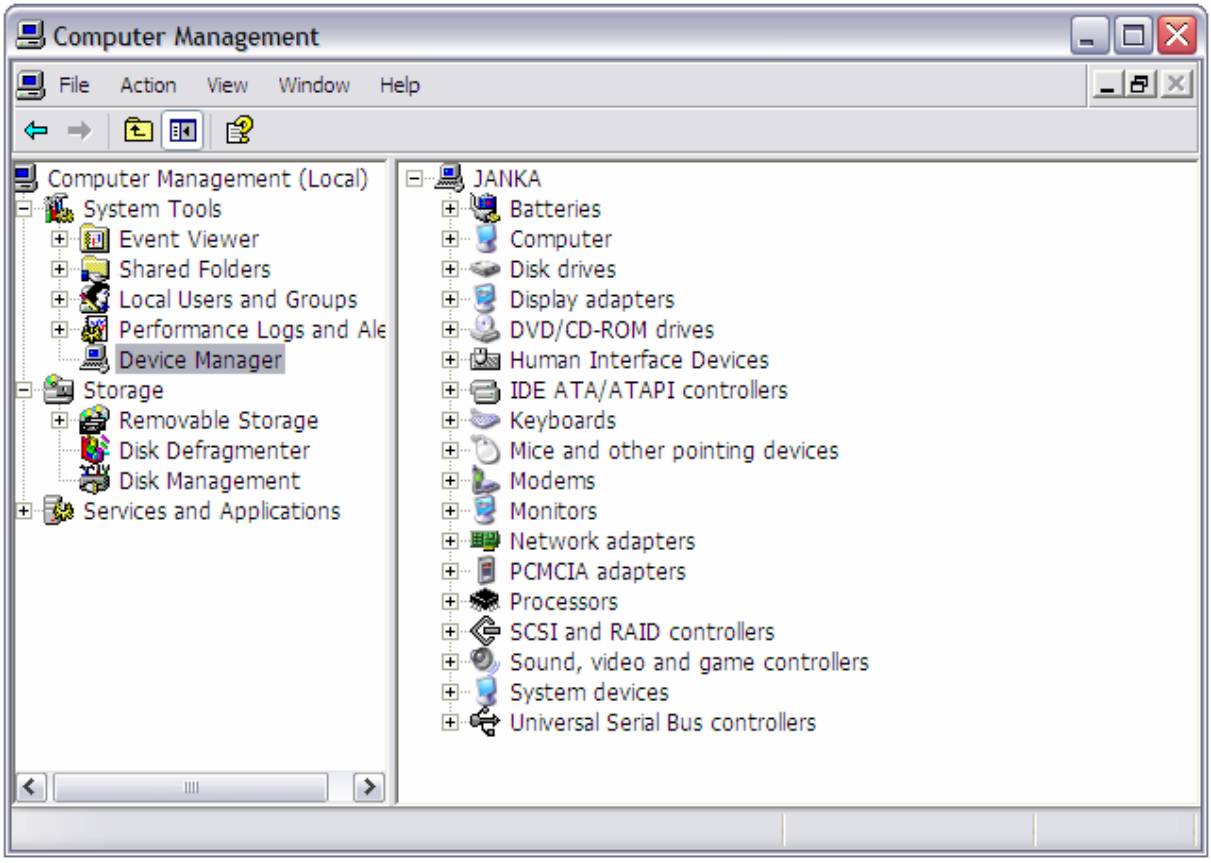
Správa zariadení

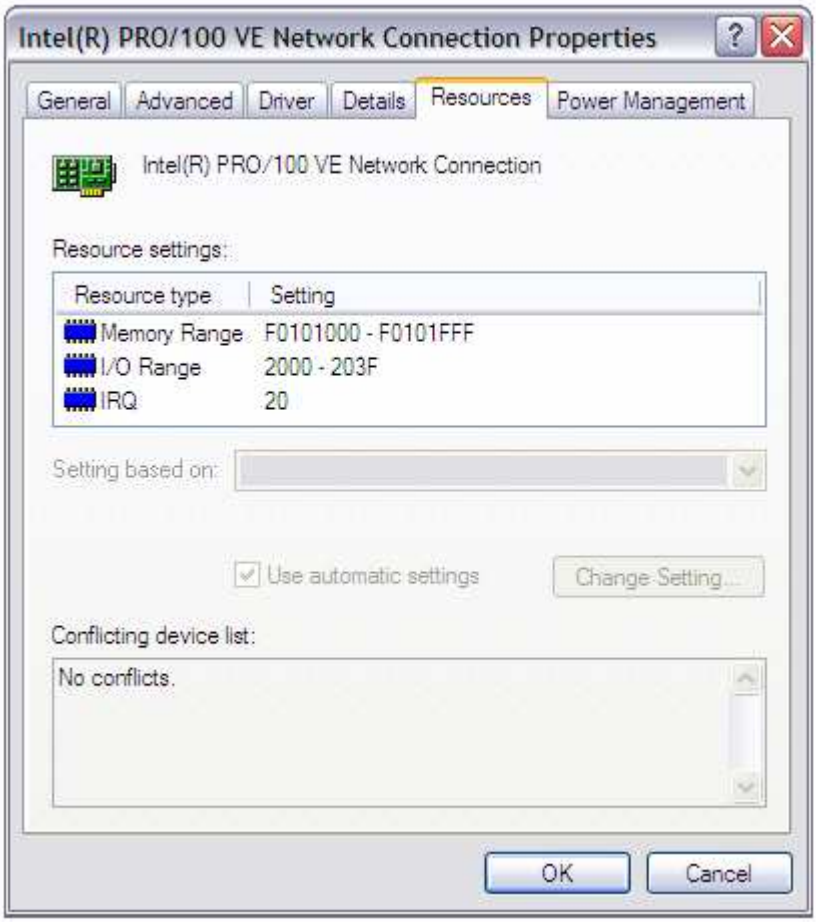
Keďže zariadenia pripojené k počítaču sú rozmanité, každé komunikuje iným spôsobom, každé môže používať len určité prerušenia a porty, na komunikáciu s nimi operačný systém používa tzv. **ovládače zariadení.**

V systéme Windows je na správu zariadení určený program, ktorý sa nazýva **Správca zariadení** (Device manager). Tento program spustíme buď cez *Ovládacie panely* alebo klikneme pravým tlačidlom myši na *Počítač* a z kontextovej ponuky zvolíme *Spravovať*. V správcovi zariadení môžeme nájsť všetky zariadenia, ktoré systém automaticky rozpoznal v systéme. O každom zariadení môžeme zistiť, aké používa porty a prerušenia, verziu ovládača a aké súbory tvoria ovládač zariadenia.



Ak sa chceme o danom zariadení dozvedieť viac, stačí v správcovi zariadení roztvoriť kategóriu zariadení a dvojklikom otvoriť vlastnosti zariadenia. Otovrí sa nám okno s niekoľkými kartami.

Na karte **Všeobecné** (General) sa môžeme dozvedieť typ zariadenia, výrobci a umiestnenie. Na karte **Prostriedky** (Resources) sa môžeme dozvedieť aké prerušenie, vstupno výstupné porty a adresy pamäte zariadenie používa. Na karte **Ovládač** (Driver) sa zasa môžeme dozvedieť poskytovateľa ovládača, dátum jeho vytovrenia, verziu a kliknutím na tlačidlo Podrobnosti ovládača aj to, z akých súborov ovládač pozostáva.



So správou zariadení úzko súvisí synchronizácia, komunikácia a kooperácia procesov. Zoberme si prístup procesov k zariadeniam, ktoré je často potrebné zdieľať.

Ak by dva procesy naraz tlačili na tlačiareň, tak by boli výsledky premiešané. V takom prípade sa to rieši dvoma spôsobmi. Prvý spôsob je pripustiť k tlačiarni jediný proces a ostatné nech tlačia tak, že pripravia tlač do súboru a potom v jedinom okamihu, keď je všetko pripravené, kontaktujú proces na tlač. Výhodou je, že okamih je krátky a proces obsluhujúci tlačiareň bude pravdepodobne pre prenos dát voľný. Ak nie je, tlačechtivý proces jednoducho chvíľku počká a skúša to znovu. Druhou možnosťou je prideľovať I/O zariadenia na požiadanie. Proces o zariadenie požiada, je mu pridelený prístup k nemu a po ukončení práce procesu je prostriedok uvoľnený pre použitie inými procesmi. Tu však vzniká nebezpečie uviaznutia.

Procesy si môžu informácie navzájom vymieňať. Zahŕňa to komunikáciu a synchronizáciu spolupráce. (*Ja ti dodám také a také údaje a ty z nich vykreslíš graf na monitore, dobre? Pozor! Údaje nasledujú* ***teraz****. 10011010111001101010101110001010. Pozor! Koniec prenosu.* ☺) Vhodnejším príkladom pre túto kapitolu je komunikácia dvoch programov pomocou zdieľanej dátovej štruktúry, či proste prístup dvoch programov k jednej premennej, poľu, položke databázy, súboru a pod. Všeobecne hovoríme o prostriedkoch (tiež o zdrojoch či zariadeniach), ktoré potrebujeme zdieľať.

Systém obsahuje konečný počet zdrojov na distribúciu medzi niekoľko súťažiacich procesov. Zdroje sú rozdelené do niekoľkých typov, každý obsahuje niekoľko identických inštancií. Zdrojmi nemusia byť len vstupno-výstupné zariadenia!

Napríklad pamäťový priestor, cykly procesora, súbory a vstupno–výstupné zariadenia (ako tlačiarne a páskové mechaniky) sú tiež príkladom zdrojov. Ak má systém dva procesory, tak zdrojový typ procesor má dve inštancie. Podobne, zdrojový typ tlačiareň môže mať päť inštancií.
Ak proces žiada o inštanciu zdrojového typu, pridelenie hociktorej inštancie daného typu uspokojí požiadavku. Ak nie, potom nie sú inštancie identické a triedy zdrojových typov neboli definované správne. Napríklad, systém obsahuje dve tlačiarne. Tieto dve tlačiarne môžu byť definované v jednej zdrojovej triede, ak nikomu nevadí, ktorá tlačiareň vytlačí daný dokument. Ale, ak jedna tlačiareň je na deviatom poschodí a druhá je na prízemí, potom ľudia na deviatom poschodí nemusia vidieť tieto tlačiarne ako ekvivalentné a odlišné zdrojové triedy musia byť definované pre každú tlačiareň.
Proces musí žiadať zdroj pred jeho použitím a musí uvoľniť zdroj po jeho použití. Proces môže vyžadovať toľko zdrojov, koľko potrebuje na vykonanie danej úlohy. Samozrejme počet žiadaných zdrojov nesmie prekračovať celkový počet zdrojov dostupných v systéme. Inými slovami, proces nemôže požadovať tri tlačiarne, ak systém obsahuje iba dve.
Pri normálnom režime operácií, proces môže použiť zdroj len v nasledujúcom poradí:
1. **Požiadavka** (*Request*): Ak požiadavke nemôže byť vyhovené hneď (napríklad, zdroj je užívaný iným procesom), potom žiadajúci proces musí čakať, až kým môže získať zdroj.
2. **Použitie** (*Use*): Proces môže používať zdroj (napríklad, ak zdroj je tlačiareň, proces môže tlačiť na tlačiarni).
3. **Uvoľnenie** (*Release*): Proces uvoľní zdroj.

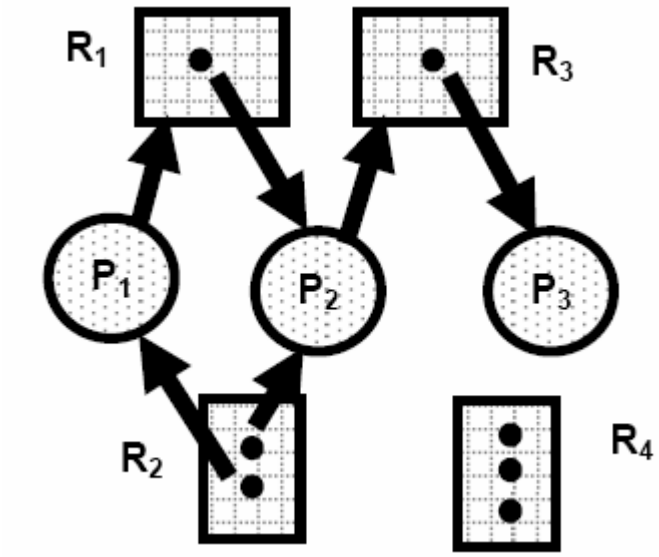
Množina procesov môže byťv **uviaznutom stave**, ak každý proces v množine čaká na udalosť, ktorá môže byť spôsobená iným procesom v množine. Udalosti, o ktoré prevažne ide, sú získavanie a uvoľňovanie zdrojov.

Na ilustráciu stavu uviaznutia, uvažujeme o systéme s tromi páskovými mechanikami. Predpokladajme, že každý z troch procesov drží jednu páskovú mechaniku. Ak každý proces požiada o inú mechaniku, tak každý proces bude v stave uviaznutia. Každý čaká na udalosť „pásková mechanika je uvoľnená“, ktorá môže byť spôsobená iba jedným z ostatných čakajúcich procesov. Tento príklad objasňuje uviaznutie vyvolané rovnakým zdrojovým typom.
Uviaznutie môžu taktiež vyvolať rôzne zdrojové typy. Napríklad, uvažujme o systéme s jednou tlačiarňou a jednou páskovou mechanikou. Predpokladajme, že proces P_i používa páskovú mechaniku a proces P_j používa tlačiareň. Ak P_i požiada o tlačiareň a P_j požiada o páskovú mechaniku, nastane uviaznutie.
Programátor, ktorý vyvíja viacvláknové aplikácie musí dávať pozor na tento problém: viacvláknové programy sú dobrými kandidátmi pre uviaznutia, pretože viaceré vlákna môžu súťažiť o zdieľané zdroje.

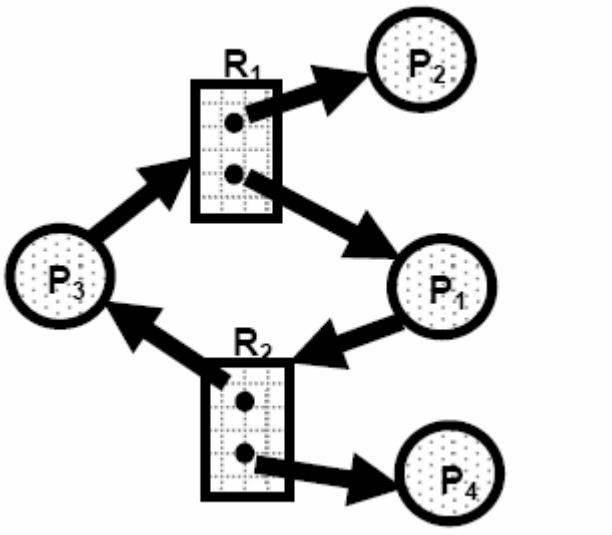
Situácia uviaznutia môže vzniknúť pri nasledujúcich štyroch podmienkach platných súčasne:
1. **Vzájomné vylúčenie** (*Mutual Exclusion*): Aspoň jeden zdroj musí byť v nezdieľateľnom móde, to znamená, iba jeden proces v čase môže používať tento zdroj. Ak ďalší proces požiada o tento zdroj, žiadajúci proces musí byť oneskorený, až kým zdroj nie je uvoľnený.
2. **Drž a čakaj** (*Hold and Wait*): Proces musí držať aspoň jeden zdroj a čakať na získanie ďalších zdrojov, ktoré sú držané ostatnými procesmi.
3. **Žiadna prednosť** (*No Preemption*): Zdroje nemôžu byť uprednostňované; to znamená, že zdroj môže byť uvoľnený iba dobrovoľne procesom, ktorý ho drží, až po tom, čo proces skončil prácu so zdrojom.
4. **Kruhové čakanie** (*Circular Wait*): Množina $\{P_0, P_1, ..., P_n\}$ čakajúcich procesov musí

existovať taká, že P_0 čaká na zdroj, ktorý drží P_1, P_1 čaká na zdroj, ktorý drží P_2, ..., P_{n-1} čaká na zdroj, ktorý drží P_n a P_n čaká na zdroj, ktorý drží P_0.

Na nasledujúcom obrázku je graf pridelenia zdrojov. Kedy môže vzniknúťuviaznutie?



V tomto prípade máme síce cyklus $P_1 \rightarrow R_2 \rightarrow P_3 \rightarrow R_1 \rightarrow P_1$, avšak, tu nie je uviaznutie. Pozorujeme, že proces P_4 môže uvoľniť svoju inštanciu
zdrojového typu R_2. Tento zdroj môže potom byť pridelený P_1, čo zlomí cyklus.



Principiálne môžeme jednať s uviaznutím jedným z troch spôsobov:
• Môžeme použiť protokol na prevenciu alebo zabránenie uviaznutia zabezpečujúci, že systém sa nikdy nedostane do stavu uviaznutia.
• Môžeme systému povoliť vstúpiť do stavu uviaznutia, detekovať ho a vyliečiť.
• Môžeme ignorovať tento problém celkom a predstierať, že žiadne uviaznutia sa nikdy nevyskytnú. Toto riešenie je použité vo väčšine operačných systémov, vrátane UNIX.

Prevencia uviaznutia je množina metód na zabezpečenie toho, aby aspoň jedna z nutných podmienok nenastala. Tieto metódy predchádzajú uviaznutiam obmedzovaním, ako žiadosti na zdroje môžu byť robené.
Zabránenie uviaznutia vyžaduje, aby operačnému systému boli dané dodatočné informácie

obsahujúce, ktoré zdroje bude proces žiadať a používať počas svojho života. S týmito dodatočnými znalosťami môžeme rozhodnúť pre každú požiadavku, či proces má alebo nemá čakať. Na rozhodnutie, či daná žiadosť má byť uspokojená alebo musí čakať, systém musí uvažovať zdroje práve voľné, zdroje práve používané všetkými procesmi a budúce žiadosti a uvoľnenia každého procesu.
Detekcia a liečenie uviaznutia. Ak systém nepoužíva prevenciu uviaznutia ani zabránenie uviaznutia, potom uviaznutie môže vzniknúť. V tomto prostredí systém môže poskytovať algoritmus, ktorý preskúma stav systému a určí, či vzniklo uviaznutie a algoritmus na vyliečenie uviaznutia (ak uviaznutie naozaj nastalo).

Manuálne metódy. Ak systém nezabezpečí, že uviaznutie nikdy nenastane a taktiež neposkytuje mechanizmus na detekciu a liečenie uviaznutia, tak môže nastať situácia, kedy systém je v stave uviaznutia, avšak nemá žiadny spôsob na rozpoznanie toho, čo sa prihodilo. V tomto prípade, nedetekované uviaznutie bude mať za následok zhoršovanie výkonu systému, pretože prostriedky sú držané procesmi, ktoré nemôžu bežať a preto stále viac a viac procesov, ktoré požiadajú o systémové prostriedky, vstúpi do stavu uviaznutia. Eventuálne, systém přestane fungovať a bude potrebné vykonať manuálny reštart.

Napriek tomu, že táto metóda sa nezdá byť tým najsprávnejším ošetrením stavu uviaznutia, je používaná v niektorých operačných systémoch. V mnohých systémoch sa uviaznutia vyskytujú len zriedka (povedzme, raz za rok). Táto metóda je lacnejšia, ako zložitá prevencia, zabraňovanie alebo detekcia a liečenie uviaznutí, ktoré musia byť používané nepretržite. Taktiež, za určitých okolností systém je zamrznutom stave, ale nie v stave uviaznutia. Ako príklad, uvažujme proces najvyššej priority bežiaci v reálnom čase a nikdy nevracajúci riadenie späť operačnému systému. Preto musia mať systémy zabudované funkcie manuálnej obnovy pre takýto stav zamrznutia a môžu jednoducho použiť tieto metódy aj pre liečenie uviaznutia.

Zdroj:
Studenovsky: Operačné systémy
Schmotzer: Operačné systémy
http://maturitazinf.mrazovci.eu/