Parcijalni ispit

PITANJA

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_I-prva\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.Sta je trap i kada nastaje? - 2018 OKT

Trap je instrukcija koja izaziva softverski prekid ako u toku izvrsavanja procesa nastane greska ili izuzetak, tada OS čuva stanje prekinutog procesa, prelazi u mod kernela i poziva funkciju kernela (rutinu, system call handler) koja implementira sistemski poziv.

//Trap je naredba koja dovodi do sinhronog prekida i nastaje nakon smestanja argumenata na stek kako bi se preslo iz user moda u kernel mod za dalje izvrsavanje.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2.Na koji nacin sistemski poziv uzrokuje prelazak u kernel mod izvrsavanja? - 2018 SEP

Sistemski poziv se obavlja u okviru korisničkog programa pozivom funkcije iz standardne biblioteke za odgovarajući programski jezik (API).

U okviru ove funkcije se argumenti smeštaju na stek, zatim se poziva trap naredba pomocu koje dolazi do prelaska u kernel mod.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

3.Koja je prednost multiprogramiranja? Objasniti ukratko. - 2018 SEP

Maksimalna iskoriscenost procesorskog vremena.Npr ako se desi neki poziv(U/i ili neka operacija) koja ne zahteva CPU Os dodeljuje cpu drugom procesu i tako ne gubi vreme.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

4.Sta je proces? - 2018 JAN

Proces je osnovni koncept operativnog sistema i predstavlja program u izvršavanju na CPU. Gledano sa strane OS-a, proces je osnovna jedinica izvršavanja i najmanji entitet koji se može planirati, dodeliti i izvršavati na procesoru.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

5.Sta je multiprogramiranje? - 2018 JAN

https://www.youtube.com/watch?v=pn2G0HBVEbc

Os upisuje vise procesa u memoriju bira 1 koji se izvrasa,ako on pozove neki U/I os dodeli procesor drugom procesu pa dobijamo iluziju da se procesi izvrsavaju istovremeno.Multiprogramiranje predstavlja mogucnost sistema da radi sa vise procesa.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

6.Koja su tri cilja dizajna operativnog sistema? - 2017 OKT

Ima tri osnovna cilja:

Pogodnost - Da omogući lako i pogodno korišćenje računara

Efikasnost - Da obezbedi efikasno korišćenje i upravljanje resursima računara

Mogućnost razvoja - Da obezbedi osnovu za efikasan razvoj, testiranje i uvođenje novih funkcija sistema.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

7.Zasto je bolje razviti mikrokernel (OS) od monolitnog kernel-a? - 2017 OKT

Zato sto je mikrokernel pouzdaniji ako dodje do greske u korisnickim aplikacijama nece doci do pada sistema, laksi je za nadogradnju, lakse se prenosi na druge platforme i lakse se podesava funkcionisaje na distribuiranim sistemima.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

8.Na koji nacin OS obezbedjuje multiprogramiranje, koje komponente OS su ukljucene i kako? - 2017 OKT

Multiprogramiranje (multitasking) – Operativni sistem istovremeno smešta u memoriju više poslova; u jednom trenutku samo jedan od poslova se izvršava na CPU, ukoliko se blokira izvršenjem U/I operacije (npr. čitanje podataka sa diska), aktivira se planiranje poslova

Ukljucene su komponenta za upravljanje memorijom, komponenta za izvrsenje procesa, komponenta za rad sa U/I uredjajima, upravljanje datotekama.

Zavisi od potrebe procesa npr. ne mora network komponenta ako nema pristupa internetu.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

9.Na koji nacin i kroz koje funkcije operativni sistem vrsi multipleksiranje resursa racunarskog sistema? Na koji nacin se dele (multipleksiraju) sledeci resursi, u vremenu, prostoru ili oba: CPU, memorija, disk, mrezna kartica, stampac, tastatura, displey? - 2017 APR

Multipleksiranje resursa se može odraditi na dva načina: u prostoru i u vremenu.

U vremenu - različiti programi ili koriste resurs naizmenicno.

U prostoru - svaki program ili korisnik dobija jedan deo resursa.

CPU,stampac,tastatura,mrezna kartica - vreme

memorija,disk- prostor

display- oba

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

10.Navesti i ukratko objasniti razlike i slicnosti izmedju sistemskog poziva i prekida. Sta se menja u PCB procesa i na koji nacin prilikom ovih dogadjaja? Koji atributi u PCB ostaju nepromenjeni (navesti barem 3). - 2016 DEC

Sistemski poziv dovodi do sinhronog prekida i predstavlja nacin da se predje u kernel mod. Dok je prekid asinhroni dogadjaj koji se desava na hardveru i on suspenduje trenutni rad procesora kako bi se obradio.

Menjaju se kontekst(registri procesora),stanje procesa,program counter. Menja se tako sto se podaci azuriraju i PCB se smesta u odgovarajucu listu.

Ne menja se kod programa,pokazivac na memoriju,lista datoteka,lista U/I uredjaja.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

11.Uporedite strukture OS-a: monolitnu, mikrokernel i slojevitu po sledecim kriterijumima: performanse, pouzdanost, prosirljivost. Kako bi ste implementirali novi drajver uredjaja u svakoj od ovih struktura? -2016 OKT2

Preformanse: monolitna odlicna ostale losije9

Pouzdanost: Slojevita i mikrokernel su pouzdanije dok je monolitna losa

Prosirljivost: monolitna losa dok mikro i slojevita imaju dobru prosirljivost

Mikrokernel - implementirao bih kao korisnicki servis koji bi bitne funkcionalnosti izvrsio u kernelu

Slojevita- kao sloj koji bi pozivao odgovarajuce funkcije slojeva nizih nivoa

Monolitna- kao procedura koji se izvrsava u kernelu

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

12.Koje su razlike izmedju procesa i niti? Zasto je korisno koristiti niti, kad vec postoji multiprocesiranje, tj. procesi mogu kreirati nove procese koji se konkurento/paralelno izvrsavaju? Objasniti. -2016 OKT1

Svaki proces ima sopstveni adresni prostor dok niti dele adresni prostor procesa u okviru koga se nalaze.

Korisno je zato sto je komunikacija izmedju niti mnogo brza od komunikacije izmedju procesa, brze se kreiraju niti i brza je obrada prekida kod niti nego kod procesa.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

13.Sta je sistemski poziv, i u kom modu rada se izvrsava (mod kernela i korisnicki mod). Objasniti nacin izvrsavanja sistemskog poziva na primeru exec sistemskog poziva za izvrsavanje novog izvrsnog koda procesa. - 2016 SEP1

Sistemski poziv je interfejs između aplikativnih/sistemskih programa i operativnog sistema. Omogućava pristup funkcijama operativnog sistema od strane korisničkih programa. Obavlja u okviru korisničkog programa pozivom funkcije iz standardne biblioteke za odgovarajući programski jezik (API) i izvrsava se u kernel modu.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

//ovo je ispitno ne mora da se radi

14. Koja komponenta operativnog sistema:

1. upravlja organizacijom podataka na SD kartici
2. vodi racuna da se svi startovani programi sinhronizuju u pristupu deljenoj memoriji
3. obezbedjuje prevodjenje virtuelne u fizicku adresu
4. planira pristup blokovima hard diska
5. poziva funkciju drajvera uredjaja - 2016 JUN
6. upravljanje UI uredjajem
7. upravljanje memorijom
8. upravljanje memorijom
9. upravljanje UI uredjajem
10. upravljanje drajverom

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

15. Koja je razlika izmedju sistemskog poziva i bibliotecke funkcije (npr. read() sistemski poziv i fscanf() bibliotecka funkcija). Objasniti ovu razliku u nacinu izvrsavanja sistemskog poziva na primeru fork sistemskog poziva za kreiranje novog procesa. - 2016 APR

Sistemski poziv je interfejs između aplikativnih/sistemskih programa i operativnog sistema. Omogućava pristup funkcijama operativnog sistema od strane korisničkih programa. Obavlja u okviru korisničkog programa pozivom funkcije iz standardne biblioteke za odgovarajući programski jezik (API) i izvrsava se u kernel modu.

fscanf() je bibliotecka funkcija koja se poziva u USER modu, odakle se posredstvom sistemsog poziva prelazi u kernel mod i poziva funkcija read() da bi se obavilo citanje. Funkciju read() nije moguce pozvati iz USER moda direktno, vec samo preko funkcije fscanf().

Kada se u toku izvrsenja koda procesa naidje na sistemski poziv fork, prelazi se u kernel mod da bi se kreirala kopija tekuceg procesa, odakle se ponovo vraca u USER mod i dalje izvrsenje se nastavlja u jednom od dva procesa.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

16. Koje su razlike izmedju procesa i niti? Zasto je bilo korisno uvodjenje niti, kad vec postoji multiprocesiranje, tj. procesi mogu kreirati nove procese koji se konkurento/paralelno izvrsavaju? Objasniti. - 2016 MAR

Ima vec odgovor na ovo.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

17. Ukratko objasniti razlike izmedju korisnickog i kernel moda racunarskog sistema. U kom modu se izvrsava i zasto:

1. Rutina za obradu prekida
2. Planer (rasporedjivac) procesa
3. Vi editor
4. Algoritam zamene stranice
5. Funkcija sin(x) - 2016 JAN
6. Drajver uredjaja
7. Dispecer
8. Algoritam planiranja diska
9. Firefox

U korisničkom modu izvršavaju se korisnički programi. U ovom režimu rada nije dozvoljen pristup određenim mem. oblastima kao I izvršavanje određenog skupa mašinskih instrukcija. Kernel OS-a se izvršava u kernel modu a iz koga se može pristupiti zaštićenim mem. oblastima i izvršavati privilegovane mašinske instrukcije. Prelazak iz jednog u drugi režim rada uzrokuje sistemski poziv, prekid i zamka (trap).

a)Kernel mod

b)Kernel mod

c)Korisnički mod

d)Kernel mod

e)Korisnički mod

f)Kernel

g)Kernel

h)Kernel

i)Korisnicki

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_II-druga\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.Navesti tri konkretne akcije/dogadjaja pri kojim aktivni proces prekida svoje izvrsenje, tj. napusta stanje Izvrsava se (Running stanje) - 2018 OKT

Ako dodje do prekida,ako se pojavi proces viseg prioriteta i ako istekne timeout interval, ako se proces zavrsi,U/I greska, nema dovoljno memorije, ubijen od OS-a, zavrsen na zahtev roditelja koji ima privilegije-odaberi 3

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2.Sta je PCB (Process Control Block) i koje informacije sadrzi? - 2018 SEP

PCB je struktura podataka u operativnom sistemu koja sadrzi informaicije o procesu. Informacije u okviru upravljačkog bloka procesa (PCB) mogu se grupisati u tri kategorije: Identifikatori procesa, Informacije o stanju procesora, Informacije za upravljanje procesom.

PCB sadrzi:

Id procesa,

Program counter,

Prioritet,

Stanje

CPU registre(kontekst procesa),

Pokazivac na virtuanlu memoriju koja mu je dodeljena,

U/I uredjaje koje koristi,

Otvorene datoteke

...

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

3.Sta cini sliku procesa u glavnoj memoriji? - 2018 SEP

Slika procesa (Process image) predstavlja skup koji čine program, podaci, magacin (stek, stack), i atributi procesa (PCB) smešteni u glavnu memoriju (delom i na disku).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

4.U kojim situacijama (dogadjajima) proces prekida svoje izvrsavanje, tj. napusta stanje Izvrsava se (Running stanje)? - 2018 JAN

Pogledati II-1.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

5.Opisati ukratko korake koje OS obavlja prilikom promene procesa (process switch, context switch) - 2018 JAN

Koraci koje obavlja OS pri zameni procesa:

1) Snimiti sadržaj registara procesora, programski brojač i statusni registar procesora u PCB procesa

2) Ažurirati PCB procesa koji je do tada bio u stanju Izvršavanje (Running)

3) Premestiti PCB u odgovarajući red – Spreman, Blokiran, Spreman/suspendovan i promeniti mu stanje

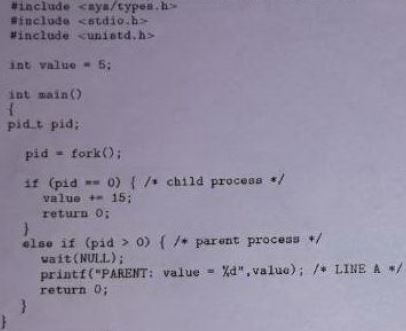
4) Izabrati sledeći proces za izvršavanje

5) Ažurirati PCB procesa koji je izabran u stanje Izvršavanje

6) Ažurirati stukture podataka za upravljanje memorijom

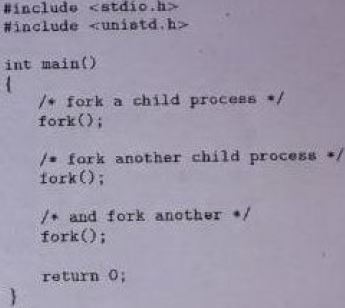
7) Obnoviti sadržaj registara procesora na vrednosti koje su postojale u trenutku kad je proces promenjen iz stanja Izvršavanje

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

6.U sledecem programu, koja vrednost je izlazna na liniji A (LINE A)? Ukratko objasniti zasto!  - 2017 OKT

ISPISUJE SE 5,NEMAS PRINT KOD DETETA DA BI BILO 20,PROBAJ PRE PISANJA!!!!

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

7.U sledecem programu, koliko se procesa generise ukljucujuci i inicijalni? Ukratko objasniti zasto! - 2017 OKT

---8 PROCESA,3 FORKA PRAVE 7 I PLUS ORIGINALNI

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

8.Koliko procesa ce se kreirati nakon izvrsenja kod a), a sta ce se desiti izvrsenjem koda b) Objasniti. - 2017 APR

1. void main(){ b) void main(){

fork(); for(;;)

fork(); fork();

exit(); } }

Koje funkcije/operacije operativni sistem obavlja prilikom sistemskog poziva fork()? Objasniti.

a)Kreirace se 2 nova i ovaj glavni znaci 4 procesa

b)Pravi koliko moze procesa da se izvrsi na procesoru,doci ce do greske kada se napuni procesor

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

9.Da li u ovim slucajevima dolazi do promene moda i zamene procesa? Obasniti zasto! Ako dolazi do zamene procesa objasniti u koje stanje prelazi atrenutno aktivni proces, ako:

1. Pozove sistemski poziv fork.
2. Nastane prekid generatora takta.
3. Startuje sistemski poziv recive() za prijem poruke od drugog procesa.
4. Pozove funkciju za mnozenje dve matrice prosledjene kao argument funkcije.
5. Izvrsi deljenje sa 0. - 2016 DEC

a) dolazi do promene moda, sto se tice zamene procesa zavisi od Os-a ali najlogicnije bi bilo da se proces koji je pozvao fork izvrsi// tako je reko na konsultacije!

b) Dolazi do promene stanja, proces prelazi u stanje Blokiran (mozda spreman),dolazi do zamene procesa, da prelazi se u kernel za obradu prekida

c) Dolazi do promene stanja, proces prelazi u stanje Blokiran recive() je uglavnom blokirajuca funkcija,dolazi do zamene moda

d) nema zamene moda nakon izvrsenja instrukcije nastavlja trenutni proces znaci nema ni switcha procesa

e) mod se menja,proces bi trebao da stane sa izvrsenjem i blokira se \*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

10.Sta je proces, a sta kontekst izvrsavanja procesa? Kako operativni sistem koristi kontekst izvrsavanja procesa? -2016 OKT2

Proces je osnovna jedinica izvršavanja i najmanji entitet koji se može planirati, dodeliti i izvršavati na procesoru. Kontekst izvršavanja procesa sadrzi registre procesora smestene za vreme izvrsavanja procesa. Koristi kontekst procesa prilikom njegovog izvrsavanja da postavi registre procesora.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

11.Navesti 4 dogadjaja koji svojim nastankom mogu uzrokovati prekid trenutno aktivnog procesa (zamenu procesa) i promenu konteksta.

-Ako dodje do prekida,ako se pojavi proces viseg prioriteta, ako istekne timeout interval, nema dovoljno memorije, ubijen od strane Os-a..

12.Zasto je vremenski kvant koji se dodeljuje procesu za izvrsenje ostao nepromenjen poslednjih 20 godina i pored toga sto se brzina procesa drasticno povecala? Koje su prednosti, a koji nedostaci veceg vremenskog kvanta u odnosu na manji? -2016 OKT1

Multitasking,Interrupt handling,User and kernel mode switching,?

Zato sto je on dovoljan da se izvrsi veliki broj procesa pogotovu sa vecom brzinom procesora i zato sto je u/i spor.Manji kvant moze dovesti da ako proces mora da izvrsi neki sistemski poziv moze se desiti da sistemski poziv potraje i da istekne vreme dodeljeno procesu za izvrsenje, a ako je preveliki kvant onda ce se procesoru dodeljivati CPU orijentisani procesi i doci ce do izgladnjivanja U/I orijentisanih procesa.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

12.Operativni sistem ima sledeca stanja procesa: spreman, izvrsava se, blokiran, novi i zavrsen. U kom stanju je bio i u koje stanje prelazi proces (nit) za svaki od sledecih slucajeva? Objasniti

1. Istekao je dodeljen vremenski kvant.
2. Izvrsava instrukciju Test\_and\_Set dok je argument instrukcije ima vrednost 0.
3. Pozvao je recive funkciju za prijem poruke.
4. Izvrsio je funkciju exit(0).
5. Tokom izvrsavanja startovan je proces viseg prioriteta. - 2016 SEP 1

a) Izvrašavanje->Spreman

b) Izvršavanje->Blokiran ili ne menja stanje nismo sigurni

c) Izvršavanje->Blokiran

d) Izvršavanje->Završen

e) Izvršavanje->Spreman

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

13.Operativni sistem ima sledeca stanja procesa: spreman, izvrsava se, blokiran, novi i zavrsen. U kom stanju je proces(nit) za svaki od sledecih slucajeva? Objasniti.

1. Pozvao je funkciju read za citanje podataka sa diska
2. Izvrsava instrukciju Test\_and\_Set do je argument funkcije ima vrednost 0.
3. Zavrsila se read funkcija i podaci su u memoriji
4. Izvrsio je funkciju signal nad monitorskom promenljivom
5. Tokom izvrsavanja na Windows OS startovan je sistemski proces. -2016 JUN

a)Izvršavanje->Blokiran

b)isti kao primer gore

c)Blokiran->Spreman

d)Nastavlja izvrsenje ne menja stanje

e)Zavisi od toga da li je blokirajuci ili ne

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

14. Pretpostavite da imate sledeci C program. Kako bi ste ga implementirali koriscenjem vise niti u istom procesu, ukoliko je programski kod izvrsne datoteke proba implementiran u funkciji proba(arg)? Dopunite kod u kome i glavna i nova nit pristupaju promenljivoj p, ali se uzajamno iskljucuju koriscenjem semafora.

main()

{

int pid;

int p=5;

pid = fork();

if(pid== 0)

excev(“proba”, arg);

} - 2016 APR

semaphore s=1;

int p=5;

main() void proba(void \*arg)

{

pthread\_t id; {

pthread\_create(&id,proba(arg),NULL); semWait(s);

semWait(s); p=p-1;

p=nesto; semSignal(s)

semSignal(s); }

} \*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

15. Navesti 4 dogadjaja koji svojim nastankom mogu uzrokovati prekid trenutno aktivnog procesa (zamenu procesa) i promenu konteksta. Zasto je vremenski kvant koji se dodeljuje procesu za izvrsenje osato skoro nepromenjen poslednjih 20 godina i pored toga sto se brzina procesora drasticno povecala? Koje su prednosti, a koji nedostaci veceg vremenskog kvanta u odnosu na manji? -2016 MAR

Isto pitanje kao 11

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

16. Sta predstavlja kontekst izvrsavanja procesa i gde se cuva. Da li se u ovim situacijama menja kontekst i kako? Objasniti.

1. Proces je pozvsao funkciju cwait() u okviru monitora.
2. Izvrsava test\_and\_set instrukciju nad promenljivom r cija je vrednost 0.
3. Upravo je zavrsio prenos bloka podataka sa diska koji je zahtevao funkciju read.
4. Planer procesa ga je izabrao za izvrsavanje u sledecem vremenskom kvantu.

-2016 JAN

Vrednost registra cuva se u PCB.

a)Izvršava->Blokiran

b)ima gore

c)Blokiran->Spreman

d)Ready->Running

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_III-treca\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.U sistemu postoje dva procesa sa po dve niti P1(T11,T12) i P2(T21, T22). Koja se nit sledeca izvrsava:

* 1. Nit T11 izvrsi sistemski poziv, a niti su implementirane na nivou korisnika (ULT).

Zato sto nit T11 izvrsava sistemski poziv, proces P1 se blokira i izvrsice se P2 a nit zavisi od biblioteke koja ih implementira.

* 1. Niti T22 istekne vremenski kvant dodeljen u biblioteci niti, a niti su implementirane na nivou korisnika (ULT).Procesor je i dalje na P2 a niti zavise od biblioteke.
  2. Nit T21 pozove funkciju semWait() nad semaforom s=0, a niti su implementirane na nivou kernerla (KLT). - 2018 OKT

T21 se blokira posto su na kernelu on bira bilo koju od spremnih niti.

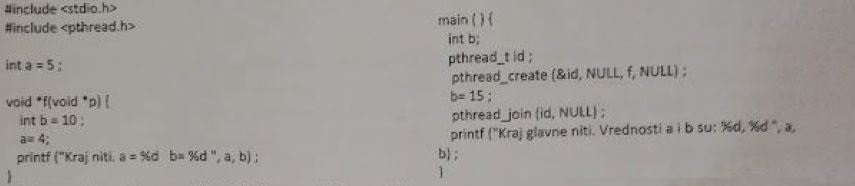
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2.Koja je razlika izmedju procesa i niti u OS koji podrzava niti na nivou kernela (KLT)? Objasniti ukratko. - 2018 SEP

U KLT procesor vidi samo niti za izvrsavanje a procesi sluze samo da bi se dodelili resursi i memorija ali se ne biraju za izvrsenje.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

3.Napisite izraz nakon izvrsenja dvo-nitnog procesa koji izvrsava navedeni kod. - 2018 SEP



pisace Kraj niti a=4, b=10 pa Kraj glavne niti a=4, b=15 pthread join ceka da se gornja nit terminira pre nego sto ispise print

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

4.Pretpostavite da izvrsite sledeci C program. Promenite kod programa koriscenjem niti tako da je programski kod izvrsne datoteke zad3 implementiran u funkciji zad3(), i da glavna i nova niti pristupaju promenljivoj res, ali se uzajamno iskljucuju koriscenjem semafora.

main()

{ int pid;

int res;

pid = fork();

if(pid==0)

execv(“zad3”, …); - 2018 JAN

Vec je uradjen!

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

5.Sta je kontekst izvrsavanja procesa i da li se i u cemu razlikuje od konteksta izvrsavanja niti? - 2017 OKT

//pitali smo reko je samo po sadrzaju da se razlikuju

Kontekst izvrsavanja procesa su registri procesora upisani u PCB za vreme njegovog izvrsavanja, razlikuju se samo po sadrzaju.TCB ima samo ono bitno za izvrsenje niti bez stek pointera, U/I uredjaja, datoteka…. TCB ima identifikatore niti i informacije za upravljanje a nema ono u sredini iz 3 podele sa slajdova tako reko jebem ga sta je mislio!

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

6.U kojim situacijama mozemo ocekivati da se brze izvrsava proces implementiran sa ULT nitima u odnosu na isti proces implementiran sa KLT nitima? - 2017 OKT

Ukoliko se zahteva mnogo promena izmedju niti u ULT-u nema prebacivanja moda (ne zahteva se privilegija kernel moda) dok se u KLT-u zahteva, pa je zbog toga ULT brzi od KLT-a.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

7.Koji su uobicajeni resursi koje dele sve niti istog procesa? - 2017 OKT

Sve niti pridružene jednom procesu dele program i resurse tog procesa. Globalne promenljive, memoriju, datoteke...

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

8. Koja je najveca prednost implementacije niti na nivou korisnika, a koji je najveci nedostatak ove implementacije? Objasniti. - 2017 APR

Najveca prednost je to sto nema promene iz user u kernel mod prilikom zamene niti, a nedostatak taj da ako bilo koja nit pozove sistemski poziv blokirace sve ostale niti tog procesa kao i proces. Jos jednan nedostatak je da niti istog procesa se ne mogu izvrsavati istovremeno na razlicitim jezgrima.

\*Drugi deo pitanja!

//ovo nece da bude 100%

Treba da uporedite citanje datoteke sa jednonitnog i visenitnog (KLT) file server-a. Ako se prijem zahteva, obrada zahteva i vracanje datoteke zahteva 12ms, ako se podaci datoteke nalaze u blok kesu, sto je slucaj u ⅓ zahteva. U ostalim slucajevima, za obradu zahteva neophodno je 75 ms tokom kojih je niti blokirana. Koliko zahteva/sekundi server moze da obradi ukoliko se radi o jednonitnom serveru, a koliko za visenitni server? Objasniti.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

9.Zamena niti u okviru istog procesa brza je kod a) ULT niti b) KLT niti? Objasniti zasto.

Najveca prednost je to sto nema promene iz user u kernel mod prilikom zamene niti.

//kod ult moda?

Navesti i ukratko objasniti konkretne dogadjaje/uslove koji zahtevaju KLT niti izmedju dva stanja, za svaki od prelazaka (strelica) od 1-6. -2016 DEC

Ready -> Running kada kernel dodeli cpu toj nit za izvrsenje

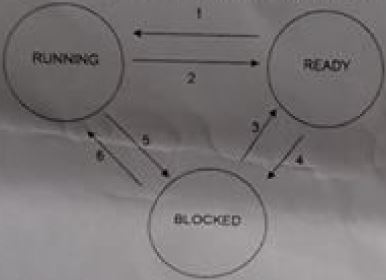
Running -> Ready ako istekne timeout interval

Running -> Blocked ako pokrene neku U/I operaciju

Blocked -> Ready ako je dobila podatke koje je cekala

Ready -> Blocked Proces viseg prioriteta zauzme resurs koji je bio dodeljen (ides da igras fudbal spreman si i dodju stariji zauzmu teren)

Blocked -> Running Ne postoji



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

10.Zasto se postizu bolje performanse u izvrsavanju procesa koji ima vise niti, od izvrsenja istog procesa koji je implementiran jednonitno? Objasniti.

Zato sto ako ima vise niti, izvrsenje procesa moze da se podeli na vise niti. Niti jednog procesa mogu se izvrsavati paralelno (konkurentno).

Nit moze biti u jednom od tri stanja: Izvrsava se (Running), Spremna (Ready) ili Blokirana (Blocked). U kom stanju su niti N1 i N2 procesa P za svaki od sledeca 3 slucaja, ako su niti implementirane u prostoru jezgra (KLT)? Objasniti.

1. Nit N1 je izvrsila funkciju read za citanje sa diska

Nit1 prelazi u blocked stanje read je sistemski poziv, a Nit2 u running ako je bila ready

1. Nit N1 je pozvala funkciju signal semafora na kome ceka nit N2 istog procesa.

Nit1 ce da se izvrsi do kraja, a Nit2 postaje ready

//Nit1 //Ni2

-radi nesto -semWait(s);

-semSignal(s); -radi nesto

-jos neki kod

1. Zavrsila se U/I operacija koju je startovala nit N1.

N1 ide u ready, N2 ne menja stanje

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

11. Proces P je dvonitni. Svaka nit procesa moze biti u jednom od tri stanja: Izvrsava se, Spremna, i Blokirana. Ako su niti implementirane u prostoru (KLT) odredite u kom su se stanju nalazile i u koje stanje prelaze niti N1 i N2 procesa P za svaki od sledecih pet dogadjaja: -2016 OKT1

1. Nit N1 je pozvala funkciju semSignal semafora Synch cija je trenutna vrednost -1 dok na semaforu Synch ceka nit N2.

synch=-1 semSignal povecava na 0 posto je synch<=0 iz queue se izbacuje nit2 sto znaci Nit2 u ready Nit1 nastavi

1. Nit N1 je pozvala funkciju write za upis u datoteku OCENE, dok N2 ceka da se zavrsi funkcija read iz datoteke USPEH.

N2 je blokiran i ceka,N1 ceka kraj upisa i nastavlja ne menja stanje

Zavrsila se U/I operacija koju je startovala nit N2.

N2 u ready,N1 nismo sigurno ne znamo gde je bila

1. Nit N1 je zavrsena dok joj je trajao vremenski kvant.

N2 u running,N1 terminated

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

12.Navesti dogadjaje koji svojim nastankom mogu uzrokovati prekid trenutno aktivnog procesa (zamena procesa) i promenu konteksta. Koje atributi procesa se menjaju u PCB-u procesa prilikom zamene procesa i na koji nacin? Objasniti. - 2016 SEP

//bilo do sad

Prekidi

Pojava procesa viseg nivoa

Isteko Timeout

Proces zvao U/I

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

13. Na koji nacin mogu da komuniciraju niti istog procesa? Navesti primer i objasniti delom (pseudo)-koda. U stistemu postoje 3 konkurenta procesa, kao sto je prikazano u sledecoj tabeli.

Procesi Niti u okviru procesa

P1 T11, T12, T13

P2 T21, T22

P3 T32

Pretpostaviti da sistem koristi round-robin planiranje (sa prekidanjem) i da se nit T12 izvrsava kada joj istekne dodeljen vremenski kvant. Ako su niti implementirane na nivou korisnika (ULT) navesti i objasniti koje niti mogu da se izvrsavaju u sledecem vremenskom kvantu. Ako su niti implementirane na nivou kenela (KLT), navesti i obrazloziti koje niti mogu biti izvrsavane u sledecem vremenskom kvantu. - 2016 JUN

**vazi**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

14. Proces (ili nit) moze biti u jednom od tri stanja: izvrsava se (Running), spreman (Ready), blokiran (Blocked), novi i zavrsen. Navesti i ukratko opisati dogadjaje koji uzrokuju prelazak u stanje spreman (Ready)? U kom je stanju i u koje stanje prelazi proces (nit) za svaki od sledecih slucajeva? Objasniti.

1. Pozvao je funkciju cwait() u okviru monitora.

proces postaje blokiran,nit isto

1. Izvrsava test\_and\_set instrukciju nad promenljivom r cija je vrednost 0.

blokira i ceka da test and set vrati vrednost pa prelazi u running

1. Upravo je zavrsen prenos bloka podataka sa diska koji je zahtevao funkcijom read.

prelazi u ready

- 2016 APR

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

15. Koje su osnovne razlike izmedju hibridne implementacije niti i implementaciji niti u prostoru jezgra (KLT). Proces P je tronitni.

Niti se kreiraju, sinhronizuju u user modu, a KLT je sve u kernelu.

Svaka nit procesa moze biti u jednom od tri stanja: Izvrsava se, Spremna i Blokirana. Ako su niti implementirane u prostoru jezgra (KLT) odredite u kom se stanju nalaze i u koje stanje prelaze niti N1, N2 i N3 procesa P za svaki od sledeca tri slucaja:

1. Nit N1 je pozvala funkciju semSignal semafora Sync cija je trenutna vrednost -1 dok na semaforu Sync ceka nit N3, dok nit N2 ne ceka.

N3 ide u ready, N2 nismo sigurno, N1 nastavi sa radom

1. Nit N1 je pozvala funkciju receive dok je N2 poslala poruku funkcjiom send, a nit N3 ceka zavrsetak operacije citanja iz datoteke Ispiti.

N3 je u blocked i ceka da predje u ready, N2 nastavlja dalje a N1 ide u blocked i ceka da stigne poruka

1. Nit N1 je pozvala funkciju write za upis u datoteku Ocene, a nit N2 ceka da istekne vremenski kvant niti N1, a N3 je u zaposlenom cekanju pozivom Test & Set instrukcije.

N1 ceka kraj upisa i nastavja, N2 je blocked i ceka ili je u running ako se misli da je ona sledeca po rasporedu za izvrsenje, N3 je u running i ceka

-2016 MAR\*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

16. Sta sadrzi upravljacki blok niti i koji atributi niti se menjaju prilikom izvrsenja sistemskog poziva od strane niti i kako?

Registar stanja i programski brojac i menja se stanje te niti azurira se TCB i pridodaje odgovarajucoj listi.2

U kom stanju su niti N1 i N2 procesa P za svaki od sledeca 2 slucaja, ako su niti implementirane na nivou korisinika (ULT) i na nivou kernela (KLT). Objasniti.

1. Nit N1 je izvrsila funkciju wait nad semaforom cija je vrednost 0. Nit1 se blokira, N2 ne znamo
2. Istekao je vremenski kvant dodeljen niti N1 u okviru algoritma planiranja implementiranog u biblioteci niti. Nit1 ide u ready, ne znamo N2 -2016 JAN \*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_IV-cetvrta\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.Prikazati kako se koriscenjem semafora moze obezbediti strogo naizmenicno

izvrsavanje 2 procesa? - 2018 OKT

Semafori se mogu koristiti za sinhronizaciju dva procesa.

Posmatramo dva procesa P1 i P2. Naredba S1 procesa P1 treba da se izvede pre naredbe S2 procesa P2. Koristi se binarni semafor synch Semafor synch se inicijalizira na 0.

void P1()

{

S1;

semSignal(synch);

…

}

void P2()

{

...

semWait(synch);

S2;

}

void main()

{

semaphore synch =0;

parbegin( P1,P2);

};

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2.Kako se signal() operacija definisana u okviru monitora razlikuje od sem\_signal() operacije definisane nad semaforom? - 2018 SEP

Glavna razlika: Ako proces u monitoru izda signal i ako nijedan proces ne čeka na toj uslovnoj promenljivoj signal je izgubljen !!

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

3.Napisati kod monitora koji obezbedjuje funkcije za pristup jednoj zajednickoj promenljivoj u memoriji koja je u formi “prekidaca”, za vise procesa koji treba da je naizmenicno ukljucuju i iskljucuju - postavljaju zajednicku promenljivu na true ili false, tako da se upis i citanje obavljaju naizmenicno. - 2018 JAN

monitor prekidac;

bool ukljucen;

cond pali,gasi;

{

ukljucen=true;

}

void Upali()  
{

if(ukljucen==true)

cwait(pali);

ukljucen=true;

csignal(gasi);

}

void Ugasi()  
{

if(ukljucen==false)

cwait(gasi);

ukljucen=false;

csignal(pali);

}

////////////////////////////////////////////

void palioc()

{

while(true)

{

Upali();

//za sta su produce i consume? Ako predstavljaju podatke iz monitora u korisnicki mod onda je drugacije

}

}

void gasioc()

{

while(true)

{

Ugasi();

}

}

void main()

{

parbegin(palioc,gasioc);

}\*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

4.Objasniti termin zaposleno cekanje (busy waiting). U kojim slucajevima se javlja zaposleno cekanje i da li postoji neko drugo “cekanje” u okviru operativnog sistema? Ako semaforske operacije semwait() i semsignal() nisu atomicne da li se pomocu njih moze obezbediti uzajamno iskljucivanje procesa? Objasniti kako da/ne. Objasniti i prikazati primerom kako se pomocu binarnog semafora moze obezbediti uzajamno iskljucivanje N procesa? - 2017 OKT

Cekanja mogu biti: cekanje semafora, cekanje monitora, cekanje neke u/i operacije...

binary\_semaphore s=1;

void Process(int i)

{

while(true){

semWait(s);

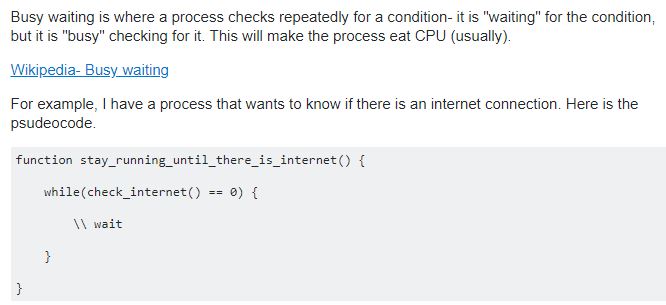
//nesto se desava

semSignal(s);

}

}

Moraju biti atomicne da bi se izvrsile do kraja inace mozemo doci u situaciju da jedan proces ne inkrementira ili dekrementira lock i da lock dobije nevalidnu vrednost.



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

5.Sta je monitor, koje su njegove osnovne karakteristike i koje su osnovne prednosti koriscenja monitora umesto semafora za uzajamno iskljucivanje procesa. Implementirati binarni semfaor koriscenjem monitora. Kako bi ste implementirali monitor koriscenjem binarnog semafora? - 2017 APR

Monitor je konstrukcija visih programskih jezika koja ima funkcionalnost identicnu semaforima i laksa je za upravljanje.

Karakteristike monitora:

1. Lokalnim promenljivama može se pristupati samo iz procedura monitora i nijednom spoljnom procedurom

2. Proces “ulazi” u monitor pozivom neke od njegovih procedura

3. U jednom trenutku samo jedan proces se može izvršavati u monitoru; svaki drugi proces koji poziva proceduru monitora se blokira i čeka da monitor postane dostupan

//monitor preko semafora nije uradjeno nemamo za parcijalni

Prednost je da monitor enkapsulira podatke i njima mozes da pristupis samo preko funkcija monitora, samo jedan proces moze u jednom trenutku da koristi funkcije monitora.

binarni preko monitora////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

monitor A;

cond uslov;

bool islocked=false;

void wait()

{

if(islocked==true)

cwait(uslov);

islocked=true;

}

void signal()

{

if(islocked==true)

{

islocked=false;

csignal(uslov);

}

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

6.Na koji nacin je u okviru monitora implementirano uzajamno iskljucivanje? Ukoliko u operativnom sistemu imamo implementirane monitore, implementirati brojacki semafor preko monitora (Monitore NSemaphore). - 2016 DEC

brojacki preko monitora

monitor A;

cond kljucevi;

int brKljuceva=neka vrednost;

void wait()

{

if(brKljuceva==0)

cwait(kljucevi);

brKljuceva--;

}

void signal()

{

brKljuceva++;

csignal(kljucevi);

}

Deljivi resursi se nalaze unutar monitora i moze im se pristupiti samo preko funkcija monitora. On uzajamno iskljucivanje obezbedjuje tako sto samo jedan proces moze da pristupi monitoru u datom trenutku.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

7. Koja je razlika izmedju binarnih i brojackih semafora? Ukoliko imate operativni sistem koji obezbedjuje binarne semafore i monitore, napisati programski kod za implementaciju brojackog semafora koriscenjem a) binarnog semafora, b)monitora. -2016 OKT1

Бинарни семафори су бинарни, могу имати само две вредности; један за представљање да се процес / нит налази у критичном делу (код који приступа заједничком ресурсу) и други треба да сачека, а други који означава да је критични одељак слободан.

С друге стране, бројање семафора узима више од две вредности, оне могу имати било коју вредност коју желите. Максимална вредност N коју узимају омогућава да N процесa / нити истовремено приступе дељеном ресурсу

brojacki preko binarnog!\*\*\*\*\*\*

binary\_semaphore que=1;

binary\_semaphore s=1;

int keys=N;

semWait(que);

semWait(s);

keys--;

if(keys>0)

semSignal(que);

semSignal(s);

//pristupi podacima

semWait(s);

keys++;

semSignal(s);

semSignal(que);

CSem(K) cs { // counting semaphore initialized to K

int val ← K; // the value of csem

BSem gate(min(1,val)); // 1 if val > 0; 0 if val = 0

BSem mutex(1); // protects val

Pc(cs) {

P(gate)

a1: P(mutex);

val ← val − 1;

if val > 0 V(gate);

V(mutex);

}

Vc(cs) {

P(mutex);

val ← val + 1;

if val = 1

V(gate);

V(mutex);

}

}

\*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

8.Koje su osnovne prednosti koriscenja monitora umesto semafora za uzajamno iskljucivanje procesa? Napisati kod kojim se obezbedjuje sinhronizacija dva procesa preko zajednicke promenljive u memoriji (koji treba da naizmenicno ukljucuju i iskljucuju “prekidac” - postavljaju zajednicku promenljivu na 0 ili 1) tako da se upis i citanje obavljaju naizmenicno, koriscenjem a) semafora, b)monitora - 2016 SEP

semaphore s1=0;

semaphore s2=0;

int promenjiva=0;

void Upali() void Ugasi()

{ {

if(promenljiva==1) if(promenljiva==0)

semWait(s2); semWait(s1);

promenljiva=1; promenljiva=0;

semSignal(s1); semSignal(s2);

} }

void main()

{

parbegin(Upali,Ugasi);

}

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

monitor b;

int promenljiva;

cond uslovzagasenje,uslovzapaljenje;

{

promenljiva=0;

}

void upali()

{

if(promenljiva==1)

cwait(uslovzapaljenje);

promenljiva=1;

csignal(uslovzagasenje);

}

void Ugasi()

{

if(promenljiva==0)

cwait(uslovzagasenje);

promenljiva=0;

csignal(uslovzapaljenje);

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

9. Implementirati brojacki semfaor koriscenjem monitora. -2016 JUN

brojacki preko monitora

monitor A;

cond kljucevi;

int brKljuceva=neka vrednost;

//neki deljivi resursi

void wait()

{

if(brKljuceva==0)

cwait(kljucevi);

brKljuceva--;

//pristup deljivim resursima

}

void signal()

{

brKljuceva++;

if(brKljuceva>0)

csignal(kljucevi);

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

10. Neka su procesi A i B implementirani na sledeci nacin:

Proces A Proces B

Wait(R) Sekcija\_B1

Wait(S) Signal(R)

Sekcija\_A1 Sekcija\_B2

Signal(S) Wait(S)

Sekcija\_A2 Sekcija\_B3

Signal(R) Signal(S)

Procesi A i B dele semafore R i S inicijalizovane na R=0 i S=1. Procesi A i B imaju vise sekcija od vise naredbi. Za zadatu implementaciju procesa A i B odgovorite sta je tacno od ponudjenih alternativa.

1. Izvrsenje sekcije Sekcija\_A1 ne moze startovati pre komplementiranja sekcije Sekcija\_B1.-tacno
2. Nije moguce istovremeno izvrsenje Sekcija\_A1 i Skecija\_B2-ne
3. Moguce je istovremeno izvrsenje Sekcija\_A1 i Sekcija\_B3-ne
4. Nije moguce istovremeno izvrsenje Sekcija\_A2 i Sekcija\_B3-ne
5. Moguce je istovremeno izvrsenje Sekcija\_A1 i Sekcija\_B3,a nije moguce istovremeno izvrsenje Sekcija\_A2 i Sekcija\_B3.-ne

-2016 APR

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

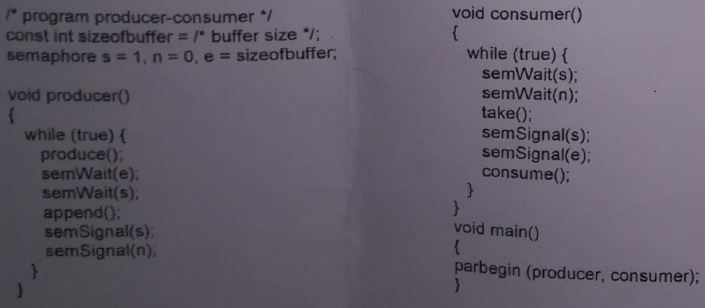
11. Koja je prednost monitora u odnosu na semaphore za implementaciju uzajamnog iskljucenja? Napisati kod za dva procesa kojim se obezbedjuje sinhronizacija dva procesa preko zajednicke promenljive u memoriji (koji treba da se naizmenicno ukljucuju i iskljucuju “prekidac” - postavljaju zajednicku promenljivu na 0 ili 1) tako da se upis i citanje obavljuju naizmenicno, koriscenjem a) semafora, b) monitora - 2016 MAR

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

12. Koja je namena binarnih, a koja brojackih semafora? Ukoliko imate operativni sistem koji obezbedjuje binarne semafore, napisati programski kod za implementaciju brojackog semafora koriscenjem binarnog. -2016 JAN

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

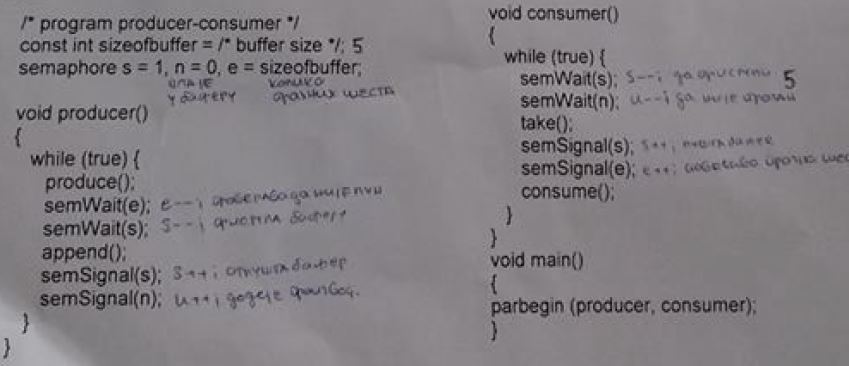
13.Da li ovim resenjem problema proizvodjac-potrosac moze nastati uzajamno blokiranje procesa proizvodjaca i potrosaca. Ako ne, objasniti na koji nacin je obezbedjeno sprecavanje/izbegavanje uzajamnog blokiranja. Ako da, objasniti u kom slucaju nastaje uzajamno blokiranje i kako ga spreciti? - 2017 APR



Nije omoguceno, dolazi do greske. S koristimo za zakljucavanje bafera, i treba da se obrnu semWait(s) i semWait(n) u consumer-u.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

14.Da li ovim resenjem problema proizvodjac-potrosac moze nastati uzajamno blokiranje procesa proizvodjaca i potrosaca. Ako ne, objasniti na koji nacin je obezbedjeno sprecavanje/izbegavanje uzajamnog blokiranja. Ako da, objasniti u kom slucaju nastaje uzajamno blokiranje i kako ga spreciti. -2016 DEC



**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

15 .Sta je semfaor? Ako su operacije semWait i semSignal implementirane na sledeci nacin objasniti ukratko sta je problem kod ovakve implementacije semafora i resite ga. Opisati kako se pomocu ovako definisanog semafora moze obezbediti uzajamno iskljucivanje dve niti, a kako se moze obezbediti da jedna nit izvrsi blok naredbi A, pre nego sto druga nit izvrsi blok naredbi B.

Semafor je integer promenljiva nad kojom su definisane tri atomicne operacije: inicijalizacija, semWait(S), semSignal(S)

void semWait(Semaphore S) { void semSignal(Semaphore S) {

while (S.count<=0); S.count = S.count +1;

S.count=S.count-1;

} }-2016 OKT2

void SemWait(s)

{

s.count--;

if(s.count<0)

//stavi ga u q i block

}

vois semSignal(s)

{

s.count++;

if(s.count<=0)

{

//aktiviraj iz steka

}

}

semaphore s=0;

void N1() void N2()

{ {

A; semWait(s);

semSignal(s); B;

} }

4. pitanje za ovaj deo

semaphore s1=1;

void Niti(int i)

{

semWait(s1);

…

semSignal(s1);

}

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_V-peta\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1.Razmotriti sistem sa 200 jedinica memorije koje su dodeljene 4 procesa na sledeci nacin: Primeniti Bankarov algoritam da bi procenili da li je sistem u bezbednom stanju.

Proces Max Zauzima

1 130 75

2 50 25

3 70 35

4 180 40 2018 OKT

**Q//koliko mu jos fali,ako ostane samo P4 nema bezbednog stanja jer ne moze da se izvrsi (neoznacen je),ovi su oznaceni ostali**

**1 55**

**2 25**

**3 35**

**4 140**

**V // trenutno raspolozivi resurs**

**//Max-C**

**//Zauzima-A**

**//R-200**

**//V-200-175=25 raspolozivi resursi**

**Bezbedna sekvenca: P2 P3 P1 P4**

**P1 55<=25 nije**

**P2 25<=25 jeste dajemo procesu P2 potrebne resurse V i on se izvrsi**

**novo V je staro v+A tog procesa 50**

**P3 35<=55 jeste radis isto kao gore V=85**

**P4 140<=85 nije**

**P1 55<=85 jeste novo V je 85+75=160**

**P4 140<=160 jeste V=160+40=200**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

2.Zasto savremeni operativni sistemi uglavnom implementiraju oporavak od uzajamnog blokiranja, a ne metode za sprecavanje i izbegavanje uzajamnog blokiranja? - 2018 SEP

Oporavak: Sistem ne pokušava da spreči pojavljivanje uzajamnog blokiranja

Strategije sprečavanja i izbegavanja uzajamnog blokiranja ograničavaju pristup resursima i nameću ograničenja procesima.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

3.Napisati deo koda za dva procesa koji se izvrsavajuci operacije nad semaforima mogu dovesti do uzajamnog blokiranja. - 2018 SEP

proces A Proces B

request R1 Request R2

lock R1 lockR2

request R2 Request R1

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

4.Zasto savremeni OS ne implementiraju tehnike sprecvavanja uzajamnog blokiranja?

OS se projektuje tako da isključi mogućnost pojave uzajamnog blokiranja. Postoje dve klase metoda, medjutim ove metode vode neefikasnom korišćenju resursa i neefikasnom izvršenju procesa.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

5.Razmotriti sistem sa 200 jedinica memorije koje su dodeljene 4 procesa na sledeci

nacin: Primeniti Bankarov algoritam da bi procenili da li je sistem u bezbednom stanju.

Proces Max Zauzima

1 120 75

2 110 50

3 70 35

4 60 15 - 2018 JAN

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

6.Da li je moguce da jedan proces bude u uzajamnom blokiranju? Ako ne objasnit zasto, a ako da objasniti kako? Razmotriti racunarski sistem sa sledecom alokacijom resursa:

A (dodeljeni resursi) C (max zahtevi procesa) V (raspolozivi resursi)

Allocation Max Avalible

A B C D A B C D A B C D

P0 0 0 1 2 0 0 1 2 1 5 2 0

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

1. Odrediti matricu neophodnih resursa svakog od procesa
2. Ako stigne zahtev od procesa P1 (0,4,2,0) odrediti da li sistem prelazi u bezbedno stanje? - 2017 OKT

//ovo nije pod b za pod b P1 Max bi bio 1 11 7 0

Q A B C D Q bi bio 0 11 7 0

P0 0 0 0 0

P1 0 7 5 0

P2 1 0 0 2

P3 0 0 2 0

P4 0 6 4 2

P0 q<=v 0 0 0 0 <= 1 5 2 0

novo V je 1 5 3 2

P2 moze novo v je 2 8 8 6

P3 moze V je 2 14 11 8

P4 moze V je 2 14 12 12

P1 moze V je 3 14 12 12

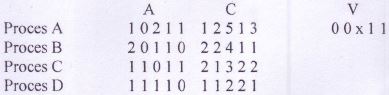
//proveri dal je isto kad saberes

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

7.Koja tri uslova moraju biti prisutna da bi nastalo uzajamno blokiranje i da li tada uzajamno blokiranje sigurno nastaje?

Uzajamno iskljucivanje, drzi i cekaj, bez prekidanja. Ne nastaje sigurno, potrebno je i kruzno cekanje.

Sistem ima 4 procesa i 5 tipova resursa koji im se dodeljuju. Trenutna alokacija resursa i maksimalni zahtevi su sledeci. Kolika je najmanja vrednost x da bi ovo stanje bilo sigurno? Objasniti.

 -2016 OKT2

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

8.Razmotrimo sistem sa 150 jedinica memorije koje su dodeljene tri procesa na sledeci nacin:

Proces Max Drzi

1 70 45

2 60 40

3 60 15

Primeniti Bankarov algoritam da bi procenili da li je sigurno zadovoljiti sledece zahteve, i ukoliko jeste navesti sigurnu sekvencu procesa:

1. Proces 4 se startuje sa maksimalnim potrebama od 60 memorijskih jedinica, dok su inicijalne potrebe 25 memorijskih jedinica.
2. Proces 4 se startuje sa maksimalnim potrebama od 60 memorijskih jedinica, dok su inicijalne potrebe 35 memorijskih jedinica.
3. Nakon procesa 4 u pitanju a) startuje se proces 5 sa maksimalnim zahtevom od 35 memorijskih jedinica dok su inicajlne potrebe 5 memorijskih jedinica. -2016 OKT1

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

9.Sta radi operativni sistem i na koji nacin kada otkrije postojanje uzajamnog blokiranja procesa?

slajd 26, 27 prezentacija 5

U sistemu se izvrsava 6 procesa: P0-P5. Ima 4 tipa resursa A(15 instanci), B(6 instanci), C(9 instanci) i D(10 instanci). U trenutku T stanje dodele i zahteva za resursima je sledece: 

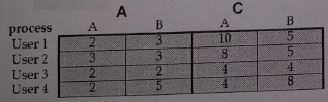
1. Proveriti stanje V vektora.
2. Izracunati matricu potrebnih resursa (C-A)
3. Ispitati da li je trenutno stanje sigurno koriscenjem Bankarovog algoritma.
4. Ako proces P3 izda zahtev za resursima (3,2,3,3) da li se ovaj zahtev moze obaviti, a da sistem ostane u sigurnom stanju? - 2016 SEP

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

10. Sistem ima tri resursa R1, R2 i R3 sa po dve instance i 5 procesa. Proces P1 drzi R2 i zahteva R1, proces P2 drzi R1, proces R3 drzi R1 i zahteva R3, proces P4 drzi R2 i R3, proces P5 drzi R3 i zahteva R2. Na osnovu grafa dodele resursa odrediti da li je sistem u uzajamnom blokiranju. Objasniti. Ako nije, dopuniti zahteve za procesa za resursima da bi sistem bio u uzajamnom blokiranju i objasniti zasto. -2016 JUN

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

11. Sta je sigurno stanje sistema? Na osnovu Bankarovog algoritma odrediti da li je sledece stanje sigurno ili ne, i objasniti zasto, ako ima dva resursa A i B ako su i raspolozivi resursi V=(10, 15), a trenutno dodeljeni resursi (A) i ukupni zahtevi procesa U1, U2, U3 i U4 za resursima (C) dati u sledecoj tablici:

****  -2016 APR

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

12. Razmotriti sistem sa 150 jedinica memorije koje su dodeljene tri procesa na sledeci nacin:

Proces Max Drzi

1 70 45

2 60 40

3 60 15

Primeniti Bankarov algoritam da bi procenili da li je sigurno zadovoljiti sledece zahteve, i ukoliko jeste navesti sigurnu sekvencu procesa:

1. Proces 4 se startuje sa maksimalnim potrebama od 60 memorijskih jedinica, dok su inicijalne potrebe 25 memorijskih jedinica.
2. Proces 4 se startuje sa maksimalnim potrebama od 60 memorijskih jedinica, dok su inicijalne potrebe 35 memorijskih jedinica.
3. Nakon procesa 4 u pitanju a) startuje se proces 5 sa maksimalnim zahtevom od 35 memorijskih jedinica dok su inicijalne potrebe 5 memorijskih jedinica. - 2016 MAR

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

13. Ukoliko su ispunjena 4 uslova za nastanak uzajamnog blokiranja da li obavezno nastane uzajamno blokiranje? Objasniti. Sistem ima 5 procesa i 4 tipa resursa koji im se dodeljuju. Trenutna alokacija resursa i maksimalni zahtevi su sledeci. Da li se procesu P3 moze dodeliti jedna instanca resursa D, a da sistem bude u sigurnom stanju? Objasniti.

A C V

A B C D A B C D A B C D

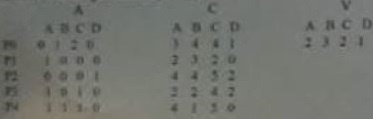
P0 0 1 2 0 3 4 4 1 2 3 2 1

P1 1 0 0 0 2 3 2 0

P2 0 0 0 1 4 4 5 2

P3 1 0 1 0 3 2 4 2

P4 1 1 1 0 4 1 3 0

 -2016 JAN (slika losa)