UVIAZNUTIE PROCESOV

Uviaznutie procesov

- Model
  - Procesy
  - Prostriedky - pamäť, CPU, súbory, V/V zariadenia, aj viacej jednotiek určitého typu
- Protokol použitia prostriedkov
  - Požiada o prostriedok.
    - Ak sa nemôže požiadavke vyhovieť ihneď (ak napr. tento prostriedok práve používa iný proces), potom proces musí čakať na uvoľnenie prostriedku.
  - Použije prostriedok.
  - Uvoľní prostriedok

Definícia uviaznutia

*Množina procesov je v stave **uviaznutia**, ak každý proces z množiny čaká na udalosť, ktorú môže vyvolať len iný proces z tejto množiny*

Nutné podmienky pre uviaznutie

1. **Vzájomné vylúčenie.** Aspoň jeden prostriedok musí byť pridelený výlučne, to znamená, že nemôže byť zdieľaný.
2. **Vlastniť a žiadať.** Musí existovať proces, ktorý má pridelený aspoň jeden prostriedok a požaduje ďalšie prostriedky, ktoré sú pridelené iným procesom.
3. **Používanie bez preempcie**. Prostriedok nemôže byť odňatý, t.j. proces môže uvoľniť prostriedok jedine dobrovoľne, keď s ním ukončí prácu.
4. **Kruhové čakanie.** Musí existovať množina *P0 , P1 . . . Pn* čakajúcich procesov takých, že *P0* čaká na prostriedok, ktorý drží *P1* , *P1* čaká na prostriedok, ktorý drží *P2, ..., Pn-1* čaká na prostriedok, ktorý drží *Pn* a *Pn* čaká na prostriedok, ktorý drží *P0* .

Pre vznik uviaznutia musia platiť všetky štyri podmienky súčasne.

Prístupy k riešeniu problému uviaznutia

- **Prevencia**
  - Uviaznutie sa nemôže vyskytnúť
- **Detekcia & Zotavenie**
  - Uviaznutie sa môže vyskytnúť, ale zotavenie je možné
- **Vyvarovať sa**
  - Uviaznutie sa môže vyskytnúť, ale sú algoritmy ako sa mu vyhnúť

Prevencia[1]

- Princíp
  - Ak zaistíme , že **trvale nebude platiť aspoň jedna z podmienok** pre uviaznutie, zaistíme aj **prevenciu** pred uviaznutím.

    Vzájomné vylúčenie

    Vlastniť a žiadať

    Používanie bez preempcie

    Kruhové čakanie

Prevencia$_2$

- **Vzájomné vylúčenie**
  - **musí platiť aj pre nezdieľateľné prostriedky**. Napr. tlačiareň nemôže byť zdieľaná medzi niekoľkými procesmi.
  - zdieľateľné prostriedky nepotrebujú vylúčenie súčasného prístupu. Pre nich procesy nečakajú.
  - **prevencia pred uviaznutím sa nedá dosiahnuť zákazom výlučného prideľovania prostriedkov**, pretože niektoré prostriedky sú svojou povahou nezdieľateľné.

Prevencia3

- **Vlastniť a žiadať**
  - Podmienka nebude nikdy platiť, ak proces , ktorý žiada o prostriedok, nevlastní žiadny iný prostriedok.
  - Možnosti:
    - proces požiada o všetky prostriedky naraz pred svojim zahájením
    - proces môže žiadať o prostriedok, len ak nevlastní žiadny iný

  Dva základné nedostatky:
    - Prvý - malé využitie prostriedkov, lebo tie môžu byť pridelené, ale sú dlho nevyužívané.
    - Druhý nedostatok je, že môže nastať starvácia.

Prevencia4

- **Zákaz preempcie**
  - Niektoré prostriedky sú preemptívne:
    - pamäť,
    - priestor na disku.
  - Selektívna preempcia.
    - prostriedky môžu byť odobraté od procesov, ktoré majú pridelené niektoré prostriedky, ale čakajú na ďalšie.

Prevencia$_5$

- **Kruhové čakanie**
  - zoradiť všetky typy prostriedkov :
    - funkcia *F*: *R* --> *N*, kde *N* je množina prirodzených čísel, **donútiť procesy požadovať prostriedky podľa vzostupného poradia číslovania.**
  - každý proces žiada o prostriedky len vo vzostupnom poradí číslovania, t.j. ak proces požiadal o *Ri*, potom môže žiadať len o prostriedky typu *Rj*, pre ktoré platí $F(Rj) > F(Ri)$.
    - Rovnaké prostriedky majú rovnaké číslo.

Ako funguje porušenie kruhového čakania?

- Predpokladajme, že kruhové čakanie pozostáva z procesov

  $P_1, P_2, ..., P_n, P_1$

- Nech P(i) čaká na prostriedok, ktorý má pridelený P(i+1) s číslom R(i)
  - P(i+1) musí mať všetky prostriedky š číslom R(i)
  - Takže musí čakať na $R(i+1) > R(i)$
  - Pretože nemôže existovať cyklus v rámci narastajúcich čísel, to znamená že taký cyklus neexistuje

Ekvivalentná stratégia

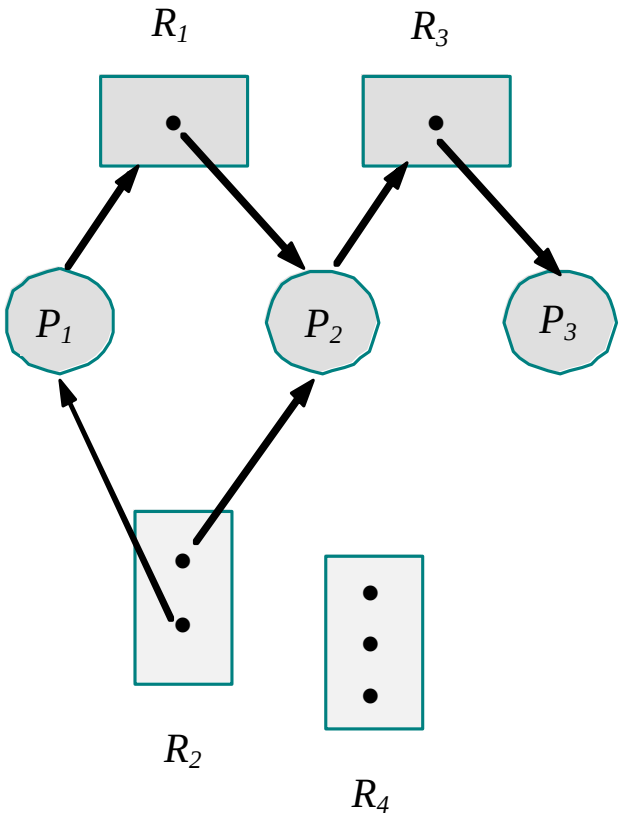
- Ekvivalentná stratégia je keď proces, ktorý žiada prostriedok s určitým číslom uvoľní všetky prostriedky s vyšším číslom.
- Typické číslovanie kopíruje prirodzené používanie prostriedkov
  - Napr. disky sú obyčajne používané pred tlačiarňami a majú priradené menšie číslo.

Detekcia & Vyhnutie sa

- Reprezentácia
  - Graf prideľovania prostriedkov
  - Čakací graf
- Algoritmy

Reprezentácia grafom

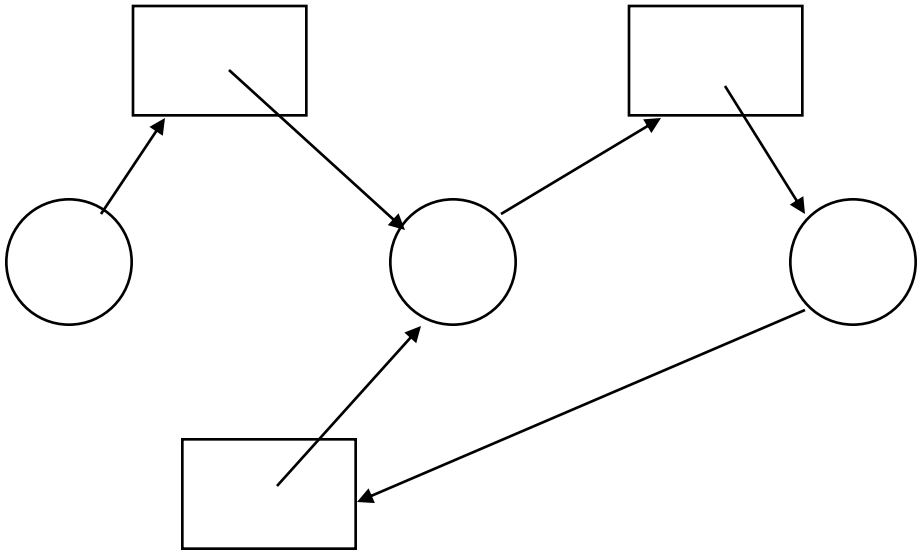
- Graf prideľovania prostriedkov(GPP)
  - 2 typy uzlov
    - Procesy
    - Prostriedky
  - Tri typy orientovaných hrán
    - Hrana požadovania
    - Hrana pridelenia



Graf prideľovania prostriedkov

Graf prideľovania prostriedkov

- Jeden prostriedok z daného typu



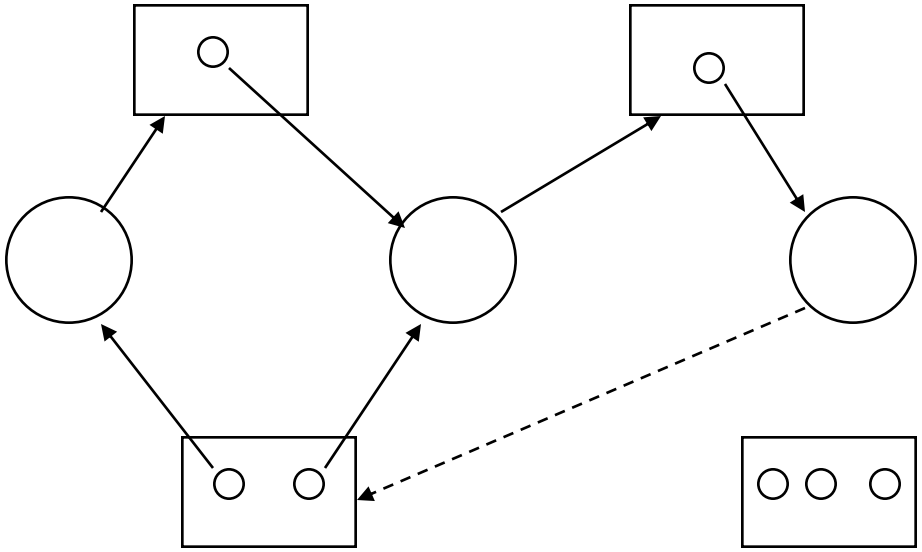
Prostriedky

Procesy

Prostriedky

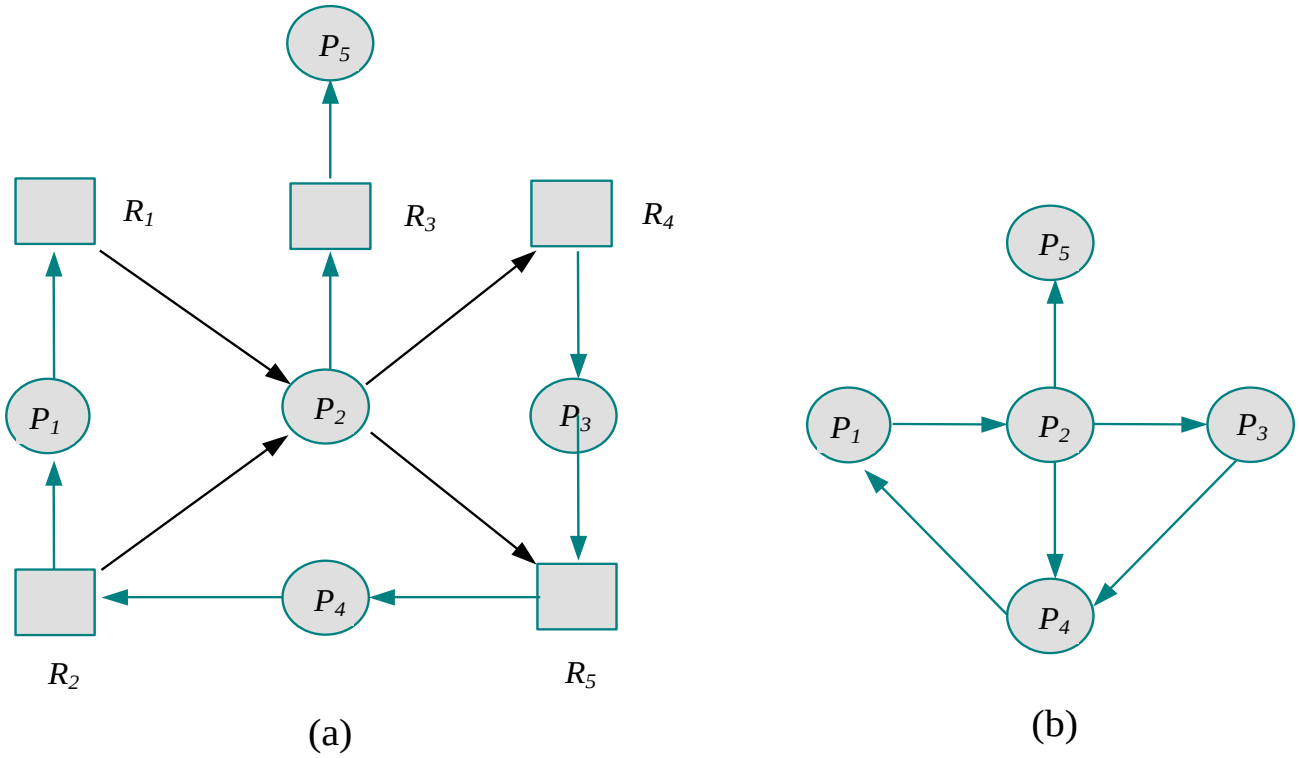
Graf prideľovania prostriedkov

- Viacej prostriedkov z daného typu



Detekcia uviaznutia

- Použijeme GPP, kde
  - sú definované hrany požiadaviek a pridelenia - ***čakací graf*** .



a) Graf prideľovania prostriedkov b) Čakací graf

Detekcia uviaznutia2

- Čakací graf
- Dva prípady
  - **jeden prostriedok** z každého typu
    - ***uviznutie je ak v grafe je slučka!!!***
    - Algoritmus pre odhalenie cyklu v grafe vyžaduje rádovo $\mathbf{n^2}$ operácií, kde **n** je počet uzlov v grafe
  - **viacej prostriedkov** z každého typu
    - ***slučka neznamená uviaznutie!!!***
    - treba použiť algoritmus na detekciu uviaznutia

Detekcia uviaznutia3

- Algoritmus pre detekciu uviaznutia
  - Prípad **viacerých prostriedkov z jedného typu**
    - údajové štruktúry
      - ***prístupné***: vektor s dĺžkou *m*, - počet prístupných prostriedkov z každého typu.
      - ***pridelené***: matica $n \times m$, - počet prostriedkov z každého typu, ktoré sú pridelené každému procesu.
      - ***zostáva***: matica $n \times m$, - počet prostriedkov z každého typu, ktoré ešte požaduje každý proces.

Algoritmus pre detekciu uviaznutia

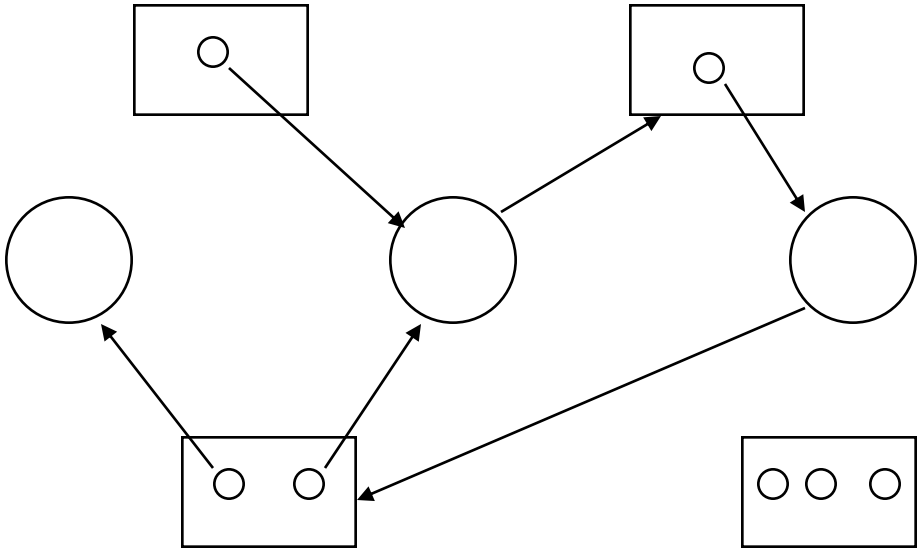
Použijeme *prístupné: vektor s dĺžkou m*
pridelené: matica $n \times m$
zostáva: matica $n \times m$

1. **Inicializujeme *pracovné* := *prístupné*.**
   Pre $i = 1, 2, ..., n$ platí *dokončené* = *false*,
   ak *pridelené*$_i \neq 0$, ináč sa *dokončené* = *true*.
2. **Nájdeme index *i* pre ktorý platí:**
   dokončené = *false* a *požadované[i]* $\leq$ *pracovné*
   Ak taký index neexistuje, ideme na krok 4.
3. **Zvýšime počet pracovných prostriedkov**
   pracovné := *pracovné* + *pridelené[i]*
   dokončené[i]:= *true*
   Ideme na krok 2.
4. Ak *dokončené[i]* = *false* pre niektoré i, $1 \leq i \leq n$, **potom systém je v stave uviaznutia.** Navyše,
   ak *dokončené[i]* = *false*, - práve proces P_i je uviaznutý.

Algoritmus vyžaduje rádovo $\boldsymbol{m \times n^2}$ operácií

Viacej prostriedkov z daného typu

- Príklad: v grafe slučka, ale nie je uviaznutie



Viacej prostriedkov z daného typu2

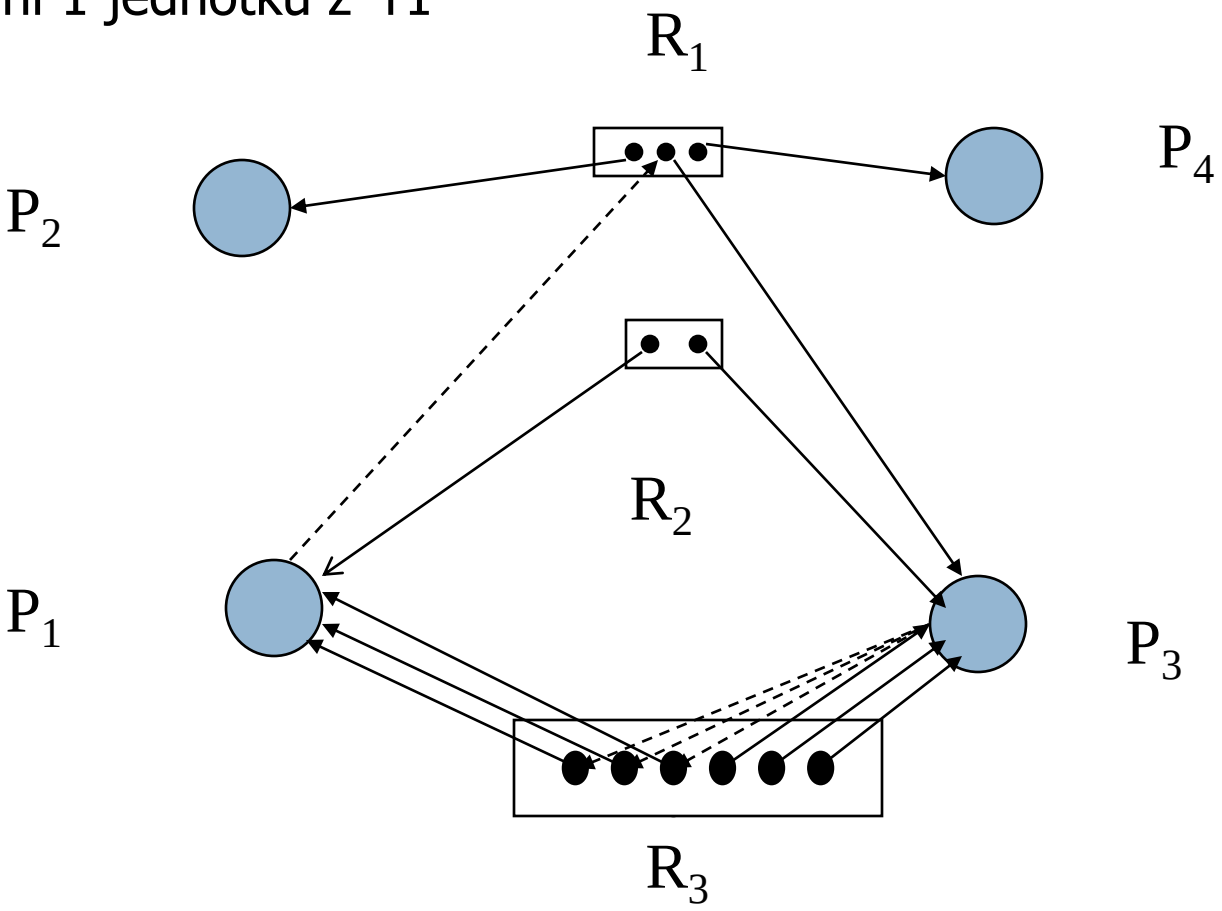
- **Príklad**
  - Všetky prostriedky 3 2 6

	Pridelené	Ďalej požadované
P1	0 1 3	1 0 0
P2	1 0 0	0 0 0
P3	1 1 3	0 0 3
P4	1 0 0	0 0 0

- **Je tu slučka, ale nie je uviaznutie!!!**
  - P1, r1, P3, r3, P1 (pozri obrázok na ďalšej strane)
- Prečo ?

Pretože existuje postupnosť vykonania procesov, ktorá nevylučuje ukončenie všetkých procesov

- skončí P4, uvoľní 1 jednotku z r1
- pridelí sa P1
- skončí P1, uvoľní 1 jednotku z r2 a 3 jednotky z r3
- pridelí sa P2
- skončí P2, uvoľní 1 jednotku z r1
- pridelí sa P3



Zmena v pridelení prostriedkov

- Príklad – zmeníme počet pridelených prostriedkov

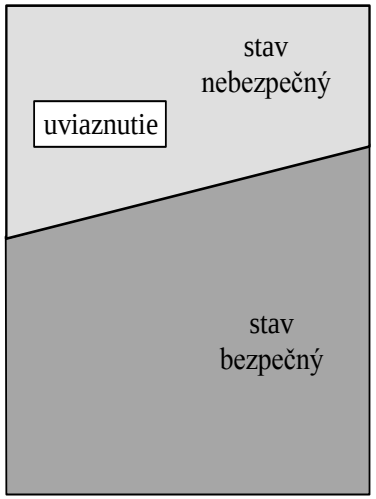
  Všetky prostriedky 3 2 6

	Pridelené	Ďalej požadované
P1	0 1 3	1 0 1
P2	1 0 0	0 0 0
P3	1 1 3	0 0 3
P4	1 0 0	0 0 0

- Na začiatku len P2 a P4 dostanú požadované prostriedky
- Výsledok: prístupné sú $|r1| = 2$ a $|r2| = |r3| = 0$
- UVIAZNUTIE medzi P1 a P3

„Bezpečná“ postupnosť

- **Postupnosť patrí k bezpečným ak:**
  - Procesy sú zoradené v takom poradí, že všetky požiadavky sú uspokojené pomocou
    - prostriedkov, ktoré sú pridelené procesu alebo
    - sú pridelené procesom, ktoré sa vyskytujú v tej postupnosti skôr
  - **Detekcia uviaznutia spočíva v kontrole existencie takej postupnosti.**
    Ak taká postupnosť neexistuje, to znamená že nastalo uviaznutie!!!



Stavové priestory stavu bezpečného, nebezpečného a stavu uviaznutia

Algoritmus bankára

Algoritmus bankára používa nasledovné dátové štruktúry:

- ***n*** je počet procesov,
- ***m*** je počet typov prostriedkov v systéme,
- ***prístupné*** : vektor s dĺžkou *m*, ktorý obsahuje počty prístupných prostriedkov z každého typu.
- ***max*** : matica *n* x *m* definuje maximálne požiadavky každého procesu.
- ***pridelené*** : matica *n* x *m* definuje počet prostriedkov každého typu, pridelených momentálne procesu P_i.
- ***zostáva*** : matica *n* x *m* , ktorá označuje prostriedky, ktoré ešte musia byť pridelené procesu.. Platí, že *zostáva[i,j] = max[i,j] - pridelené[i,j].*

Algoritmus bankára - určenie stavu systému

Kroky algoritmu pre určenie stavu systému sú nasledovné:

1. Nech ***pracovné*** a ***dokončené*** sú vektory s dĺžkou *m* resp *n*.
   Počiatočné hodnoty sú:

   pracovné:= prístupné a

   dokončené [i]:= false pre *i* =1, 2, ..., *n*.

2. Nájdeme *i* také, že *dokončené [i] = false* a
   zostáva[i] ≤ pracovné.

   Ak také *i* nie je, ideme na krok 4.

3. Priradíme *pracovné := pracovné + pridelené*$_i$

   dokončené [i] :=true

4. Ideme na krok 2.

Zložitosť: ***m*** **x** ***n***^{2} operácií pre nájdenie odpovede o stave systému.

Algoritmus bankára - vyžiadanie prostriedku

$požiadavka_i$ - vektor požiadaviek procesu P_i.

$požiadavka_i[j]=k$, - proces P_i požaduje k jednotiek prostriedku typu R_j .

Keď proces požiada o prostriedky vykoná sa nasledovné:

1. **Ak** $požiadavka_i \leq zostáva[i]$, ideme na krok 2.

   Ináč - chybový stav, proces prekročil svoje max. požiadavky.

2. **Ak** $požiadavka_i \leq prístupné_i$, ideme na krok 3.

   Ináč P_i musí čakať, pretože prostriedky nie sú prístupné.

3. Predstierame, že systém pridelil požadované prostriedky procesu P_i tak, že modifikujeme stav takto:

$$prístupné := prístupné - požiadavka_i$$
$$pridelené[i] := pridelené[i] + požiadavka_i$$
$$zostáva[i] := zostáva[i] - požiadavka_i$$

Algoritmus bankára – vyžiadanie prostriedku

Ak výsledný stav je bezpečný, transakcia sa dokončí a proces P_i dostane požadované prostriedky,

ináč proces **musí čakať** a **obnoví sa pôvodný stav.**

Príklad

Prostriedky typu

A	*B*	C
10	5	7

V čase t_0 stav systému je nasledovný

	pridelené			*max. požiadavka*			*prístupné*			*zostáva*		
	A	***B***	***C***	***A***	***B***	***C***	***A***	***B***	***C***	***A***	***B***	***C***
P_0	0	1	0	7	5	3	3	3	2	7	4	3
P_1	2	0	0	3	2	2				1	2	2
P_2	3	0	2	9	0	2				6	0	0
P_3	2	1	1	2	2	2				0	1	1
P_4	0	0	2	4	3	3				4	3	1

$$zostáva = max - pridelené$$

Postupnosť procesov : P_1, P_3, P_4, P_2, P_0 **- stav bezpečný**

Iná situácia

- Predpokladajme, že P_1 požaduje *požiadavka*$_1$ = (1, 0, 2).
- Najskôr skontrolujeme či *požiadavka*$_1 \leq$ *prístupné*

  (t.j. (1, 0, 2) ≤ (3, 3, 2)), čo je splnené.

Po pridelení prostriedkov požiadavke systém prichádza do tohto stavu:

	pridelené			*zostava*			*prístupné*		
	A	*B*	*C*	*A*	*B*	*C*	*A*	*B*	*C*
P_0	0	1	0	7	4	3	2	3	0
P_1	3	0	2	0	2	0			
P_2	3	0	2	6	0	0			
P_3	2	1	1	0	1	1			
P_4	0	0	2	4	3	1			

- **Je tento stav je bezpečný?** - vykonáme algoritmus pre určenie stavu systému
- Zistíme, že postupnosť P_1, P_3, P_4, P_0, P_2 **vyhovuje** podmienkam bezpečnosti a **pridelíme** požadované prostriedky procesu P_1.

Vlastnosti algoritmu bankára

- Pracuje s grafom prideľovania prostriedkov
- Hrana požiadavky sa konvertuje na hranu pridelenia len ak nevznikne slučka
- **Nedostatky**
  - Neberie do úvahy či z daného prostriedku je viacej jednotiek
  - Každý prostriedok sa musí brať ako samostatný uzol
  - Pridelenie je zložitejšie, lebo sa pri každej požiadavke musí prehľadať graf prideľovania

Problémy s riešením uviaznutia pomocou vyhnutia sa

- **Vyžaduje špecifikáciu budúcich požiadaviek**
  - procesy OS vo všeobecnosti **nemajú** túto informáciu
  - aplikovateľný pre špeciálne situácie
    - konštrukcie v programovacích jazykoch
      - transakčné - orientované systémy
    - známe OS komponenty

Zotavenie sa z uviaznutia

- Známe sú len všeobecné princípy
- Dve možnosti
  - **Ukončenie procesu** – nie je jednoduché
    - ukončenie všetkých uviaznutých procesov
    - ukončenie procesov jeden po druhom kým sa uviaznutie neukončí
  - **Preempcia prostriedkov**
    - výber obete
    - rollback & restart
    - prevencia starvácie

Kombinovaný prístup

- Jednotlivé metódy nie sú prijateľné pre celý systém
- Lepšie je ich skombinovať!
  - klasifikovať prostriedky a použiť iný prístup pre každý typ
    - **PCB** (process control blocks) – používa sa zoradenie prostriedkov
    - **užívateľská pamäť** -- používa sa preempcia
    - **zariadenia, ktoré sa priraďujú** – používa sa vyhnutie sa; vyžaduje explicitnú požiadavku
  - priestor na swapovanie – prideľuje sa dopredu, lebo požiadavky sú známe