

Operačné systémy

06. Uviaznutie

Ing. Martin Vojtko, PhD.



2024/2025



- 1 Prostriedky
- 2 Uviaznutie
 - Nevyhnutné podmienky vzniku
 - Modelovanie
- 3 Riešenie problému uviaznutia
 - Ignorovanie problému
 - Detekcia a náprava
 - Vyhýbanie sa
 - Prevencia
- 4 Ďalšie problémy
- 5 Zhrnutie

Prostriedky

Prostriedok (Resource)

Prostriedok

je akákoľvek časť výpočtového systému, ktorá môže byť alokovaná/získaná, použitá a uvoľnená.

- Prostriedok je akákoľvek časť výpočtového systému či už HW alebo SW. (CPU, pamäť, disk ..., dáta, tabuľky v databáze)
- Prostriedky môžu byť vo výpočtovom systéme multiplikované (4-CPU, dve tlačiarne, niekoľko diskov...)

Prostriedok

Preemptívny prostriedok

je prostriedok, ktorý môže byť odobraný procesu bez narušenia správneho fungovania procesu. (CPU, pridelený stránkový rám v pamäti)

Nepreemptívny prostriedok

je prostriedok, ktorý nemôže byť odobraný procesu bez narušenia správneho fungovania procesu. (tlačiareň, súbor)

- Preemptívnosť závisí aj od kontextu.
Napríklad ak nastavím DMA do stránkového rámu v pamäti táto nemôže byť odobraná.
- Nepreemptívne prostriedky sú tie na, ktoré je nutné hľadiť s obozretnosťou pri práci s nimi.

Alokácia prostriedku

blokujúca alokácia

v prípade, že proces nedostane prostriedok, proces je uspaný, blokový.

neblokujúca alokácia

v prípade, že proces nedostane prostriedok, proces dostane chybové hlásenie. Proces môže skúšať v slučke.

Uviaznutie

Uviaznutie (Deadlock)

Uviaznutie

je situácia pri ktorej každý proces z vybranej množiny procesov čaká na udalosť, ktorú môže vyvolať iba iný proces z množiny.

- Mutual Exclusion, ktorým sme vyriešili Race Conditions je problémom.
- Našťastie k uviaznutiu môže dôjsť len ak platia nevyhnutné podmienky uviaznutia.
- Stačí aby jedna podmienka nebola splnená a výpočtový systém **nemôže** uviaznuť!

Alokacia prostriedku s moznym uviaznutim

- Bežné OS problém uviaznutia neriešia v user móde. Je na programátorovi zabezpečiť aby jeho procesy resp. thready neuviazli.

```
1  tSem resource_A = 1; resource_B = 1;
2
3  void process_A()
4  {
5      down(&resource_A);
6      down(&resource_B);
7      use_resources();
8      up(&resource_B);
9      up(&resource_A);
10 }
11
```

```
1  void process_B()
2  {
3      down(&resource_A);
4      down(&resource_B);
5      use_resources();
6      up(&resource_B);
7      up(&resource_A);
8  }
```

```
1  void process_B()
2  {
3      down(&resource_B);
4      down(&resource_A);
5      use_resources();
6      up(&resource_A);
7      up(&resource_B);
8  }
```

Nevyhnutné podmienky vzniku uviaznutia

Vzájomné vylučovanie (Mutual Exclusion Condition)

prostriedok môže byť pridelený najviac jednému procesu.

Čiastočné pridelenie (Hold and Wait Condition)

proces s pridelenými prostriedkami môže žiadať o ďalšie prostriedky.

Nepreemptívne prostriedky (No Preemption Condition)

prostriedok môže byť uvoľnený iba procesom, ktorý ho získal.

Čakanie na prostriedky (Circular Wait Condition)

proces čaká na prostriedky využívané iným procesom.

Modelovanie uviaznutia - orientovaný graf

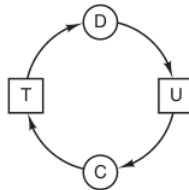
- Orientovaný graf obsahuje dva druhy vrcholov. Procesy (kruh) a Prostriedky (štvorec).
- Orientovaná hrana z prostriedku do procesu reprezentuje prostriedok používaný procesom. (a)
- Orientovaná hrana z procesu do prostriedku reprezentuje proces čakajúci na prostriedok. (b)
- Ak v grafe existuje orientovaný cyklus, v systéme môže dôjsť k uviaznutiu. (c)



(a)



(b)



(c)

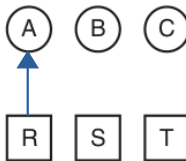
Modelovanie uviaznutia - orientovaný graf

1. A requests R
2. B requests S
3. C requests T
4. A requests S
5. B requests T
6. C requests R
deadlock



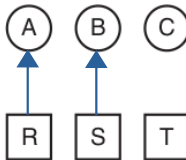
Modelovanie uviaznutia - orientovaný graf

-
1. A requests R
 2. B requests S
 3. C requests T
 4. A requests S
 5. B requests T
 6. C requests R
deadlock



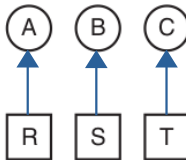
Modelovanie uviaznutia - orientovaný graf

-
1. A requests R
 2. B requests S
 3. C requests T
 4. A requests S
 5. B requests T
 6. C requests R
deadlock



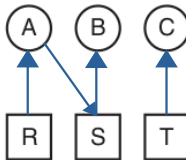
Modelovanie uviaznutia - orientovaný graf

- ▶
1. A requests R
 2. B requests S
 3. C requests T
 4. A requests S
 5. B requests T
 6. C requests R
- deadlock



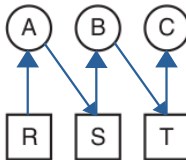
Modelovanie uviaznutia - orientovaný graf

- ➡
1. A requests R
 2. B requests S
 3. C requests T
 4. A requests S
 5. B requests T
 6. C requests R
- deadlock



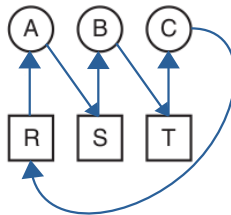
Modelovanie uviaznutia - orientovaný graf

-
1. A requests R
 2. B requests S
 3. C requests T
 4. A requests S
 5. B requests T
 6. C requests R
deadlock



Modelovanie uviaznutia - orientovaný graf

-
1. A requests R
 2. B requests S
 3. C requests T
 4. A requests S
 5. B requests T
 6. C requests R
deadlock



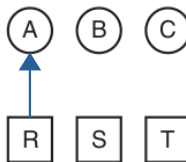
Modelovanie uviaznutia - orientovaný graf

1. A requests R
 2. C requests T
 3. A requests S
 4. C requests R
 5. A releases R
 6. A releases S
- no deadlock



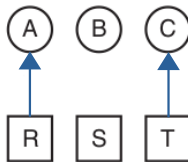
Modelovanie uviaznutia - orientovaný graf

- ▶ 1. A requests R
- 2. C requests T
- 3. A requests S
- 4. C requests R
- 5. A releases R
- 6. A releases S
- no deadlock



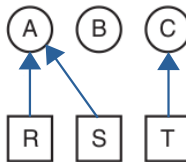
Modelovanie uviaznutia - orientovaný graf

- ▶
1. A requests R
 2. C requests T
 3. A requests S
 4. C requests R
 5. A releases R
 6. A releases S
- no deadlock



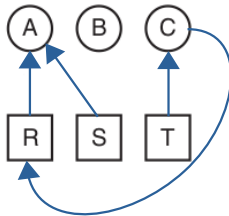
Modelovanie uviaznutia - orientovaný graf

- ➔
1. A requests R
 2. C requests T
 3. A requests S
 4. C requests R
 5. A releases R
 6. A releases S
- no deadlock



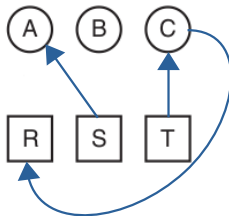
Modelovanie uviaznutia - orientovaný graf

- ➔
1. A requests R
 2. C requests T
 3. A requests S
 4. C requests R
 5. A releases R
 6. A releases S
- no deadlock



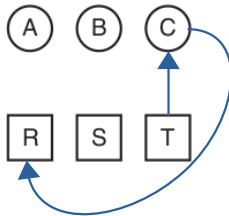
Modelovanie uviaznutia - orientovaný graf

- ➡
1. A requests R
 2. C requests T
 3. A requests S
 4. C requests R
 5. A releases R
 6. A releases S
- no deadlock



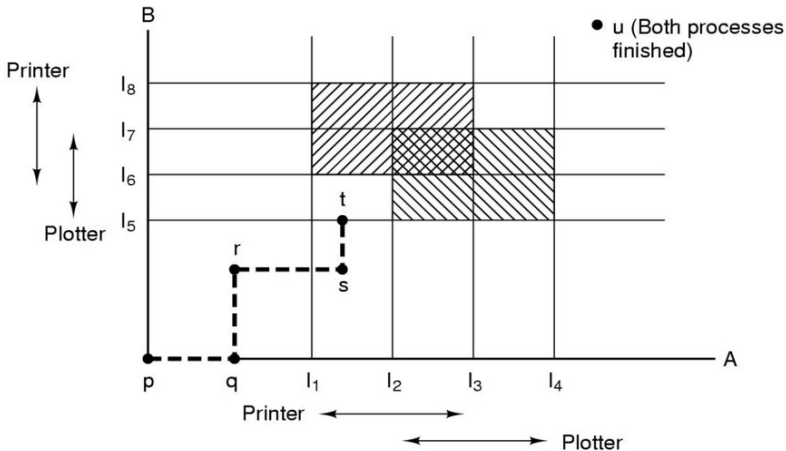
Modelovanie uviaznutia - orientovaný graf

- ➡
1. A requests R
 2. C requests T
 3. A requests S
 4. C requests R
 5. A releases R
 6. A releases S
no deadlock



Modelovanie uviaznutia - zobrazenie v čase

- V grafe vykresľujeme vykonávanie procesov na CPU.
- Čas kedy proces žiada o prostriedok je označený čiarou.
- Oblasť grafu, kde procesy súčasne používajú prostriedok je označená.

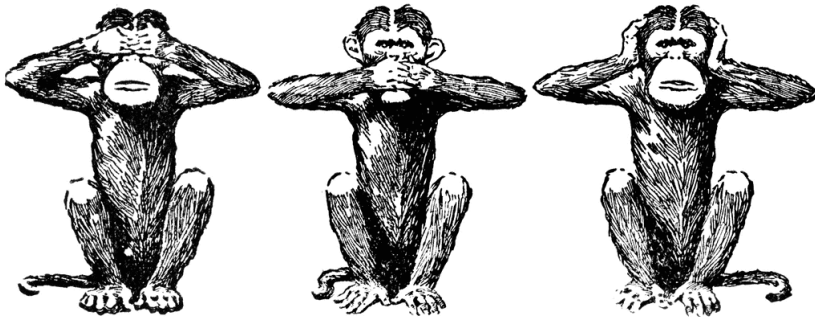


Riešenie problému uviaznutia

Riešenie problému uviaznutia

- Ignorovanie problému
- Detekcia a náprava
- Vyhýbanie sa
- Prevencia

Ignorovanie problému



Ignorovanie problému

- Ignorovanie problému sa často označuje aj ako pštrosí algoritmus (Ostrich Algorithm).
- Všetko je to o zvážení ceny a výkonu:
 - Pravdepodobnosť vzniku uviaznutia?
 - Koľko úsilia je nutné vynaložiť na nápravu?
 - Aký vplyv bude mať náprava na výkon systému?
 - Aké dôsledky má vznik uviaznutia?
- Návrh OS je často spojený s týmto prístupom.

Detekcia a náprava

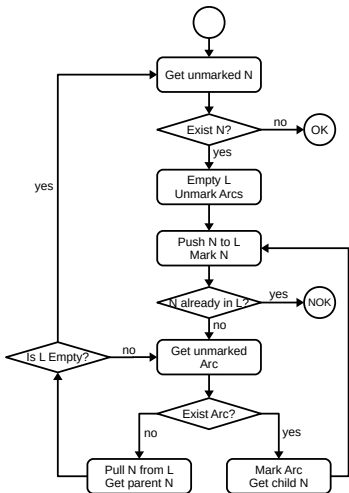
- Návrh mechanizmu, ktorý dokáže rozpoznať, že celý alebo časť výpočtového systému uviazla.
- OS musí mať informáciu o tom aké prostriedky aktuálne proces má a aké prostriedky sa požadujú.

Detekcia - jeden prostriedok z každého typu

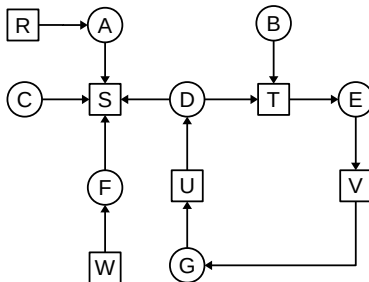
Algoritmus

- 1 Pre každý vrchol N grafu vykonaj nasledujúce kroky.
- 2 Inicializuj množinu prejdéných vrcholov L . Nastav všetky hrany ako neoznačené.
- 3 Pridaj aktuálny vrchol N do L a skontroluj či vrchol tam už nie je. Ak áno graf obsahuje cyklus. Koniec.
- 4 Ak z N vychádzajú neoznačené hrany choď do 5. Inak choď do 6.
- 5 Vyber neoznačenú hranu a označ ju. Označ cieľový vrchol hrany za aktuálny a opakuj 3.
- 6 Ak vrchol je počiatočný bod, graf neobsahuje cyklus. Koniec. Inak sme dosiahli slepú cestu. Odstráň N z L a vráť sa po hrane späť a pokračuj bodom 3.

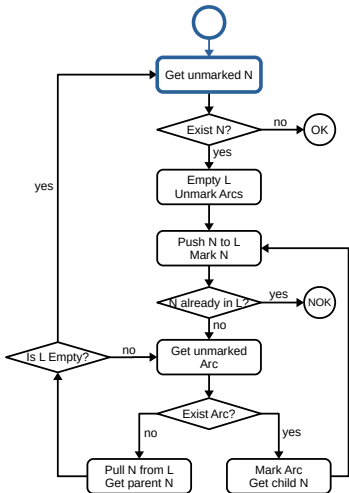
Detekcia - jeden prostriedok z každého typu



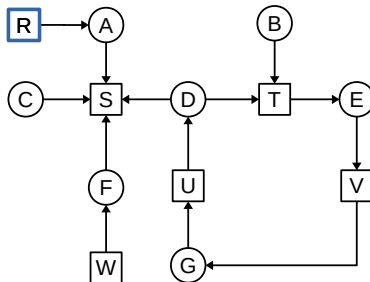
L :



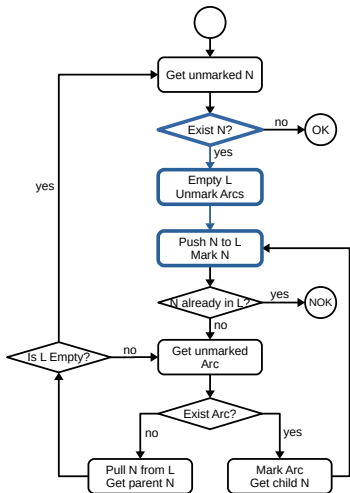
Detekcia - jeden prostriedok z každého typu



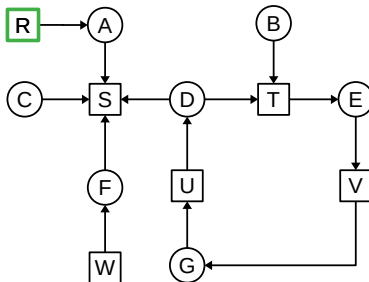
L :



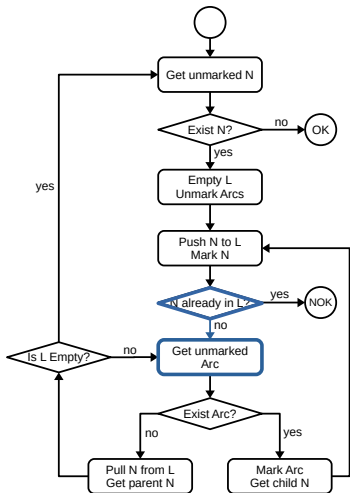
Detekcia - jeden prostriedok z každého typu



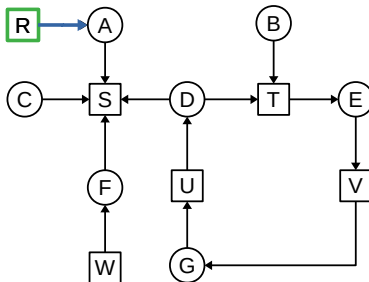
L : R,



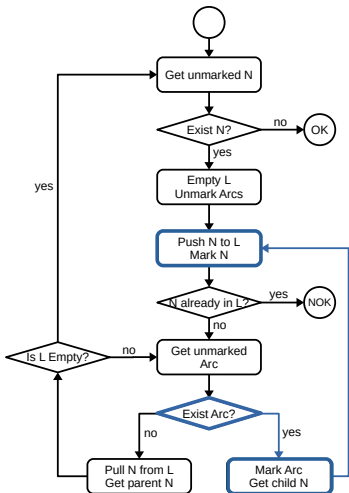
Detekcia - jeden prostriedok z každého typu



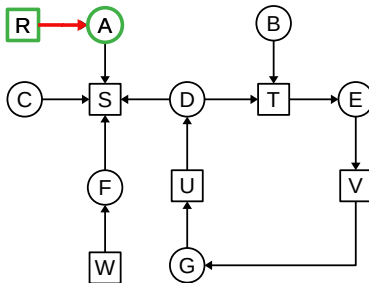
L : R,



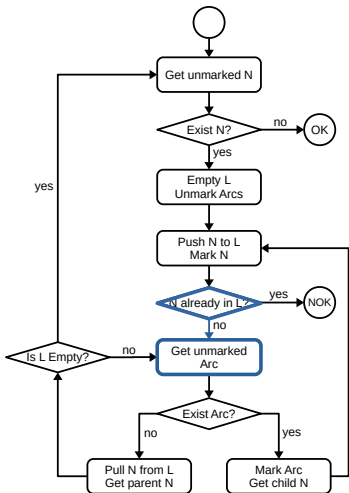
Detekcia - jeden prostriedok z kazdeho typu



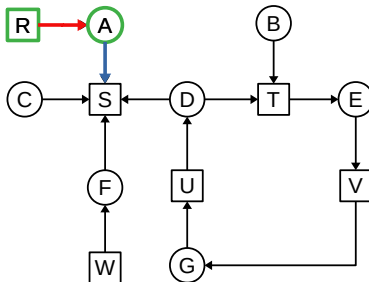
L : R, A



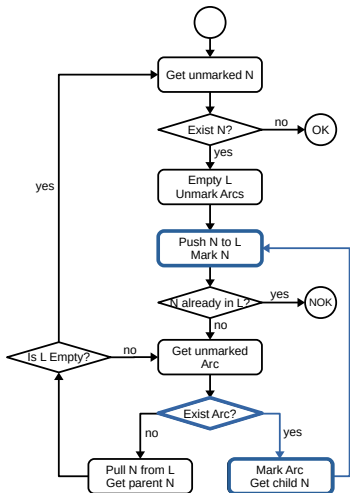
Detekcia - jeden prostriedok z každého typu



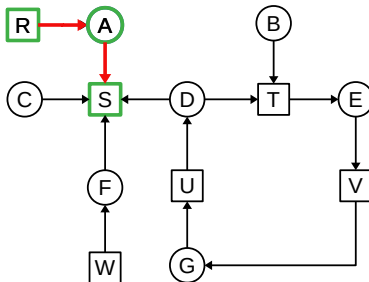
L : R, A



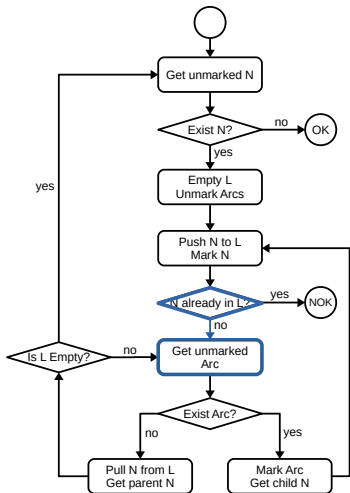
Detekcia - jeden prostriedok z kazdeho typu



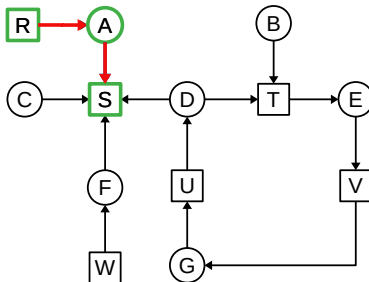
L : R, A, S



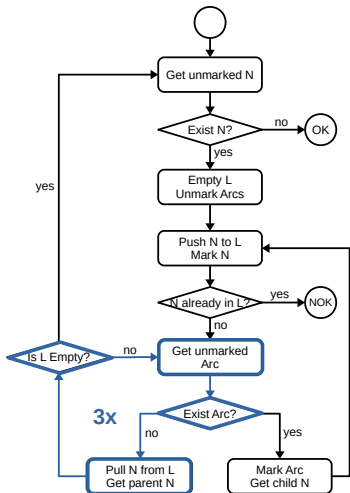
Detekcia - jeden prostriedok z každého typu



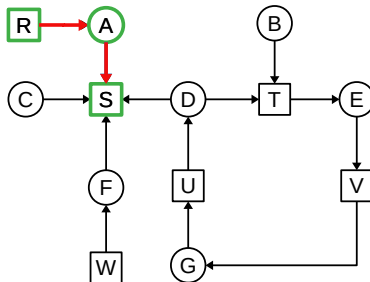
L : R, A, S



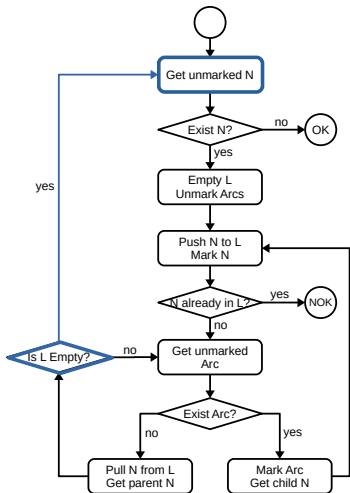
Detekcia - jeden prostriedok z každého typu



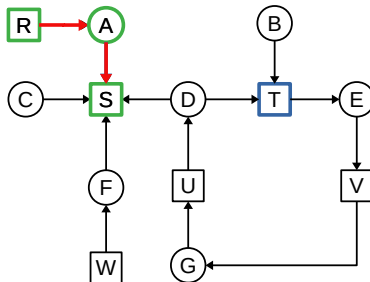
L :



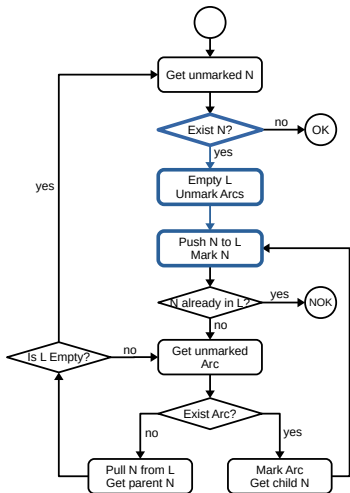
Detekcia - jeden prostriedok z každého typu



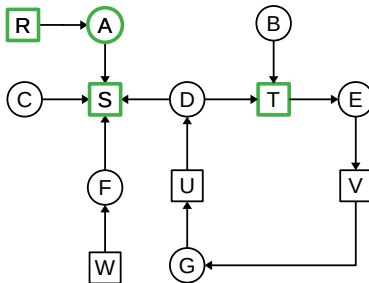
L :



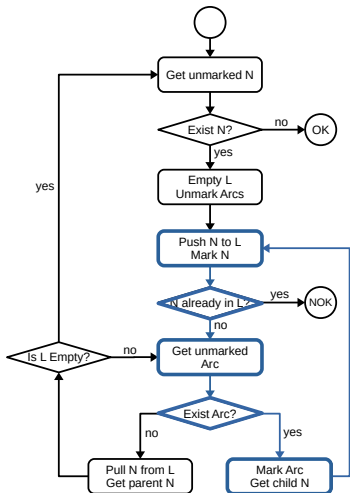
Detekcia - jeden prostriedok z každého typu



L : T

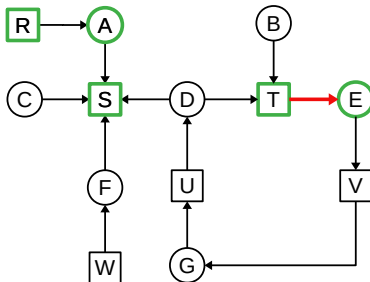


Detekcia - jeden prostriedok z každého typu

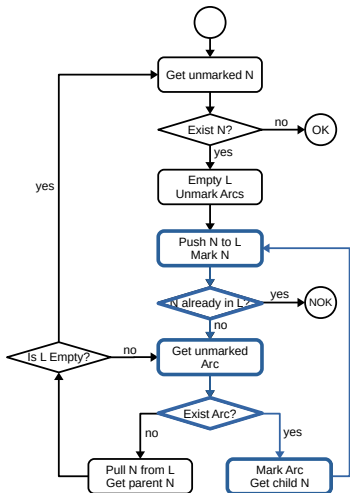


L : T, E

1

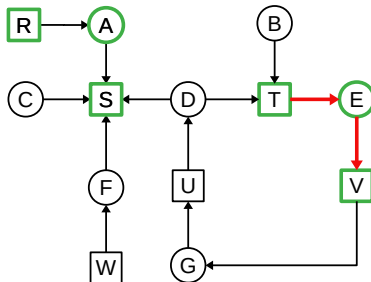


Detekcia - jeden prostriedok z každého typu

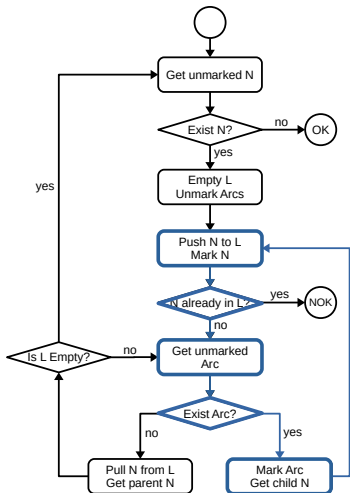


L : T, E, V

2

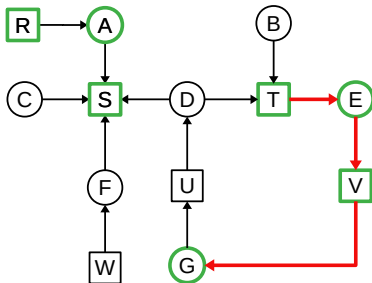


Detekcia - jeden prostriedok z každého typu

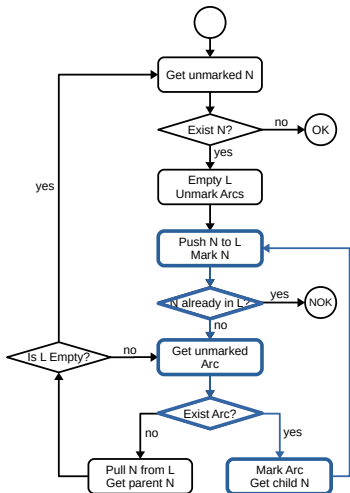


L : T, E, V, G

3

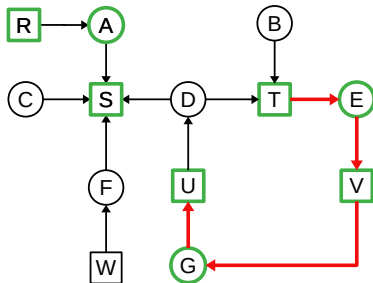


Detekcia - jeden prostriedok z každého typu

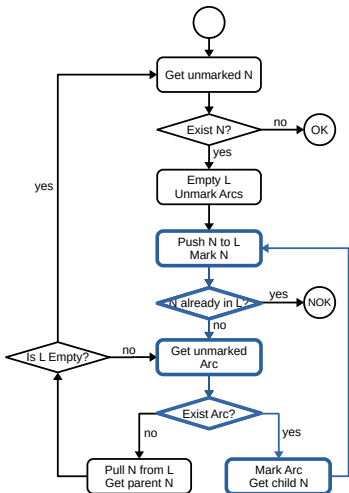


L : T, E, V, G, U

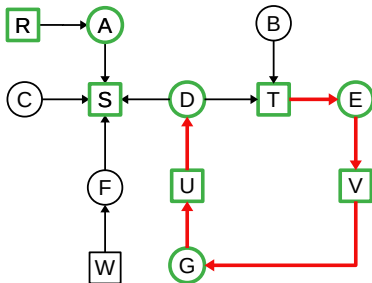
4



Detekcia - jeden prostriedok z každého typu

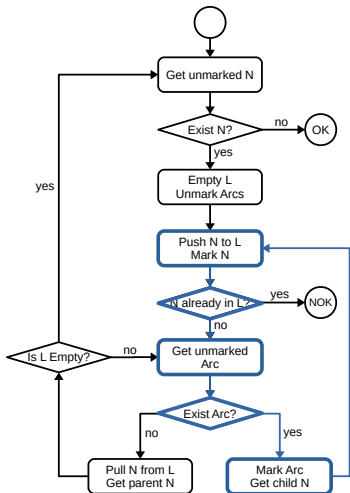


L : T, E, V, G, U, D

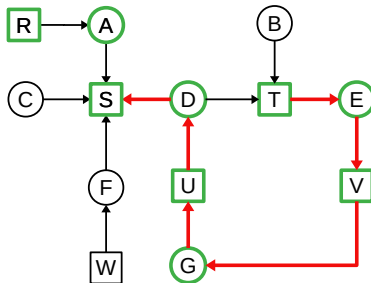


5

Detekcia - jeden prostriedok z každého typu

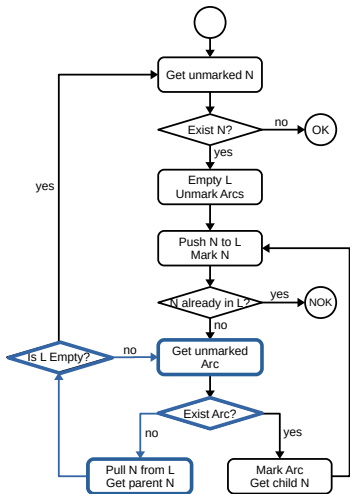


L : T, E, V, G, U, D, S



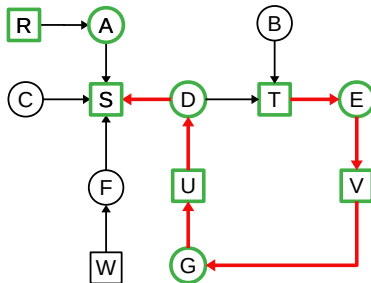
6

Detekcia - jeden prostriedok z každého typu

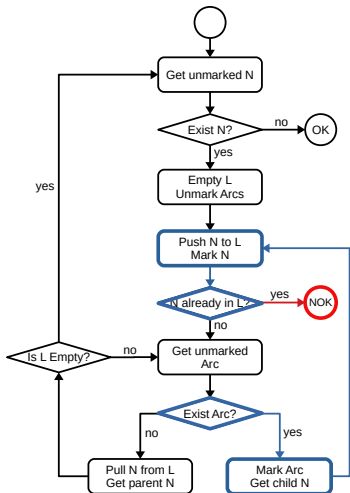


L : T, E, V, G, U, D

6

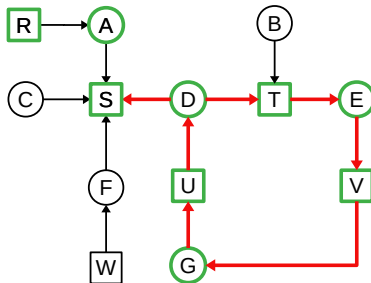


Detekcia - jeden prostriedok z každého typu



L : **T**, E, V, G, U, D, **T**

7




Detekcia - viac prostriedkov z každého typu

- Majme n procesov a m tried prostriedkov.
- E_m je vektor všetkých prostriedkov jednotlivých tried.
- A_m je vektor všetkých dostupných prostriedkov jednotlivých tried.
- C_{nm} je matica všetkých alokovaných prostriedkov jednotlivých tried.
- R_{nm} je matica všetkých žiadaných prostriedkov jednotlivých tried.
- Platí invariant $(\sum_{i=1}^n C_{ij}) + A_j = E_j$

Resources in existence
($E_1, E_2, E_3, \dots, E_m$)

Current allocation matrix




C_{11}	C_{12}	C_{13}	\dots	C_{1m}
C_{21}	C_{22}	C_{23}	\dots	C_{2m}
\vdots	\vdots	\vdots		\vdots
C_{n1}	C_{n2}	C_{n3}	\dots	C_{nm}

Row n is current allocation
to process n

Resources available
($A_1, A_2, A_3, \dots, A_m$)

Request matrix



R_{11}	R_{12}	R_{13}	\dots	R_{1m}
R_{21}	R_{22}	R_{23}	\dots	R_{2m}
\vdots	\vdots	\vdots		\vdots
R_{n1}	R_{n2}	R_{n3}	\dots	R_{nm}

Row 2 is what process 2 needs

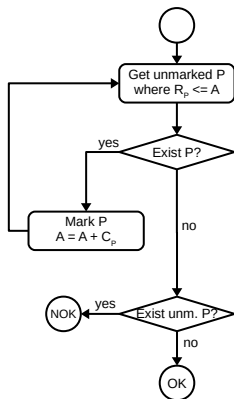
Detekcia - viac prostriedkov z každého typu

- Detekčný algoritmus založený na porovnávaní vektorov.
- Hovoríme, že vektor $A_m \leq B_m$ ak $\forall i, 1 \leq i \leq m : A_i \leq B_i$

Algoritmus

- 1 Vyhľadaj neoznačený proces i pre ktorý platí, že riadok $R_i \leq A$.
- 2 Ak taký riadok existuje tak $A = A + C_i$. Vráť sa do bodu 1.
- 3 Ak taký riadok neexistuje algoritmus končí. Všetky neoznačené procesy uviazli.

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

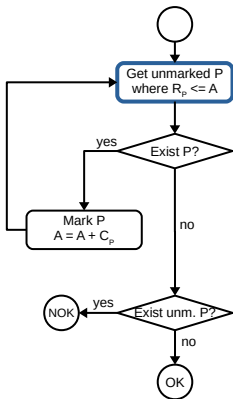
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

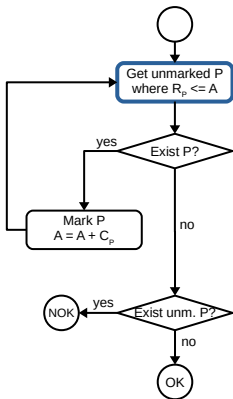
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

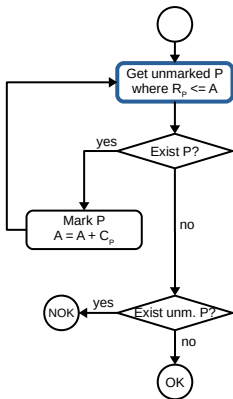
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

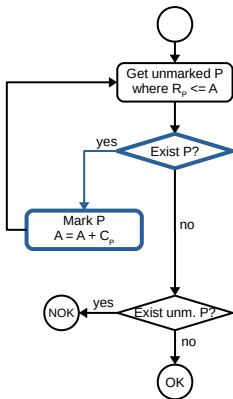
$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 2 \ 2 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

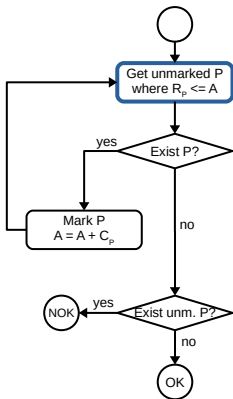
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 2 \ 2 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

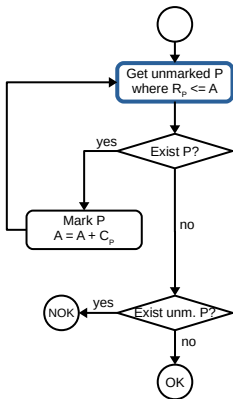
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 2 \ 2 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

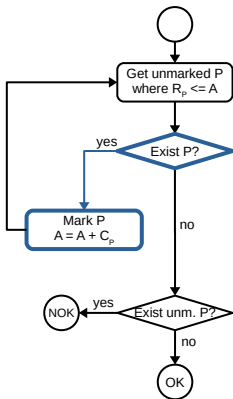
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (4 \ 2 \ 2 \ 1)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

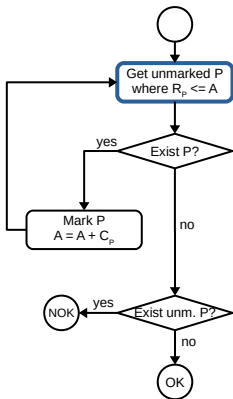
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

$$A = (4 \ 2 \ 2 \ 1)$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

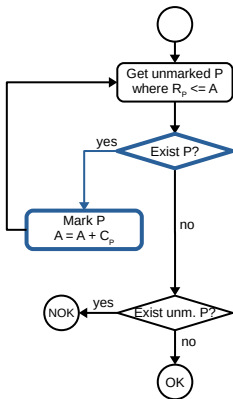
$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{bmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (4 \ 2 \ 3 \ 1)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

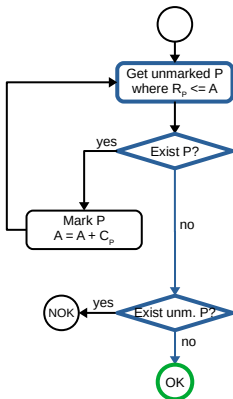
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (4 \ 2 \ 3 \ 1)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

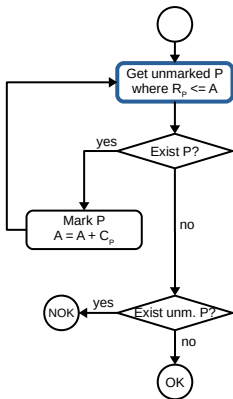
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

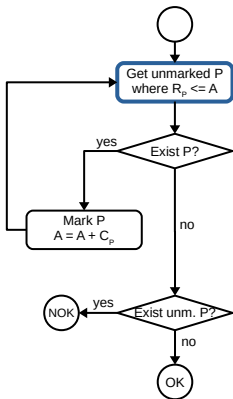
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

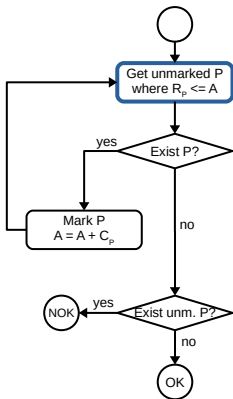
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z každého typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

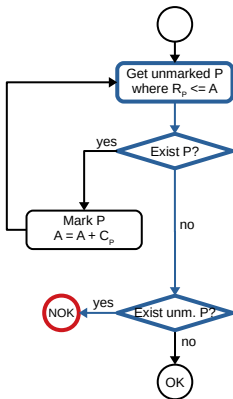
$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Detekcia - viac prostriedkov z kazdeho typu



$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{vmatrix}$$

$$E = (4 \ 2 \ 3 \ 1)$$

Current allocation matrix

$$C = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{vmatrix}$$

$$A = (2 \ 1 \ 0 \ 0)$$

Request matrix

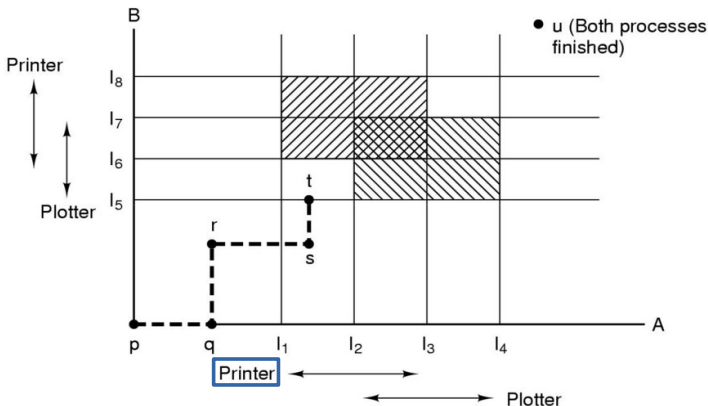
$$R = \begin{vmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \end{vmatrix}$$

Náprava (Recovery)

- Preempcia prostriedku:
 - Vyberie sa proces, ktorému sa odoberie prostriedok.
 - Výber zohľadňuje mieru rizika.
 - Napríklad ak proces používa tlačiareň, tak používateľ dostane neúplný dokument.
- Rollback procesu:
 - Kritické procesy v pravidelných intervaloch odkladajú stav do pamäti.
 - stav (check-point) sa môže prepisovať alebo sa vždy vytvorí nový.
 - Počas uviaznutia sa vyberie proces, ktorému sa odoberie prostriedok.
 - Vybraný proces sa spustí z uloženého check-pointu.
 - Napríklad ak proces používa tlačiareň, tak používateľ dostane dokument s duplikovaným obsahom.
- Kill procesu:
 - Vyberie sa proces ktorý je zabitý systémom.
 - Vražda jedného procesu nemusí stačiť.
 - Po ustálení systému sa proces môže spustiť znova.
 - Napríklad ak proces používa tlačiareň, tak používateľ dostane dokument s duplikovaným obsahom.

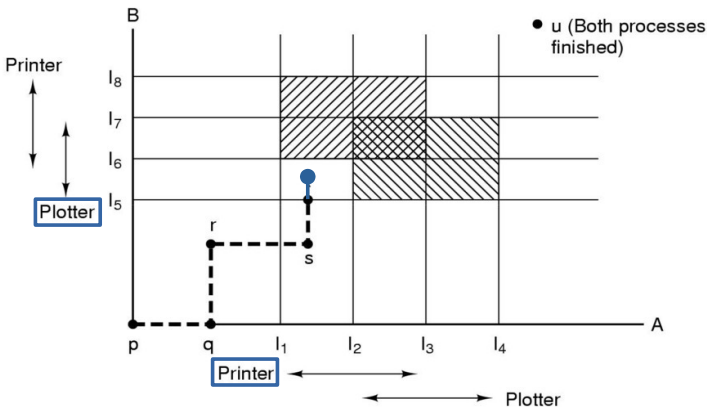
Vyhýbanie sa

- Prostriedky sú cielene pridelovane tak, aby nedošlo k uviaznutiu.
- V každom časovom bode je nutné zabezpečiť aby stav pridelenia prostriedkov zostal bezpečný.



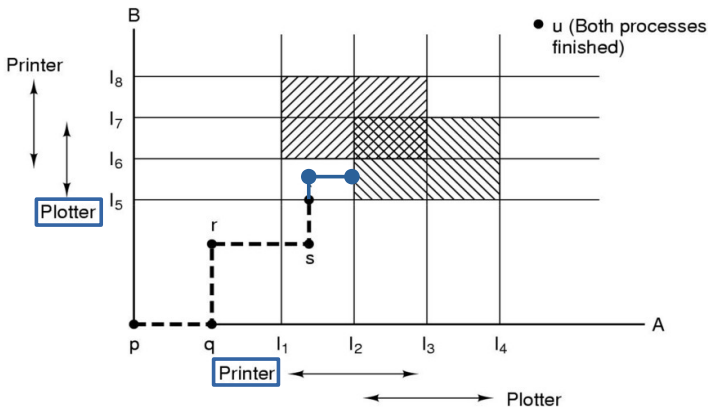
Vyhýbanie sa

- Prostriedky sú cielene pridelované tak, aby nedošlo k uviaznutiu.
- V každom časovom bode je nutné zabezpečiť aby stav pridelenia prostriedkov zostal bezpečný.



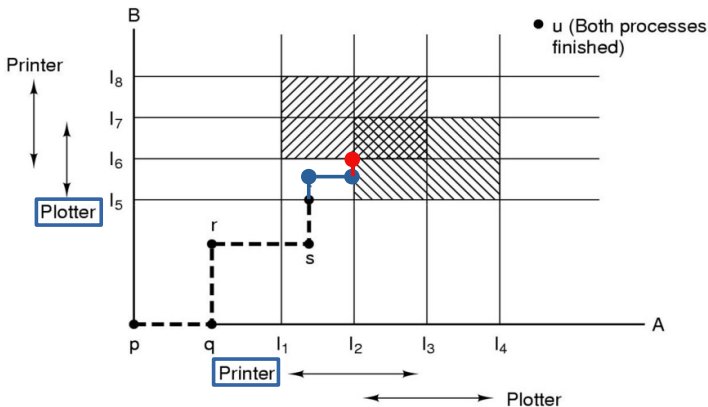
Vyhýbanie sa

- Prostriedky sú cielene pridelovane tak, aby nedošlo k uviaznutiu.
- V každom časovom bode je nutné zabezpečiť aby stav pridelenia prostriedkov zostal bezpečný.



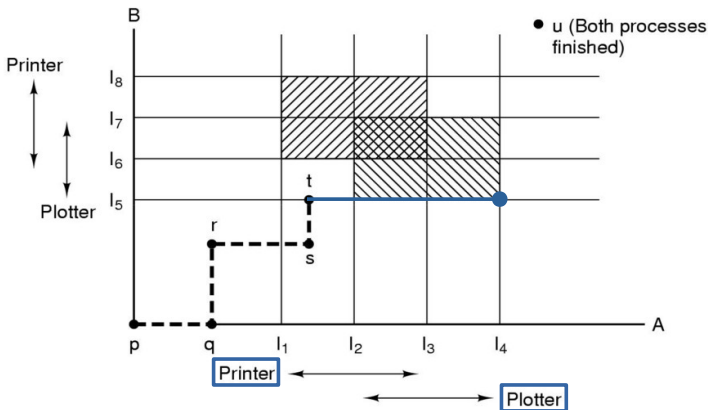
Vyhýbanie sa

- Prostriedky sú cielene pridelované tak, aby nedošlo k uviaznutiu.
- V každom časovom bode je nutné zabezpečiť aby stav pridelenia prostriedkov zostal bezpečný.



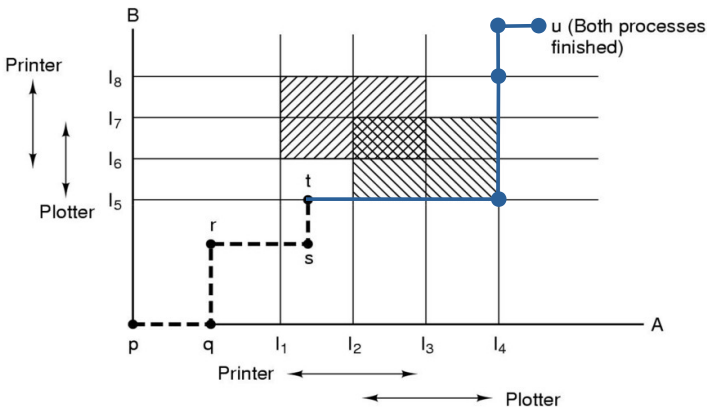
Vyhýbanie sa

- Prostriedky sú cielene pridelovane tak, aby nedošlo k uviaznutiu.
- V každom časovom bode je nutné zabezpečiť aby stav pridelenia prostriedkov zostal bezpečný.



Vyhýbanie sa

- Prostriedky sú cielene pridelovane tak, aby nedošlo k uviaznutiu.
- V každom časovom bode je nutné zabezpečiť aby stav pridelenia prostriedkov zostal bezpečný.



Bankárov algoritmus

- Bankár má pridelený balík peňazí.
- Cieľom je mať vždy dostatok financií aby Bankár vedel uspokojiť požiadavky každého.
- Na základe stavu financií a požiadaviek zákazníkov rozhoduje komu pridelí požadovanú čiastku.
 - Najprv vyhodnotí aktuálny stav.
 - Potom rozhodne či môže prideliť požadované prostriedky.
- Odmietnutý zákazníci dostanú peniaze v čase keď je stav financií bezpečný.
- Rozlišujeme Bankárov algoritmus pre jeden typ prostriedku a pre n typov prostriedkov.

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

OK	has	max
A	3	9
B	2	4
C	2	7
Free		3

NOK	has	max
A	3	9
B	2	4
C	2	7
Free		2

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	4	4
C	2	7
Free	1	

	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	-	-
C	2	7
Free	5	

	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	-	-
C	7	7
Free	0	

	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	-	-
C	-	-
Free	7	

	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	9	9
B	-	-
C	-	-
Free	1	

	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

OK	has	max
A	3	9
B	2	4
C	2	7
Free	3	

	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	2	4
C	2	7
Free	3	

NOK	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	2	4
C	2	7
Free	3	

	has	max
A	3	9
B	4	4
C	2	7
Free	0	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	2	4
C	2	7
Free	3	

	has	max
A	3	9
B	-	-
C	2	7
Free	4	

Bankárov algoritmus - 1 typ prostriedku

- Zistenie, či aktuálny stav je bezpečný.

	has	max
A	3	9
B	2	4
C	2	7
Free	3	

NOK	has	max
A	3	9
B	2	4
C	2	7
Free	2	

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedok - rozhodnutie či ho dostane.

	has	max
A	3	9
B	2	4
C	2	7
Free	3	

A : request 1

B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedok - rozhodnutie či ho dostane.

	has	max
A	4	9
B	2	4
C	2	7
Free	2	

A : request 1

B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedkov - rozhodnutie či ho dostane.

	has	max
A	4	9
B	4	4
C	2	7
Free	0	

A : request 1

B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedok - rozhodnutie či ho dostane.

	has	max
A	4	9
B	-	-
C	2	7
Free	4	

A : request 1

B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedkov - rozhodnutie či ho dostane.

	has	max
A	3	9
B	3	4
C	2	7
Free		2

A : request 1

B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedkov - rozhodnutie či ho dostane.

	has	max
A	3	9
B	4	4
C	2	7
Free		1

A : request 1

B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedkov - rozhodnutie či ho dostane.

	has	max
A	3	9
B	-	-
C	7	7
Free		0

A : request 1
B : request 1

Bankárov algoritmus - 1 typ prostriedku

- Proces žiada prostriedok - rozhodnutie či ho dostane.

	has	max
A	9	9
B	-	-
C	-	-
Free	1	

A : request 1

B : request 1

Bankárov algoritmus - n typov prostriedkov

- V podstate algoritmus je totožný s algoritmom použitým na detekciu uviaznutia pre n typov prostriedkov.
- Algoritmus sa aplikuje pre každú novú žiadosť procesu o prostriedky.
- Ak algoritmus nájde slučku tak proces prostriedky nedostane.

	Process	Tape drives	Plotters	Printers	Blu-rays
A	3	0	1	1	
B	0	1	0	0	
C	1	1	1	0	
D	1	1	0	1	
E	0	0	0	0	

	Process	Tape drives	Plotters	Printers	Blu-rays
A	1	1	0	0	
B	0	1	1	2	
C	3	1	0	0	
D	0	0	1	0	
E	2	1	1	0	

 $E = (6342)$ $P = (5322)$ $A = (1020)$

Vyhýbanie sa - problémy

- Bankárov algoritmus je úžasný.
- je to príklad teoretického riešenia...
- ...ktoré sa v praxi nedá použiť.
- Pre jeho úspešné nasadenie musíme:
 - poznať všetky prostriedky v systéme. Prostriedky však môžu pribúdať a ubúdať.
 - uviesť koľko a akých prostriedkov bude proces potrebovať ešte pred jeho spustením.
 - uviesť koľko procesov budeme mať. Procesy však vznikajú a zanikajú.
- Vo všeobecnosti existuje len málo systémov, ktoré by sa aktívne snažili vyhýbať uviaznutiu.

Prevenca

- Na zabránenie vzniku uviaznutia stačí aby len jedna z nutných podmienok neplatila.
- Útok na Mutual Exclusion condition
- Útok na Hold-and-Wait condition
- Útok na No-preemption condition
- Útok na Circular Wait condition

Prevenca - Mutual Exclusion condition

- Vzájomné vylučovanie, chráni pred vstupom do kritickej oblasti.
- Procesy čakajúce na vstup do KO sú buď blokované alebo cyklia.
- Proces, ktorý je v KO exkluzívne vlastní prostriedok.
- Ak navrhujeme systém tak, že nie je potrebné vylučovanie máme vyhrané.
- Návrh systému teda vyžaduje mať čo najmenej procesov, ktoré prístupujú k prostriedku.

Daemon

Všeobecným riešením problému je použitie Daemon procesu. Daemon vždy prístupuje len k prostriedku, ktorý riadi (Nežiada o iné). Prostriedok ideálne môže používať len Daemon. Ostatné procesy posielajú požiadavky na spracovanie Daemon-ovy prostredníctvom vyhradeného miesta v pamäti.

Prevenca - Hold-and-Wait condition

- Procesy sa môžu dostať do situácie kedy vlastnia jeden prostriedok a žiadajú o druhý (dining philosophers).
- Riešením je získanie všetkých potrebných prostriedkov pred spustením procesu.
- Ak sa to nepodarí proces nebude spustený a čaká.
- Problém je, že nevieme aké prostriedky proces bude chcieť. Ak by sme vedeli, tak použijeme Bankárov algoritmus.

Prevenca - No-preemption condition

- Problematické sú ne-preemptívne prostriedky.
- Ak navrhнем proces tak, že prostriedok ktorý mu bol pridelený môžeme odobrať a potom zas vrátiť, máme vyhrané.
- Riešenie je založené rovnako na Daemon-och.

Odloženie na disk

Proces, ktorý pracuje s prostriedkom odkladá výsledky do pamäte. Napríklad do súboru v priečinku na disku. Daemon číta hotové súbory v priečinku a odosiela ich na prostriedok. Všeobecne môže dôjsť k situácii kedy sa disk preplní a dôjde k uviaznutiu ale je to menej pravdepodobné.

Prevenca - Circular Wait condition

- Ak procesy navzájom čakajú na prostriedky, ktoré vlastní druhý proces dochádza k uviaznutiu.
- Ak obmedzíme systém tak, že proces môže vlastniť iba jeden prostriedok máme vyhrané.
- V skutočnosti to je príliš reštriktívna požiadavka.

Usporiadanie prostriedkov

Všeobecným a používaným riešením je zabezpečiť správne poradie žiadania prostriedkov. Ak proces A požiadava prostriedok 1 a súčasne 2, tak proces B nesmie požiadať o prostriedok 2 a potom o 1. Systém si udržiava takéto informácie a ak dôjde k takejto situácii tak je vyvolaná výnimka. Následne takto odhalený problém sa opraví ako bug fix.

Prevenca

- Mutual Exclusion condition - Prostriedok pridelený jednému riadiacemu procesu.
- Hold-and-Wait condition - Proces žiada všetky prostriedky naraz pri spustení.
- No-preemption condition - Sprav prostriedky preemtívnymi.
- Circular Wait condition - Usporiadaj zamykanie prostriedkov.

Ďalšie problémy

Dvoj-fázové zamykanie

- Vo veľkých databázových systémoch je často nutné, že proces pristupuje k rôznym záznamom.
- Problémom je, že zamknutie celej databázy a následné upravenie záznamu je neefektívne.
- Riešením je zamykanie po záznamoch čo môže viesť k uviaznutiu.

Dvoj-fázové zamykanie

kedy sa proces v prvej fáze pokúsi zamknúť všetky potrebné záznamy naraz. Následne ak sa to podarí tak prechádza do druhej fázy. Ak sa v prvej fáze nepodarí získať všetky záznamy tak záznamy, ktoré získal uvoľní a skúša to znovu.

Livelock

- Dvoj-fázové zamykanie funguje na princípe uvoľnenia už získaných prostriedkov.
- Je to slušné ale ak sú všetci slušní môže dôjsť k situácii kedy procesy dookola uvoľňujú prostriedky.
- Asi každý sa už dostal do situácie kedy sa vyhýbal inému na dva, tri, štyri krát, že?

Livelock

je situácia kedy procesy nie sú uviaznuté ale napriek tomu nedokážu ukončiť svoju činnosť pretože v cykle uvoľňujú a znova žiadajú rovnaké prostriedky.

- Problém môže byť postačujúco vyriešený náhodným časom čakania pred ďalším pokusom o získanie prostriedku.

Starvation (vyhladovanie)

- Vždy ak existuje nejaká politika plánovania procesov alebo aj politika nápravy uviaznutia môže dôjsť k situácii kedy nejaké procesy čakajú príliš dlho na prostriedok.
- Napríklad ak OS vykonáva nápravu uviaznutia vybraná obeť môže byť vždy ten istý proces.
- Napríklad ak Bankár zamietne pridelenie prostriedkov, obeťou bude pravdepodobne proces, ktorý ich žiada príliš veľa.

Starvation

je situácia kedy proces kvôli nevyváženosti riadiaceho algoritmu sa nedostáva k prostriedku. Týmto problémom často trpia plánovacie algoritmy alebo algoritmy ktoré zamedzujú vzniku uviaznutia.

- Problém môže byť postačujúco vyriešený jedine úpravou algoritmu alebo návrhu výpočtového systému.

Zhrnutie

Zhrnutie

- Uviaznutie systému je komplexným problémom vo výpočtových systémoch.
- Je veľmi ťažké odhaliť samotnú príčinu vzniku uviaznutia. Takéto problémy sa veľmi zle ladia.
- Je dokonca tak náročné mu zabrániť, že mnohé OS ich prakticky ignorujú.
- Často je na programátoroch samotných zabezpečiť výpočtový systém. Použitím techník prevencie.
- Najbežnejším princípom, ktorý sa používa je číslovanie a fixné usporiadanie prostriedkov.

Čo robiť do ďalšej prednášky

- Test v piatok CPU 25.10 9:00 - 12:00
- Každý otestujte či sa viete prihlásiť do ais.
- Prečítať kapitolu 3.1, 3.2, 3.3, 3.7 z Tanenbauma.