Proč synchronizace procesů?

- Aby se zabránilo současnému přístupu několika procesů do kritické sekce.
- Kritická sekce je část kódu která přistupuje ke sdílenému prostředku
  - soubor - čtení / zapisování
  - zařízení - tiskárna, chytré spotřebiče

Problém Producent vs Konzument

- Producent je proces který vytváří data (zapisování do souboru)
- Konzument je proces který pracuje s těmi daty (ukládá, zobrazuje na obrazovku, analyzuje)
- mezi nimi je vyrovnavací paměť (buffer) aby si mohli data předat - má omezenou kapacitu
  - producent dává hodnoty do bufferu a konzument je bere a zpracovává
- je tam `count` - počítá položky v bufferu
- problém
  - když producent a konzument přistoupí najednou k count
  - přistoupí najednou: count = 3
    - producent -> count++ (3+1 = 4)
    - konzument -> count-- (3-1 = 2)
  - jiné hodnoty - zaseknutí se procesu
- řešení
  - zesynchronizování přístupu k buffru a count
  - semafory a tak... - popsáno níže

Běžící proces	Akce	Výsledek
Producent	R0 = count	R0 = 3
	R0 += 1	R0 = 4
Konzument	R1 = count	R1 = 3
	R1 -= 1	R1 = 2
Producent	count = R0	count = 4
Konzument	count= R1	**count = 2**

Kritická Sekce

- část strojového kódu která pracuje se sdíleným prostředkem
- hrozí přístup více vláken
  - **podmínky - pro přístup k KS**
    - Dva procesy nesmí být ve stejné KS najednou.
    - Rychlost procesoru nesmí mít vliv na řešení KS. - hazardy
    - Proces mimo KS nesmí blokovat proces který do něj má vstoupit.
    - Proces nesmí čekat nekonečně dlouho na přístup k KS.

Aktivní a pasivní čekání

Aktivní
- proces se neustále ptá jestli může přistoupit k KS
- plytvání času

Pasivní
- proces se zeptá o přístup
  - když nemůže přisoupit - je uspán do času kdy se uvolní
  - hrozí dead lock - uvaznutí ve spánku
  - problém - chtějí si navzájem vyhovět - 2 lidé a úzká chodba

Synchronizace procesů

- Problém Producent vs. Konzument
- Kritická sekce
  - Charakteristika a podmínky pro ošetření KS
  - Aktivní vs. pasivní čekání
- Sdílený prostředek
- Řešení KS
  - Zákaz přerušení
  - Zamykací proměnná
  - Přesné střídání
  - Petersonovo řešení
  - Atomická instrukce
  - Sleep() a Wakeup()
  - Semafory a transakce
- Klasické synchronizační problémy

Synchronizační problémy

- Producent Konzument
- Čtenáři písáři
  - cílem je povolit čtení více čtenářům, když pisaři spí
  - a povolit psání jednomu pisaři když čtenáři spí
  - problém
    - pisář může čekat nekonečně dlouho když stále přichází noví čtenáři
- Věřící filozofové
  - každý z nich střídá činnost - jí, přemýšlí
  - mezi nimi je vždy jedna vidlička
  - aby mohl jíst potřebuje dvě vidličky - levá pravá
    - vidličku nejde sdílet
  - problém
    - Když všichni filozofové najednou zvednou levou lžíčku a čekají na pravou, a nikdo se nedočká.
- Spicí holič
  - holič pokud nemá zákazníka - usne
  - když přijde zákazník - vzbudí ho
  - když dokončí střih jde zkontrolovat čekárnu
  - problém
    - Holič dostřihá a jde do čekárny. V tom přijde nový proces a holič je ještě ve stávu střihu, tak jde do čekárny a holič v čekárně nikoho neviděl tak se uspal.

Zákaz přerušení

- po vstoupení procesu do KS se zakáže přerušení toho procesu
  - procesor nebude přepínat mezi procesy když tam už někdo je
  - tím pádem tam nemůže žáden další proces vstoupit
- proces sí udělá svojí práci se sdíleným prostředkem
- až když skončí práci - povolí přerušení
- problém
  - když proces nepovolí přerušení - zamrzení systému
  - blokuje to celý systém - procesy které by mohly běžet nemužou
  - funguje to jen na jednom jádře
- je to starší řešení

Zamykací proměnná

- je tam proměnná lock, která říká zda je KS volná nebo obsazená
- mění znaménko - 1 nebo 0
  - operace if() a lock=1 nejsou atomické - neprovedou se najednou
- problém
  - proces1 zjistí - volno
  - připravuje se ke vstoupení
    - v okamžiku se proces2 zeptá
      - proces1 tam zatím není tak řekne - volno
  - oba vstoupí do KS zároveň

Přesné střídání

- je tam proměnná turn, která říká který proces povoluje k ppřistoupení
- např.:
  - turn=1 - proces1 může vstoupit do KS
  - turn=0 - proces0 může vstoupit do KS
  - když proces1 je v KS - proces0 čeká
- po dokončení se dále může vykonávat v nekritické sekci
- problém
  - každý proces dostává turn
  - ty procesy, které nepotřebují KS blokují další procesy, které potřebují

Petersonovo řešení

- řeší problém přesného střídání
- má proměnnou která zjistí a říká jestli proces potřebuje KS
- proces vstoupí do KS jen když reálně potřebuje řešit něco v ks

Atomická instrukce

- zajišťuje aby nějaký proces projel celý najednou
- např.: x++ - je to pár kroků
  - načti x
  - přičti 1
  - ulož do paměti
- může do toho vstoupit jiný proces a přeruší se
- proměnná lock - uzavření procesu - aby se vykonál celý
- procesor při vykonávání instrukce uzamkne datovou sběrnici

Sleep / Wake

- je to systémové volání, které uspí nebo probudí daný proces
- když counter není shodný s countem jaký má proces - proces se uspí
- když systémové volání zavolá proces - probudí se
- výhoda - nezatěžujou procesor - spí

Semafor

- má pouze 2 hodnoty - 1 nebo 0 (červená, zelená)
- procesy přicházejí a semafor za jeden krok
  - pustí procesy do fronty semaforu a jeden proces si veme do KS
- je tam také fronta ready
  - je pro procesy které jsou připravené a čekají jen na přidělení CPU