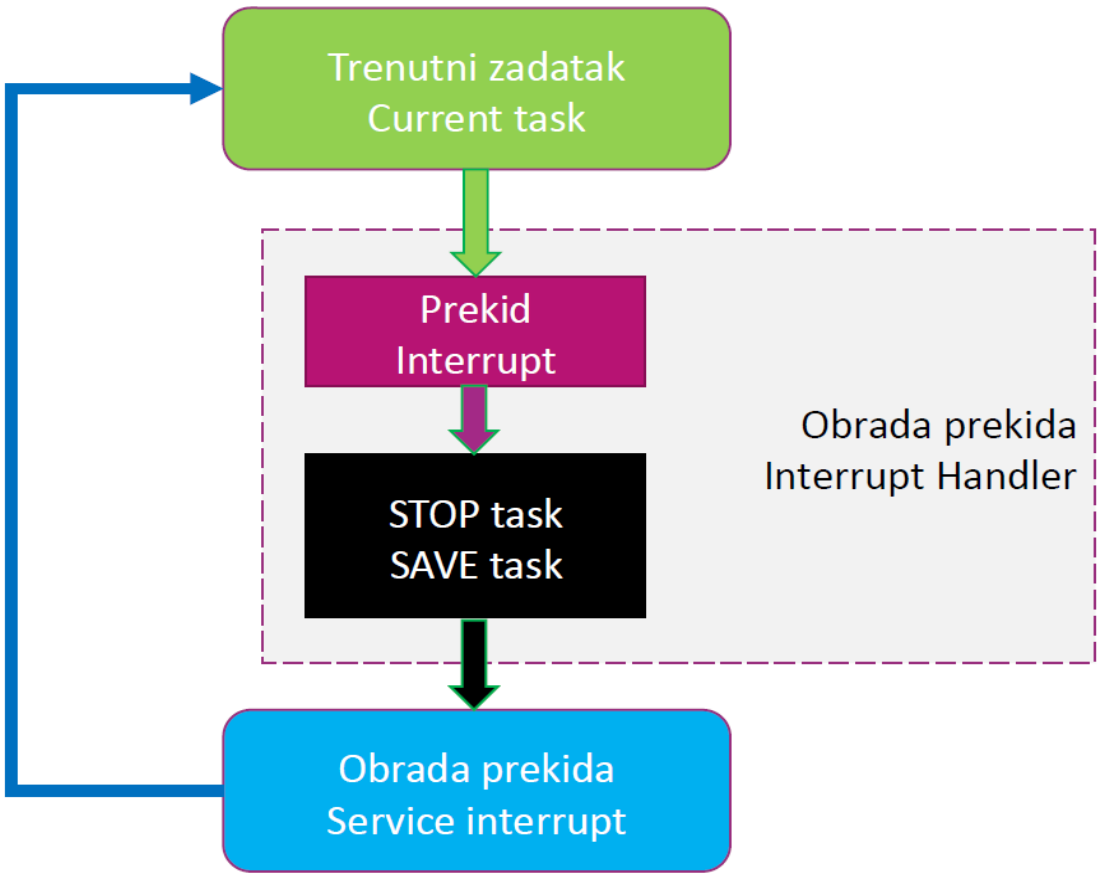
OPERATIVNI SUSTAVI

ISHOD 1

1. Što je prekid?
    a. Prekid je signal koji se šalje od računalne komponente ili programa prema operacijskom sustavu. Uzrokuje da OS privremeno zaustavi redovne procese i počne obrađivati zahtjev za prekid (interrupt handler).

2. Kako nastaje prekid i kako se obrađuje?
    a. Prekid nastaje kada hardverski ili softverski signal zatraži pažnju procesora. Postupak obrade uključuje:
        i. Zaustavljanje trenutnog zadatka.
        ii. Spremanje stanja procesora (registara, programskog brojača).
        iii. Pozivanje interrupt handlera koji rješava zahtjev.
        iv. Vraćanje na prethodni zadatak.



3. Koja je uloga IRQ-a?
    a. IRQ (Interrupt Request) je hardverski signal koji obavještava procesor da je potrebna njegova pažnja. Svaka komponenta ima dodijeljen IRQ broj, a prekidni sustav koristi interrupt handler za obradu signala.

4. Zašto su prekidi važni?
    a. Omogućuju multitasking i brzi odgovor na događaje.
    b. Daju korisniku bolju kontrolu nad računalom.
    c. Trajanje prekida je obično vrlo kratko (0,1% - 2% CPU vremena). Ako traje dulje od 5%, to može ukazivati na hardversku grešku.

5. Što je interrupt handler?
    a. To je specijalizirani dio softvera koji rješava zahtjeve za prekidima. On obrađuje prekid i vraća kontrolu operacijskom sustavu.

6. Kako su prekidi prioritetizirani?
    a. Prekidi imaju razine prioriteta. Viši prioritetni prekid može prekinuti niži prioritetni.

7. Koji su primjeri prekida?
    a. Hardverski:
        i. Pritisak tipke
        ii. Hardware error (paper jam)
        iii. Disk spreman za prijenos podataka.
    b. Softverski:
        i. Sistemski poziv
        ii. Programska greška (npr. segmentation fault).

8. Što se događa ako nema prekida?
    a. Bez prekida, računalo ne bi moglo reagirati na događaje u stvarnom vremenu, što bi značilo da treba čekati dovršetak zadatka prije nego što započne novi.

ISHOD 2

1. Što je proces? Kako se razlikuje od programa?
    a. Proces je instanca programa koja se izvršava.
    b. Program je pasivni skup uputa pohranjen na disku, dok je proces aktivni entitet s vlastitim resursima, kao što su memorija, CPU vrijeme i stanje.

2. Koji su osnovni elementi procesa?
    - Programski kod (instrukcije koje proces izvodi).
    - Podaci procesa, uključujući varijable i strukture.
    - Stanje procesa (trenutno stanje izvršavanja).
    - Programski brojač (lokacija sljedeće instrukcije).
    - Stog (za funkcijske pozive i povratne adrese).
    - Skup registara (trenutno stanje CPU-a za proces).
    - Resursi dodijeljeni procesu (npr. datoteke, I/O uređaji).

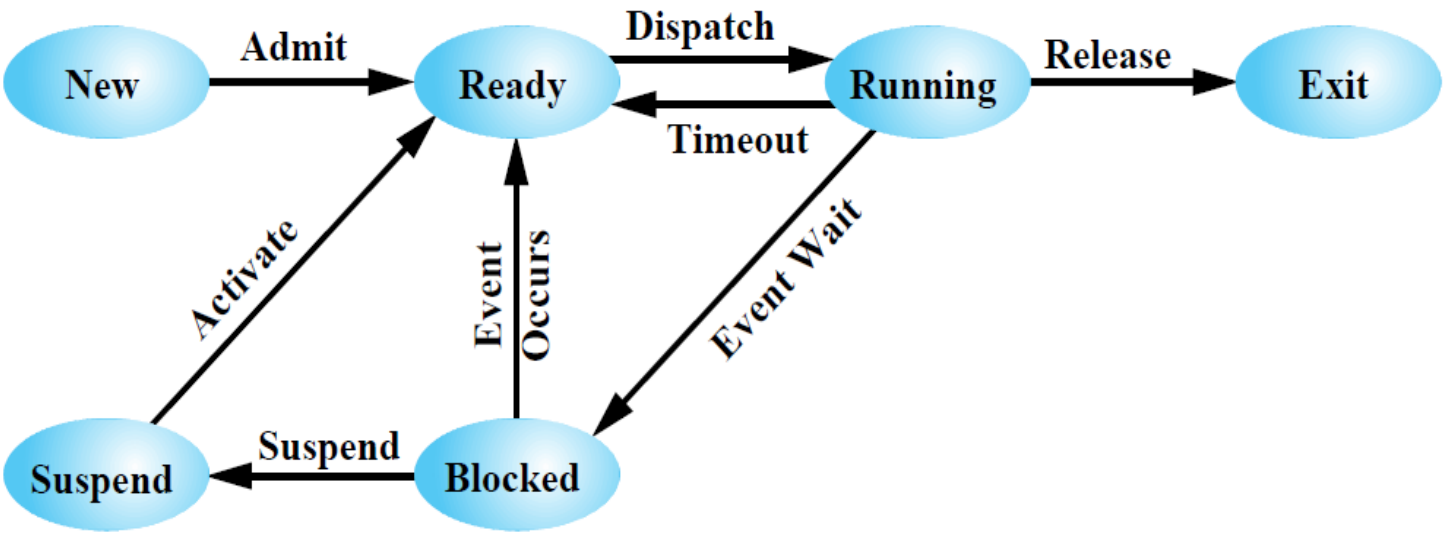
3. Kako operacijski sustav alocira resurse za procese?
    - OS koristi mehanizme poput tablica procesa, memorijskih tablica, i tablica uređaja za upravljanje resursima.
    - Svakom procesu alocira izolirani adresni prostor i upravlja pristupom resursima putem kontrolnih blokova procesa (PCB).

4. Kako nastaje proces?
    a. Proces se kreira kada:
        - OS učita program u memoriju.
        - Dodijeli resurse (memoriju, stog, I/O).
        - Inicijalizira kontrolni blok procesa (PCB).
        - Procesi se često kreiraju pomoću funkcije poput fork().

5. Objasnite ulogu funkcije fork() u stvaranju procesa.
    - fork() je sistemski poziv koji kreira novi proces (dijete) kopirajući trenutno stanje roditeljskog procesa.
    - Novi proces dobiva vlastiti PCB i izvršava iste instrukcije kao roditelj.
    - Povratne vrijednosti fork() razlikuju roditelja (pozitivna vrijednost) od djeteta (0).

6. Što je roditeljski, a što dječji proces?
    - Roditeljski proces je proces koji poziva fork() kako bi kreirao novi proces.
    - Dječji proces je novi proces kreiran od roditeljskog procesa. Dijeli neke resurse, ali ima svoj zaseban PCB.

7. Koje su moguće stanja procesa? Objasnite model s pet stanja procesa.
    - Moguća stanja procesa:
        - New: Proces je kreiran, ali još nije spreman za izvršavanje.
        - Ready: Proces čeka na CPU.
        - Running: Proces trenutno koristi CPU.
        - Blocked: Proces čeka na I/O operaciju ili događaj.
        - Exit: Proces je završio izvršavanje.
    - OS upravlja prijelazima između ovih stanja ovisno o događajima poput prekida, završetka ili zahtjeva za resursima.



8. Koji su razlozi za stvaranje i gašenje procesa?
    a. Razlozi za stvaranje:
        i. Pokretanje novog programa.
        ii. Korisnički zahtjev (npr. otvaranje aplikacije).
        iii. Servisni proces kreiran od OS-a (npr. kontrola ispisa).
    b. Razlozi za gašenje:
        i. Normalan završetak.
        ii. Greške (npr. dijeljenje s nulom, pristup nedopuštenoj memoriji).
        iii. Prekoračenje vremenskog ograničenja.
        iv. Odluka roditeljskog procesa ili OS-a.

9. Kako se upravlja suspendiranim procesima?
    a. Suspendirani procesi se premještaju iz glavne memorije na disk radi oslobađanja resursa.
    b. OS može ponovno aktivirati suspendirani proces kad resursi postanu dostupni.
    c. Proces ostaje suspendiran dok ga ne aktivira agent (npr. OS ili roditeljski proces).

10. Što je PCB (Process Control Block) i koja je njegova uloga?
    a. PCB je podatkovna struktura koju OS koristi za praćenje svih informacija o procesu.
    b. Uloga PCB-a:
        i. Pohranjuje stanje procesa (registre, programski brojač, stog).
        ii. Upravljanje memorijom i resursima.
        iii. Omogućuje OS-u prebacivanje između procesa.

11. Koje informacije o procesu sadrži PCB?

Identifikator procesa (PID).
Stanje procesa.
Podaci o rasporedu (prioritet, redoslijed čekanja).
Pokazivači na memorijske tablice i resurse.
Informacije o interprocesnoj komunikaciji.
Informacije o korištenju resursa (otvorene datoteke, I/O uređaji).

12. Objasnite koncept "zombi" procesa.
    - Zombi proces je proces koji je završio izvršavanje, ali je ostao u tablici procesa jer roditelj još nije preuzeo njegove informacije.
    - Ako roditelj ne pozove funkciju poput wait(), proces ostaje zombi.

13. Što je "orphan" proces i kako operacijski sustav upravlja takvim procesima?
    a. Orphan proces je proces čiji je roditelj završio prije njega.
    b. OS automatski dodjeljuje jezgru (npr. init proces) kao novog roditelja za upravljanje orphan procesom.

14. Što je dretva i kako se razlikuje od procesa?
    a. Dretva (thread) je osnovna jedinica izvršenja unutar procesa. Dretve dijele adresni prostor i resurse procesa, dok procesi imaju vlastite resurse i adresni prostor. Dretve omogućuju paralelno izvršavanje unutar istog procesa.

15. Koje su ključne prednosti korištenja dretvi?
    a. Brže stvaranje i prebacivanje između dretvi nego između procesa.
    b. Dijeljenje memorije i resursa unutar istog procesa smanjuje troškove komunikacije.
    c. Poboljšava performanse na višejezgrenim procesorima omogućujući paralelizam.

16. Što je dretva i koje su osnovne značajke dretvi?
    a. Dretva je osnovna jedinica izvršavanja u operacijskom sustavu.
    b. Dijeli resurse sa svojim procesom, uključujući memorijski prostor i datoteke.
    c. Osnovne značajke uključuju:
        i. Svaka dretva ima vlastiti programski brojač, registre, i izvršni stog.
        ii. Dretve omogućuju paralelno izvršavanje unutar procesa.
        iii. Dijele adresni prostor i resurse procesa kojem pripadaju.

17. Koja je razlika između procesa i dretve?
    a. Proces je jedinica koja upravlja resursima poput memorije i datoteka, dok je dretva osnovna jedinica izvršavanja unutar procesa.
    b. Svaki proces ima vlastiti adresni prostor, dok dretve unutar istog procesa dijele zajednički adresni prostor.
    c. Prebacivanje između procesa zahtijeva više resursa (kontekstno prebacivanje) nego prebacivanje između dretvi.

18. Koje su ključne prednosti korištenja dretvi u operacijskim sustavima?
    - Brže prebacivanje konteksta u usporedbi s procesima.
    - Manje resursa potrebno za stvaranje i upravljanje dretvama.
    - Jednostavnija međusobna komunikacija (dijele adresni prostor i resurse).
    - Paralelizacija unutar aplikacija poboljšava učinkovitost.

19. Koje su ključne operacije povezane s promjenama stanja dretvi?

Spawn	Stvaranje nove dretve
Block	Dretva prelazi u stanje čekanja na resurse
Unblock	Dretva se vraća u spremno stanje
Finish	Završetak izvršavanja dretve

20. Koje su moguće promjene stanja dretvi?

Running	Dretva se trenutno izvršava
Ready	Dretva je spremna za izvršavanje, ali čeka CPU
Blocked	Dretva čeka resurs ili događaj

21. Objasnite koncept sinkronizacije dretvi. Zašto je potrebna?
    - Sinkronizacija dretvi koristi se za sprječavanje konflikata prilikom pristupa zajedničkim resursima. Potrebna je kako bi se:
        - Izbjegle utrke za resurse.
        - Očuvao integritet podataka.
        - Osigurala ispravna komunikacija i koordinacija između dretvi.

22. Kakva je razlika između korisničkih dretvi (ULT) i jezgrinih dretvi (KLT)?
    a. Korisničke dretve (ULT): Upravljanje dretvama obavlja aplikacija. Jezgra nije svjesna njihova postojanja.
    b. Jezgrine dretve (KLT): Upravljanje dretvama obavlja jezgra OS-a. KLT omogućuju istinsku paralelizaciju na višeprocesorskim sustavima.

23. Koje su prednosti i nedostaci korisničkih dretvi?
    - Prednosti:
        - Prebacivanje između dretvi je brže jer ne zahtijeva ulazak u jezgrin način rada.
        - Raspored može biti specifičan za aplikaciju.
        - Funkcioniraju na bilo kojem operacijskom sustavu.
    - Nedostaci:
        - Jedna blokirajuća operacija može zaustaviti sve dretve unutar procesa.
        - Ne iskorištavaju prednosti višeprocesorskih sustava jer jezgra raspoređuje cijeli proces, a ne pojedine dretve.

24. Kako se upravlja dretvama u višedretvenom okruženju?
    - Upravljanje uključuje:
        - Raspodjelu CPU vremena
        - Sinkronizaciju pristupa resursima
        - Upravljanje životnim ciklusom dretvi
    - Moderni operacijski sustavi omogućuju višedretvenost putem jezgrinih i kombiniranih modela (poput KLT-a u kombinaciji s ULT-ovima).

25. Što je višedretvenost i koje su njene prednosti?
    - Višedretvenost omogućava izvršavanje više dretvi unutar jednog procesa, čime se povećava učinkovitost programa. Prednosti uključuju:
        - Bolju iskoristivost procesorskog vremena.
        - Bržu obradu jer dretve mogu paralelno raditi na različitim zadacima.
        - Poboljšanu responzivnost aplikacija.

26. Kako se višedretvenost implementira na jednojezgrenim, a kako na višejezgrenim procesorima?
    - Na jednojezgrenim procesorima koristi se prebacivanje konteksta kako bi se simulirala istovremena izvršenja.
    - Na višejezgrenim procesorima različite dretve zaista mogu paralelno raditi na različitim jezgrama, što dodatno povećava brzinu izvršavanja.

27. Što je kritični odsječak i kako ga zaštititi u višedretvenom okruženju?
    - Kritični odsječak je dio koda koji pristupa zajedničkim resursima. Zaštita se postiže:
        - Mehanizmima međusobnog isključivanja poput semafora ili mutexa.
        - Upotrebom algoritama za sinkronizaciju kao što su Dekkerov ili Petersonov algoritam.

28. Što je "jacketing" u kontekstu dretvi i kako prevladava nedostatke korisničkih dretvi?
    - "Jacketing" je tehnika pretvaranja blokirajućeg sistemskog poziva u neblokirajući. Ovo omogućuje korisničkim dretvama da nastave raditi čak i kada neka dretva unutar procesa pozove blokirajuću operaciju.

29. Koja je razlika između modela 1:1, M:1, 1:M i M:N odnosa dretvi i procesa?
    - 1:1: Svaka dretva izvršenja je jedinstveni proces s vlastitim adresnim prostorom i resursima
    - M:1: Proces definira adresni prostor i dinamičko vlasništvo resursa .
      U tom procesu može se stvoriti i izvršiti više dretvi
    - 1:M: Dretva može migrirati iz jednog procesnog okruženja u drugo.
      To omogućuje lako premještanje dretvi između različitih sustava
    - M:N: Kombinira prethodna M:1 i 1:M model; određeni broj korisničkih dretvi mapira se na određeni broj jezgrinih dretvi.

ISHOD 3

1. Koji su ciljevi planiranja procesorskog vremena?
    - Ciljevi planiranja procesorskog vremena uključuju:
        - Maksimiziranje iskorištenosti procesora: Održavanje procesora što više zauzetim.
        - Minimiziranje vremena čekanja: Smanjenje vremena koje procesi provode u redovima.
        - Minimiziranje vremena odziva: Osiguravanje brzih odgovora korisnicima u interaktivnim sustavima.
        - Postizanje pravednosti: Ravnomjerna raspodjela vremena između procesa kako bi se izbjeglo izgladnjivanje.
        - Postizanje predvidljivosti: Stabilan rad sustava bez velikih oscilacija u performansama.

2. Koja je razlika između dugoročnog, srednjoročnog i kratkoročnog planiranja procesorskog vremena?
    - Dugoročno planiranje: Odlučuje koji procesi ulaze u sustav za obradu. Cilj je kontrolirati opterećenje sustava i osigurati adekvatan broj aktivnih procesa.
    - Srednjoročno planiranje: Određuje koji procesi u glavnoj memoriji ostaju aktivni ili se premještaju u sekundarnu memoriju. Koristi se za upravljanje višezadaćnošću.
    - Kratkoročno planiranje: Odabire sljedeći proces za izvršavanje na procesoru. Ovo se događa vrlo često i u milisekundama, obično na temelju algoritama poput Round Robina ili SJF-a.

3. Koje su glavne razlike između kriterija orijentiranih na korisnika i onih orijentiranih na sustav u raspoređivanju procesorskog vremena?
    - Korisnički orijentirani kriteriji: Fokusirani su na iskustvo korisnika, npr. minimalno vrijeme odziva, poštenje i predvidljivost.
    - Sustavno orijentirani kriteriji: Fokusirani su na učinkovitost sustava, npr. visoka iskorištenost procesora, velika propusnost i nisko vrijeme čekanja.

4. Što podrazumijeva pojam "vrijeme odziva" i zašto je važno u interaktivnim sustavima?
    - Vrijeme odziva je interval između podnošenja zahtjeva i prvog odgovora sustava. U interaktivnim sustavima, brzo vrijeme odziva ključno je za korisničko iskustvo, jer omogućuje sustavu da odgovori na zahtjeve korisnika bez uočljivih kašnjenja.

5. Objasnite kako funkcionira algoritam First Come, First Serve (FCFS) te koje su njegove prednosti i nedostaci.
    - FCFS izvršava procese redoslijedom kojim dolaze u red čekanja.
    - Prednosti:
        - Jednostavna implementacija.
        - Pravedan prema redoslijedu dolaska.
    - Nedostaci:
        - Može dovesti do dugog čekanja ako proces koji je na redu traje dugo (efekt "konvoja").
        - Nije prilagođen za interaktivne sustave jer ne uzima u obzir hitnost zahtjeva.

6. Što je Round Robin (RR) algoritam i kako se određuje vremenski kvant (quantum)?
    - Round Robin dijeli procesorsko vrijeme na jednake intervale (kvante) koje dodjeljuje procesima redoslijedom.
      Ako proces ne završi tijekom svog kvanta, stavlja se na kraj reda.
    - Određivanje vremenskog kvanta:
        - Prevelik kvant može rezultirati slabim odzivom u interaktivnim sustavima.
        - Premali kvant uzrokuje učestale promjene konteksta, što smanjuje učinkovitost.

7. U čemu se razlikuju algoritmi Shortest Job First (SJF) i Shortest Remaining Time First (SRTF)?
   - SJF:
     - Neiznuđen (non-preemptive).
     - Proces s najkraćim ukupnim vremenom izvršenja ima prioritet.
   - SRTF:
     - Iznuđen (preemptive).
     - Proces s najkraćim preostalim vremenom izvršenja ima prioritet.
   - SRTF može prekinuti trenutno izvršavanje procesa ako novi proces s kraćim preostalim vremenom uđe u red čekanja.

8. Koja je razlika između preemptivnih i nepreemptivnih algoritama raspoređivanja?
   - Nepreemptivni:
     - Jednom kada proces započne izvršavanje, ostaje na procesoru sve dok se ne završi ili blokira.
       Pogodniji za batch sustave.
   - Preemptivni:
     - Procesor može prekinuti trenutni proces i dodijeliti ga drugom procesu, npr. pri dolasku hitnijeg procesa.
       Bolje za interaktivne sustave i sustave s višezadaćnošću.

9. Što znači "fair-share scheduling" i kako osigurava pravednu raspodjelu procesorskog vremena?
   - Fair-share scheduling dodjeljuje procesorsko vrijeme proporcionalno udjelima koji su unaprijed određeni za korisnike ili grupe procesa.
     Cilj je da korisnici koji već koriste veći udio resursa ne dobiju dodatne prednosti, dok oni s manjim udjelima dobiju veći prioritet.

10. Koji su kriteriji za optimizaciju performansi procesora pri kratkoročnom planiranju?
    - Propusnost: Maksimiziranje broja dovršenih procesa po jedinici vremena.
    - Vrijeme čekanja: Smanjivanje vremena koje procesi provode u redovima.
    - Vrijeme obrta: Ukupno vrijeme od dolaska procesa do završetka izvršenja.
    - Vrijeme odziva: Brzina početnog odgovora na zahtjev.

11. Kako kriteriji poput "propusnosti" i "iskorištenosti procesora" utječu na odabir algoritma raspoređivanja?
    - Propusnost: Algoritmi koji brzo dovršavaju procese, poput SJF-a, povećavaju propusnost.
    - Iskorištenost procesora: Algoritmi koji minimiziraju vrijeme neaktivnosti procesora, poput RR-a s dobro odabranim kvantom, maksimiziraju iskorištenost.

12. Kako raspoređivanje procesorskog vremena može utjecati na predvidljivost sustava?
    - Algoritmi koji osiguravaju stabilno vrijeme odziva i ujednačeno ponašanje, poput RR-a, doprinose predvidljivosti. Velike varijacije u vremenu čekanja i izvršenja mogu signalizirati neoptimalan algoritam ili preopterećenje sustava.

13. Zašto je važno postići balans između propusnosti i vremena odziva pri odabiru algoritma?
    - Visoka propusnost može smanjiti vrijeme odziva u sustavima s velikim brojem kratkih procesa. Međutim, davanje prednosti propusnosti može rezultirati izgladnjivanjem dugotrajnih procesa. Balans osigurava da sustav ostane učinkovit, a korisničko iskustvo zadovoljavajuće.