OPERATIVNI SISTEMI

V - Zastoj

V - Zastoj

S A D R Ž A J

5.1 - Sistemski model i osobine zastoja

- u višeprocesnoj okolini više procesa se mogu međusobno takmičiti za konačan broj resursa
- proces mora zahtevati resurs pre korišćenja, kao što ga mora i osloboditi nakon korišćenja
- u toku rada, proces može zahtevati više resursa
- u normalnom režimu rada, proces može da koristi resurs samo na jedan od sledeća tri načina
  - zahtev (*request*) - ukoliko resurs nije raspoloživ proces mora čekati njegovo oslobađanje
  - korišćenje (*use*)
  - oslobađanje (*release*) - obavezno nakon što proces završi sa korišćenjem resursa

5.1 - Sistemski model i osobine zastoja

- ukoliko proces P1 zahteva neraspoloživ resurs R1 on ulazi u stanje WAIT i postaje blokiran
- postavlja se pitanje da li blokirani proces P1 može zauvek ostati u tom stanju?
- ova pojava je moguća i prikazana je u sledećem primeru
  - pre ulaska u stanje WAIT procesu P1 je dodeljen i drugi resurs R2
  - resurs R2 ostaje neraspoloživ za druge procese
  - resurs R1 je već ranije dodeljen na korišćenje drugom procesu P2
  - proces P2 u toku vremena prelazi u stanje čekanja na neraspoloživ resurs R2
  - u ovoj situaciji niko ne oslobađa svoje resurse a traži nove - procesi ostaju zaglavljeni
- ovakva situacija naziva se **zastoj** (*deadlock)*
- zastoj treba prevashodno pokušati izbeći
- ako je ipak sistem doveden u stanje zastoja on se mora oporaviti

5.1 - Sistemski model i osobine zastoja

Uslovi pod kojima nastupa zastoj

- za nastajanje zastoja potrebno je da istovremeno budu ispunjena sledeća četiri uslova

1. **međusobno isključenje**
   - samo jedan proces u jednom trenutku može koristiti resurs ili jednu njegovu instancu
   - drugi proces koji zahteva taj resurs (instancu) mora da čeka dok se resurs ne oslobodi
2. **nema pretpražnjenja**
   - resurs se ne može nasilno oduzeti i predati drugom procesu
   - proces koji ga koristi mora da završi posao i oslobodi resurs
3. **uslov zadržavanja resursa i čekanja na drugi** (*hold and wait)*
   - proces drži jedan resurs
   - proces istovremeno čeka na dobijanje resursa koga koristi neki drugi proces

5.1 - Sistemski model i osobine zastoja

Uslovi pod kojima nastupa zastoj

- za nastajanje zastoja potrebno je da istovremeno budu ispunjena sledeća četiri uslova (nastavak)

4. **kružno čekanje** (*circular wait)*
   - postoji skup procesa {P0, P1,..., Pn} koji čekaju na resurse u kružnom poretku
   - Proces P0 čeka na resurs koga drži proces P1
   - Proces P1 čeka na resurs koga drži proces P2
   - ...
   - Proces Pn-1 čeka na resurs koga drži proces Pn
   - Proces Pn čeka na resurs koga drži proces P0

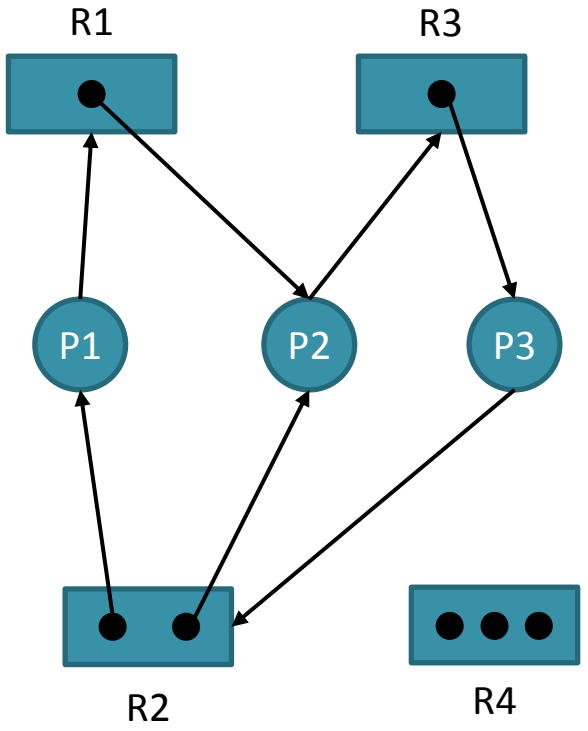
5.1 - Sistemski model i osobine zastoja

Graf dodeljenih resursa

- graf dodeljenih resursa (*resource allocation graph)* se sastoji od skupa objekata i skupa strelica E
- skup objekata se sastoji od
  - skupa svih **aktivnih procesa** u sistemu: P = {P1, P2, ..., Pn}
  - Skupa svih **raspoloživih resursa:** R = {R1, R2, ..., Rm}
- skup strelica se sastoji od
  - **strelica zahteva** (Pi $\rightarrow$ Rj)
    - proces Pi zahteva jednu instancu resursa Rj i čeka na nju
    - strelica se dodaje u graf uvek kada proces traži resurs
  - **strelica alokacije, tj. dodele** (Rj $\rightarrow$ Pi)
    - resurs Rj je dodeljen procesu Pi
    - strelica se dodaje u graf uvek kada se resurs dodeli procesu
- ako graf **ne sadrži** kružni tok, zastoja sigurno nema

5.1 - Sistemski model i osobine zastoja

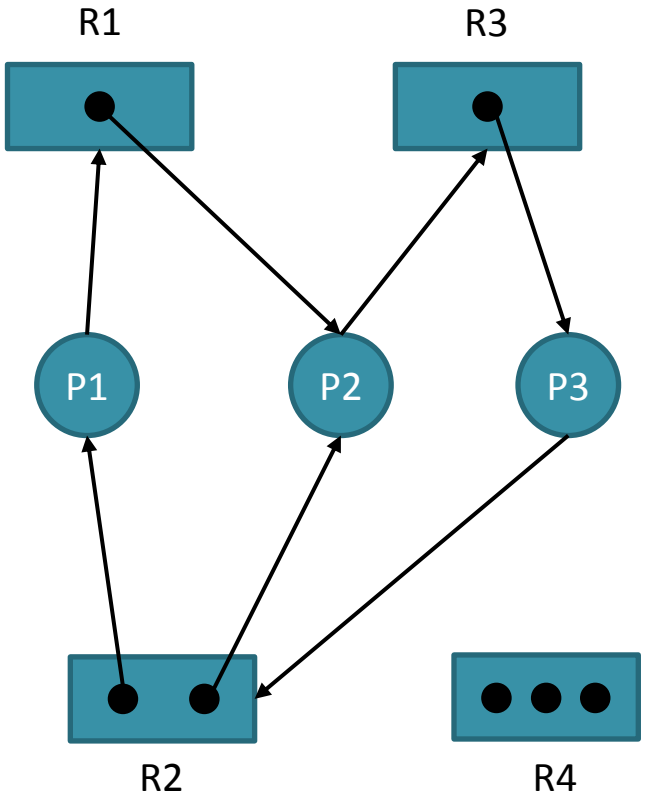
- Graf dodeljenih resursa
- ako graf sadrži bar jedan kružni tok moguće su dve situacije
  - ako svi resursi u kružnom toku sadrže tačno jednu instancu, zastoj se dogodio
  - ako resursi u kružnom toku sadrže više instanci može se dogoditi da zastoja nema
- primer
  - P = {P1, P2, P3}
  - R = {R1, R2, R3, R4}
  - resursi R1, R2, R3 i R4 redom imaju 1, 2, 1 i 3 instance
  - E = {P1 $\rightarrow$ R1, P2 $\rightarrow$ R3, P3 $\rightarrow$ R2, R1 $\rightarrow$ P2, R2 $\rightarrow$ P1, R2 $\rightarrow$ P2, R3 $\rightarrow$ P3}
  - da li je zastoj moguć?



5.1 - Sistemski model i osobine zastoja

Graf dodeljenih resursa

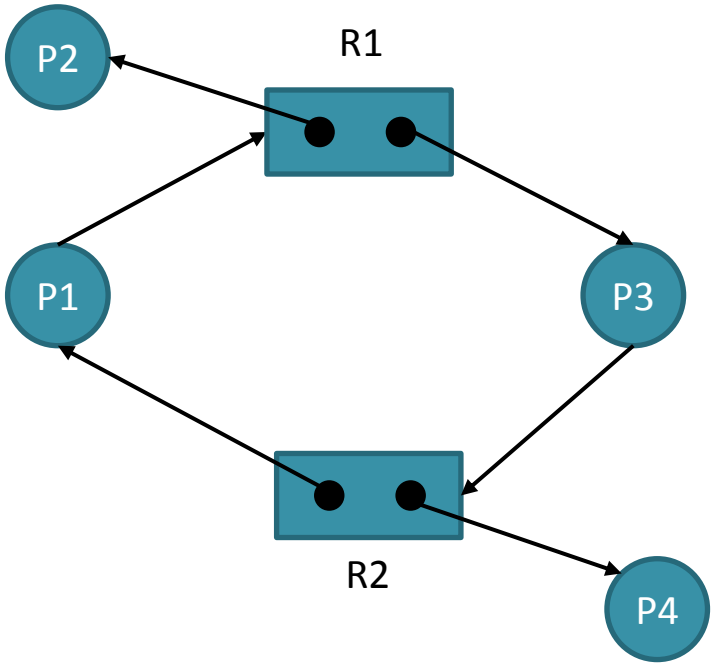
- na grafu postoje dve kružne putanje koje mogu izazvati zastoj
  - P1 → R1 → P2 → R3 → P3 → R2 → P1
  - P2 → R3 → P3 → R2 → P2
  - procesi P1, P2 i P3 su u zastoju
  - proces P2 čeka na R3 koga drži P3
  - proces P3 traži R2 koga drže P1 i P2
  - proces P2 je blokiran sa procesom P3 oko resursa R3
  - proces P1 traži R1 koga drži P2



5.1 - Sistemski model i osobine zastoja

Graf dodeljenih resursa

- kružni tok na grafu ne znači obavezno zastoj
- uočavamo kružni tok: P1 → R1 → P3 → R2 → P1
- oba resursa imaju više instanci tako da nema zastoja
- mogući scenario
  - proces P4 može da završi svoj posao i da oslobodi jednu instancu resursa R2
  - oslobođena instance resursa R2 se može dodeliti procesu P3
  - time se prekida krug i eliminiše zastoj



5.2 - Metode upravljanja zastojem

- problem upravljanja zastojem može se rešavati na tri načina
  - **sprečavanjem i izbegavanjem** zastoja (*deadlock prevention and avoidance)*
    - metode koje obezbeđuju da sistem nikada ne uđe u stanje zastoja
  - **detekcijom i oporavkom** od zastoja (*deadlock detection and recovery)*
    - metode koje dozvoljavaju sistemu da uđe u stanje zastoja
    - to stanje naknadno detektuju i oporavljaju sistem
  - **ignorisanjem problema** zastoja
    - neki OS se pretvaraju da se te pojave ne dešavaju
    - u tom slučaju se ne koriste prethodne dve metode

5.2 - Metode upravljanja zastojem

Prevencija zastoja

- zastoj se dešava ako su sva četiri prethodno navedena uslova istovremeno ispunjena
- zastoj se sprečava metodama koje obezbeđuju da se **najmanje jedan uslov ne ispuni**
- **međusobno isključenje**
  - resurse može da koristi samo jedan proces u jednom trenutku
  - obavezan uslov za sve nedeljive resurse (štampač na primer)
  - za deljive resurse nije obavezan
    - jedan metod sprečavanja je izbegavanje uslova međusobnog isključenja za sve deljive resurse

5.2 - Metode upravljanja zastojem

Prevencija zastoja

- **uslov zadržavanja resursa i čekanja na drugi**
- može se izbeći na dva načina
  - svaki proces mora da traži i alocira sve svoje resurse pre početka izvršavanja
  - proces može da traži resurs samo pod uslovom da ne drži ni jedan drugi
    - to znači da proces može koristiti samo jedan resurs u jednom trenutku
    - time se značajno smanjuje korišćenje U/I resursa
- **nema pretpražnjenja**
  - ovaj uslov se može poništiti korišćenjem sledećeg protokola
    - proces koji traži novi neraspoloživ resurs otpušta sve zauzete resurse i prelazi u WAIT
    - proces može preći u stanje READY tek kada povrati sve resurse koje je posedovao i resurse koje je tražio na korišćenje
    - o ovom slučaju proces ne može preći u stanje WAIT sa zauzetim resursima
  - drugi način bi bio nasilno oduzimanje resursa od drugih procesa koji su blokirani a drže te resurse

5.2 - Metode upravljanja zastojem

Prevencija zastoja

- **kružno čekanje**
  - može se izbeći na sledeći način
    - svakom resursu se dodeli broj *N* iz skupa prirodnih brojeva N=F(Ri)
    - procesi mogu zahtevati resurse u strogo rastućem redu
    - proces koji je zahtevao resurs Ri može zahtevati resurs Rj samo ako je F(Rj)>F(Ri)
    - poslednji proces u nizu ne može da se vrati unazad i traži resurs koga drži prvi proces

5.3 - Izbegavanje zastoja

- izbegavanje zastoja je moguće ako OS **unapred** ima informacije o resurima koje će procesi zahtevati
- sistem koji zna šta procesi žele može da napravi redosled opsluživanja zahteva tako da se izbegne zastoj
- najprostija šema izbegavanja zastoja
  - od svakog procesa se traži da deklariše **najveći broj potrebnih resursa** svakog tipa
  - na osnovu toga se može konstruisati algoritam koji će sprečiti stanje zastoja
  - algoritam dinamički ispituje **trenutno stanje dodeljenih resursa**
    - broj raspoloživih resursa
    - broj dodeljenih resursa
    - broj maksimalno traženih resursa u jednom trenutku vremena
  - tako se osigurava da sistem nikada ne uđe u stanje kružnog čekanja

5.3 - Izbegavanje zastoja

Bezbedno stanje

- kada proces traži neki resurs, sistem mora da proceni da li će dodela tog resursa ostaviti sistem u bezbednom stanju (*safe state)*
- stanje je **bezbedno** ako sistem može može alocirati resurse svakom procesu u nekom poretku i još uvek izbeći zastoj
- bezbedno stanje je stanje bez zastoja
- stanje koje **nije bezbedno**
  - ne mora biti zastoj
  - može težiti ka zastoju
  - u tom stanju procesi mogu lako dovesti sistem u zastoj
- sistem je u bezbednom stanju ako postoji bezbedna sekvenca svih procesa

5.3 - Izbegavanje zastoja

Bezbedno stanje

- sekvenca <P1, P2, ..., Pn> je **bezbedna** ako za svaki proces Pi u svakom trenutku važi
  - resursi koje Pi može još tražiti mogu da se zadovolje iz trenutno raspoloživih resursa
  - to uključuje i resurse koji pripadaju procesima Pj pokrenutim pre procesa Pi (j<i)
  - ako resursi nisu trenutno raspoloživi Pi čeka da svi Pj završe poslove i oslobode resurse
- **primer**
- tri procesa (P0, P1 i P2) imaju na raspolaganju 12 resursa istog tipa
- inicijalno je P0 dobio 5 resursa, P1 i P2 su dobili po 2 resursa
- maksimalne potrebe procesa P0, P1 i P2 su redom 10, 4 i 9 resursa

5.3 - Izbegavanje zastoja

Bezbedno stanje

- **primer**
- ispitivanje bezbednosti sekvence procesa <P1,P0,P2>
- u tabeli je prikazano stanje angažovanih i raspoloživih resursa tokom izvođenje procesa po navedenom redosledu

Korak	Sekvenca procesa	Broj angažovanih resursa od strane procesa			
		P0	P1	P2	Slobodni resursi
0	inicijalno stanje	5	2	2	3
1	nakon aktiviranja P1	5	4	2	1
2	nakon završetka P1	5	0	2	5
3	nakon aktiviranja P0	10	0	2	0
4	nakon završetka P0	0	0	2	10
5	nakon aktiviranja P2	0	0	9	3

5.3 - Izbegavanje zastoja

Bezbedno stanje

- **primer**
- inicijalno postoje 3 slobodna resursa
  - P1 može da zadovolji svoje maksimalne potrebe (uzima još 2 resursa)
  - P1 završava posao i oslobađa 4 resursa
- ostaje 5 slobodnih resursa
  - P0 može da zadovolji svoje maksimalne potrebe (uzima još 5 resursa)
  - P0 završava posao i oslobađa 10 resursa
- ostaje 10 slobodnih resursa
  - P2 može da zadovolji svoje maksimalne potrebe (uzima još 7 resursa)

5.3 - Izbegavanje zastoja

Bezbedno stanje

- **primer**
- iz prethodih podataka vidljivo je da je sekvenca procesa <P1,P0,P2> bezbedna
  - svaki proces je u momentu svog izvođenja imao dovoljan broj resursa na raspolaganju
- situacija bi međutim bila drugačija da je u inicijalnom stanju P2 dobio 3 resursa
  - sistem bi bio tada doveden u nestabilno stanje
  - nakon završetka rada proces P1 bi vratio resurse koje je koristio
  - tada bi na raspolaganju u sistemu bilo ukupno 4 resursa
  - to je nedovoljno da proces P0 započne ra radom, jer je njemu za to potrebno 5 dodatnih resursa
  - nastao bi zastoj

5.3 - Izbegavanje zastoja

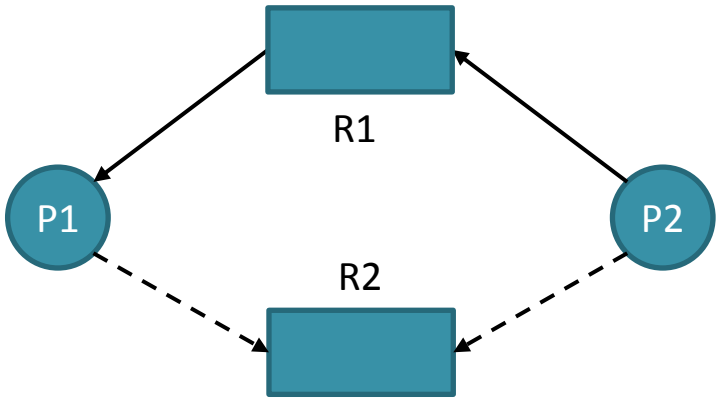
Graf dodele resursa za izbegavanje zastoja

- pored prethodno navedenih strelica zahteva i strelica dodele, uvodi se **strelica mogućih zahteva** (*claim edge)*
- ova strelica znači da proces može zahtevati resurs u nekom budućem trenutku
- u grafu se predstavlja isprekidanom linijom
- zahtevanje i oslobađanje resursa
  - u trenutku u kome proces zahteva resurs
    - mogući zahtev postaje stvarni zahtev za resurs
    - strelica mogućeg zahteva menja se strelicom zahteva
  - kada proces oslobodi resurs strelica dodele menja se strelicom mogućih zahteva
- svi mogući zahtevi moraju biti poznati pre nego što proces otpočne izvršavanje

5.3 - Izbegavanje zastoja

Graf dodele resursa za izbegavanje zastoja

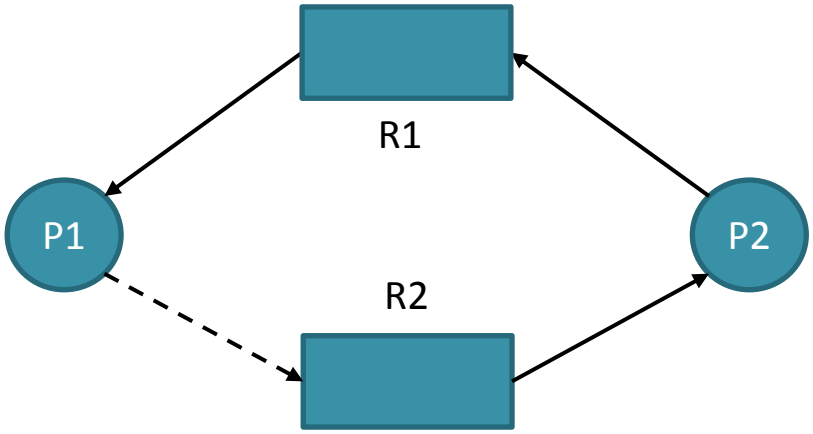
- to će omogućiti da se nacrta **graf sa mogućim zahtevima procesa**
  - na bazi tog grafa se odlučuje da li će sistem dozvoliti zahtev ili ne
  - ukoliko se ispunjenjem zaheva formiraju kružni tokovi, sistem se može dovesti u nebezbedno stanje pa se procesu ne ispunjava zahtev
- u priloženom primeru se vidi da je resurs R2 slobodan, ali i da postoje mogući zahtevi od procesa P1 i P2



5.3 - Izbegavanje zastoja

Graf dodele resursa za izbegavanje zastoja

- ako bi proces P2 izdao zahtev za korišćenjem R2, sistem to ne bi smeo da dozvoli
  - u slučaju da sistem to dozvoli, dodeljuje se resurs R2 procesu P2 i pojaviće se kružni tok
  - ako i P1 zatraži R2
    - dolazi do zastoja



5.3 - Izbegavanje zastoja

Bankarski algoritam

- prethodno opisani graf koji sadrži strelice mogućih zahteva nije podesan za resurse koji imaju **više instanci**
- u tim situacijama za izbegavanje zastoja koristi se **bankarski algoritam** (*Banker's algorithm)*
- algoritam se može primeniti ukoliko su ispunjeni sledeći uslovi
  - resursi imaju uglavnom više instanci
  - svaki proces unapred deklariše najveći broj instanci svakog resursa koji želi da koristi
  - kada proces zahteva resurse sistem procenjuje da li će posle toga ostati u stabilnom stanju
    - ako ostaje u stabilnom stanju proces će dobiti resurs
    - u suprotnom proces mora da sačeka da drugi procesi oslobode neke resurse
  - proces koji dobije resurse mora da ih vrati u nekom konačnom vremenu

5.3 - Izbegavanje zastoja

Bankarski algoritam

- broj procesa: n, broj resursa: m
- strukture podataka za bankarski algoritam
  - **vektor rapoloživosti:** raspoloživo[j], j∈[0,m]
    - ako je raspoloživo[j]=k, tada je k instanci resursa Rj raspoloživo
  - **matrica maksimalnih zahteva:** maksimalno[n,m]
    - ako je maksimalno[i,j] = k, tada proces Pi može tražiti ukupno k instanci resursa Rj
  - **matrica alokacije:** dodela[n,m]
    - ako je dodela[i,j] = k, tada je proces Pi trenutno dobio k instanci resursa Rj
    - vrsta matrice alokacije je vektor dodela[i] (svi resursi koje je proces Pi dobio)
  - **matrica potreba:** potreba[n,m]
    - ako je potreba[i,j] = k, tada proces Pi može tražiti još k instanci resursa Rj
    - vrsta matrice potreba je vektor potreba[i] (svi resursi koje proces Pi može tražiti)

5.3 - Izbegavanje zastoja

Bankarski algoritam

- važi jednakost: potreba[i,j] = maksimalno[i,j] - dodela[i,j]
- uvodimo definiciju: za dva vektora X i Y od n elemenata važi X<Y ako je X[i]<Y[i] za svako $i \in [1,n]$
- neka su **rad** i **kraj** vektori dužine m i n, respektivno
- sledeći algoritam **određuje da li se sistem nalazi u bezbednom stanju**

1. inicijalizacija
   - rad = raspoloživo (vektor raspoloživih resursa)
   - kraj[i] = 0 za $i \in [1,n]$ (opisuje završetak procesa i)
2. pronalaženje procesa Pi koji može da zadovolji svoje potrebe, odnosno procesa za koji važi
   - (2.a) kraj[i] = 0
   - (2.b) potreba[i] $\leq$ rad
   - ako nema takvih procesa idi na korak 4

5.3 - Izbegavanje zastoja

Bankarski algoritam

3. procesu se dodeljuju svi potrebni resursi nakon čega on završava aktivnost i vraća ih u sistem
   - rad = rad + dodela[i] (oslobađanje resursa)
   - kraj[i] = 1
   - ići na korak 2
4. ako je kraj[i] = 1 za sve i, svi procesi mogu da završe posao pa je sistem u stabilnom stanju

- algoritam zahteva m.n2 operacija kako bi proverio da li je sistem u stabilnom stanju

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer provere bezbednog stanja**
- u sistemu se nalaze tri procesa (P0, P1 i P2) i resurs A sa 12 instanci
- stanje sistema je dato sledećom tabelom

proces	dodela	maksimalno	potreba
P0	5	10	5
P1	2	4	2
P2	2	9	7

- resurs A ima **3 slobodne instance**
- uvek se polazi od procesa sa manjim zahtevima
- kasnije se rešavaju problemi najzahtevnijih procesa

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer provere bezbednog stanja**
- nakon toga proces P0 uzima još pet instanci resursa, a zatim ih vraća

proces	dodela	maksimalno	potreba	raspoloživo
P0	10	10	0	0
P1	4	4	0	
P2	2	9	7	

proces	dodela	maksimalno	potreba	raspoloživo
P0	KRAJ			10
P1	KRAJ			
P2	2	9	7	

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer provere bezbednog stanja**
- na kraju, proces P2 uzima još sedam instanci resursa, koje po završetku vraća

proces	dodela	maksimalno	potreba	raspoloživo
P0	KRAJ			3
P1	KRAJ			
P2	9	9	7	

proces	dodela	maksimalno	potreba	raspoloživo
P0	KRAJ			12
P1	KRAJ			
P2	KRAJ			

- sekvenca <P1, P0, P2> dovešće do zadovoljenja potreba svih procesa
- prema tome, sistem je u bezbednom stanju

5.3 - Izbegavanje zastoja

Bankarski algoritam

- Kada proces zahteva resurs, primenjuje se **algoritam za obradu zahteva i dodelu resursa**
- sa $zahtev_i$ označićemo vektor trenutnih zahteva procesa Pi
  - ako je $zahtev_i[j]=k$ tada proces Pi traži k instanci resursa Rj
- pre ulaska u algoritam za izbegavanje zastoja operativni sistem proverava
  - da li proces traži više od onoga što je specificirao ($zahtev_i \geq potreba_i$)
    - ako traži, prijavljuje se greška i odbija se zahtev
  - da li su resursi raspoloživi ($zahtev_i \leq$ raspoloživo)
    - ako nisu, OS uvodi proces u stanje čekanja

5.3 - Izbegavanje zastoja

Bankarski algoritam

- ako je $zahtev_i \geq potreba_i$ i $zahtev_i \leq$ raspoloživo obavlja se kvazi-dodela resursa
  - raspoloživo = raspoloživo - $zahtev_i$ (broj raspoloživih resursa)
  - dodela = dodela + $zahtev_i$ (broj dodeljenih resursa)
  - $potreba_i$ = $potreba_i$ - $zahtev_i$ (potrebe procesa)
- nakon zamišljene dodele resursa ispituje se da li sistem ostaje u bezbednom stanju
- ako ostaje, sistem će procesu Pi dodeliti resurse (alokacija postaje stvarna)
  - u protivnom proces Pi mora da čeka

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer algoritma za obradu zahteva i dodelu resursa**
- u sistemu se nalazi pet procesa (P0 do P4) i četiri resursa (A, B, C i D)
- u trenutku t_0 je raspoloživo = (1,5,2,0)
- stanje sistema u trenutku t_0 opisano je sledećom tabelom

	dodela				maksimalno				potreba			
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

- da li je sistem u bezbednom stanju?
- da li će sistem da odobri procesu P1 $zahtev_i$ = (0, 4, 2, 0)?

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer algoritma za obradu zahteva i dodelu resursa**
- 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)
- sekvenca je određena tako što se bira proces sa najmanjim potrebama pod uslovom da one mogu da se zadovolje raspoloživim resursima
  - moguće da su zbirne potrebe nekog procesa manje ali da je potreba u odnosu na pojedinačni resurs veća od raspoloživog stanja, tada se razmatra naredni proces sa najmanjim potrebama

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer algoritma za obradu zahteva i dodelu resursa**
- nakon što je potvrđeno da je sistem je u bezbednom stanju, razmatra se odobrenje zahteva procesa P1 $zahtev_i$ = (0, 4, 2, 0)
- najpre se proverava da li je $zahtev_i$ < raspoloživo
  - pošto je (0,4,2,0) < (1,5,2,0) uslov je ispunjen
- obavlja se kvazi-dodela resursa na osnovu zahteva
  - procesu P1 pridodajemo (0,4,2,0)
  - nakon dodele je raspoloživo =(1,1,0,0)

	dodela				maksimalno				potreba			
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

5.3 - Izbegavanje zastoja

Bankarski algoritam

- **primer algoritma za obradu zahteva i dodelu resursa**

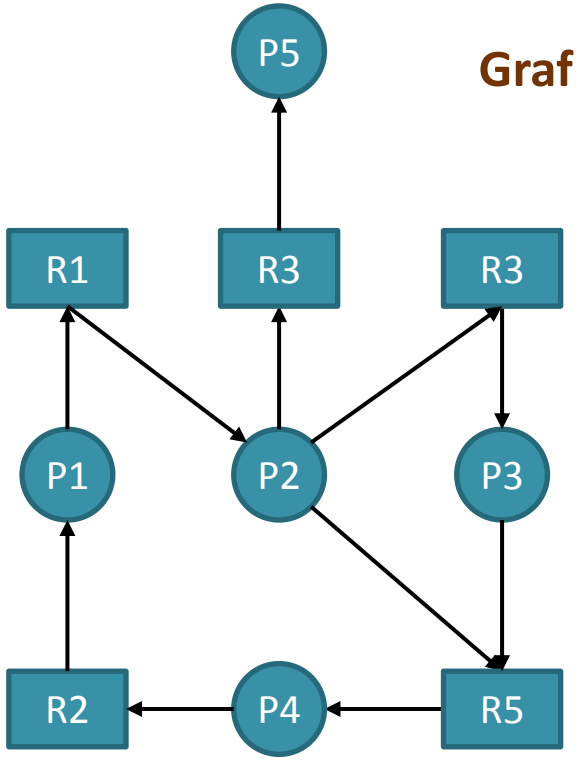
	dodela				maksimalno				potreba			
Proces	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	0	0	0	0
P1	1	4	2	0	1	7	5	0	0	3	3	0
P2	1	3	5	4	2	3	5	6	1	0	0	2
P3	0	6	3	2	0	6	5	2	0	0	2	0
P4	0	0	1	4	0	6	5	6	0	6	4	2

  - sistem nakon kvazi-dodele ostaje u bezbednom stanju, jer sledeća sekvenca procesa zadovoljava uslove stabilnosti
    - P0 uzima (0,0,0,0) a zatim vraća (0,0,1,2) → raspoloživo =(1,1,1,2)
    - P2 uzima (1,0,0,2) a zatim vraća (2,3,5,6) → raspoloživo =(2,4,6,6)
    - P3 uzima (0,0,2,0) a zatim vraća (0,6,5,2) → raspoloživo =(2,10,9,8)
    - P1 uzima (0,3,3,0) a zatim vraća (1,7,5,0) → raspoloživo =(3,14,11,8)
    - P4 uzima (0,6,4,2) a zatim vraća (0,6,5,6) → raspoloživo =(3,14,12,12)
- sistem **odobrava** zahtev za dodelom resursa

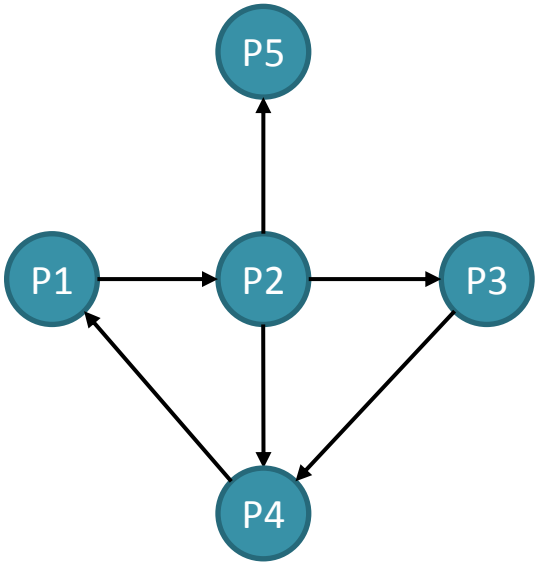
5.4 - Detekcija i oporavak od zastoja

- ako sistem ne primenjuje algoritme za sprečavanje ili izbegavanje zastoja, tada su zastoji sasvim mogući, čak i vrlo verovatni
- u tom slučaju sistem mora da obezbedi dve stvari
  - algoritam za **detekciju zastoja** (ispituje da li se zastoj dogodio)
  - algoritam za **oporavak iz stanja zastoja**
- u slučaju da svi resursi imaju samo jednu instancu, za detekciju zastoja se koristi **graf čekanja na resurse** (*wait-for graph)*
  - konstruiše se od grafa dodeljenih resursa
    - čvorovi grafa su samo procesi (nema resursa)
    - strelice se crtaju samo između procesa koji čekaju jedan drugog za resurse

5.4 - Detekcija i oporavak od zastoja



Graf čekanja na resurse



- postojanje zastoja utvrđuje se ispitivanjem postojanja **kružnih tokova**
  - ako kružni tok postoji to je zastoj
  - algoritam se periodično poziva da ispita postojanje kružnih tokova
  - za ispitivanje je potrebno n^2 operacija, pri čemu je n broj strelica u grafu čekanja

5.4 - Detekcija i oporavak od zastoja

Detekcija zastoja u slučaju da resursi imaju više instanci

- ukoliko resursi imaju više instanci za detekciju zastoja se koristi algoritam sličan bankarskom
- strukture podataka
  - **vektor rapoloživosti:** raspoloživo[j], je[0,m]
    - ako je raspoloživo [j]=k, tada je k instanci resursa Rj raspoloživo,
  - **matrica alokacije:** dodela [n,m]
    - Ako je dodela[i,j] = k, tada je proces Pi trenutno dobio k instanci resursa Rj
  - **matrica trenutnih zahteva:** zahtev[n,m]
    - Ako je zahtev [i,j] = k, tada proces Pi trenutno traži k instanci resursa Rj

5.4 - Detekcija i oporavak od zastoja

Detekcija zastoja u slučaju da resursi imaju više instanci

- neka su **rad** i **kraj** vektori dužine m i n, respektivno
- sledeći algoritam **određuje da li se sistem nalazi u stanju zastoja**

1. inicijalizacija
   - rad = raspoloživo
   - kraj[i] = {0, ako je dodelai $\neq$ 0; inače je 1} za i$\in$[1,n]
2. pronalaženje procesa Pi koji može da zadovolji svoje potrebe, odnosno procesa za koji važi
   - (2.a) kraj[i] = 0
   - (2.b) zahtevi $\leq$ rad
   - ako nema takvih procesa idi na korak 4
3. u ovom koraku se resursi procesa vraćaju u sistem
   - rad = rad + $dodela_i$ (oslobađanje resursa)
   - kraj[i] = 1
   - ići na korak 2

5.4 - Detekcija i oporavak od zastoja

Detekcija zastoja u slučaju da resursi imaju više instanci

- sledeći algoritam **određuje da li se sistem nalazi u stanju zastoja** (nastavak)

4. ako je kraj [i] = 0 za sve i$\in$[1,n], sistem je u zastoju (preciznije proces Pi je u zastoju)

- algoritam zahteva m.n2 operacija kako bi proverio da li je sistem u stanju zastoja

5.4 - Detekcija i oporavak od zastoja

Detekcija zastoja u slučaju da resursi imaju više instanci

- **primer**
- sistem sa pet procesa (P0 do P4) i tri resursa: A (7 instanci), B (2 instance) i C (6 instanci)
- u trenutku t_0 nema raspoloživih resursa, odnosno raspoloživo = (0,0,0)
- stanje sistema u trenutku t_0 opisano je sledećom tabelom

Vektor	dodela			zahtev		
Resurs	A	B	C	A	B	C
P0	0	1	0	0	0	0
P1	2	0	0	2	0	2
P2	3	0	3	0	0	0
P3	2	1	1	1	0	0
P4	0	0	2	0	0	2

5.4 - Detekcija i oporavak od zastoja

Detekcija zastoja u slučaju da resursi imaju više instanci

- **primer**
- kada se primeni algoritam za detekciju
  - dokazuje se da postoji sekvenca <P0, P2, P3, P1, P4> koja dovodi do rezultata kraj [i]=1 za sve procese
  - to znači da sistem **neće** biti doveden u stanje zastoja
- interesantno
  - sistem u trenutku t_0 nema raspoloživih resursa
  - počevši od procesa koji najmanje traže (P0 i P2) može se obaviti dodela resursa bez dovođenja do zastoja

5.4 - Detekcija i oporavak od zastoja

Detekcija zastoja u slučaju da resursi imaju više instanci

- **primer**
- ukoliko uvedemo malu izmenu: proces P2 traži jednu instancu resursa C
  - proces P0 ima najmanje prohteve, završava, vraća resurse → raspoloživo = (0,1,0)
  - nakon toga nemamo proces koji može da zadovolji svoje potrebe
  - dakle, sistem je u stanju zastoja
    - procesi P1, P2, P3, i P4 su zaglavljeni

Vektor	dodela			zahtev		
Resurs	A	B	C	A	B	C
P0	0	1	0	0	0	0
P1	2	0	0	2	0	2
P2	3	0	3	0	0	**1**
P3	2	1	1	1	0	0
P4	0	0	2	0	0	2

5.4 - Detekcija i oporavak od zastoja

Oporavak od zastoja

- posle detekcije zastoja potrebno je izvršiti oporavak
- dve metode za oporavak od zastoja koje se najčešće koriste su
- **prekid izvršenja procesa u zastoju**
  - oslobađaju se svi resursi koje je zauzimao proces koji se prekida
  - postoje dve tehnike:
    - prekid izvršenja **svih** zaglavljenih procesa
    - prekid izvršenja procesa **do razbijanja kružnog toka**
- **nasilno oduzimanje resursa od procesa koji su u zastoju**
  - tehnika koja ne prekida izvršenje procesa
  - određenom procesu (ili procesima) u zastoju se oduzimaju resursi
  - odabir žrtve sa najmanje gubitaka