OPERACIJSKI SUSTAVI MI1 – OGLEDNA PITANJA

Prvi ishod:

- Što osigurava vezu između aplikacijskih programa i sklopovlja?

  Hal.dll

- Kakva sve mogu biti sučelja?

  Korisnička sučelja mogu biti GUI ili Command Line

- Podjela operacijskih sustava.

  Korisnički operacijski sustavi i poslužitelji

- Koje su prednosti/nedostaci hijerarhijske izgradnje OS-a?

  Prednosti – lagana modifikacija dijelova sustava i primjena na različite modele sklopovlja
  Nedostaci – povećano vrijeme pristupa do određene operacije, sporo izvođenje, vrijeme i cijena razvoja

- Opišite slijed podizanja operacijskog sustava.

- Za koje operacije se prekida petlja čekalica?

  Prekida se kada dođe neki drugi proces

- Opišite i dajte primjer sistemskog poziva.

- Opišite i dajte primjer prekida.

  Svaki klik miša ili tipke na tikovnici će uzrokovati prekid kako bi obavijestio OS o specifičnom događaju koji se mora obraditi

- Opišite i dajte primjer iznimke.

  Iznimka je neki poziv s reškom, npr. ako dođe do dijeljenja s nulom ili do stack overflowa, tada će doći do izvršavanje iznimke

- Opišite i objasnite razliku između monolitne strukture i mikrojezgre.

  Monolitna struktura se razlikuje od mikrojezgre tako što se kod monolitne strukture upravljački programi nalaze unutar kernela

- Kako izgleda model jednostavnog računala?

  Von Neumannov model računala

- Što je interrupt request, kako se on postavlja, što je plug and play tehnologija?

  Interrupt request je poziv za prekid

  Postavlja se interrup request ID-om

  Plug and play tehnologija je tehnologija koja ne traži konfiguraciju prije prvog korištenja

- Što se dešava kod svakog poziva potprograma?

  Kontekst se prebacuje na potprogram pa se potprogram počne izvršavati

- Što je to promjena konteksta?

  Spremanje stanja aktualne dretve u memoriju prije izvršavanje druge

- Kakva je struktura stoga?

  First in last out

- Kako izgleda poziv procedure call-by-value (prednosti i nedostaci)?

  Call-by-value funckija stvara kopiju vrijednosti i nju obrađuje

- Kako izgleda poziv procedure call-by-reference (prednosti i nedostaci)?

  Call-by-reference funckija prosljeđuje adresu vrijednosti i radi s originalom

- Koja je razlika između WIN32API i .NET ili Jave sa stanovišta brzine izvođenja?

  WIN32 će se uvijek izvršavati prije

- Da li za vrijeme analize prekida (određivanja izvršavanja) može doći prekid, zašto?

  Ne može jer kod zabranjuje nove prekide u izvršavanju zbog toga što se u slučaju previše prekida nikada neće izvršiti do kraja

- Objasni algoritam za izbor prekida (logični ne kod)?

  Onemogućuje se dolazak novih prekida, zatim se sortiraju prekidi po prioritetima te se nakon toga prekidi omoguće i kreće s izvršavanjem

- Računalni sustav ima 5 pristupnih sklopova sa 5 prioriteta : P1-najveći, P2,P3,P4 i P5 najmanji. Veći prioritet prekida trenutni prekid. Prekidi traju: P1 – 1t, P2 – 2t, P3 – 2t,P4 – 1t i P5 – 2t. Prekidi se pojavljuju redom P4 u 1t, P2 u 7t, P3 u 2t,P1 u 3t, P4 u 4t i P5 u 6t . Nacrtajte obradu

Drugi ishod:

- Koja je razlika program, proces, dretva, nit?

  Program je statični niz instrukcija, proces je program u izvođenju, dok svaki proces ima barem jednu dretvu koja se sastoji od više niti

- Zašto na OS-u koristimo procese?

  Kako bi osigurali višezadaćnost

- Što radi naredba fork()?

  Naredbom fork() kreiramo novi proces koji postaje child process pozivatelja

- Što je PID?

  PID je Process ID koji služi za razlikovanje svih aktivnih procesa prilikom

izvršavanja nekih operacija na njima

- Kako se pokreće proces?

  Pokreće se naredbom fork() (Za UNIX) ili CreateProcess() (Za Windows) u command liniji ili programom iz grafičkog sučelja

- Objasni promjene stanja procesa (kada i zašto prelazi u pojedina stanja).

  U stanju „New“ se nalazi novokreirani proces koji prelazi u stanje „Ready“ kada čeka svoje izvršavanje, odnosno čeka slobodno procesorsko vrijeme, zatim prelazi u stanje „Running“ kada se proces izvršava koji nakon toga prelazi u stanje „Waiting“ ako čeka izvršavanje nekog vanjskog događaja (najčešće I/O operacije) ili u stanje „Terminated“ ako je završio s izvršavanjem

- Objasni nasljedna svojstva procesa?

  Svaki proces može biti proces roditelj koji proizvodi svoju djecu koja zatim od roditelja nasljeđuju kopiju svih njegovih varijabli

  Jezgra sustava (init proces) ima PID 0 i nema roditelja, dok svi ostali imaju roditelje

- Kakav je to zombie proces, kako nastaje? Ž

  Zombie proces je proces čiji ga roditelj ne sakupi po završetku izvršavanja, odnosno kada roditelj prvi „umre“

- Vidljivost varijabli između dva procesa, nasljeđivanje varijabli.

  Procesi međusobno ne mogu vidjeti svoje varijable

- Navedi jednu od starijih metoda za među procesnu komunikaciju i objasni.

  Zajedničko korištenje shareane memorije – problem nastaje zbog toga što ne znamo kad će koji proces koristiti tu memoriju

- Navedi jednu od novijih metoda za među procesnu komunikaciju i objasni, naglasi prednost.

  IPC (Inter Process Communication), komuniciranje poruka kod koje je prednost kontrola korištenja memorije između procesa

- Koji je nedostatak RPC rješenja?

  Oslanja se na mrežnu komunikaciju za rad

- Dan je kod za pokretanje procesa… koliko je pokrenuto novih procesa, što se događa ako sustav vrati 0, što se događa ako sustav vrati pozitivan broj, što se događa ako sustav vrati -1

Treći ishod:

- Koja je prednost dretve nad procesom?

  Brža promjena konteksta, međusobno dijeljenje memorijske lokacije, jeftinija po

pitanju resursa, omogućuje paralelizam unutar jednog programa

- Što dretve dijele, a što ne dijele međusobno?

  Dretve dijele memorijsku lokaciju, ali međusobno ne vide vlastiti stog i registre procesora

- Objasni promjena stanja dretvi (ne treba crtati ali treba objasniti)

  Isto kao i kod procesa

- Što je to slijed dretvi čemu on služi?

  Slijed dretva je redoslijed kojim će se dretve izvršavati ovisno o tome jesu li međusobno zavisne ili nisu – ako su zavisne mora se utvrditi redoslijed koji će se te dretve izvršavati

- Kada je varijabla za dvije dretve nezavisna?
  Kada dretve samo čitaju tu varijablu

- Kakve su to cikličke dretve, navedi primjer?

  To su dretve koje se ponavljaju, npr. čekalice

- Objasni zašto nastaje kritični odsječak.

  Nastaje zato što više dretvi želi pristupiti istoj varijabli

- Objasni primjer rješavanja problema sa varijablom zastavica, što ne valja?

- Objasni primjer rješavanja problema sa varijablom pravo, što ne valja?

- Objasni logiku rada Dekkerovog postupka (ne treba pisati algoritam nego logički objasniti kako radi)?

- Objasni razliku Dekkerovog i Petersonovog algoritma?

- Objasni prednost Lamportovog algoritma?

- Što radi mutex?

  Mutex je „brava“ ili „ključ“ kojeg dretve čekaju kako bi pristupile kritičnom odsječku

- Što može semafor, a ne može mutex?

  Semafor ima više stanja, a mutex samo 2 stanja (nulu i jedinicu)

- Objasni korištenje semafora za notifikaciju?

  Služe za signalizaciju među dretvama, npr. dretva A čeka dretvu B da obavi neku operaciju i postavi semafor, zatim dretva A uzima rezultat dretve B kao ulazni parametar

- Objasni korištenje semafora za brojanje?

  Koristi se u programima u kojima će se određeni broj dretvi izvršiti, a ostatak

blokirati

- Objasni razliku binarnog i općeg semafora (za koje vrijednosti pušta/ne pušta)

  Binarni semafor ima 2 stanja, a opći više

- Koja je razlika između iznuđene i neiznuđene višezadaćnosti?

  Iznuđene višezadaćnosti – Round Robin, Shortest Remaining Job First – dretva se može prekinuti

  Neiznuđene višezadaćnosti – FIFO, Shortest Job First – dretva se ne može prekinuti osim ako sama ne dodijeli CPU vrijeme drugoj dretvi

- Navedi primjer algoritma kod neiznuđene višezadaćnosti (prednosti, nedostaci)?

  FIFO, SJF – efikasnija i lakša implementacija, ali otvara mogućnost beskonačno duge operacije

- Navedi primjer algoritma kod iznuđene višezadaćnosti (prednosti, nedostaci)?

  Round Robin, SRJF – sporija zbog više promjena konteksta, ali spriječava smrzavanje vremenskim sklopom koji daje pravo izvršavanja jedne dretve na određeno vrijeme

- Algoritam XXXXX koristi iznuđene ili neiznuđene višezadaćnosti, objasni?

  Treba objasniti zadani algoritam…

- Koji su ciljevi izbora višezadaćnosti?

  Efikasnost i vrijeme odziva

- Objasni cilj minimiziranje vremena odziva kod algoritama za višezadaćnosti?

  Cilj minimiziranja vremena odziva je da se neka operacija ili posao izvrši u što kraćem vremenu

- Objasni cilj pravednosti kod algoritama za višezadaćnosti?

  Cilj pravednosti je dugo neizvršenoj dretvi dati pravo izvršavanja van pravila redoslijeda

- Objasni cilj maksimiziranje broja korisnih operacija kod algoritama za višezadaćnosti?

  Cilj maksimiziranja broja korisnih operacija je povećati iskoristivost procesora, odnosno vremena kada procesor obavlja neki koristan posao u nekom vremenu

- Objasniti logiku rada strategije XXX za raspoređivanje?

  Treba objasniti zadani algoritam (RR, FIFO, SRJF, SJF)….

- Što je to izgladnjivanje, zašto je ono opasno?

  Ako se cilj pravednosti ne izvrši može doći do izgladnjivanja dretve, što znači da se nikada neće moći izvršiti zbog stalnog dolaska novih dretvi s većim prioritetom

- Prikaži sva stanja XXXXX algoritma do njegova pražnjenja , ako je trenutno stanje reda D1=3t, D2=1t, D3=2t i u 0t trenutku je CPU slobodan. Naknadno u 2t dolazi nova dretva u red D4=2t.