četiri uređaja (*Primary Master*, *Primary Slave*, *Secondary Master*, *Secondary Slave*). Na SCSI disk se mogu vezati 15 uređaja.

A.2. U čemu je osnovna razlika u ponašanju između uređaja vezanih na isti kanal IDE kontrolera (na primer, *Primary Master* i *Primary Slave*) i uređaja vezanih na različite kanale (na primer, *Primary Master* i *Secondary Master*)?

Uredaji vezani na različite kanale mogu da primaju ili šalju podatke računaru istovremeno. Na jednom kanalu može biti samo jedan uredaj aktivan u jednom trenutku vremena.

A.3. a. Kako se određuju prioriteti uređaja vezanih na SCSI kontroler?

b. Koji prioritet treba dodeliti sistemskom disku?

(a) Prioritet svakog uređaja određen je njegovim identifikacionim brojem koji se postavlja preko preklopnika na uređaju.

(b) Prioritet uređaja sa ID=0 je najviši i treba ga dodeliti sistemskom disku.

A.4. Objasnite čime su prouzrokovani limiti od 512MB i 8.4GB.

PC i ATA arhitekturom.

A.5. Kojom specijalnom datotekom je prestavljen (a) IDE *Primary Slave*, (b) četvrti SCSI disk?

(a) /dev/hdb,

(b) /dev/sdd.

A.6. Na jednom disku treba kreirati šest sistema datoteka. Koliko se najmanje logičkih particija mora kreirati?

Na disku se mogu kreirati tri primarne particije i jedna extended, a u okviru nje tri logičke. Na taj način može se kreirati šest sistema datoteka. To znači da su potrebne najmanje tri logičke particije.

A.7. Navedite nekoliko značajnijih programa kojima se mogu kreirati particije na disku.

Značajniji Linux programi za particionisanje diskova su fdisk, cfdisk, sfdisk i parted.

A.8. Koju funkciju obavlja program fdformat?

Obavlja formatiranje niskog nivoa, odnosno pripremu magnetne površine disketa.

A.9. Ako se razmatraju magnetne trake, šta predstavljaju *rewind* a šta *non-rewind* nodovi?

Rewind nodovi: kada se *backup* snimi na uređaj preko *rewind* noda, po završetku snimanja traka se premotava na početak. Nakon toga, ukoliko se novi *backup* snima na isti medijum, prethodni biva uništen (prepisan).

Non-rewind nodovi: kada se *backup* snimi na uređaj preko *non-rewind* noda, po završetku snimanja traka se ne premotava na početak. Na naj način se omogućava da na jednom medijumu (traci) bude smešteno nekoliko *backup*-a.

A.10. Kojim programom se proverava ispravnost površine magnetnog medijuma?

Ispravnost površine magnetnog medijuma proverava se programom badblocks.

A.11. Čemu služi program mkfs?

Program mkfs služi za kreiranje sistema datoteka.

A.12. Navedite komandu za montiranje prve primarne particije četvrtog SCSI diska na direktorijum /data.

```
mount /dev/sdd1 /data
```

A.13. Šta se dešava sa trenutnim sadržajem direktorijuma koji nije prazan ukoliko se isti iskoristi kao *mount-point* za aktiviranje novog sistema datoteka?

Sadržaj direktorijuma je privremeno nevidljiv. Kada se sistem datoteka deaktivira, podaci sa mount-point direktorijuma postaju ponovo vidljivi.

A.14. Koji sistem datoteka se najpre aktivira prilikom podizanja UNIX operativnog sistema ? Da li se taj sistem datoteka može deaktivirati?

Root sistem datoteka se prvi aktivira prilikom podizanja UNIX operativnog sistema i ne može se naknadno deaktivirati.

A.15. Šta je opisano datotekama /etc/fstab i /etc/mtab?

Sistemi datoteka koji se aktiviraju automatski prilikom podizanja operativnog sistema (*auto-mount*) opisani su u datoteci /etc/fstab (*filesystem table*). Ova datoteka sadrži statičke informacije o sistemima datoteka. Datoteke /etc/mtab (*mount table*) sadrži informacije o sistemima datoteka koji su aktivirani, odnosno montirani na aktivno UNIX stablo.

A.16. Kako se svim korisnicima sistema može omogućiti da aktiviraju konkretan sistem datoteka (na primer, /dev/hda2)? Mogućnost davanja lozinke *superuser*-a je isključena.

Davanjem dozvola svim korisnicima za montiranje sistema datoteka u datoteci /etc/fstab. Korisnici će moći da aktiviraju samo one sisteme datoteka za koje je u polju fs_mntops navedena opcija user.

A.17. Navedite dve varijante komande za deaktiviranje sistema datoteka koji se nalazi u prvoj logičkoj particiji Primary Master diska, a montiran je na direktorijum /data.

```
umount /dev/hda5
umount /dev/data
```

A.18. a. Zašto pri radu sa programom fsck sistem datoteka čiji se integritet proverava mora biti deaktiviran?

b. Koja je funkcija lost+found direktorijuma?

(a) Program fsck zaobilazi standardne rutine za pristup sistemima datoteka i tretira disk kao uređaj, tako da sistem datoteka, čiji se integritet provera, ne sme biti aktiviran. Jedino root sistem datoteka može biti aktiviran u režimu čitanja prilikom provere integriteta (ovo se obavlja prilikom podizanja sistema).

(b) Prilikom kreiranja novog sistema datoteka, mkfs kreira direktorijum lost+found. U ovaj direktorijum fsck smešta pokazivače ka *i-node* strukturama koje nisu evidentirane ni u jednom direktorijumu, ili na njih ukazuju oštećene *dir-info* strukture. Program fsck ne može da sazna pravo ime oštećenih objekata, tako da im dodeljuje imena na osnovu broja indeksnog čvora, u obliku #inode-number. Na ovaj način se povećava stepen pouzdanosti sistema – datoteka, čiji *i-node* nije evidentiran ni u jednom direktorijumu se ne briše, već se notira u lost+found direktorijumu, tako da se kasnije može upotrebiti ukoliko ima korisnog sadržaja.

A.19. a. U čemu je smisao tehnike vođenja dnevnika transakcija?

b. Da li se integritet journaling sistema datoteka proverava nakon nasilnog obaranja sistema?

(a) Gubitak integriteta sistema datoteka najčešće se javlja kao posledica nasilnog zaustavljanja sistema, odnosno promena u objektima sistema datoteka koje nisu blagovremeno ažurirane u tabeli indeksnih čvorova, i može za posledicu imati gubitak podataka. Opasnost od gubitka podataka umanjuje se uvođenjem dnevnika transakcija koji prati aktivnosti vezane za promenu meta-data oblasti, odnosno i-node tabele, i objekata sistema datoteka. Dnevnik (*journal*, *log*) se ažurira pre promene sadržaja objekata i prati relativne promene u sistemu datoteka u odnosu na poslednje stabilno stanje. Transakcija se zatvara po obavljenom upisu i može biti ili u potpunosti prihvaćena ili odbijena. U slučaju oštećenja, izazvanog npr. nepravilnim gašenjem računara, sistem datoteka se može lako rekonstruisati povratkom na stanje poslednje prihvaćene transakcije.

(b) Integritet journaling sistema datoteka se ne proverava nakon nasilnog obaranja sistema.

A.20. Čemu služe programi df i du?

Program df (*disk free*) prikazuje količinu slobodnog prostora na jednom ili više sistema datoteka.

Program du (*disk used*) prikazuje količinu prostora na sistemu datoteka koju zauzimaju direktorijumi, počev od tekućeg direktorijuma ili od direktorijuma koji je naveden kao parametar.

A.21. Navedite komandu kojom se prva dva bloka diska (*master boot record*) mogu zapisati na disketu (montirana na direktorijum /mnt/floppy) kao datoteka oldboot.rec.

```
dd if=/dev/hda of=/mnt/floppy/oldboot.rec bs=512 count=2
```

A.22. Navedite komande za kreiranje i aktiviranje *swap* prostora u formi datoteke /swap_file veličine 250MB.

```
dd if=null of=/swap_file count=250M
mkswap /swap_file
swapon /swap_file
```

A.23. Navedite komande za kreiranje i aktiviranje *swap* prostora u formi particije /dev/hda3. Pretpostaviti da je particija veličine 250MB već kreirana programom fdisk.

```
mkswap /dev/hda3
```

```
swapon /dev/hda3
```

A.24. Koji način realizacije *swap* prostora daje bolje performanse i zašto?

Swap prostor u formi posebne particije funkcioniše brže nego *swap* datoteka, jer se zaobilaze rutine za pristup objektima sistema datoteka.

A.25. Objasnite sekvencu aktiviranja sistema datoteka i formiranja aktivnog UNIX stabla prilikom podizanja sistema.

Najpre se na root direktorijum aktivnog UNIX stabla aktivira root sistem datoteka u režimu čitanja (*read-only*) radi provere integriteta. Root sistem datoteka se, zatim, reaktivira u režimu čitanja i pisanja. Posle toga se redom aktiviraju svi *auto-mount* sistemi datoteka opisani u /etc/fstab.

A.26. Šta se nalazi na direktorijumu /dev?

Specijalne datoteke koje predstavljaju uređaje (*device nodes*).

A.27. Šta se nalazi na direktorijumima /etc i /var?

/etc – većina konfiguracionih datoteka.

/var - datoteke koje se menjaju prilikom regularnog funkcionisanja sistema, poput *spool* direktorijuma i log datoteka.

A.28. Šta se nalazi na direktorijumima /bin, /sbin, /usr/bin i /usr/sbin?

/bin - najčešće korišćene komande koje mogu koristiti regularni korisnici. Nalazi se u sistemskoj putanji svih korisnika.

/sbin - komande namenjene superuseru, ali ih po potrebi mogu koristiti i obični korisnici ukoliko im se za to daju dozvole. /sbin se ne nalazi u putanji regularnih korisnika, ali se nalazi u putanji superusera.

/usr/bin - većina korisničkih programa (user binaries). Nalazi se u sistemskoj putanji svih korisnika.

/usr/sbin - komande za administraciju raznih serverskih programa i ostale komande koje nisu neophodne za rad u jednokorisničkom režimu rada (superuser binaries). /usr/sbin se ne nalazi u putanji regularnih korisnika, ali se nalazi u putanji superusera.

A.29. Šta je /proc sistem datoteka?

/proc sistem datoteka omogućava lak pristup strukturama podataka kernela, na osnovu čega se mogu dobiti informacije o sistemu (npr. o procesima, odakle potiče i naziv).

/proc nije sistem datoteka u pravom smislu te reči, već simbolička predstava ovih struktura jednim kvazi-sistemom datoteka. Nijedna datoteka sa direktorijuma proc ne zauzima mesto na disku - sve ove datoteke kreira kernel u operativnoj memoriji računara.

Većina datoteka proc sistema datoteka data je u formatu koji je prilagođen korisniku, odnosno u formatu koji je jednostavan za čitanje i razumevanje. Sadržaj ovih datoteka može se pregledati pomoću standardnih alata za rad sa datotekama.

A.30. Kako se na realnom sistemu datoteka može snimiti slika operativne memorije (*memory dump*)?

Kopiranjem datoteke /proc/kcore (na primer cp /proc/kcore /tmp/mem_image).

Korisnici i grupe

A.31. Objasnite razliku između sistemskih i regularnih korisnika. Koji sistemski korisnik može da se prijavi na sistem?

Sistemski korisnici, koji nastaju prilikom instalacije operativnog sistema, služe za specijalne namene, a ne za prijavljivanje na sistem. Jedini sistemski korisnik koji se može prijaviti na sistem jeste *superuser root.* Root ima sve privilegije i služi isključivo za administraciju sistema.

Regularne korisnike kreira superuser. Regularni korisnici služe za prijavljivanje na sistem.

A.32 U kojoj su datoteci opisani svi korisnici sistema? Navedite polja koja opisuju jednog korisnika.

Svi korisnici sistema opisani su u datoteci /etc/passwd. Ova datoteka je realizovana u vidu tabele u kojoj je svaki korisnik opisan jednom linijom (zapisom) kog čine sledeća polja:

- korisničko ime (username),
- šifrovana lozinka,

- UID (numerički identifikator korisnika),
- GID (numerički identifikator primarne grupe),
- opisni podaci (puno ime, brojevi telefona itd...),
- putanja ličnog direktorijuma (*home*),
- komandni interpreter (*shell*).

A.33. Koji korisnik može kreirati druge korisnike?

Root.

A.34. a. Šta je to primarna grupa korisnika i gde se definiše?

b. Gde se definišu ostale grupe kojima korisnik pripada?

(a) Korisnik mora pripadati najmanje jednoj grupi koja je navedena u datoteci /etc/passwd. Ta grupa se naziva primarna grupa korisnika i njoj se formalno dodeljuju datoteke koje taj korisnik iskopira ili kreira. Primarna grupa korisnika može biti sistemska ili regularna.

(b) Ostale grupe kojima korisnik pripada koriste se za regulisanje kontrole pristupa (pripadnost korisnika ostalim grupama reguliše se pomoću datoteke /etc/group).

A.35. Šta je datoteka /etc/shadow i pomoću kog programa se može kreirati tako da bude funkcionalna?

U datoteci /etc/shadow, kojoj može da pristupi samo root, nalaze se šifrovane lozinke svih korisnika sistema i informacije o vremenskim ograničenjima. Datoteka /etc/shadow se može kreirati pomoću komande pwconv, koja na osnovu postojeće /etc/passwd i eventualno /etc/shadow kreira novu shadow datoteku.

A.36. Odakle se kopiraju datoteke koje predstavljaju inicijalno okruženje novokreiranog korisnika?

Datoteke koje čine inicijalno okruženje novokreiranog korisnika kopiraju se iz direktorijuma /etc/skel.

A.37. Koja datoteka predstavlja globalnu konfiguracionu datoteku za sve korisnike?

/etc/profile

A.38. Zašto se datoteke /etc/passwd i /etc/group modifikuju komandama vipw i vigr, a ne, na primer, običnim editorima teksta, kao što su joe ili jed ?

vipw i vigr zatvaraju datoteke /etc/passwd i /etc/group za upis, tako da druge komande ne mogu istovremeno da promene njen sadržaj.

A.39. a. U čemu je razlika između RUID i EUID?

b. Šta prikazuje komanda id, a šta komanda whoami?

(a) RUID je ID korisnika koji je inicijalno prijavljen na sistem i ne menja se tokom rada, bez obzira na to da li je korisnik pokrenuo komandu su i pomoću nje se prijavio pod drugim imenom,

EUID je jednak RUID ukoliko korisnik nije pokrenuo komandu su, odnosno UID korisnika pod čijim imenom je privremeno prijavljen, ukoliko je izvršena zamena identiteta komandom su.

(b) Komande who am i i id prikazuju RUID i EUID korisnika respektivno.

A.40. Na koji način se regularni korisnik može privremeno prijaviti na sistem kao root (pod predpostavkom da je već prijavljen na sistem kao regularan korisnik i da zna lozinku *superuser*-a)?

Pomoću komande su.

Vlasnički odnosi i prava pristupa

A.41. Definišite vlasničke kategorije (a) vlasnika, (b) grupe i (c) ostatka sveta.

(a) Vlasnik (*owner*) najčešće je korisnik koji je kreirao objekat, osim ukoliko superuser ne promeni vlasništvo. U tom slučaju, vlasnik je korisnik kome je vlasništvo dodeljeno.

(b) Grupa (*group*) je korisnička grupa kojoj je datoteka formalno priključena. Objekti sistema datoteka mogu pripadati samo jednoj grupi, a to je najčešće primarna grupa korisnika koji je objekat kreirao. Superuser naknadno može promeniti pripadnost objekta grupi.

(c) Ostali (*others*, *public*) su svi oni korisnici koji nisu ni vlasnik objekta, niti pripadaju grupi kojoj objekat pripada.

A.42. Objasnite problem unije vlasničkih kategorija.

U slučajevima unije vlasničkih kategorija (vlasnik datoteke pripada grupi kojoj pripada datoteka), vlasništvo se određuje po pravilima konkretnog UNIX sistema. Kod starijih UNIX sistema, pristupna prava vlasnika koji pripada grupi kojoj pripada objekat određena su unijom vlasničkog i grupnog prava. Kod UNIX sistema koji prate POSIX standard (POSIX *compliant*), kao što je Linux, pristupna prava vlasnika koji pripada grupi kojoj pripada i datoteka određena su isključivo vlasničkim pravom.

A.43. Šta znači pravo upisa u odnosu na datoteku?

Pravo upisa omogućava korisniku da modifikuje sadržaj datoteke nad kojom je to pravo dato.

A.44. Koja su prava neophodna korisniku da bi mogao da obriše datoteku i nad kojim objektom ta prava moraju biti definisana?

Korisnik može obrisati datoteku ukoliko mu je dato pravo upisa nad direktorijumom u kome se ta datoteka nalazi.

A.45. Šta znači pravo izvršavanja u odnosu na direktorijum?

Pravo izvršavanja nad direktorijumom omogućava korisnicima da se na taj direktorijum pozicioniraju i da mogu da pročitaju kontekst objekata koji se u tom direktorijumu nalaze.

A.46. Ko može promeniti prava pristupa datoteke ili direktorijuma?

Vlasnik datoteke ili root.

A.47. Ko može promeniti vlasništvo datoteke ili direktorijuma?

root.

A.48. Gde se čuvaju informacije o vlasniku, grupi i pristupnim pravima objekata sistema datoteka?

Informacije o vlasniku, grupi i pristupnim pravima objekata sistema datoteka čuvaju se u indeksnom čvoru (*i-node*) tog objekta.

A.49. Koja je funkcija (a) SUID i (b) SGID flegova?

(a) Kada korisnik pokrene program kome je postavljen setuid bit, realni vlasnički odnosi se menjaju efektivnim. Korisnik koji izvršava program u procesu koji nastaje dobija identitet vlasnika izvršnog programa.

(b) Grupni bit (setgid) postavlja se i funkcioniše na sličan način - realna grupa korisnika se u procesu zamenjuje efektivnom grupom, odnosno grupom kojoj izvršni program sa postavljenim setgid bitom pripada.

A.50. Objasniti značenje specijalnog atributa i (*immutable*).

Specijalni atribut i (*immutable*) znači da datoteka ne može biti modifikovana ili obrisana, ne može joj se promeniti ime niti se može kreirati link koji ukazuje na tu datoteku.

Rad sa datotekama iz komandne linije

A.51. Navedite osnovne funkcije komandnog interpretera.

Komandni interpreter interpretira komandnu liniju i po potrebi pokreće programe. Dodatno, *shell* omogućava redirekciju ulaza i izlaza, povezivanje komandi u pipeline, zamenu imena datoteka, rukovanje promenljivim i kontrolu okoline. Kao posebna funkcija komandnog interpretera može se izdvojiti shell programiranje.

A.52. Šta je (a) redirekcija, a šta (b) pipe?

(a) UNIX komande primaju podatke sa standardnog ulaza (stdout), rezultate izvršenja šalju na standardni izlaz (stderr), a poruke o greškama na standardni uređaj za greške (stderr). Većina UNIX komandi koristi tastaturu kao standardni ulaz, a monitor kao standardni izlaz i uređaj za greške.

UNIX omogućava da se ulaz i izlaz komande preusmere. To znači da komanda može čitati ulazne podatke iz npr. datoteke (redirekcija ulaze), odnosno da se izlaz komande može sačuvati kao datoteka ili odštampati radi kasnijeg čitanja (redirekcija izlaza).

(b) Pipeline funkcioniše na sledeći način: standardni izlaz komande sa leve strane znaka pipe -cev (|) postaje standardni ulaz komande sa desne strane znaka. Znak pipe zahteva komandei sa leve i sa desne strane, a razmaci između znaka i komande su proizvoljni

A.53. Čemu služi <Ctrl-D> kontrolni karakter u Bourne-again shellu?

<Ctrl-d> označava kraj datoteke. Koristi se za napuštanje programa koji podatke čitaju sa standardnog ulaza (tastatura).

A.54 Šta je alias i kako se definiše?

Alias je način dodele kraćeg imena pomoću kog se određena komanda, ili niz komandi, može pozvati iz komandnog interpretera.

Za korn i bash alias se dodeljuje na sledeći način:

```
alias aliasname=value
```

Na primer:

```
alias h=history
alias home="cd;ls"
alias ls="ls -l"
```

A.55. Kako se prilagođava odziv komandnog interpretera?

Svaki korisnik može promeniti svoj shell prompt pomoću lokalne promenljive PS1 (prompt string). Ove promene važe samo za tekući shell. Kada se korisnik ponovo prijavi na sistem, dobiće podrazumevani prompt.

A.56. Koje su korisničke inicijalizacione datoteke i gde se nalaze?

Postoje dve vrste shell inicijalizacionih datoteka:

Globalne inicijalizacione datoteke, nalaze se u /etc direktorijumu (na primer /etc/profile). Ove datoteke postavljaju globalno okruženje za sve korisnike sistema i može ih modifikovati samo superuser.

Korisničke inicijalizacione datoteke nalaze se u home direktorijumu korisnika(~/.bash_profile, ~/.bashrc) i postavljaju okruženje specifično za datog korisnika. Ove datoteke osim superusera može modifikovati i sam korisnik.

A.57. Navedite tri načina za dobijanje pomoći.

```
man command,
info command,
command –help.
```

Command je ime komande (programa) za koju tražite pomoć. Alternativno, www.tldp.org ili neki drugi *web* sajt mogu da posluže kao dobar izvor informacija.

A.58. Kako se može najjednostavnije pronaći lokacija neke komande, odnosno programa, u aktivnom UNIX stablu?

Komandama which i whereis.

A.59. a. Da li UNIX prepoznaje datoteke na osnovu ekstenzija?

b. Kako se može odrediti tip datoteke?

(a) UNIX generalno ne prepoznaje datoteke na osnovu ekstenzija. Na primer, tar arhiveru je svejedno da li korisnik zadaje komandu: tar xvfz arhiva.divx ili tar xvfz arhiva.tar.gz sve dok je argument tar arhiva komprimovana gnu zip programom za kompresiju.

(b) Tip datoteke se može odrediti pomoću komande file.

A.60. Čemu služi test magičnih brojeva?

Testom magičnih brojeva određuje se programski paket kojim je datoteka kreirana. Svaki program prilikom kreiranja datoteka upisuje neke kontrolne informacije (*overhead*) koje ovaj test posmatra kao identifikator tipa datoteke, odnosno magični broj. Bilo koja datoteka sa nepromenjenim identifikatorom, koji se nalazi na fiksnom ofsetu od početka datoteke, može biti opisana na ovaj način. Magični brojevi se smeštaju u datoteke /usr/share/magic.mgc i /usr/share/magic i prilikom izvršavanja ovog testa redom traže na fiksnom ofsetu u datoteci. Ukoliko se određeni magični broj poklopi sa magičnim brojem datoteke, program file prekida dalje izvršenje i na ekranu ispisuje informaciju o programu kojim je datoteka kreirana.

A.61. Navesti komandu za kopiranje kompletnog sadržaja direktorijuma /home u direktorijum /backup/home (tako da se datoteka /home/jsmith/1.txt kopira u /backup/home/jsmith/1.txt).

```
cp -r /home /backup
```

A.62. Šta UNIX radi kada korisnik zada sledeću komandu `cp /tmp/a* /home/jsmith/b*` ?

UNIX prijavljuje grešku, jer se ne može kopirati grupa datoteka uz istovremenu promenu imena kopijama. Na primer:

```
cp e* /tmp/a*
```

```
cp: copying multiple files, but last argument `/tmp/a*' is
not a directory
```

A.63. Navedite komandu za pomeranje grupe datoteka čije ime počinje dvocifrenim brojem sa direktorijuma /tmp na direktorijum /backup/tmp.

```
mv /tmp/[0-9][0-9]* /backup/tmp
```

A.64. Koji su uslovi potrebni za kopiranje, a koji za pomeranje datoteke dat1 sa direktorijuma dir1 na direktorijum dir2?

Za kopiranje: r (dat1), w (dir2).

Za pomeranje: r (dat1), w (dir1), w (dir2).

Ovo su minimalni uslovi. Potpuni skup uslova se u oba slučaja dobija se proširivanjem skupa pravima x (dir1) i x (dir2).

A.65. a. Objasnite značenje promenljive umask.

b. Koja prava pristupa ima kopija datoteke, ako su pristupna prava izvorišne datoteke 666, a promenljiva umask ima vrednost 027?

(a) Inicijalna prava pristupa dodeljuju se na osnovu vrednosti promenljive umask, koja je specifična za svakog korisnika.Vrednost promenljive umask postavlja se prilikom prijavljivanja na sistem, a naknadno se može promeniti.

Inicijalna prava datoteka i direktorijuma određuju se različitim formulama. Inicijalnaprava direktorijuma predstavlja vrednost 777 od koje je oduzeta vrednost promenljive umask: perm(dir)=777-umask. To znači da su za vrednost promenljive umask=027 inicijalna prava za direktorijum 750.

Datotekama se inicijalno ne dodeljuju prava izvršavanja ni u jednoj vlasničkoj kategoriji, tako da se podrazumevana prava računaju po nešto složenijoj formuli. Vrednost 666 se logički množi sa komplementom promenljive umask: perm(file)= 666&(^umask), čime se formiraju prava za datoteku. To znači da su za vrednost promenljive umask=027 inicijalna prava za datoteku 640.

(b) 664.

A.66. Šta je (a) hard, a šta (b) simbolički link?

(a) Hard link je alternativno ime datoteke, odnosno jedna file-info struktura koja ukazuje na i-node datoteke.

(b) Simbolički link je objekat sistema datoteka u pravom smislu te reči: simbolički link ima svoj i-node, u kome se nalazi pokazivač na blok podataka u kome je smešten pokazivač na originalnu datoteku.

A.67. a. Da li korisnik može obrisati datoteku ako postoji hard link na nju?

b. Da li korisnik može obrisati datoteku ako postoji simbolički link na nju?

(a) Ne. Datoteka se može obrisati samo ako nema hard linkova, tj. ako se obriše neki hard link, onda se broj hard linkova umanjuje za 1. Kada je broj hard linkova 1, to znači da datoteka ima samo jedno ime, tj. da na njen i-node ukazuje samo jedna file-info struktura i tada se može obrisati.

(b) Da. Broj simboličkih linkova nije od uticaja na brisanje datoteke.

A.68. Navedite komandu za brisanje direktorijuma /oldfiles koji nije prazan.

```
rm -r -f /oldfiles
```

A.69. Navedite komandu kojom se traže sve datoteke koje pripadaju korisniku jsmith veličine najmanje 512 bajtova na celom aktivnom stablu.

```
find / -user jsmith – size +512c -print
```

A.70. U kom obliku treba zadati komandu ls da bi se na ekranu prikazale i skrivene datoteke?

```
ls -a
```

A.71. U čemu je razlika između komandi cat i less?

Komanda cat prikazuje sadržaj datoteke filename na ekranu, bez zaustavljanja nakon svakog punog ekrana.

Komanda less se koristi za prikazivanje sadržaja datoteke ekran po ekran i dozvoljava dvosmernu navigaciju po datoteci (unapred i unazad).

A.72. Navedite komande za prikazivanje prve tri i poslednje tri linije datoteke /etc/passwd.

```
head -3 /etc/passwd
tail -3 /etc/passwd
```

A.73. Navedite komandu za brojanje reči i karaktera u datoteci /tmp/tekst1.

```
wc -wc /tmp/tekst1
```

A.74. Navedite komandu koja pronalazi i na ekranu prikazuje sve razlike između datoteka /tmp/dat1 i /tmp/dat2, a pri tome ne uzima u obzir razlike između malih i velikih slova i suvišne blanko karaktere.

```
diff -iw /tmp/dat1/tmp/dat2
```

A.75. Navedite komandu koja uređuje datoteku /tmp/spisak1.txt po mesecima u drugoj koloni u opadajućem poretku, a rezultat upisuje u datoteku /tmp/spisak2.txt.

```
sort -M +1 /tmp/spisak1.txt > /tmp/spisak2.txt
```

A.76. Navedite komandu kojom se u datoteku /tmp/spisak3.txt izdvajaju sve linije datoteke /tmp/spisak1.txt koje počinju nizom karaktera the ili The.

```
grep '^[Tt]he' /tmp/spisak1.txt > /tmp/spisak3.txt
```

A.77. Navedite komandu kojom se u datoteku /tmp/spisak4.txt izdvajaju sve linije datoteke /tmp/spisak1.txt koje u sebi sadrže najmanje jedan od sledećih nizova karaktera: run1, run2, fun1, fun2.

```
grep '[rf]un[12] ' /tmp/spisak1.txt > /tmp/spisak4.txt
```

A.78. Navedite komandu kojom se u datoteku /tmp/spisak4.txt izdvajaju sve linije datoteke /tmp/spisak1.txt koje sadrže jedno ili više uzastopnih pojavljivanja slova p.

```
grep 'pp*' /tmp/spisak1.txt > /tmp/spisak4.txt
```

A.79. Navedite sekvencu karaktera kojom se napušta vi editor, ako se korisnik koji u njemu radi trenutno nalazi u režimu za unos teksta.

```
<Esc> <:> <q> <!>
```

A.80. Navedite sekvencu karaktera kojom se u vi editoru traži niz karaktera jsmith počev od tekuće pozicije do kraja datoteke.

```
/jsmith
```

Shell programiranje

A.81. Navedite dve komande kojima se može pokrenuti shell program myscript iz tekućeg direktorijuma.

`./myscript` (pod uslovom da je korisniku koji pokreće komandu dato pravo x nad datotekom myscript)

`bash ./myscript` (bez obzira da li je korisniku koji pokreće komandu dato pravo x nad datotekom myscript ili ne)

A.82. Čemu služi linija #!/bin/bash na početku shell programa?

Specificira komandni interpreter u kome će se program izvršavati.

A.83. Kako se na ekranu može prikazati vrednost promenljive PS1?

```
echo $PS1
```

A.84. Navedite komandu kojom se podatak unet sa tastature upisuje u promenljivu prom1.

```
read prom1
```

A.85. a. Kako se prenose argumenti sa komandne linije u shell program?

b. Čemu služi promenljiva $# ?

(a) Za shell program ss3, koji je pozvan pomoću linije `ss3 arg1 arg2 arg3`, prvi komandni argument je arg1, drugi arg2, a treći arg3:

(b) Vrednost promenljive $# je broj komandnih argumenta (u ovom slučaju tri).

A.86. Šta je izlazni status komande i čemu služi?

Posle izvršenja, Linux komande vraćaju vrednost na osnovu koje se može odrediti da li je komanda izvršena uspešno ili ne. Ako je povratna vrednost 0, komanda je izvršena uspešno. Ako je povratna vrednost različita od 0 (tj. veća od 0), komanda se nije uspešno završila, a taj broj predstavlja neku vrstu dijagnostičkog statusa koja se naziva izlazni status. Shell programerima je ova vrednost raspoloživa preko sistemske promenljive $?.

A.87. a. Objasnite upotrebu tilda proširenja.

b. Šta radi komanda cd ~\doc?

(a) Tilda prefiks se tretira kao potencijalno ime korisnika za prijavljivanje na sistem. Tilda prefiks se zamenjuje po sledećim pravilima: ako je login-name nulte dužine, tilda se zamenjuje vrednošću HOME shell promenljive, a ako je HOME promenljiva nepostavljena, tilda se zamenjuje homedirektorijumom korisnika koji izvršava taj komandi interpreter.

(b) cd ~\doc pozicionira korisnika na poddirektorijum doc koji se nalazi u njegovom home direktorijumu.

A.88. Šta na ekranu ispisuje sledeća komanda: echo $`who am i;pwd ?

Ispisuje ime korisnika koji je prijavljen na sistem i tekući direktorijum. Na primer:

```
echo $`who am i;pwd`
jsmith  ttyp0  Nov 29 19:43 (slayer.internal.vets.edu.yu)
/tmp
```

A.89. Neka je promenljivoj ime dodeljena vrednost jsmith. Šta na ekranu ispisuje sledeća komanda: echo $(($ime!=jsmith))?

vrednost 0

A.90. U šta se proširuje sledeća komanda:

$ mkdir /home/jsmith/{data,video,mp3}?

Proširuje se u tri komande:

```
mkdir /home/jsmith/data,
mkdir /home/jsmith/video,
mkdir /home/jsmith/mp3.
```

Backup i arhiviranje. Instalacija softverskih paketa.

A.91. U čemu je razlika između rezervne kopije podataka (*backup*) i arhive?

Korisnici sami kreiraju rezervne kopije svojih podataka, a *backup* radi sistem administrator na nivou celog sistema. Arhiviranje je proces kreiranja kopije značajnih korisničkih datoteka na drugom medijumu u značajnim trenucima vremena. Pod terminom kopije podataka (*backup*) podrazumeva se postupak kreiranja kopije čitavog sistema datoteka ili fundamentalnih direktorijima i datoteka. *Backup* treba da obuhvati podatke koji obezbeđuju integritet celog sistema ili neke funkcionalne celine, kao što je, na primer, baza podataka ili sistem za elektronsku poštu.

A.92. Šta radi komanda `tar cvfz proba.tar.gz a.a b.b c.c`?

Komanda kreira komprimovanu arhivu proba.tar.gz i u nju smešta datoteke a.a, b.b i c.c

A.93. Kako se može prikazati sadržaj arhive kreirane u prethodnom pitanju?

```
tar tvfz proba.tar.gz
```

A.94. a. Navedite komandu koja bezuslovno dodaje datoteku b.b u arhivu proba.tar.

b. Gde se smešta datoteka b.b?

(a) `tar avf proba.tar b.b`

(b) Datoteka b.b se smešta na kraj arhive.

A.95. Objasnite režime rada komande cpio.

U izlaznom načinu rada (*copy-out*) komanda cpio arhivira datoteke, odnosno kopira ih u arhivu. U ulaznom režimu rada (*copy-in*) komanda cpio vrši ekstrakciju datoteke iz arhive, a u posebnom obliku (*copy-test*) lista sadržaj cpio arhive. U prolaznom režimu rada (*copy-pass*) komanda cpio kopira datoteke sa jednog direktorijuma na drugi, kombinujući ulazni i izlazni režim rada bez realnog korišćenja arhive.

A.96. Navedite komandu koja programom cpio arhivira sve datoteke iz tekućeg direktorijuma u datoteku /tmp/arch1.

```
ls -1 | cpio -ocvB > /tmp/arch1
```

A.97. Šta tačno radi komanda find / –mount –print | cpio –pduvm /newdisk? Pretpostaviti da je na direktorijum /mnt/newdisk montiran prazan sistem datoteka veličine veće od root sistema datoteka.

Komanda kreira kopiju root sistema datoteka na direktorijumu /newdisk.

A.98. Navedite sintaksu komande rpm za (a) instaliranje, (b) uklanjanje i (c) nadoradnju paketa.

(a) `rpm -i pack.rpm`

(b) `rpm -e pack.rpm`

(c) `rpm -U pack.rpm`

A.99. Navedite sintaksu komande apt-get za (a) instaliranje, (b) uklanjanje i (c) nadoradnju paketa.

(a) `apt-get install package`

(b) `apt-get remove package`

(c) `apt-get update; apt-get upgrade`

A.100. Ukratko opisati postupak instalacije .tar.gz paketa.

Tarball paketi se moraju ručno raspakovati. Posle toga se pokreće odgovarajući instalacioni program (na primer, sh install.sh) ili program za prevođenje

izvornog koda (*make*). U okviru paketa obično se nalazi datoteka koja opisuje dalji postupak instalacije.

Mrežno okruženje

A.101. a. Navedte primer IP adrese u klasi A, B i C.

b. Šta predstavljaju adrese 127.0.0.1 i 255.255.255.255 ?

(a) klasa A: 10.10.10.10

klasa B: 140.10.10.10

klasa C: 200.10.10.10

(b) 127.0.0.1 je adresa lokalne petlje, a 255.255.255.255 je broadcast adresa.

A.102. Šta je opisano datotekama (a) /etc/services i (b) /etc/protocols?

(a) Datoteku /etc/protocols koriste razni dijagnostički alati radi prevođenja broja protokola u simboličko ime (prilikom prikazivanja poruka na ekranu).

(b) Datotekom /etc/services opisani su servisi i portovi na kojima ti servisi pružaju usluge klijentima.

A.103. Šta znači linija `hosts: dns [!UNAVAIL=return] files` navedena u datoteci /etc/nssswitch.conf?

Znači da se razrešavanje imena obavlja preko DNS servera, a ukoliko je on nedostupan, preko lokalnih konfiguracionih datoteka (etc/hosts).

A.104. Čemu služi program ifconfig?

Komanda ifconfig koristi se za konfigurisanje mrežnih interfejsa rezidentnih u kernelu. Komandom ifconfig mogu se postaviti parametri poput IP adrese, maske podmreže i broadcast adrese, a takođe se može prikazati i trenutna konfiguracija mrežnog interfejsa, kao i MAC adresa mrežne kartice. Komanda ifconfig koristi se prilikom podizanja sistema radi postavljanja parametara mrežnog interfejsa. Nakon toga se najčešće koristi u dijagnostičke svrhe.

A.105. a. Pomoću kog programa se može proveriti da li je mreža preopterećena?

b. Na osnovu čega se izvodi taj zaključak?

(a) Može se zaključiti pomoću izveštaja o mrežnom interfejsu koji prikazuje program netstat (netstat -i). Izveštaj o mrežnom interfejsu obuhvata informacije o korišćenju mreže i broju kolizija.

(b) Mreža nije preopterećena ukoliko je broj kolizija, odnosno vrednosti RX-DRP i TX-DRP reda veličine 1-2% ukupnog broja paketa. U protivnom, mreža je preopterećena.

A.106. a. Šta su inetd i xinetd?

b. Šta je vezivanje servisa za IP adresu?

c. Šta je redirekcija zahteva?

(a) inetd i xinetd su wrapper-i, odnosno obvojnice. Nakon zahteva za uspostavljanjem konekcije na nekom portu, wrapper pokreće odgovarajući program koji pruža servis na tom portu i predaje mu kontrolu.

(b) Vezivanje (*binding*) servisa sa nekom IP adresom omogućava da se servisu može pristupiti isključivo preko IP adrese s kojom je povezan. Na ovaj način se različiti servisi mogu izvršavati preko različitih mrežnih interfejsa, što je korisno ukoliko se na sistemu nalazi više mrežnih adaptera ili ukoliko je sistem konfigurisan sa više IP adresa. Na primer, nesigurni servisi kao što je Telnet ne moraju se isključiti - servis se može vezati za IP adresu kojoj se može pristupiti samo sa lokalne mreže (interno), ali ne i sa Interneta.

(c) Redirekcijom se zahtevi za uspostavljanje konekcije na određenom portu preusmeravaju na drugi računar, odnosno na drugu IP adresu i port. Na primer, zahtev za uspostavljanjem konekcije na TCP portu 80 može biti preusmeren na Web server.

A.107. Šta je određeno linijom `/share/doc -ro -access=ws1:ws2:ws3` u datoteci /etc/exports NFS servera?

Ova linija određuje dostupnost konkretnog direktorijuma preko NFS-a. Direktorijum /share/doc je dostupan kao NFS sistem datoteka sa računara ws1, ws2 i ws3 u režimu čitanja (read-only).

A.108. Navedite komandu za montiranje NFS sistema datoteka /share/doc sa servera fserver1 na direktorijum /doc.

```
mount -t nfs fserver1://share/doc /doc
```

A.109. Koje sistemske datoteke ulaze u sastav NIS baze?

U podrazumevanom stanju, u sastav NIS baze ulaze sledeće datoteke: passwd - svi korisnici sistema, group - sve korisničke grupe, hosts - parovi IP adresa i imena računara, ethers - parovi MAC adresa i imena računara, networks - parovi IP adresa i imena mreža, services - imena IP portova, protocols - imena IP protokola i netgroup - pripadnost korisnika mrežnim grupama.

A.110. a. Navedite dva načina za pokretanje i zaustavljanje Apache web servera.

b. Koji je od ta dva načina bolji i zašto?

(a) Korišćenje apachectl skript datoteke i datoteke /etc/init.d/httpd.

(b) Metod za pokretanje httpd servera koji se preporučuje je korišćenje apachectl skripta. Ovaj skript podešava neke promenjive okruženja koje su neophodne za pravilnofunkcionisanje httpd servera u nekim okruženjima. Skript *apachectl* prosleđuje argumente date u komandnoj liniji prilikom njegovog pokretanja samom httpd-u, što omogućava korišćenje bilo koje *httpd* opcije sa *apachectl* skriptom. Neke opcije čije je prisustvo u datom okruženju uvek potrebno moguce je specificirati i u samoj datoteci *apachectl*.

Štampanje i administracija štampača

A.111. a. U čemu je razlika između lokalnih i udaljenih štampača?

b. Šta su mrežni štampači?

(a) U odnosu na računar sa koga se šalje zahtev za štampu, štampač može biti lokalni ili udaljeni - remote. Lokalni štampači su svi štampači povezani na paralelni ili serijski port računara, sa kog se šalje zahtev za štampu. Udaljeni štampači su svi štampači povezani na drugi server ili radnu stanicu, koji su dostupni sa računara sa kog se šalje zahtev za štampu.

(b) U mrežne štampače ugrađeni su NIC adapteri, odnosno mrežne kartice koje omogućavaju direktno vezivanje štampača na mrežne uređaje (Hub

ili Switch), pri čemu se veza najčešće ostvaruje UTP kablovima kategorije 5 i 5e.

A.112. Šta je red za štampač i u kom se direktorijumu obično nalazi? Šta predstavlja ime štampača?

Red za štampač (*Printer Queue*) je specijalan direktorijum ili datoteka smeštena na hard disku radne stanice ili mrežnog servera. Pošto štampači obično nemaju hard disk za skladištenje dokumenata, svi zahtevi se najpre smeštaju u red za štampač. Ako je štampač slobodan, zahtev se odmah štampa. Ako je štampač zauzet, zahtev čeka u redu sve dok se štampač ne oslobodi. Na UNIX sistemima, svaki štampač ima svoj red koji se nalazi na disku radne stanice ili mrežnog servera na koji je štampač povezan. Redovi za štampač se obično nalaze u direktorijumu /var/spool kao datoteke /var/spool/lp0 i /var/spool/lp2.

A.113. Navedite osnovne komande za štampanje po System V i BSD standardu.

System V: lp

BSD: lpr

A.114. Šta prikazuje komanda `lpq –Pnewprinter@nicotine`?

Prikazuje statusni izveštaj o štampaču newprinter vezanom na računar nicotine.

A.115. Šta radi komanda `lprm –Pprinter1 –user jsmith`?

Komanda uklanja sve poslove za štampu koje je korisnik jsmith poslao na štampač printer1.

A.116. Navedite primer komande kojom se pokreće direktno štampanje redirekcijom izlaza.

```
cat mydoc.txt > /dev/lp0
```

A.117. Navedite komandu za dodavanje dodavanje HP DeskJet štampača koji je povezan na paralelni port računara.

```
lpadmin -p HPDeskJet -E -v parallel:/dev/lp0 -m deskjet.ppd
```

A.118. Navedite komandu za brisanje štampača printer2.

```
lpadmin -x printer2
```

A.119. Navesti komandu koja dozvoljava korisnicima user1 i user2 da štampaju na štampaču printer2, a svim ostalim zabranjuje da štampaju na tom štampaču.

```
lpadmin -p printer2 -u allow:user1, user2
```

A.120. Navedite komandu kojom se svim korisnicima sistema dozvoljava da na štampaču printer2 štampaju najviše 100 stranica nedeljno.

```
lpadmin -p printer2 -o job-quota-period=604800 -o job-page-
limit=100
```

Procesi

A.121. Objasnite šta je *daemon* proces. Navedite primer *daemon* procesa.

Daemon proces je proces kog je pokrenuo kernel i koji se izvršava u pozadini. Na primer, lpd (*line printer scheduler*) se najčešće pokreće pri podizanju sistema s namenom da prihvata poslove za štampu i da njima upravlja. Ukoliko nema zahteva za štampu, lpd se izvršava, ali je neaktivan. Kada neko pošalje zahtev za štampu, lpd postaje aktivan dok se zahtev ne odštampa.

A.122. Navedite komandu kojom se može najlakše odrediti PID procesa tuxracer.

```
ps -ef | grep tuxracer
```

A.123. U čemu je razlika između signala TERM i KILL?

Osim signala KILL i STOP, proces može da "uhvati" (catch) signal, odnosno da izvrši neku drugu akciju umesto podrazumevane. Proces koji hvata signal može da odluči koju će akciju da izvrši kao posledicu datog signala.

TERM (terminate) je signal za uništenje procesa. Proces koji primi TERM signal može regularno da završi svoj rad (da najpre završi obradu podataka i upiše rezultate na disk, a zatim da prekine izvršenje).

KILL je signal za neopozivo uništenje procesa - proces ne može da ignoriše ili blokira ovaj signa.

A.124. Šta radi komanda `nohup sort lista1 > lista2 &` ?

Izvršavanje procesa u pozadini je posao u kome korisnik direktno ne učestvuje. Ukoliko se korisnik odjavi sa sistema, UNIX će poslati signal HUP svim procesima koje je inicirao shell, odnosno uništiće sve procese koji su nastali kao posledica izvršenja korisničkih komandi. To znači da se posle odjavljivanja korisnika prekida izvršenje svih programa koje je korisnik pokrenuo.

Ukoliko korisnik želi da pokrene program koji će nastaviti izvršenje i posle odjavljivanja korisnika sa sistema, potrebno je da obezbedi imunitet programa na HUP signal. Za to se koristi komanda nohup (no hang-up). Program se dalje izvršava normalno, ali je imun na neke signale, uključujući HUP. Posle odjavljivanja korisnika sa sistema, program nastavlja svoje izvršenje, ali više ne koristi terminal kao standardni izlaz.

Posle zadavanja komande

```
nohup sort lista1 > lista2 &
```

korisnik može da se odjavi sa sistema, a proces koji je iniciran komandom sort će nastaviti svoje izvršenje. Posle ponovnog prijavljivanja na sistem, korisnik može da pogleda rezultat sortiranja u datoteci lista2.

A.125. a. Navedite komandu kojom se može promeniti prioritet procesa čiji je PID 1500 sa 20 na 30.

b. Da li će se posle toga proces brže ili sporije izvršavati?

(a) `renice 30 1500,`

(b) Povećava se nice vrednost, što znači da prioritet procesa pada, tako da proces radi sporije.

A.126. Čemu služe komande fg i bg?

U komandnom interpreteru koji podržava kontrolu poslova, korisnik može pomoću komande fg premestiti izvršenje posla u prvi plan, posle čega posao postaje procesna grupa.

Komanda bg premešta suspendovani posao u pozadinu. Posao se najčešće premešta u pozadinu posle suspenzije, odnosno slanja kontrolnih karaktera za suspendovanje procesne grupe koja se izvršava u prvom planu. Posle premeštanja u pozadinu posao nastavlja sa izvršenjem u pozadini. Posao dalje ostaje u pozadini dok se ne izvrši, dok ne primi neki signal od korisnika, ili ne pokuša da izvrši ulazno/izlaznu operaciju vezanu za terminal.

A.127. U čemu je osnovna razlika između korišćenja komande at i cron daemon procesa?

Komanda at omogućava korisnicima da zakažu izvršenje komandi, odnosno binarnih i shell programa u određenom trenutku vremena.

Pomoću alata crontab korisnici mogu zakazati periodično izvršenje komande, odnosno izvršenje u specificiranim intervalima.

Podizanje i obaranje sistema

A.128. Navedite dva načina za konfigurisanje LILO boot loadera.

LILO se može konfigurisati na dva načina:

- pomoću interaktivnog programa /usr/sbin/liloconfig, koji zahteva da se odgovori na nekoliko jednostavnih pitanja,
- modifikacijom konfiguracione datoteke /etc/lilo.conf i pokretanjem programa lilo, koji ažurira informacije *boot* sektora sa informacijama u datoteci.

A.129. a. Šta je init ?

b. Kojim je nivoom izvršenja predstavljen jednokorisnički režim rada?

(a) Init je prvi proces korisničkog nivoa koga pokreće kernel, a zbog svog značaja za pravilno funkcionisanje sistema u literaturi se pominje pod imenom "*super daemon*". Init je zadužen za:

- pokretanje procesa getty (tako da korisnici mogu da se prijave na sistem) i upravljanje terminalima. Init obezbeđuje ispravno funkcionisanje programa getty i prekida njegov rad ukoliko administrator zabrani prijavljivanje na sistem. Na sistemima sa grafičkim okruženjem, init obezbeđuje pravilno funkcionisanje grafički orijentisanog login programa,
- implementaciju nivoa izvršenja (runlevels),
- usvajanje procesa siročića (orphans).

(b) Nivoom izvršenja 1.

A.130. Šta su inicijalizacione rc datoteke ? Kada se shell programi u rc direktorijumima izvršavaju ?

Prilikom prelaska u novi nivo izvršenja sistem izvršava komande, odnosno shell programe, koji se nalaze u /etc/rc?.d direktorijumima. Ove datoteke su označene zajedničkim imenom inicijalizacione rc datoteke, ili skraćeno - rc datoteke.

Sve rc datoteke, odnosno shell programi nalaze se u direktorijumu /etc/init.d. U /etc/rc?.d direktorijumima smešteni su linkovi na ove datoteke.

Konfigurisanje jezgra

A.131. Šta su moduli jezgra i šta se njima omogućava?

Programski moduli omogućavaju dodavanje funkcionalnosti jezgru bez ponovnog prevođenja izvornog koda jezgra. Na primer, ukoliko postoji potreba za povezivanjem SCSI uređaja na sistem, programski modul za podršku odgovarajućeg SCSI interfejsa se može učitati u jezgro jednom komandom (insmod). Inače, ukoliko se koristi monolitni kernel, Linux zahteva da se celo jezgro ponovo prevede sa uključenom podrškom za taj uređaj. S obzirom na veliki broj hardverskih uređaja koje Linux podržava, programskim modulima je omogućeno da:

- jezgro ostane relativno malo (mikrokernel arhitektura),
- korisnici na relativno jednostavan način dodaju podršku za svoj hardver, bez ponovnog podizanja sistema.

A.132. Navedite komande za učitavanje i uklanjanje modula za podršku ReiserFS sistema datoteka.

```
insmod reiserfs
rmmod reiserfs
```

A.133. Koji se problemi ne mogu rešiti modulima jezgra?

- na procesoru postoji greška (bug) koja povremeno izaziva krah sistema,
- na sistem je instaliran novi disk kontroler koji ima podršku u programskom modulu, ali se sistem ne može podići sa diska vezanog na taj kontroler ukoliko modul nije deo minimalnog kernela,

- pojavljuje se problem sigurnosti konkretne verzije kernela, .
- korisnici sistema se žale da ne mogu da pristupe nekim sistemima datoteka, ukoliko pre toga ne upotrebe komande insmod i mount,
- nadogradnja jezgra novom verzijom.

A.134. Kako se pokreće okruženje sa menijima iz koga se može konfigurisati izvorni kod novog jezgra ?

```
su -
cd /usr/src/kernel-source-2.4.19
make menuconfig
```

A.135. Navedite komande za prevođenje izvornog koda jezgra Linux sistema.

```
make dep; make clean; make zImage; make modules; make modules_install
```

A.136. Šta je /vmlinuz ?

Simbolički link na sliku kernela. Na primer:

```
ls -l /vmlinuz
lrwxrwxrwx   1   root   root    23   maj 25 2004   /vmlinuz -> boot/vmlinuz-2.4.26-686
```

Disk kvote

A.137. Gde je implementirana podrška za disk kvote?

Implementirana je na nivou kernela.

A.138. Šta se treba dodati u liniju `LABEL=/home /home ext2 defaults 1 2` datoteke /etc/fstab da bi kvote bile postavljene za korisnike, a šta za grupe?

`LABEL=/h /home ext2 defaults, usrquota 1 2` (za korisnike):

`LABEL=/home /home ext2 defaults, grpquota 1 2` (za grupe)

A.139. U čemu je razlika između soft i hard limita ? Šta je grace period?

Soft limit određuje najveću količinu prostora na sistemu datoteka koju korisnik može da iskoristi za smeštanje svojih datoteka. *Hard limit* je apsolutno ograničenje. Korisnik ne može ni na koji način da pređe ovaj limit. *Hard limit* vrednost ima smisla samo ako je postavljen parametar *grace period*. Parametrom *grace period* (period milosti) postavlja se vremenska granica pre nasilne primene vrednosti parametra soft limit.

Jednostavno rečeno, *soft limit* je rastegljiv do granice definisane *hard limit*-om u periodu *grace period*.

1.140. Čemu služi program *qoutacheck*?

Alat quotacheck analizira potrošnju datoteka i direktorijuma na odgovarajućem sistemu datoteka i na osnovu toga kreira odgovarajuće datoteke quota.user i quota.group.

Sigurnost i zaštita

A.141. Navedite dve osnovne mere povećanja opšte sigurnosti sistema na nivou BIOS-a.

1. Zabrana podizanja operativnog sistema sa flopi diska i CD-ROM uređaja,
2. Postavljanje lozinke za pristup BIOS konfiguraciji.

A.142. Čemu služe direktive *password* i *restricted* u datoteci /etc/lilo.conf?

Direktiva password=pass štiti sliku kernela lozinkom. Ukoliko se direktiva password navede, korisnik koji želi da podigne sistem moraće da navede lozinku pass. LILO lozinke su osetljive na mala i velika slova.

Ukoliko je direktiva restricted navedena u datoteci /etc/lilo.conf, LILO će zahtevati lozinku samo u slučajevima učitavanja slike kernela sa dodatnim parametrima iz LILO komandne linije (na primer: linux single). Direktiva se navodi zajedno sa direktivom password za svaku sliku kernela posebno.

Napomena: korišćenje direktive password bez direktive restricted je loša praksa. U tom slučaju je nemoguće izvesti reboot proceduru sa udaljenog računara, jer LILO zahteva lozinku koja se može uneti samo sa tastature servera.

A.143. Šta je opisano datotekom /etc/securetty?

Opisani su "sigurni terminali", odnosno terminali sa kojih root može da se prijavi na sistem. Sa ostalih terminala, superuser može da se prijavi samo ako koristi program su.

A.144. Kako se može zabraniti korišćenje kombinacije tastera Ctrl-Alt-Del?

Zaustavljanje sistema kombinacijom tastera Ctrl-Alt-Del može se najlakše zabraniti pretvaranjem sledeće linije u datoteci /etc/inittab u komentar:

```
# ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Posle toga je potrebno zadati komandu:

```
/sbin/init q
```

A.145. Kako se poruka digitalno potpisuje?

Prilikom potpisivanja, najpre se jednosmernom hash funkcijom izračuna hash vrednost h poruke p, koja se posle toga šifruje algoritmom sa javnim ključem:

- kreiranje hash vrednosti H(p)=h1,
- potpisivanje (šifrovanje hash vredosti privatnim ključem) E(k1,h1)=s.

A.146. Šta radi komanda gpg -sear jsmith /backup/dat1?

Komanda šifruje datoteku /backup/dat1 javnim ključem korisnika jsmith i potpisuje šifrat privatnim ključem korisnika jsmith. Od korisnika koji zadaje komandu zahteva se da unese passphrase svog privatnog ključa.

A.147. Šta je demilitarizovana zona?

Demilitarizovana zona (DMZ) je deo mreže koji ne pripada privatnoj mreži, a nije direktno povezan na Internet. Najčešće, ovo područje je između rutera za pristup Internetu i interne mreže, odnosno javni deo privatne mreže.

A.148. Čemu služi NAT tabela?

NAT tabele se koriste za prevođenje mrežne adrese (izvorišno ili odredišno polje) IP paketa u Linux kernelu 2.4.

A.149. Čemu služi *mangle* tabela?

Mangle tabelom menjaju se svojstva paketa koja nemaju direktan uticaj na pakete (na primer polja *Type Of Service* i *Time To Live*).

A.150. Kako se vrši filtriranje paketa kroz lance?

Paket koji stigne do mrežne barijere prosleđuje se na odgovarajući drajver za uređaj u kernelu. Paket dalje prolazi kroz seriju koraka u kernelu pre nego što se šalje pravoj aplikaciji (lokalno) ili preusmerava ka drugom računaru.

Na primer, paket koji dolazi sa mreže, a koji je namenjen drugom računaru na mreži, prolazi kroz sledeće korake: paket ulazi u mrežni interfejs (na primer eth0), mangle prerouting, nat prerouting, odluka o rutiranju, mangle forward, filter forward, mangle postrouting, nat postrouting, paket se šalje na mrežni interfejs (na primer eth1).

Paket se može zaustaviti na bilo kom lancu iptables mrežne barijere. Lanci i tabele su jedinstveni za sve mrežne interfejse.

B. Primer testa sa prvog dela ispita

U ovom dodatku je dat primer testa za polaganje prvog dela ispita (iz januarskog roka 2003. godine) sa tačnim odgovorima i svim potrebnim objašnjenjima. Studenti na testu biraju tačan odgovor od nekoliko ponuđenih. Svaki tačan odgovor donosi dva poena, netačan 0, a nonsens se kažnjava sa -2 poena (sprečava "sportsku prognozu"). Za prolaz je potrebno 60 poena.

B.1. Programi za kreiranje sistema datoteka pod UNIX operativnim sistemom deluju na određeni deo diska i obavljaju sledeće funkcije:

a. fdisk deluje na jedan ili više sistema datoteka i kreira im granice, a mkfs deluje samo na jedan sistem datoteka i logički kreira taj sistem datoteka,

b. fdisk deluje na ceo disk i kreira granice, a mkfs deluje na jedan ili više sistema datoteka i logički kreira sve obuhvaćene sisteme datoteka,

c. fdisk deluje na ceo disk i kreira granice, a mkfs deluje na jedan sistem datoteka i logički kreira taj sistem datoteka,

d. fdisk deluje samo na jedan sistem datoteka i logički kreira taj sistem datoteka, a mkfs deluje na ceo disk i kreira granice.

Tačan odgovor: c.

Objašnjenje: fdisk kreira granice u kojima mkfs kreira fajl sisteme. Što će reci, administrator zadaje komande tipa fdisk disk-node i mkfs partition-node. Argument za fdisk je uvek nod za ceo disk. Na primer, zadaćete komandu: `fdisk /dev/hda`, ukoliko želite da izdelite primary master na particije. Argument za mkfs je uvek nod za particiju u kojoj želite da kreirate fajl sistem (najverovatnije morate da navedete i tip i još neke parametre sistema datoteka). Na primer, zadaćete komandu: `mkfs -t ext3 /dev/hdb5` ukoliko želite da kreirate ext3 sistem datoteka u prvoj logičkoj particiji *primary slave* diska.

B.2. Prilikom podizanja UNIX operativnog sistema, aktivno UNIX stabla se formira po sledećem redu:

a. / direktorijum, swap, root fs, *auto-mount* sistemi datoteka, ručno aktivirani sistemi datoteka,

b. / direktorijum, root fs, ručno aktivirani sistemi datoteka, *auto-mount* sistemi datoteka,

c. / direktorijum, root fs, *auto-mount* sistemi datoteka, ručno aktivirani sistemi datoteka,

d. / direktorijum, swap, root fs, ručno aktivirani sistemi datoteka, *auto-mount* sistemi datoteka.

Tačan odgovor: c.

Nonsens: a, d (swap se ne montira na aktivno stablo)

Objašnjenje: posle punjenja memorije kernelom root fs se aktivira u *read-only* režimu. Posle toga se proverava integritet root sistema datoteka, ukoliko root fs nije neki od journaling sistema datoteka ili ukoliko drugačije nije eksplicitno naglašeno (npr. u polju fs_passno u datoteci /etc/fstab za root fs upisano je 0). Zatim se root FS ponovo aktivira (*re-mount*) u režimu čitanja i pisanja. Posle toga se aktiviraju svi *auto-mount* sistemi datoteka, tj. svi sistemi datoteka koji su opisani u datoteci /etc/fstab i za koje u polju *options* nije naveden parametar *noauto*. Posle toga, superuser (i u određenim slučajevima, regularni korisnici) mogu aktivirati neke druge sisteme datoteka. Kratko rečeno: prvo se aktivira

root fs, zatim sve što je opisano u /etc/fstab, i na kraju ono što korisnici aktiviraju.

B.3. Na jednom aktivnom UNIX i aktivnom DOS/Windows 3.x/9x/ME operativnom sistemu za lokalne sisteme datoteka važi sledeće:

a. i na *Windows* i na UNIX sistemima se svi sistemi datoteka aktiviraju u procesu podizanja operativnog sistema i ostaju stalno aktivni,

b. na *Windows*-u se samo logički disk C: aktivira pri podizanju operativnog sistema; prilikom podizanja UNIX sistema sktivira se samo root fs,

c. na *Windows*-u se svi sistemi datoteka aktiviraju pri podizanju operativnog sistema; prilikom podizanja UNIX sistema aktiviraju se swap i root fs.

d. na *Windows*-u se samo logički disk C: aktivira pri podizanju operativnog sistema; na UNIX sistemima se svi sistemi datoteka aktiviraju u procesu podizanja operativnog sistema.

Tačan odgovor: c.

Objašnjenje: DOS/Windows 3.x/9x/ME aktiviraju sve lokalne FAT sisteme datoteka prilikom podizanja operativnog sistema. Posle toga nema dodatnog aktiviranja lokalnih FAT sistema datoteka. Windows NT/2K/XP/2K3 aktivira lokalne sisteme datoteka (FAT i NTFS), shodno tome kako je administrator naglasio *Logical Disk Management* servisu. Posle toga, administrator i ovlašćeni korisnici mogu montirati neki lokalni sistem datoteka sa hard diska na neki *mount-point* direktorijum ili mapirati bilo koji logički disk na lokalni ili mrežni sistem datoteka ili CD/DVD uređaj. Prilikom podizanja UNIX operativnog sistema, root fs mora da se aktivira. Root fs se montira na root direktorijum aktivnog UNIX stabla i ostaje stalno aktivan dok je sistem podignut. Takođe, jedna swap particija mora biti aktivirana prilikom podizanja operativnog sistema. Nakon toga se pomoću komandi mount i umount (ili pomoću programa kao što je KwikDisk) obavlja aktiviranje/deaktiviranje sistema datoteka. Što se tiče *swap*-a, ukoliko se swap formira kao datoteka, swap particija se može deaktivirati.

Napomena: Pod DOS/Windows 3.x/9x/ME operativnim sistemima mogu se aktivirati neki specifični sistemi datoteka pomoću drajvera (kao što su drajveri za Norton DiskReet ili Secure Filesystem). Ovakvi fajl sistemi se generalno mogu deaktivirati. Međutim, podrška za ove FS ne isporučuje se uz sam operativni sistem, već kao poseban programski paket ili deo programskog

paketa (npr. Norton Utilities), tako da se može smatrati da DOS/Windows 3.x/9x/ME ne mogu dodatno aktivirati lokalne sisteme datoteka nakon podizanja operativnog sistema.

B.4. Aktiviranje swap particije i aktiviranje sistema datoteka je:

a. identična procedura i radi se istim komandama,

b. identična procedura, ali se radi se različitim komandama

c. različita procedura, ali se radi istim komandama,

d. različita procedura koja se radi različitim komandama.

Tačan odgovor: d.

Nonsens: a, b, c.

B.5. Komanda mount na UNIX sistemu obavlja sledeću aktivnost:

a. montira prethodno aktivirani sistem datoteka na *mount-point* direktorijum,

b. mapira logički disk na prethodno neaktivirani sistem datoteka,

c. mapira logički disk na prethodno aktivirani sistem datoteka.

d. montira prethodno neaktivirani sistem datoteka na *mount-point* direktorijum,

e. deaktivira sistem datoteka.

Tačan odgovor: d.

B.6. Komanda za reparaciju sistema datoteka pod UNIX-om je:

a. df i može se koristiti bez obzira na stanje sistema datoteka,

b. mkfs i može se koristiti bez obzira na stanje sistema datoteka,

c. badblocks i može se koristiti bez obzira na stanje sistema datoteka,

d. fsck i može se koristiti bez obzira na stanje sistema datoteka,

e. fsck i može se koristiti samo na sistemima datoteka koji su aktivirani,

f. fsck i može se koristiti samo na sistemima datoteka koji nisu aktivirani.

Tačan odgovor: f.

Objašnjenje: integritet sistema datoteka može se proveriti pomoću programa fsck (*filesystem check*). Program fsck će otkloniti manje "kvarove" u sistemu datoteka i upozoriti korisnika na postojanje problema koji se ne mogu otkloniti.

Program fsck zaobilazi standardne rutine za pristup sistemima datoteka i tretira disk kao uređaj, tako da sistem datoteka čiji se integritet provera ne sme biti aktiviran.

B.7. Postupak kreiranja i aktiviranja swap prostora u formi datoteke obuhvata sledeće korake (komande):

a. kreiranje prazne datoteke, mkswap, swapoff,

b. kreiranje prazne datoteke, mkswap, mount,

c. kreiranje prazne datoteke, swapon, mkswap,

d. kreiranje prazne datoteke, mkswap, swapon,

e. kreiranje prazne datoteke, mkswap, fsck,

f. kreiranje prazne datoteke, fdisk, mkswap, swapon,

g. kreiranje prazne datoteke, fdisk, swapon.

Tačan odgovor: d.

Nonsens: b.

Swap datoteka se kreira na sledeći način: najpre se u postojećem sistemu datoteka kreira prazna datoteka bez rupa željene veličine, a zatim se u okviru nje kreira logička struktura swap prostora. Swap datoteke se mogu aktivirati i deaktivirati na isti način kao i swap particije, komandama swapon i swapoff, pri čemu se kao parametri programa navode imena swap datoteka sa apsolutnom putanjom.

```
dd if=null of=/swap_file count=100M
mkswap /swap_file
swapon /swap_file
```

B.8. Ukoliko vam se pojave fizički oštećeni sektori na disku sa podacima, postupak oporavka diska sa očuvanjem postojećih podataka obuhvata sledeće korake (komande):

a. badblocks koja detektuje i oporavlja defekte na celom disku,

b. badblocks koja detektuje i oporavlja defekte na oštećenom sistemu datoteka,

c. badblocks koja detektuje defekte, a zatim scandisk koja oporavlja ceo disk na bazi liste defekata generisane od strane programa badblocks,

d. badblocks koja detektuje defekte, a zatim fsck koja oporavlja oštećeni sistem datoteka na bazi liste defekata generisane od strane programa badblocks,

e. badblocks koja detektuje defekte, a zatim mkfs koja oporavlja oštećeni sistem datoteka na bazi liste defekata generisane od strane programa badblocks,

g. badblocks koja detektuje defekte, a zatim df koja oporavlja ceo disk na bazi liste defekata generisane od strane programa badblocks.

Tačan odgovor: d.

Komanda badblocks se može koristiti za proveru ispravnosti površine diska koju sistem datoteka zauzima. Kao rezultat izvršenja komande na ekranu se prikazuju brojevi svih neispravnih blokova diska koji se analizira - ova lista se kasnije može iskoristiti kao parametar programa fsck čime se lista neispravnih blokova aranžira u meta-data strukturi sistema datoteka. Sledeći primer ilustruje komande kojima se ova provera može izvesti:

```
badblocks /dev/sda2 > bad-block-list
fsck -t ext2 -l bad-block-list /dev/sda2
```

B.9. Koliko obaveznih ulaznih parametara zahtevaju komande mount i umount zahtevaju?

a. mount - tačno 2 ili 0, umount - najmanje 1,

b. mount - tačno 2 ili 0, umount - najmanje 2,

c. mount - najmanje 2, umount - tačno 1,

d. mount - tačno 2, umount - najmanje 1,

e. mount - tačno 2 ili 0, umount - tačno 1,

f. mount - ili 2 ili 0, umount - najmanje 3.

Tačan odgovor: e.

Objašnjenje: sintaksa komande mount je: `mount` (prikazuje koji su sistemi datoteka aktivirani i na koje su direktorijume montirani) ili `mount filesystem-node MPD` (montira sistem datoteka specificiran blok specijalnom datotekom filesystem-node na direktorijum MPD). Ukupno obaveznih parametara za mount: 0 ili 2.

Sintaksa komande umount je: `umount filesystem-node` ili `umount MPD`. U oba slučaja, umount deaktivira fajl sistem, pri čemu se navodi odgovarajuća specijalna datoteka ili *mount-point* direktorijum. Ukupno obaveznih parametara za umount: 1

B.10. Za prikazivanje količine slobodnog prostora na jednom ili više sistema datoteka i prikazivanje količine prostora na sistemu datoteka koju zauzimaju direktorijumi respektivno se koriste sledeće komande:

a. fsck i du,

b. badblocks i df,

c. fdisk i fsck,

d. du i df,

e. df i du,

f. mkfs i df.

Tačan odgovor: e.

Objašnjenje: pogledajte rezultate izvršenja sledećih komandi

```
# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sdb3             372M   34M  319M  10% /
/dev/sda1              45M  7.4M   35M  17% /boot
/dev/sdb2             973M   24M  899M   3% /home
/dev/sda3             1.7G  1.4G  268M  85% /shares
# du -h /home
16k     /home/lost+found
24k     /home/nemanja
7.0M    /home/dragan
24k     /home/bora
7.1M    /home
```

B.11. Za nodove kojima su predstavljeni sistemi datoteka i mount-point direktorijume (MPD) važe sledeće osobine:

a. nodovi se nalaze isključivo na direktorijumu /dev, a imena nodova su proizvoljna; MPD može biti bilo koji UNIX direktorijum (poželjno je da bude prazan),

b. nodovi se mogu nalaziti na bilo kom direktorijumu, a imena nodova su proizvoljna; MPD može biti bilo koji UNIX direktorijum (poželjno je da bude prazan),

c. nodovi se nalaze isključivo na direktorijumu /dev, a imena nodova su strogo određena; MPD može biti bilo koji UNIX direktorijum (poželjno je da bude prazan),

d. nodovi se mogu nalaziti na bilo kom direktorijumu, a imena nodova su strogo određena; MPD može biti bilo koji UNIX direktorijum (poželjno je da bude prazan),

e. nodovi se nalaze isključivo na direktorijumu /dev, a imena nodova su strogo određena; MPD može biti bilo koji UNIX direktorijum (mora biti prazan),

f. nodovi se mogu nalaziti na bilo kom direktorijumu, a imena nodova su strogo određena; MPD može biti bilo koji UNIX direktorijum (mora biti prazan).

Tačan odgovor: c.

Objašnjenje: sistemi datoteka su predstavljeni blok specijalnim datotekama koje se nalaze na direktorijumu /dev (npr. /dev/sda2, /dev/hda5). Imenom specijalne datoteke određen je tip diska (SCSI/IDE), redni broj uređaja (IDE *Primary Slave* ili treći SCSI Disk) i broj particije. *Mount-point* direktorijumi mogu biti bilo koji direktorijumi, ali je preporučljivo da budu prazni. Ukoliko nisu prazni, njihov sadržaj će biti privremeno nevidljiv (sve dok se sistem datoteka ne demontira).

B.12. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda2 on / type ext2 (rw)
/dev/hda1 on /DOS type msdos (rw)
/dev/hda4 on /u type ext2 (rw)
none on /proc type proc (rw)
```

Pod predpostavkom da je *primary master* jedini disk, da na disku nema extended particije i da je swap realizovan u formi particije, odredite ispravnu vezu između particija i sistema datoteka:

a. hda4: root, hda3: swap, hda2: UNIX user, hda1: FAT,

b. hda4: UNIX user, hda3: swap, hda2: root, hda1: FAT,

c. hda4: UNIX user, hda3: FAT, hda2: root, hda1: swap,

d. hda4: swap, hda3: UNIX user, hda2: root, hda1: FAT.

Tačan odgovor: b.

Objašnjenje: /dev/hda4 je ext2 sistem datoteka montiran na direktorijum /u, što će reći da je to UNIX user fs (korisnički sistem datoteka). /dev/hda2 je ext2 sitem datoteka montiran na / aktivnog stabla, što će reći da je to root fs. /dev/hda1 je FAT sistem datoteka montiran na direktorijum /DOS. Pod datom pretpostavkom da je /dev/hda jedini disk i da na njemu ne postoji extended particija, zaključuje se da je /dev/hda3 swap particija.

Ukoliko se eksplicitno ne naglasi da ne postoji extended particija, onda /dev/hda3 može biti extended, tako da swap može biti smešten u npr. /dev/hda5, /dev/hda6 itd.

B.13. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda4 on /DOS type msdos (rw),
/dev/hda3 on / type ext2 (rw),
/dev/hda1 on /u type ext2 (rw),
server12:/baza1 on /baza1 type nfs (rw),
none on /proc type proc (rw).
```

Odredite ispravnu vezu između mount-point direktorijuma i sistema datoteka koji su na te direktorijume montirani.

a. /: local-FAT, /DOS: local-FAT, /u : local-UNIX, /baza1: local-SWAP,

b. /: local-UNIX, /DOS: local-SWAP, /u: local-UNIX, /baza1: local-UNIX ,

c. /: local-UNIX, /DOS: local-FAT, /u: local-UNIX, /baza1: remote NFS,

d. /: local-UNIX, /DOS: local-FAT, /u: remote NFS, /baza1: local-UNIX.

Tačan odgovor: c.

Objašnjenje: /dev/hda4 je FAT sistem datoteka montiran na direktorijum /DOS. /dev/hda3 je ext2 sistem datoteka montiran na / aktivnog stabla, što će reći da je to root fs. /dev/hda1 je ext2 sistem datoteka montiran na direktorijum /u, što će reći da je to UNIX user fs. Na direktorijum /baza1 montiran je mrežni sistem datoteka (remote NFS) server12:/baza1 (računar server12, direktorijum baza1).

B.14. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda4 on /DOS  type msdos (rw),
/dev/hda1 on / type ext2 (rw),
/dev/hda2 on /u type ext2 (rw),
none on /proc type proc (rw).
```

Pod pretpostavkom da je primary master jedini disk i da na njemu ne postoji extended particija, odredite gde se nalazi swap particija:

a. Swap particija se može logički odrediti i to je /dev/hda1,
b. Swap particija se može logički odrediti i to je /dev/hda2,
c. Swap particija se može logički odrediti i to je /dev/hda3,
d. Swap particija se može logički odrediti i to je /dev/hda4,
e. Swap particija se ne može logički odrediti.

Tačan odgovor: c.

B.15. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda1 on / type ext2 (rw),
/dev/hda4 on /DOS type msdos (ro),
none on /proc type proc (rw).
```

Odredite ispravnu vezu između particija i sistema datoteka:

a. hda4: Swap, hda3: FAT, hda2: UNIX user, hda1: root ,
b. hda4: FAT, hda3: nepoznat, hda2: nepoznat, hda1: root,
c. hda4: FAT, hda3: swap, hda2: swap, hda1: root,
d. hda4: swap, hda3: FAT, hda2: UNIX user, hda1: root.

Tačan odgovor: b.

Objašnjenje: uzevši u obzir da nije eksplicitno navedeno da je *primary master* jedini disk i da nije eksplicitno navedeno da na njemu nema drugih particija, zaključuje se:

- /dev/hda4 je FAT sistem datoteka montiran na direktorijum /DOS,
- /dev/hda1 je ext2 sistem datoteka montiran na / aktivnog UNIX stabla, što će reći da je to root fs.

Ne može se zaključiti da li je swap particija /dev/hda2, /dev/hda3, u nekoj logičkoj particiji ili na nekom drugom disku.

Napomena: čak i kada bi bilo eksplicitno naglašeno da je *primary master* jedini disk i da na njemu nema drugih particija, ne bi se moglo odrediti da li je swap na /dev/hda2 ili /dev/hda3. Zato je nepoznato šta se nalazi na /dev/hda2 i /dev/hda3.

B.16. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda1 on / type ext2 (rw),
/dev/hdc5 on /mnt type ext2 (rw),
/dev/hda4 on /DOS type msdos (ro),
none on /proc type proc (rw).
```

Na osnovu toga odredite broj i vrstu diskova na serveru:

a. minumum 2 IDE diska, IDE *Primary Master* i IDE *Primary Slave*

b. minumum 2 IDE diska, IDE *Primary Master* i IDE *Secondary Master*

c. minimum 2 SCSI diska

d. tačno 2 IDE diska, IDE *Primary Master* i IDE *Secondary Master*

e. samo 1 IDE disk, IDE Primary Master

Tačan odgovor: b.

Objašnjenje: pošto je na root direktorijum aktivnog stabla montiran ext2 fs sa particije /dev/hda1, a na direktorijum /DOS montiran FAT fs sa particije /dev/hda4 zaključujemo da postoji IDE *primary master* (/dev/hda). Pošto je na direktorijum /mnt montiran ext2 fs sa particije /dev/hdc5, zaključujemo da postoji IDE *secondary master* (/dev/hdc). Na osnovu ove komande ne može se odrediti da li postoje i neki drugi diskovi na sistemu. Zato je odgovor minimum dva diska: IDE *primary master* i IDE *secondary master*.

B.17. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda1 on / type ext2 (rw)
/dev/hdb6 on /mnt type ext2 (rw)
/dev/hda4 on /home type ext2 (ro)
/dev/hda5 on /temp type ext2 (rw)
none on /proc type proc (rw)
```

Na osnovu toga odrediti vezu FileSystem-a i particija u kojoj se nalaze:

a. hda1:extended, hda4:primary, hda5:extended, hdb6:extended

b. hda1:primary, hda4:extended, hda5:extended, hdb6:extended

c. hda1:primary, hda4:primary, hda5:extended, hdb6:extended

d. hda1:primary, hda4:primary, hda5:extended, hdb6:primary

Tačan odgovor: c.

Objašnjenje: /dev/hdax ili /dev/sdax su:

- primarne (ili extended) particije, ako je x=1,2,3,4. Ako se na osnovu izvršenja komande mount vidi da je npr. na /dev/hda2 montiran neki sistem datoteka, onda je /dev/hda2 primarna particija. U suprotnom, /dev/hda2 može biti primarna ili extended particija.
- logičke particije, ako je x=5,6,7...

To znači da se sistemi datoteka /dev/hda1 i /dev/hda4 sigurno nalaze u primarnim particijama, a /dev/hda5 i /dev/hdb6 u logičkim, tj u extended particiji.

B.18. Rezultat izvršenja komande mount na jednom UNIX serveru je

```
/dev/hda1 on / type ext2 (rw)
/dev/hda4 on /home type ext2 (ro)
/dev/hdc1 on /baza1 type ext2 (rw)
/dev/hdc7 on /baza2 type ext2 (rw)
none on /proc type proc (rw)
```

Na osnovu toga odredite na koliko je dela izdeljen *primary master*.

a. tačno 4 dela, a postojanje extended particije nije sigurno,

b. tačno 4 dela, a postoji extended particija,

c. tačno 2 dela, a ne postoji extended particija,

d. minimum 3 dela, bez extended particije.

Tačan odgovor: a.

Objašnjenje: ako postoji /dev/hda1 i /dev/hda4, onda sigurno postoje i /dev/hda2 i /dev/hda3. Međutim, na aktivno UNIX stablo nije aktiviran ni jedan sistem datoteka sa particija /dev/hda2 i /dev/hda3, tako da se ne može odrediti šta se na njima nalazi. Zaključujemo da je *primary master* izdeljen na tačno četiri dela (4 primarne/extended particije) ne računajući logičke particije, ali i da se ne može odrediti da li extended particija postoji ili ne postoji.

Napomena: slično tome, za secondary master se može zaključiti da je izdeljen na najmanje dva dela od kojih je jedan sigurno extended particija.

B.19. Rezultat izvršenja komande mount na jednom UNIX serveru je:

```
/dev/hda1 on / type ext2 (rw)
/dev/hda4 on /home type ext2 (ro)
/dev/hdc1 on /baza1 type ext2 (rw)
/dev/hdc7 on /baza2 type ext2 (rw)
none on /proc type proc (rw)
```

Na osnovu toga odredite koliko se sistema datoteka nalazi u četvrtom delu *primary master* diska:

a. tačno jedan,

b. najmanje jedan,

c. najmanje dva,

d. najmanje 1, a broj se ne moze tačno odrediti.

Tačan odgovor: a.

Objašnjenje: u četvrtom delu diska (/dev/hda4) nalazi se ext2 sistem datoteka aktiviran na direktorijum /home.

B.20. Svaka linija u datoteci /etc/passwd sadrži sledeće informacije (u tačnom redosledu):

a. ime:password:shell:UID:GID:komentar:home

b. ime:password:UID:GID:komentar:home:shell

c. ime:password:UID:GID:komentar:shell:home

d. ime:UID:GID:komentar:home:shell:password

e. ime:home:shell:password:UID:GID:komentar

Tačan odgovor: b.

Objašnjenje: datoteka /etc/passwd je tabela u kojoj svaka linija (zapis) opisuje tačno jednog korisnika na sledeći način.

```
ime:password:UID:GID:komentar:HOME:Shell
```

Polja imaju sledeće značenje: ime – korisničko ime, password – lozinka korisnika, koja se obično smešta u datoteku /etc/shadow, UID (User ID) – jedinstveni identifikator korisnika, GID (Group ID) – jedinstveni identifikator primarne grupe korisnika, komentar – kraći opis korisnika, home – home direktorijum korisnika (npr /home/cartman), shell – komandni interpreter korisnika (npr /bin/bash).Na primer:

```
jsmith:x:1152:1002:John Smith III:/home/jsmith:/bin/bash
```

B.21. Sistemski korisnici UNIX sistema (kao što su lp, bin i mem) imaju sledeće osobine:

a. nastaju u toku instalacije UNIX sistema i služe da se korisnici prijave na sistem,

b. nastaju u toku instalacije UNIX sistema i imaju specijalnu namenu (ne služe za prijavljivanje korisnika na sistem),

c. nastaju posle instalacije UNIX sistema i služe da se korisnici prijave na sistem,

d. nastaju posle instalacije UNIX sistema i imaju specijalnu namenu (ne služe za prijavljivanje korisnika na sistem).

Tačan odgovor: b.

Nonsens: c.

Obaveštenje: sistemski korisnici, kao što su root, daemon i slični nastaju u toku instalacije konkretnog UNIX sistema. Osim *superuser*-a (root) koji može da se prijavi na sistem i obavi neke administrativne poslove, ostali sistemski korisnici ne mogu da se prijave na sistem.

B.22. Regularni korisnici UNIX sistema imaju sledeće osobine:

a. nastaju u toku instalacije UNIX sistema i služe da se korisnici prijave na sistem,

b. nastaju u toku instalacije UNIX sistema i imaju specijalnu namenu (ne služe za prijavljivanje korisnika na sistem),

c. nastaju posle instalacije UNIX sistema i služe da se korisnici prijave na sistem,

d. nastaju posle instalacije UNIX sistema i imaju specijalnu namenu (ne služe za prijavljivanje korisnika na sistem).

Tačan odgovor: c.

Nonsens: b.

Obaveštenje: regularne korisničke naloge kreira root posle instalacije UNIX sistema i oni služe da bi se korisnici prijavili na sistem.

Napomena: neke Linux distribucije pred kraj instalacije nude mogućnost kreiranja regularnih korisničkih naloga. Treba uzeti u obzir da je u tom momentu većina datoteka koje čine tu Linux distribuciju iskopirana na sistem, tako da se to može prihvatiti kao kreiranje korisnika nakon instalacije sistema.

B.23. Svaka datoteka mora:

a. pripadati bar jednoj grupi, a ta grupa može biti ili regularna ili sistemska,

b. pripadati bar jednoj grupi, a ta grupa mora biti sistemska,

c. pripadati bar jednoj grupi, a ta grupa mora biti regularna,

d. ne mora pripadati ni jednoj grupi,

e. pripadati tačno jednoj grupi, a ta grupa mora biti sistemska,

f. pripadati tačno jednoj grupi, a ta grupa moze biti ili regularna ili sistemska,

g. pripadati tačno jednoj grupi, a ta grupa mora biti regularna

Tačan odgovor: f.

Objašnjenje: korisnik mora pripadati najmanje jednoj grupi koja je navedena u datoteci /etc/passwd. Ta grupa se naziva primarna grupa korisnika i njoj se formalno dodeljuju datoteke koje taj korisnik iskopira ili kreira. Primarna grupa korisnika može biti sistemska ili regularna. Ostale grupe kojima korisnik

pripada koriste se za regulisanje kontrole pristupa (pripadnost korisnika ostalim grupama reguliše se pomoću datoteke /etc/group).

B.24. Vlasnika i grupu datototeke odredićete najlakše pomoću:

a. komande man,
b. komande info,
c. komande ls -a ,
d. komande ls -l ,
e. komande ls –c,
f. direktnim čitanjem indeksnog čvora te datoteke.

Tačan odgovor: d.

Objašnjenje: sledeći primer govori dovoljno.

```
# ls -l Fbi_1.31.gz
-rwxr--r-- 1 nmacek admin 92194 maj 18 2004 Fbi_1.31.gz
```

Vlasnik je nmacek, grupa je admin.

B.25. Korisnik u odnosu na bilo koju datoteku ili direktorijum može biti:

a. vlasnik i ostatak sveta istovremeno,
b. vlasnik i grupa istovremeno,
c. grupa i ostatak sveta istovremeno,
d. vlasnik, grupa i ostatak sveta istovremeno,
e. ne može biti nista istovremeno.

Tačan odgovor: b.

Nonsens: a, c, d.

Objašnjenje: prava pristupa određuju se prema vlasničkoj kategoriji kojoj korisnik pripada. U slučajevima unije vlasničkih kategorija (vlasnik datoteke pripada grupi kojoj pripada datoteka) vlasništvo se određuje po pravilima konkretnog UNIX sistema. U svakom slučaju, korisnik se u odnosu na objekat u sistemu datoteka može naći u 3+1 vlasničkoj poziciji: vlasnik (owner), grupa (group), ostatak sveta (others), i vlasnik i grupa istovremeno.

B.26. Odaberite potpun skup komandi koje može da izvrši korisnik pozicioniran nad direktorijumom za koji mu je dato pravo upisa (*write*):

a. cp, rm

b. rm, mv

c. cp, mv, ls

d. rm, mv, >filename, mkdir, ls -l

e. vi (ili drugi editor za kreiranje novih datoteka)

Tačan odgovor: d.

Objašnjenje: *write* pravo dato nad direktorijumom znači da korisnik može da izmeni sadržaj direktorijuma, tj da dodaje nove datoteke i poddirektorijume, menja imena postojećim i briše postojeće datoteke i direktorijume. Odnosno, korisnik može da izvrši rm i obriše datoteku ili direktorijum, mv i promeni ime datoteke ili direktorijuma, >filename i kreira novu datoteku, mkdir i kreira novi direktorijum. Kako je korisnik još i pozicioniran nad tim direktorijumom, to znači da ima pravo i da izvrši ls -l.

B.27. Pravo čitanja (*read*) dato nad direktorijumom znači da korisnik može da izvrši sledeću komandu:

a. cp

b. ls -l

c. ls

d. ls -ld

e. ls -al

f. rmdir

Tačan odgovor: d.

Objašnjenje: *read* pravo znači da korisnik može da pročita sadržaj direktorijuma, tj. da pročita šta piše u *file-info* strukturama koje čine jedan direktorijumski blok. To se radi pomoću komande ls.

Napomena: ls -ld, ls -al i ls -l ne mogu se izvršiti ukoliko nad direktorijumom nije dato pravo izvršavanja (x).

B.28. Komanda za promenu prava pristupa datoteke ili direktorijuma je:

a. chown, a može je izvršiti samo vlasnik ili root ,

b. chown, a može je izvršiti samo vlasnik,

c. chown, a može je izvršiti samo root,

d. chmod, a može je izvršiti samo vlasnik ili root ,

e. chmod, a može je izvršiti samo vlasnik,

f. chmod, a može je izvršiti samo root ,

g. chgrp, a može je izvršiti samo vlasnik ili root ,

h. chgrp, a može je izvršiti samo vlasnik,

i. chgrp, a može je izvršiti samo root.

Tačan odgovor: d.

Objašnjenje: komandom chmod superuser ili vlasnik mogu menjati prava pristupa datoteke ili direktorijuma. Regularan korisnik ne može izmeniti prava pristupa nekog objekta ukoliko nije njegov vlasnik. Za razliku od toga, promena vlasničkih odnosa zahteva root privilegije – vlasničke odnose može promeniti samo *superuser*.

B.29. Izdvojeni su bitni elementi iz datoteka /etc/passwd i /etc/group jednog UNIX sistema. Isečak datoteke /etc/passwd je:

```
fox1:x:500:500::/home/fox1:/bin/bash
student:x:502:502::/home/student:/bin/bash
```

Isečak datoteke /etc/group je:

```
users::100:fox1,student
fox1::500:fox1
```

Na serveru se nalazi direktorijum source:

```
drwxrwxr-x 1 fox1 users 2048 Mar 4 20:57 source
```

Odredite koje akcije korisnik student može izvršiti nad direktorijumom source.

a. student ne može izlistati sadržaj direktorijuma, ne može kreirati poddirektorijume, može preći na direktorijum,

b. student ne može izlistati sadržaj direktorijuma, može kreirati nove datoteke, može preći na direktorijum,

c. student može izlistati sadržaj direktorijuma, može obrisati postojeće datoteke, može preći na direktorijum,

d. student može izlistati sadržaj direktorijuma, ne može obrisati postojeće datoteke, može preći na direktorijum.

Tačan odgovor: c.

Objašnjenje: korisnik student je član grupe users, kojoj pripada i direktorijum source. U odnosu na datoteku source, korisnik student je "grupa", tj. na njega se primenjuje grupno pravo, a to je u ovom slučaju rwx. Dozvole rwx garantuju da student može izlistati sadržaj direktorijuma (ls), obrisati datoteke u njemu (rm) ili preći na direktorijum (cd).

B.30. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# id student7
uid=507(student7) gid=400(vetes) groups=400(vetes),
401(rt), 402(I_grupa)
# ls -ld my_doc
drw-r-xr-x 2 student7 rt 585 Mar 4 20:57 my_doc
```

Pod pretpostavkom da je konkretan UNIX neusklađen sa POSIX-om po pitanju unije vlasničkog i grupnog prava, odredite koje akcije korisnik student7 može izvršiti nad direktorijumom my_doc.

a. student7 može izlistati sadržaj direktorijuma, može kreirati nove datoteke, ne može preći na direktorijum,

b. student7 može izlistati sadržaj direktorijuma, može menjati imena datotekama u direktorijumu, može preći na direktorijum,

c. student7 može izlistati sadržaj direktorijuma, ne može menjati imena datotekama u direktorijumu, može preći na direktorijum,

d. student7 ne može izlistati sadržaj direktorijuma, ne može kreirati nove datoteke, može preći na direktorijum,

e. student7 ne može izlistati sadržaj direktorijuma, može kreirati nove datoteke, ne može preći na direktorijum.

Tačan odgovor: b.

Objašnjenje: korisnik student7 je član grupe rt, kojoj pripada i direktorijum my_doc. U odnosu na direktorijum my_doc, korisnik student7 je "vlasnik" i "grupa", tj. na njega se primenjuje unija vlasničkog i grupnog prava (pretpostavljamo neusklađenost sa POSIX-om), a to je u ovom slučaju

rw+rx=rwx. Dozvole rwx garantuju da student7 može izlistati sadržaj direktorijuma (ls), promeniti imena datoteka u njemu (mv) ili preći na direktorijum (cd).

B.31. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# groups student1
student1: vets rt II_grupa
# ls -l proba
-rw-rw-r-- 1 student4 nrt 585 Mar 4 20:57 proba
```

Odredite koje akcije korisnik student1 može izvršiti nad datotekom proba.

a. student1 može čitati sadržaj datoteke, ne može je obrisati, ne može je izvršavati,

b. student1 može čitati sadržaj datoteke, može joj izmeniti sadržaj, ne može je izvršavati,

c. student1 može čitati sadržaj datoteke, može joj izmeniti sadržaj, može je izvršavati,

d. student1 može čitati sadržaj datoteke, ne može je obrisati, može je izvršavati,

e. student1 može čitati sadržaj datoteke, ne može joj izmeniti sadržaj, ne može je izvršavati,

f. student1 može ne čitati sadržaj datoteke, može je obrisati, može je izvršavati.

Tačan odgovor: e.

Objašnjenje: korisnik student1 nije član grupe nrt, kojoj pripada direktorijum my_doc. U odnosu na direktorijum my_doc, korisnik student7 nije ni "vlasnik" ni "grupa", tj. na njega se primenjuje pravo za ostatak sveta, a to je u ovom slučaju r--. Dozvola r-- garantuje da student1 može pročitati sadržaj datoteke, ali kako nema prava w i x on joj ne može izmeniti sadržaj niti je može izvršavati.

B.32. Izdvojeni su bitni elementi iz datoteka /etc/passwd i /etc/group jednog UNIX sistema. Isečak datoteke /etc/passwd je:

```
fox1:x:500:500::/home/fox1:/bin/bash
student1:x:501:501::/home/student1:/bin/bash
```

```
student2:x:502:502::/home/student2:/bin/bash
student3:x:503:503::/home/student3:/bin/bash
student4:x:504:504::/home/student4:/bin/bash
student5:x:505:505::/home/student5:/bin/bash
```

Isečak datoteke /etc/group je:

```
mem::3:root,mem
users::100:fox1,student2,student5
fox1::500:fox1
studenti::501:student1,student2,student3,student4,s
tudent5
```

Na serveru se nalazi direktorijum my_doc:

```
drwxr-xr-x 2 student4 users 585 Mar 4 20:57 my_doc
```

U direktorijumu my_doc, se nalazi datoteka my_file:

```
-rw-rw-r-- 1 student4 users 585 Mar 4 20:57 my_file
```

Odredite koje akcije korisnik student2 može izvršiti nad datotekom my_file.

a. student2 može pročitati i menjati sadržaj datoteke i može obrisati datoteku,

b. student2 može pročitati sadržaj datoteke i može obrisati datoteku, ali ne može menjati sadržaj datoteke,

c. student2 može pročitati sadržaj datoteke, ne može menjati sadržaj datoteke i ne može obrisati datoteku,

d. student2 može pročitati i menjati sadržaj datoteke, ali ne može obrisati datoteku,

e. student2 ne može pročitati niti menjati sadržaj datoteke, ali može obrisati datoteku.

Tačan odgovor: d.

Objašnjenje: korisnik student2 je član grupe users, kojoj pripadaju direktorijum my_doc i datoteka my_file koja se u tom direktorijumu nalazi. U odnosu na direktorijum my_doc, korisnik student2 je "grupa", tj. na njega se primenjuje grupno pravo, a to je u ovom slučaju r-x. Dozvole r-x garantuju da student2 može pročitati sadržaj direktorijuma i da se može pozicionirati na direktorijum, ali ne može menjati sadržaj direktorijuma (na primer, ne može obrisati datoteku my_file koja se u njemu nalazi). U odnosu na datoteku my_file student2 je

grupa i na njega se primenjuje grupno pravo rw-, što će reći da može da pročita i promeni sadržaj te datoteke.

B.33. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# id student5
uid=505(student5) gid=400(vetes) groups=400(vetes),
401(rt), 402(I_grupa)
# ls -ld my_doc
drwxrwx--- 2 student4 rt 585 Mar 4 20:57 my_doc
# ls -l my_doc/my_file
-rw-r----- 1 student4 vetes 585 Mar 4 20:57 my_file
```

Pod pretpostavkom da se korisnik student5 nalazi na direktorijumu my_doc, odrediti koje od sledećih komandi taj korisnik može da izvrši:

1. >new, 2. mv my_file /tmp, 3. rm my_file, 4. ls -l my_file, 5. vi my_file

a. 1. da 2. ne 3. da 4.da 5. da

b. 1. da 2. da 3. ne 4.da 5. da

c. 1. da 2. da 3. da 4.ne 5. da

d. 1. ne 2. da 3. da 4.da 5. da

e. 1. da 2. da 3. da 4.da 5. ne

f. 1. ne 2. da 3. ne 4.da 5. da

g. 1. ne 2. ne 3. ne 4.da 5. da

Tačan odgovor: e.

Objašnjenje: korisnik student5 je član grupa rt i vetes, kojoj pripadaju i direktorijum my_doc i datoteka my_file, respektivno. Korisnik student5 je u odnosu na direktorijum my_doc "grupa", tj. na njega se primenjuje grupno pravo, a to je u ovom slučaju rwx (korisnik student5 ima sva prava nad tim direktorijumom). U odnosu na datoteku my_file, korisnik student5 je grupa i na njega se primenjuje grupno pravo r--, što će reći da može samo da pročita sadržaj te datoteke. Od navedenih komandi, student5 može da izvrši sledeće:

- `>new` (kreiranje nove datoteke, zahteva w pravo nad direktorijumom),

- `mv my_file /tmp` (zahteva r nad my_doc, w nad my_doc, w nad /tmp, r nad my_file, što je sve dato),
- `rm my_file` (zahteva w nad my_dir),
- `ls -l my_file` (zahteva r i x nad my_dir).

Korisnik student5 ne može vi editorom da izmeni sadržaj datoteke my_file, jer u odnosu na nju nema pravo upisa.

B.34. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# id student3
uid=503(student3) gid=400(vetes) groups=400(vetes),
401(rt), 402(I_grupa)
# ls -l my_doc
drwxrwxrwt 2 root  root 585 Mar 4 20:57 my_doc
# ls -l my_doc/my_file
-rw-r-----  1 student3 rt 585 Mar 4 20:57 my_file
```

Pod pretpostavkom da server na kome su ove komande izvršene ne prati POSIX specifikaciju po pitanju unije vlasničkog i grupnog prava, odredite koje akcije korisnik student3 može izvršiti nad datotekom my_file.

a. student2 može pročitati i menjati sadržaj datoteke i može obrisati datoteku,

b. student2 može pročitati sadržaj datoteke i može obrisati datoteku, ali ne može menjati sadržaj datoteke,

c. student2 može pročitati sadržaj datoteke, ne može menjati sadržaj datoteke i ne može obrisati datoteku,

d. student2 može pročitati i menjati sadržaj datoteke, ali ne može obrisati datoteku,

e. student2 ne može pročitati niti menjati sadržaj datoteke, ali može obrisati datoteku.

Tačan odgovor: a.

Objašnjenje: korisnik student3 je član grupe rt kojoj pripada i datoteka my_file. Korisnik student3 je u odnosu na direktorijum my_doc "ostatak sveta", tj. na njega se primenjuje pravo rwt, odnosno rwx sa *sticky* bitom. Korisnik može da čita i menja sadržaj direktorijuma my_doc (uz ograničenje da može da briše

samo svoje datoteke koje nameće *sticky* bit), kao i da se pozicionira na taj direktorijum. U odnosu na datoteku my_file, korisnik student3 je vlasnik i grupa i na njega se primenjuje unija vlasničkog i grupnog prava rw + r = rw- (pod predpostavkom da konkretan UNIX ne prati POSIX specifikacije), što će reći da može da pročita i izmeni sadržaj te datoteke. Korisnik student3 može pročitati i izmeniti sadržaj datoteke my_file, a može je i izbrisati jer je njen vlasnik.

Napomena: u slučaju da je datoteka dodeljena nekom drugom korisniku, student3 ne bi mogao da je obriše, bez obzira na pravo x koje ima nad direktorijumom my_doc.

B.35. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# id student2 <enter>
uid=502(student2) gid=400(vetes) groups=400(vetes),
401(rt), 402(I_grupa)
# ls -l my_file
-rw----rw- 1 student4 vetes 585 Mar 4 20:57 my_file
```

Koje od sledećih komandi superuser root može uspešno da izvrši ?

1. `chown student2 my_file`
2. `chgrp rt my_file`
3. `chmod u=,g+rx,o+x my_file`

Oredite ispravan odziv sistema.

a. 1. Permission denied, 2. Permision denied, 3. AR=--w-r-xrwx

b. 1. owner=student2, 2. group=rt, 3. AR=----rwxrwx

c. 1. owner=student2, 2. Permision denied, 3. AR=----r--rwx

d. 1. Permission denied, 2. Permision denied, 3. AR=----r-xr-x

e. 1. owner=student2, 2. group=rt, 3. AR=----r-xrwx

f. 1. owner=student2, 2. group=rt, 3. AR=-r--r-xrwx

Tačan odgovor: e.

Objašnjenje: samo vlasnik ili root mogu da promene pristupna prava datoteke, a samo root može da promeni vlasničke odnose. Pošto root izvršava komande, sve tri će biti uspešno izvršene, a posle toga će:

- novi vlasnik biti student2,

- nova grupa biti rt,
- nova prava pristupa biti 057 (---r-xrwx).

B.36. Data je datoteka sa pristupnim pravima -rw-r-xrwx. Ukoliko kopiranje datoteke izvršava korisnik za kog je vrednost promenljive umask=521, kopija datoteke će imati sledeća pristupna prava:

a. -rwxrwxrwx

b. -r----xr--

c. --w-r-xrw-

d. -rw-r-xrw-

e. --w-r-xrwx

f. --w-rwxrw-

g. --wxr-xrw-

Tačan odgovor: c.

Objašnjenje:
```
datoteka = 657 = rw-r-xrwx
umask    = 521 = r-x-w---x
kopija   = 156 = --xr-xrw-
```

B.37. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# umask
200
# ls -l file1
-rwxrwxrwx 1 root root ... 1235 file1
```

Korisnik student1 (primarna grupa rt2001) zadaje sledeće dve komande jednu za drugom.

```
# cp file1 /tmp/file2
# ls -l /tmp/file2
```

Rezultat izvršenja druge komande je:

a. `-rwxr-xr-x 1 root root ... 1235 file2`

b. `-r-xrwxrwx 1 student1 rt2001 ... 1235 file2`

c. `-rwxr-xr-x 1 root rt2001 ... 1235 file2`

d. `-rwxr--r-- 1 student1 root ... 1235 file2`

e. `-rwxr--r-- 1 student1 rt2001 ... 1235 file2`

f. `-rwxr-xr-x 1 student1 rt2001 ... 1235 file2`

g. `-rwxrwxr-x 1 student1 rt2001 ... 1235 file2`

h. `-r-xrwxrwx 1 student1 root ...  1235 file2`

Tačan odgovor: b.

Objašnjenje: korisnik student1 ima pravo čitanja nad datotekom file1 i pravo upisa nad direktorijumom /tmp (podrazumevana prava data nakon instalacije za /tmp su rwxrwxrwt).

```
datoteka = 777 = rwxrwxrwx
umask    = 200 = -w-------
kopija   = 577 = r-xrwxrwx
```

Vlasnik nove datoteke (file2) je korisnik student1, jer je on inicirao kopiranje. Iz istih razloga se datoteka file2 dodeljuje njegovoj primarnoj grupi (rt2001).

B.38. Za hard i simboličke linkove i sisteme datoteka na kojima se nalaze originalne datoteke (odnosno datoteke nad kojima će biti kreiran link) važi:

a. I hard i simbolički linkovi se mogu kreirati isključivo na istim sistemima datoteka na kojima se nalazi originalna datoteka,

b. Hard linkovi se mogu kreirati isključivo na fizički istim sistemima datoteka, a simbolički linkovi na istim ili različitim,

c. Hard linkovi se mogu kreirati isključivo na istim sistemima datoteka, a simbolički linkovi na različitim,

d. I hard i simbolički linkovi se mogu kreirati na fizički istim sistemima datoteka na kojima se nalazi originalna datoteka ili na drugim sistemima datoteka.

Tačan odgovor: b.

Objašnjenje: hard link je *file-info* struktura (jedan zapis u direktorijumu) koja ukazuje na i-node originalne datoteke. Pošto link i original imaju isti i-node, moraju se nalaziti na fizički istom sistemu datoteka (hard link se ne sme nalaziti na drugoj particiji ili na drugom disku). Ne mogu se linkovati datoteke sa mrežnog sistema datoteka (NFS), direktorijumi ili nepostojeće datoteke. Simbolički linkovi su objekti sistema datoteka koji ukazuju na druge objekte

(prečice). Svaki simbolički link koristi poseban i-node i jedan blok podataka u sistemu datoteka. Simbolički linkovi se mogu kreirati nalaziti na fizički istom ili različitom sistemu datoteka, odnosno na istoj ili drugoj particiji (disku). Takođe, mogu se linkovati datoteke sa mrežnog sistema datoteka (NFS), direktorijumi, kao i nepostojeće datoteke.

B.39. Na jednom UNIX sistemu su izvršene sledeće komande:

```
# ls -l admin admin1
-rwxrwxr-x 1 root root ... 15834 admin
lrwxrwxrwx 1 student1 rt2001 ... 12 admin1
# umask
011
# su student2
$ cp  admin1  admin2 <enter>
$ ls -l admin2
```

Rezultat izvršenja poslednje komande je:

a. admin2 nije formirana, student1 nema prava da obavi cp

b. `-rwxr-xr-x 1 root root  ... 15834 admin2`

c. `-rwxr-xr-x 1 root root  ... 12 admin2`

d. `-rwxr-xr-x 1 student1 rt2001 ... 15834 admin2`

e. `-rwxr-xr-x 1 student1 rt2001 ... 12 admin2`

f. `lrwxrwxrwx 1 student1 rt2001 ... 12 admin2`

g. `-rwxrw-r-- 1 student1 rt2001 ... 15834 admin2`

h. `-rwxrwxr-x 1 student1 rt2001 ... 15834 admin2`

Tačan odgovor je g.

Objašnjenje: svaka akcija izvršena nad linkom (osim brisanja) odnosi se na originalnu datoteku. Komanda cp će kreirati kopiju originalne datoteke (admin), vlasnik će biti korisnik koji je inicirao kopiranje (student1), grupa je primarna grupa tog korisnika (rt2001), a prava pristupa su pristupna prava originalne datoteke umanjena za prava specificirana promenljivom umask (775-011=764).

B.40. Odredite minimalni skup prava potreban za kopiranje datoteke:

a. r i w nad datotekom,

b. w nad datotekom, r nad odredišnim direktorijumom,

c. r nad datotekom, x nad odredišnim direktorijumom,

d. r nad datotekom, w nad odredišnim direktorijumom, r nad izvorišnim direktorijumom,

e. r nad datotekom, w nad odredišnim direktorijumom,

f. r nad datotekom.

Tačan odgovor: e.

Da biste iskopirali datoteku, neophodna su prava koja vam dozvoljavaju da:

- pročitate sadržaj datoteke (read nad originalnom datotekom),
- izmenite sadržaj odredišnog direktorijuma (write nad odredišnim direktorijumom).

B.41. Komanda mv a* /tmp/b* obavlja sledeću aktivnost:

a. promenu imena jedne datoteke,

b. promenu imena i pomeranje jedne datoteke u drugi direktorijum,

c. pomeranje grupe datoteka u drugi direktorijum,

d. pomeranje podstabla,

e. ne obavlja ništa, komanda je neispravna.

Tačan odgovor: e.

Nonsens: a.

Objašnjenje: komanda mv ne može se koristite za pomeranje grupe datoteka sa promenom imena (slično važi i za kopiranje datoteka komandom cp).

```
tulip:~# mv a* /tmp/a*
mv: when moving multiple files, last argument must be a
directory
Try `mv --help' for more information.
```

B.42. Odaberite potpun skup prava potreban za izvršavanje sledeće komande:

```
mv /sdir/file /ddir:
```

a. r (file), r (/sdir), w (/ddir), (/sdir), x (/ddir)

b. r (file), x (/sdir), w (/ddir), (/sdir), r (/ddir)

c. r (file), w (/sdir), w (/ddir), r (/sdir), x (/ddir)

d. r (file), r (/sdir), w (/sdir), (/sdir), x (/ddir)

e. r (file), w (/sdir), w (/ddir), (/sdir), x (/ddir)

f. w (file), w (/sdir), w (/ddir), (/sdir), x (/ddir)

g. x (file), w (/sdir), w (/ddir), (/sdir), x (/ddir)

h. r (file), w (/sdir), r (/ddir), (/sdir), x (/ddir)

Tačan odgovor: e.

Objašnjenje: da biste pomerili datoteku, neophodna su prava koja vam dozvoljavaju da:

- pročitate sadržaj datoteke (read nad originalnom datotekom),
- izmenite sadržaj originalnog direktorijuma (write nad originalnim direktorijumom),
- izmenite sadržaj odredišnog direktorijuma (write nad odredišnim direktorijumom).

Ovo je minimalni set uslova – potpuni skup parava formira se proširivanjem minimalnog seta uslova dozvolama za pozicioniranje nad odredišnim i izvorišnim direktorijumom (x nad odredišnim i izvorišnim direktorijumom).

B.43. Na jednom UNIX sistemu izvršene su sledeće komande.

```
# ls -ld my_dir
drwxrwxrwt 5 root root ... 4096 my_dir
# cd my_dir
# ls -l my_file
-rwxr-xr-x 1 student1 root ... 1024 my_file
# su student1
$ rm my_file
```

Odredite da li korisnik student1 može obrisati datoteku my_file.

a. student1 može obrisati datoteku my_file,

b. student1 ne može obrisati datoteku my_file,

c. student1 može obrisati datoteku my_file pod uslovom da je član grupe root,

d. nema dovoljno podataka da se odredi da li student1 može obrisati datoteku my_file.

Tačan odgovor: a.

Objašnjenje: korisnik student1 je vlasnik datoteke my_file koja se nalazi na direktorijumu my_doc za koji je postavljen sticky bit. To znači da student1 ima pravo da obriše datoteku my_file.

B.44. Korisnik student1 je zadao sledeću komandu:

find / -user student1 -print -exec chmod 666 {} \;

Komanda find, koju zadaje korisnik student1 u ovom obliku, traži:

a. sve datoteke kojima je vlasnik student1 ali ne može da im menja prava pristupa,

b. sve datoteke kojima je vlasnik student1 i menja im prava pristupa,

c. sve datoteke koje se zovu student1 i menja im prava pristupa,

d. sve datoteke koje se zovu student1 ali ne može da im menja prava pristupa.

Tačan odgovor: b.

Obaveštenje: ova komanda traži sve datoteke u aktivnom stablu (početni direktorijum pretrage je /) koje pripadaju korisniku student1 i nad njima izvršava akciju izmene prava pristupa (prava pristupa se mogu zameniti jer komandu izvršava student1, koji je vlasnik tih datoteka).

B.45. Komanda rmdir briše direktorijum:

a. u svakom slucaju,

b. samo ako je direktorijum prazan,

c. ako je direktorujum potpuno prazan ili sadrži poddirektorijume, ali ne i datoteke,

d. samo ako je direktorijum prazan ili je nad njim postavljen SUID bit.

Tačan odgovor: b.

Objašnjenje: komanda rm briše direktorijum samo ako je prazan.

```
tulip:~# rmdir testing
rmdir: `testing': Directory not empty
```

Direktorijum koji nije prazan briše se komandom:

```
# rm -r -f directory
```

B.46. Postupak administriranja novog *floppy* diska, u formi sistema datoteka na UNIX sistemu obavlja se sledećim komandama (komande su navedene redom kojim se zadaju):

a. mount -> fdisk -> mkfs

b. fcsk -> mount -> cd

c. fdformat -> format -> mkfs -> mount

d. fdformat -> mkfs -> mount

e. fdisk -> mount -> mkfs

f. fdisk -> mkfs -> mount

Tačan odgovor: d.

Objašnjenje: prvo se obavi *low-level* formatiranje diskete, zatim se na disketi kreira fajl sistem i na kraju se disketa montira na aktivno UNIX stablo.

```
# fdformat -v /dev/fd0H1440
# mkfs -t ext2 -c /dev/fd0H1440
# mount /dev/fd0 /floppy
```

B.47. Data je datoteka a.a:

```
-rw-r--r-- 1 root root 18 Nov 27 09:16 a.a
```

Data je tar arhiva na disketi koja sadrži identičnu datoteku a.a:

```
-rw-r--r-- root/root 18 2002-11-27 09:16 a.a
```

Nakon izvršenja komande `tar cvf /dev/fd0 a.a` dogodiće se sledeće:

a. datoteka u arhivi nije starija, nema nikakvog transfera,

b. datoteka u arhivi je starija, doći će do transfera, a nova datoteka a.a se smešta na početak arhive,

c. datoteka u arhivi je starija, doći će do transfera, a nova datoteka a.a se u arhivu smešta preko postojeće,

d. datoteka u arhivi je starija, doći će do transfera, a nova datoteka a.a se smešta na kraj arhive,

e. stara arhiva se uništava i kreira se nova sa jednom datotekom a.a,

f. komanda samo proverava da u arhivi postoji datoteka a.a bez transfera.

Tačan odgovor: e.

Objašnjenje: tar arhiveru zadat je parametar c (create) tako da se postojeća arhiva na disketi uništava i kreira nova u koju se smešta datoteka a.a.

B.48. Postavljeni zahtev za stampu koji zbog greske u stampacu ne moze da se štampa:

a. ostaje uvek u redu za štampu, a može ga obrisati samo onaj korisnik koji je zahtev poslao na štampu,

b. ostaje uvek u redu za štampu, a može ga obrisati ili root ili korisnik koji je zahtev poslao na štampu,

c. ostaje uvek u redu za štampu, a može ga obrisati sistemski korisnik lp,

d. zahtev se briše sam ako štampač otkaže.

Tačan odgovor: b.

Objašnjenje: svaki korisnik može da ukloni svoje poslove iz reda, ali ne i poslove drugih korisnika, dok superuser može da ukloni bilo koji posao iz reda. Posao ostaje u redu dok ga vlasnik ili root iz reda eksplicitno ne uklone. Poslovi se brišu iz reda komandama lprm (System V) i cancel (BSD). Pre brisanja, posla potrebno je odrediti broj posla pomoću komande lpstat.

B.49. Na jednom UNIX sistemu izvršene su sledeće komande:

```
# ls -l dir1/a.a dir2/a.a
-rw-r--r-- 1 root root 18 Nov 27 09:30 dir1/a.a
-rw-r--r-- 1 root root 18 Nov 27 09:30 dir2/a.a
# cd dir1
# find . -print |cpio -pduvm <dir2>
```

Rezultat izvršenja poslednje zadate komande je:

a. nema nikakvog transfera

b. doći će do transfera, bez čuvanja postojeće datoteke

c. doći će do transfera, ali će se prethodno formirati rezervna kopija postojeće datoteke

d. komanda ima pogresnu sintaksu

Tačan odgovor: b.

Nonsens: d.

Objašnjenje: program cpio u ovom slučaju radi u *copy-pass* režimu i zbog flega u (*unconditional*) obavlja bezuslovni transfer.

B.50. Process getty:

a. proverava par korisničko ime i lozinka sa podacima u datotekama /etc/passwd i /etc/shadow

b. čeka na korisnika da unese korisničko ime i poziva proces login

c. prihvata korisničke komande sa terminala

d. prihvata zahteve za štampu sa terminala

Tačan odgovor: b.

Nonsens: d.

Objašnjenje: najpre proces init pokreće po jednu instancu procesa getty za svaki terminal sa kog je korisnicima dozvoljeno prijavljivanje na sistem. Proces getty na ekranu štampa poruku koja je upisana u datoteku /etc/issue, a zatim osluškuje terminal i čeka da korisnik unese korisničko ime. Proces getty, zatim, pokreće login proces kome kao parametar prenosi korisničko ime. Proces login zahteva od korisnika da unese lozinku, a zatim proverava par korisničko ime - lozinka sa podacima u datoteci /etc/passwd i /etc/shadow. Ukoliko je lozinka validna, login pokreće komandni interpreter koji je za datog korisnika naveden u /etc/passwd, a u suprotnom prekida izvršenje, posle čega init pokreće novu instancu getty procesa. Kada korisnik završi rad i odjavi se sa sistema, komandni interpreter prekida izvršenje i vraća kontrolu procesu init.

C. Primer testa sa drugog dela ispita

U ovom dodatku je dat primer testa za polaganje drugog dela ispita. Drugi deo ispita obuhvata zadatke iz sledećih oblasti:

- raspoređivanje procesa,
- komunikacija između procesa - sinhronizacioni mehanizmi,
- zastoj,
- upravljanje memorijom,
- virtuelna memorija,
- sistemi datoteka i
- algoritmi za raspoređivanje disk zahteva i RAID sistemi.

C.1. Raspoređivanje procesa

Na sistemu postoji pet procesa (P1, P2, P3, P4, P5) koji su u potpunosti CPU-*bound* (ne koriste ulazno izlazne uređaje). Vremena nailaska u sistem (*arrival time*) i vremena izvršavanja na procesoru (*burst time*) data su u milisekundama. Prioritet 5 je najviši, a prioritet 1 najniži.

Proces	Prioritet	Vreme nailaska	Vreme izvršavanja
P1	2	100	40
P2	5	110	20
P3	1	130	50
P4	4	160	10
P5	3	200	30

Vreme potrebno za zamenu konteksta procesa može se zanemariti. Odredite vremena potrebna za kompletiranje procesa (*turnaround time*) ako se procesor dodeljuje na osnovu sledećih algoritama:

a. Round Robin sa vremenskim kvantumom Q = 10.

b. Raspoređivanje na osnovu prioriteta sa pretpražnjenjem

c. FCFS

d. Shortest Job First (bez pretpražnjenja)

e. SRTF (shortest remaining time first), odnosno SJF sa pretpražnjenjem.

(a) Round Robin:

Preostalo vreme potrebno za izvršavanje procesa:

Vreme	P1	P2	P3	P4	P5
100	40				
110	30	20			
130	20	10	50		
160	10	0	40	10	
190	0	0	30	0	
200	0	0	20	0	30
240	0	0	0	0	10
250	0	0	0	0	0

Vremena potrebna za kompletiranje procesa:

P1: 190, P2: 160, P3: 240, P4: 190, P5: 250

(b) Raspoređivanje na osnovu prioriteta sa pretpražnjenjem

Preostalo vreme potrebno za izvršavanje procesa:

Vreme	P1	P2	P3	P4	P5
100	40				
110	30	20			
130	30	0	50		
160	0	0	50	10	
170	0	0	50	0	
200	0	0	20	0	30
230	0	0	20	0	0
250	0	0	0	0	0

Vremena potrebna za kompletiranje procesa:

P1: 160, P2: 130, P3: 250, P4: 170, P5: 230

(c) FCFS:

Vremena potrebna za kompletiranje procesa:

P1: 140, P2: 160, P3: 210, P4: 220, P5: 250

(d) SJF bez pretpražnjenja:

Vremena potrebna za kompletiranje procesa:

P1: 140, P2: 160, P3: 220, P4: 170, P5: 250

(e) SRTF sa pretpražnjenjem:

Vremena potrebna za kompletiranje procesa:

P1: 160, P2: 130, P3: 220, P4: 170, P5: 250

C.2. Komunikacija između procesa

a. Koja je posledica moguća ukoliko procesi nesinhronizovano pristupaju zajedničkim podacima? Navedite primer koji ilustruje nekonzistentnost deljene promenljive u slučaju da sinhronizacioni mehanizmi međusobnog nisu pravilno implementirani.

b. Ukratko opišite razliku između primene semafora i monitora u sinhronizaciji izvršenja procesa.

c. Objasnite na konkretnom primeru kako neispravna upotreba semafora može dovesti do zastoja.

d. Proširite osnovnu definiciju semafora tako da se pri korišćenju izbegne mogućnost da proces bude zauzet čekanjem.

(a) t = 100 msec: Proces P1 upisuje vrednost u memorijsku lokaciju A. Proces P2 treba posle nekog vremena ovu vrednost da uveća i upiše na istu lokaciju. Proces P1 proverava vrednost lokacije A na svakih 100 msec. Ukoliko je vrednost uvećana, proces P1 je preuzima i nastavlja dalje sa radom.

t = 110 msec: Proces P3 prepisuje vrednost lokacije A ne znajući da to ne treba da uradi (programer nije zaštitio deljenu promenljivu semaforom ili nekim drugim sinhronizacionim mehanizmom)

t = 200 msec: Proces P2 preuzima neispravnu vrednost promenljive.

(b) Monitori su sinhronizacioni mehanizmi višeg nivoa od semafora. Korisnici nemaju direktan pristup promenljivima u monitoru niti vide interne procedure monitora.

(c) void invalidsync {wait (mutex); do something; wait (mutex);}

(d) Semafor treba definisati kao strukturu koju čine vrednost semafora (koja može biti i negativna) i semaforski red (lista pokazivača na procese koji čekaju na semaforu). Proces koji izvršava *wait* operaciju nad semaforom čija je vrednost 0 se blokira i prevodi u semaforski red. Proces oslobađa procesor, koji se predaje nekom drugom procesu koji nije blokiran.

C.3. Zastoj

a. Posmatrajte sledeći segment koda iz aplikacije orijentisane ka bankarstvu, koji prenosi specificirani iznos sa izvorišnog računa na odredišni. Prilikom prenosa, aplikacija zaključava izvorišni i odredišni račun (koristeći neki sinhronizacioni mehanizam takav da su operacije zaključavanja i otpuštanja resursa nedeljive) i menja stanja oba računa. Svi računi su numerisani. Pretpostavite da veći broj procesa može izvršavati ovaj kod simultano.

```
void transfer (  double balance[], int source,
                 int destination, double amount)
{
   if (source < destination) {
        lock(source); lock (destination);
   }
   else {
        lock (destination); lock (source);
   }
   balance [source] -= amount;
   balance [destination] += amount;
   unlock (source); unlock (destination);
}
```

Da li aplikacija može izazvati zastoj deadlock ? Ako ne može, objasnite zašto. Ukoliko može, navedite sekvencu događaja koji mogu izazvati zastoj.

Aplikacija ne može izazavati zastoj. Resursi, odnosno računi, su numerisani, procesi uvek prvo zaključavaju resurs sa manjim rednim brojem od resursa sa većim rednim brojem. To znači da je razbijena mogućnost kružnog čekanja (circular-wait), odnosno da do zastoja ne može doći.

C.4. Zastoj

Na jednom hipotetičkom serveru nalazi se 6 uređaja za rad sa trakama, 3 plotera, 4 štampača i 2 CD-ROM uređaja. Trenutno stanje sistema opisano je sledećom tabelom:

	Traka	Ploter	Štampač	CD
P1	3, 1	0, 1	1, 0	1, 0
P2	0, 0	1, 1	1, 0	0, 2
P3	1, 3	1, 1	1, 0	0, 0
P4	1, 0	1, 0	0, 1	1, 0
P5	0, 2	0, 1	0, 1	0, 0

gde prva cifra u svakom paru označava broj instanci resursa koji proces trenutno koristi, a druga broj dodatnih instanci tog resursa koje su potrebne procesu. Na primer, proces P1 trenutno koristi 3 instance magnetnih traka a zahteva još jednu.

a. Pod pretpostavkom da svaki proces traži maksimum resursa, navedite sekvencu koja će dovesti do izvršenja svih procesa.

b. Navedite sekvencu koja će dovesti do zastoja.

(a) Slobodno: 1 traka, 0 plotera, 1 printe, 0 CD

Izvrši proces P4.

Slobodno: 2 trake, 1 ploter, 1 printer, 1 CD

Izvrši proces P1.

Slobodno: 5 traka, 1 ploter, 2 printera, 2 CD

Izvrši proces P2.

Slobodno: 5 traka, 2 plotera, 3 printera, 2 CD

Izvrši proces P3.

Slobodno: 6 traka, 3 plotera, 4 printera, 2 CD

Izvrši proces P5.

(b) Slobodno: 1 traka, 0 plotera, 1 printer, 0 CD

Otpočni izvršenje proces P5 do tačke dok ne zahteva još jednu traku i jedan štampač. Svi procesi žele još resursa koji nisu slobodni - sistem je u zastoju.

C.5. Upravljanje memorijom

Pretpostavite da u memoriji postoje redom sledeće šupljine: 20K, 30K, 17K. Koristeći *best-fit* algoritam, zahtevi za dodelom memorije

koji stižu u sledećem mogu se ispuniti: 17K, 20K, 20K. Međutim, *first-fit* i *worst-fit* algoritmi ne dovode do ispunjenja svih zahteva.

a. Navedite niz zahteva za dodelom memorije koji se mogu opslužiti korišćenjem *first-fit* algoritma, ali ne i korišćenjem *best-fit* ili *worst-fit* algoritama za dodelu memorije

b. Navedite niz zahteva za dodelom memorije koji se mogu opslužiti korišćenjem *worst-fit* algoritma, ali ne i korišćenjem *best-fit* ili *first-fit* algoritama za dodelu memorije

(a) 10K, 10K, 30K, 17K

(b) 10K, 20K, 20K, 17K

C.6. Upravljanje memorijom

Posmatrajte tabelu segmenata jednog procesa i tabelu stranica jednog procesa na sistemu sa stranicama veličine 2K

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Page	Frame
0	1
1	4
2	3
3	7
4	12

a. Odredite fizičke adrese koje odgovaraju sledećim logičkim adresama datim pomoću parova <segment, offset>:

<0,430> <1,10> <2,500> <3,400> <4,112>

Ponuđeni odgovori:

1. 649 2310 1727 illegal illegal
2. illegal 2310 649 1727 illegal
3. 649 2310 illegal 1727 illegal
4. 1727 2310 649 illegal illegal

b. Odredite fizičke adrese koje odgovaraju sledećim logičkim adresama:

251 3129 10000 800 0

Ponuđeni odgovori:

1.	2229	9273	25657	2048	2848
2.	2229	9273	25657	2848	2048
3.	2229	9273	illegal	2848	2048
4.	2229	illegal	25657	illegal	2048

(a) Tačan odgovor: 3.

(b) Tačan odgovor: 2.

C.7. Virtuelna memorija

Dat je sistem virtuelne memorije koji se sastoji od šest fizičkih okvira. Svi okviri su u početnom trenutku prazni. Dat je sledeći niz od 20 referenci za korišćenjem stranica koje se pojavljuju sledećim redom:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6.

a. Koliko PF prekida nastaje ukoliko se zamena stranica vrši po FIFO algoritmu ?

b. Koliko PF prekida nastaje ukoliko se zamena stranica vrši po LRU algoritmu ?

(a) 10.

(b) 7.

C.7. Sistemi datoteka

a. Prilikom provere konzistentnosti sistema datoteka izgrađena je sledeća tabela zauzetosti blokova u području podataka:

Blok	0	1	2	3	4	5	6	7
Zauzet	1	1	0	0	2	1	0	1
Slobodan	0	1	1	0	0	0	1	0

Šta je ovde nekonzistentno? Objasnite kako se svaka od ovih nekonzistentonsti može ispraviti?

b. Evidencija o slobodnim blokovima može se voditi pomoću mape bitova ili liste slobodnih blokova. Za prazan disk, kapaciteta 1GB, sa veličinom bloka 1KB, odredite veličinu mape bitova za opis slobodnih blokova.

c. Za prazan disk, kapaciteta 1GB, sa veličinom bloka 1KB, odrediti veličinu linearne liste za opis slobodnih blokova, pod uslovom da se brojevi blokova opisuju najmanjim mogućim brojem bita.

d. Za prazan disk, kapaciteta 1GB, sa veličinom bloka 1KB, odrediti veličinu linearne liste za opis slobodnih blokova, pod uslovom da se brojevi blokova opisuju sa 32 bita.

(a) Blok 1 je i zauzet i slobodan. Blok 1 se uklanja iz liste slobodnih blokova.

Blok 3 nije ni zauzet ni slobodan. Blok 3 se dodaje u listu slobodnih blokova.

Postoji nekonzistentnost vezana za blok 4 (prema tabeli, blok 4 je dvostruko zauzet). Podaci sa bloka 4 moraju se pomeriti u prazan blok, a kontrolni blok jedne od datoteka koja ukazuje na blok 4 mora biti ažuriran pokazivačem na taj blok.

(b) broj bitova = broj slobodnih blokova,

broj slobodnih blokova = kapacitet diska / veličina bloka,

broj slobodnih blokova = 1GB / 1KB = 1.048.576,

broj bitova = broj slobodnih blokova = 1.048.576,

broj bajtova = broj bitova / 8 = 131.072,

Veličina mape bitova je 131.072B, odnosno 128KB.

(c) Lista je realizovana u vidu povezanih blokova koji sadrže po n pokazivača na slobodne blokove diska.

broj slobodnih blokova = kapacitet diska / veličina bloka,

broj slobodnih blokova = 1GB / 1KB = 1.048.576,

broj bitova za opis broja bloka = $\log_2$ (broj blokova) = 20 bitova,

opisanih blokova u bloku diska = veličina bloka (u bitima) / broj bitova za opis broja bloka,

opisanih blokova u bloku diska = 1024B · 8 / 20 = 409,6 = 409 (ne mogu se iskoristiti poslednjih 12 bita),

Na kraju bloka mora se naći ukazivač na sledeći blok koji opisuje slobodne blokove diska.

To znači da jedan blok diska opisuje 408 slobodnih blokova.

Veličina liste slobodnih blokova = broj slobodnih blokova / broj slobodnih blokova koje opisuje jedan blok diska

Veličina liste slobodnih blokova = 1.048.576/408 = 2.570,04

Veličina liste slobodnih blokova je 2.571 blokova odnosno 2.571KB.

(d) Lista je realizovana u vidu povezanih blokova koji sadrže po n pokazivača na slobodne blokove diska.

broj bitova za opis broja bloka = 32 bita,

opisanih blokova u bloku diska = veličina bloka (u bitima) / broj bita za opis broja bloka,

opisanih blokova u bloku diska = 1024B · 8 / 32 = 256.

Na kraju bloka mora se naći ukazivač na sledeći blok koji opisuje slobodne blokove diska.

To znači da jedan blok diska opisuje 255 slobodnih blokova.

Veličina liste slobodnih blokova = broj slobodnih blokova / broj slobodnih blokova koje opisuje jedan blok diska.

Veličina liste slobodnih blokova = 1.048.576/255 = 4.112,06

Veličina liste slobodnih blokova je 4.113 blokova odnosno 4.113KB.

C.8. Raspoređivanje disk zahteva

Posmatrajte disk sa 100 cilindara označenih brojevima od 0-99. Glave za čitanje i pisanje se trenutno nalaze cilindru 19, a prethodno su bile pozicionirane na cilindru 18. Trenutni raspored zahteva u redu za rad sa diskom je: 77, 53, 80, 10, 15, 5, 16, 67. Odredite redosled opsluživanja disk zahteva ukoliko se za raspoređivanje zahteva koriste sledeći algoritmi:

a. First Come, First Served (FIFO)

b. Shortest Seek Time First

c. LOOK

d. C-LOOK (koji opslužuje cilindre u rastućem redosledu)

(a) 77, 53, 80, 10, 15, 5, 16, 67

(b) 16, 15, 10, 5, 53, 67, 77, 80

(c) 53, 67, 77, 80, 16, 15, 10, 5

(d) 53, 67, 77, 80, 5, 10, 15, 16

Literatura

[1] M. J. Bach, *The Design of the UNIX Operating System*, Prentice Hall (1987).

[2] Kvaternik, R.: *Uvod u operativne sisteme, četvrto izdanje*, Informator, Zagreb (1988).

[3] A. M. Lister, R. D. Eager, *Fundamentals of Operating Systems, Fifth Edition*, The Macmillan Press Ltd (1993).

[4] A. S. Tannenbaum, A. S. Woodhull, *Operating System Design and Implementation*, Second Edition, Prentice Hall (1997).

[5] Stallings, W.: *Operating Systems, Fourth Edition*, Prentice Hall (2000).

[6] A. Danesh, *Red Hat Linux*, Mikro knjiga (2000).

[7] A. S. Tannenbaum, *Modern Operating Systems*, Prentice Hall (2001).

[8] G. Mourani, *Securing and Optimizing Linux: The Ultimate Solution*, Open Network Architecture Inc. (2001).

[9] W. Stanfield, R. D. Smith, *Linux System Administration, Second Edition*, Sybex (2001).

[10] T. Collings, K. Wall, *Red Hat Linux Networking & System Administration*, Hungry Minds Inc. (2002)

[11] A. Silberschatz, P. B. Galvin, G. Gagne, *Operating System Concepts, Sixth Edition (Windows XP Update)*, John Wiley & Sons, Inc (2003).

[12] Obradović, S.: *Osnovi računarske tehnike i programiranja, četvrto izdanje*, Viša elektrotehnička škola, Beograd (2003).

[13] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: UNIX i Linux*, Viša elektrotehnička škola, Beograd (2004).

[14] B. Đorđević, D. Pleskonjić, N. Maček, *Operativni sistemi: koncepti*, Viša elektrotehnička škola, Beograd (2004).