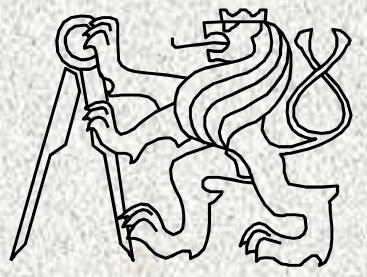
VLÁKNA



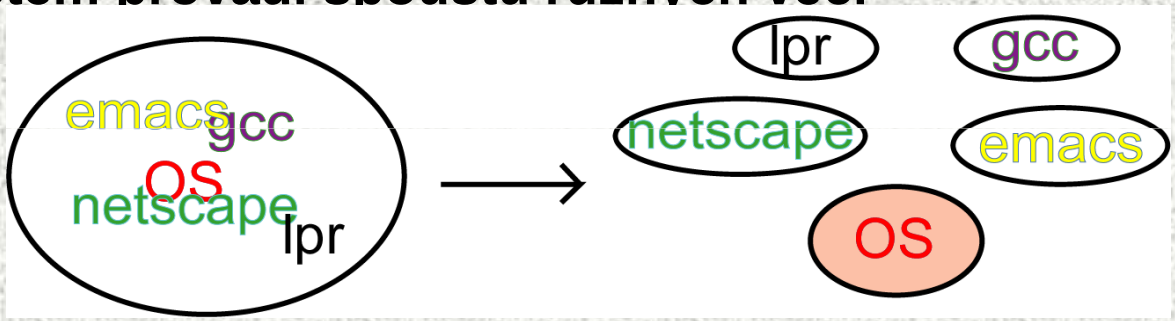
A0B36PR2-Programování 2

Fakulta elektrotechnická
České vysoké učení technické

Procesy

- Veškerý běžící software v systému je organizován jako množina "sekvenčně" běžících procesů.

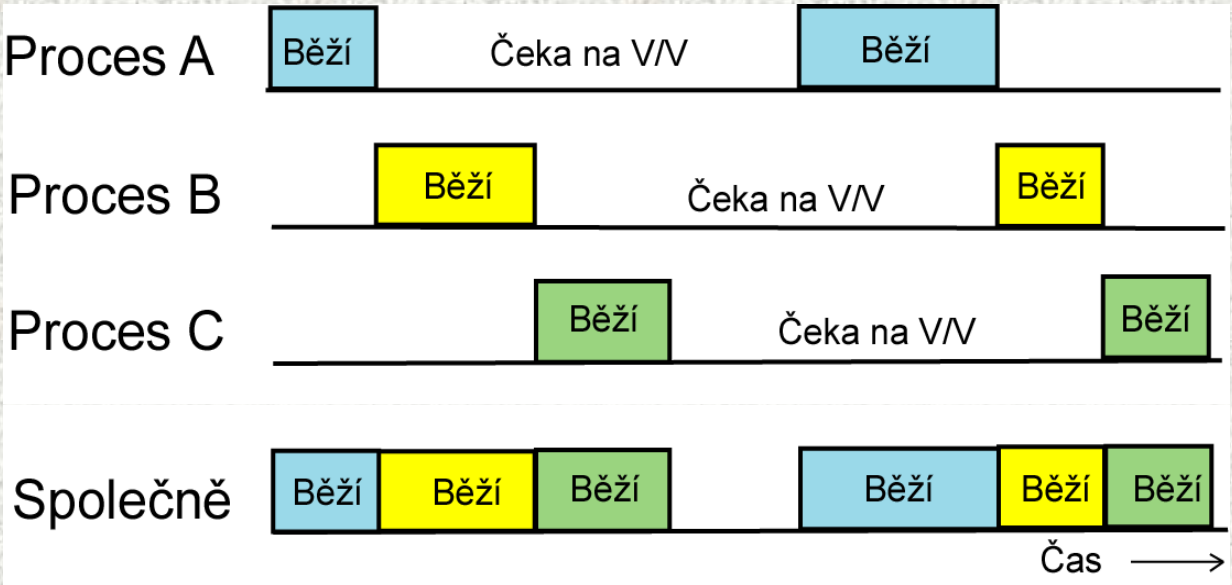
- Proč procesy?
  - **Systém provádí spoustu různých věcí**



1. Jednoduchost - jak to zjednodušit?
   - **Z každé jednotlivé akce udělat izolovaný proces.**
   - **OS se zabývá v jednom okamžiku pouze jednou věcí.**
   - **Univerzální trik pro správu složitých problémů:**
     - **dekompozice problému!**

Procesy (2)

2. Proč procesy? Rychlost



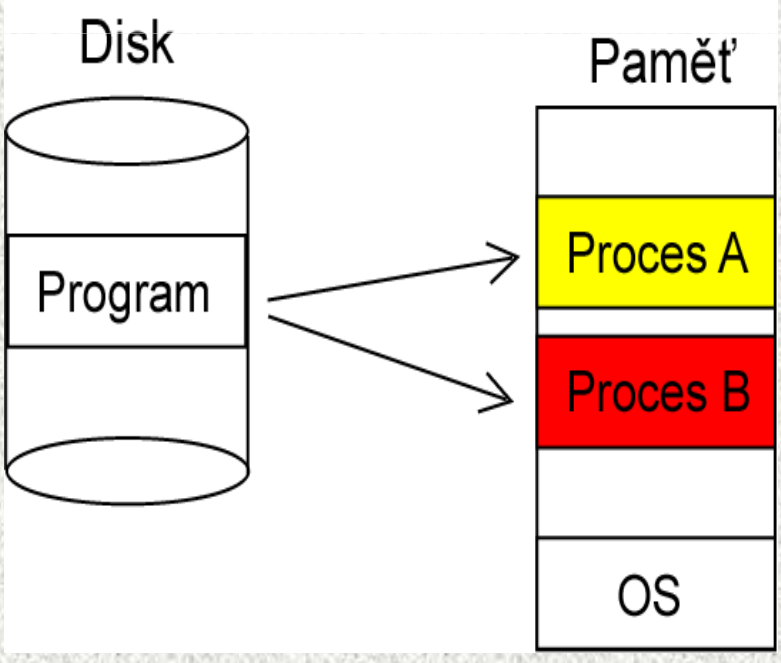
3. Proč procesy? V/V „paralelismus“

- **Zatímco jeden proces čeká na dokončení V/V operace jiný proces může používat CPU.**
- **Překrývání zpracování: dělá z 1 CPU “více CPU”.**
- **Reálný paralelismus – více procesorů!**

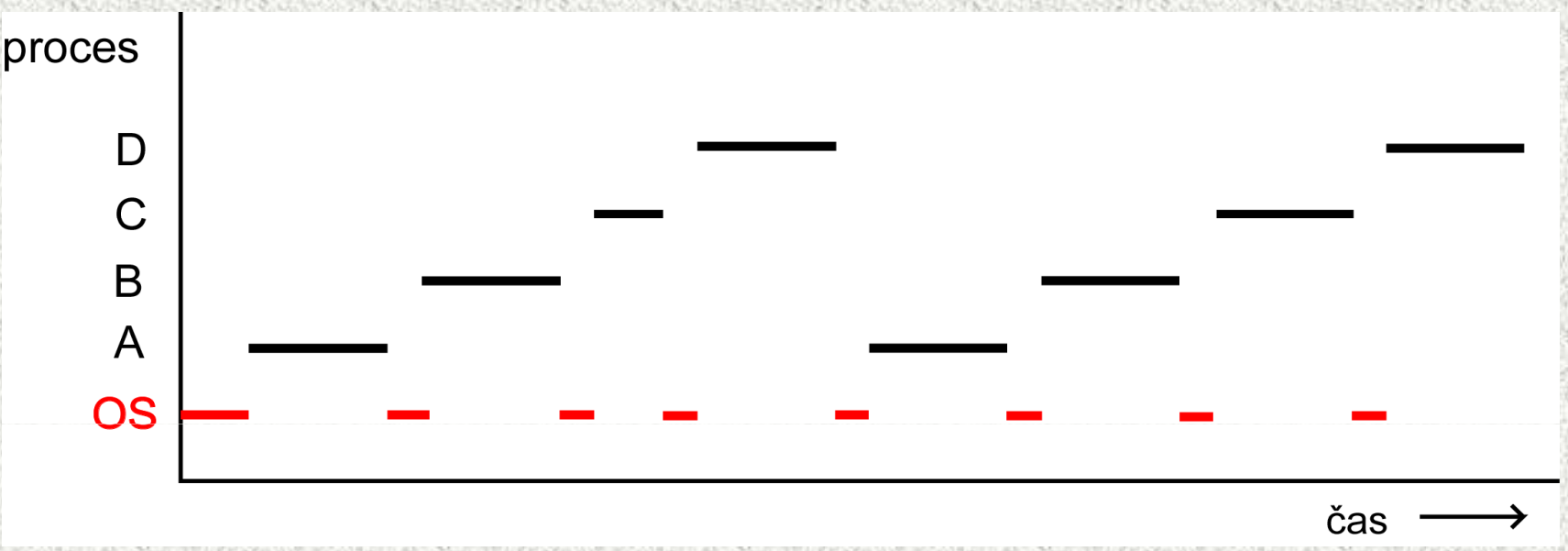
Program versus proces

- Co je program?
  - **posloupnost instrukcí a data uložená v souboru na disku, pasivní.**
- Co je proces?
  - **abstrakce spuštěného programu, zahrnující aktuální hodnoty registrů a proměnných.**

- Příklad: spustíme dvakrát editor.
  - **stejný program,**
  - **ale dva různé procesy.**



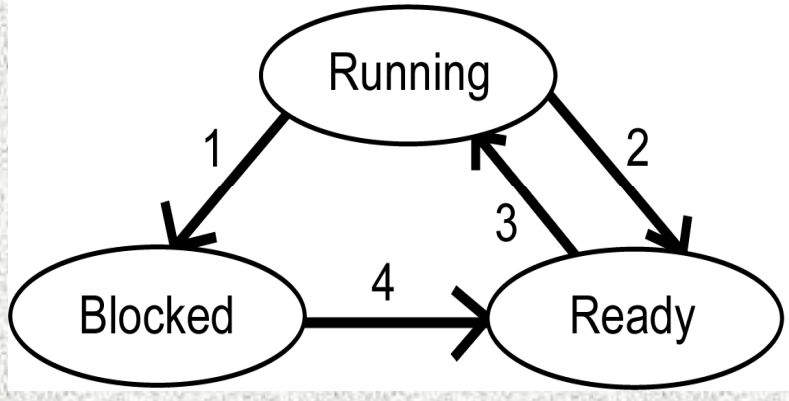
Přepínání kontextu



- CPU v krátkých časových intervalech (řádově milisekundy) přechází od vykonávání jednoho procesu k vykonávání instrukcí druhého procesu.
- Kdo určuje, který další proces poběží?

OS

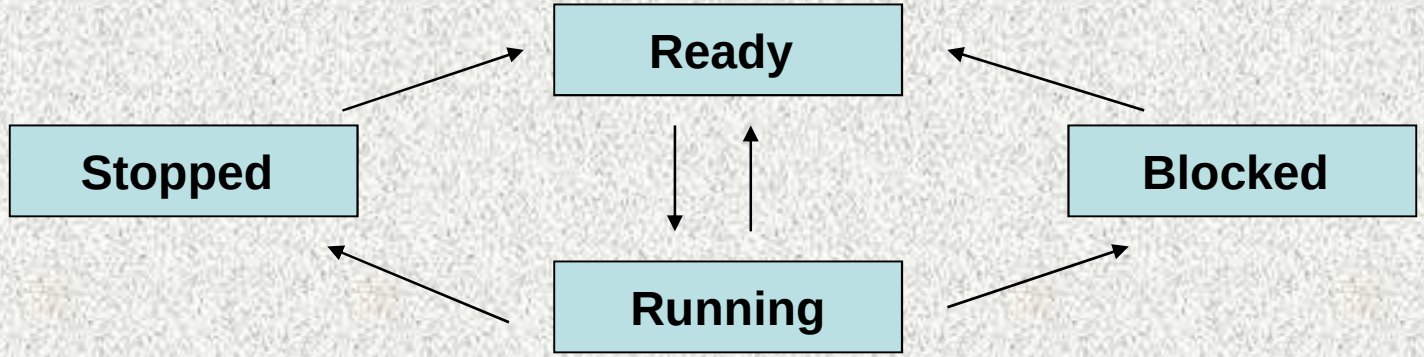
Stavy procesu



1. Čekání na událost nebo prostředek.
2. Vypršelo časové kvantum.
3. Nastal naplánovaný čas.
4. Nastala událost nebo je prostředek k dispozici.

- Stavy procesu
  - **Running = proces pravě používá CPU.**
  - **Blocked = proces čeká na externí událost nebo na prostředek.**
  - **Ready = (pozastavené) proces je připraven a čeká na přidělení CPU.**
  - **Stoped = proces ukončen, není možné pokračovat**

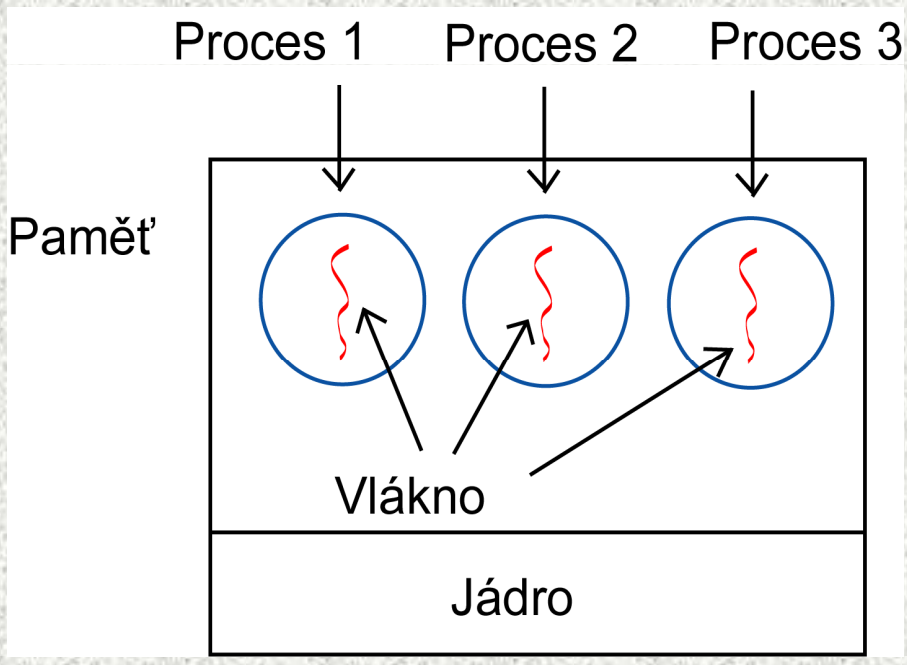
Stavy procesu



- Stavy procesu
  - **Running = proces pravě používá CPU.**
  - **Blocked = proces čeká na externí událost nebo na prostředek.**
  - **Ready = (pozastavené) proces je připraven a čeká na přidělení CPU.**
  - **Stopped = proces ukončen, není možné pokračovat**

Proces versus vlákno

- Proces model
  - **Každý proces alokuje příslušné prostředky (adresové prostor obsahující kód, data a zásobník procesu, otevřené soubory, potomky, reakce na signály, ...)**
  - **Jeden proces = jedno vlákno výpočtu.**



Priority, synchronizace

- Princip priority
  - Vlákno předává řízení dobrovolně, pak vláknu s nejvyšší prioritou
  - Vlákno s vyšší prioritou může přerušit vlákno s prioritou nižší – *preemptivní paralelní zpracování*
- Princip synchronizace
  - Vlákno čeká na dokončení jiného vlákna
  - Problém sdílení datových prostorů
  - Řeší se synchronizační metodou
    - Vlákno v synchronizační metodě nad daným objektem zabrání všem ostatním vláknům volat další synchronizační metodu nad týmž objektem

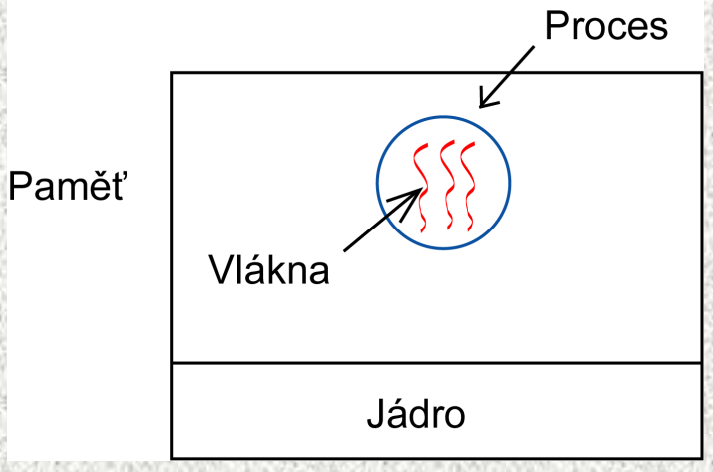
Proces versus vlákno (2)

- Vláknový model – jeden proces – více vláken
  - **Odděluje alokaci prostředků a samotný výpočet.**
  - **Proces slouží k alokaci společných prostředků.**
  - **Vlákna jsou jednotky plánované pro spuštění na CPU.**
- *Příklad: práce na pozadí, vstup/výstup, simulace, producent/konzument*

- **Vlákno má (na rozdíl o procesu)**
  - **svůj vlastní program counter (pro uchování informace o výpočtu),**
  - **registry (pro uchování aktuálních hodnot),**
  - **zásobník (který obsahuje historii výpočtu),**
  - **lokální proměnné,**
  - **ale ostatní prostředky jsou sdílené.**

Vláknový model

- Jednotlivá vlákna v daném procesu nejsou nezávislá tak jako jednotlivé procesy.
- Všechny vlákna v procesu sdílí stejný adresový prostor, stejné otevřené soubory, potomky, reakce na signály, …
- Multithreading
  - **Procesy se spouští implicitně pouze s jedním vláknem.**
  - **Toto vlákno může vytvářet další vlákna pomocí knihovní funkce**
  - **Když chce vlákno skončit, může se opět ukončit pomocí knihovní funkce**

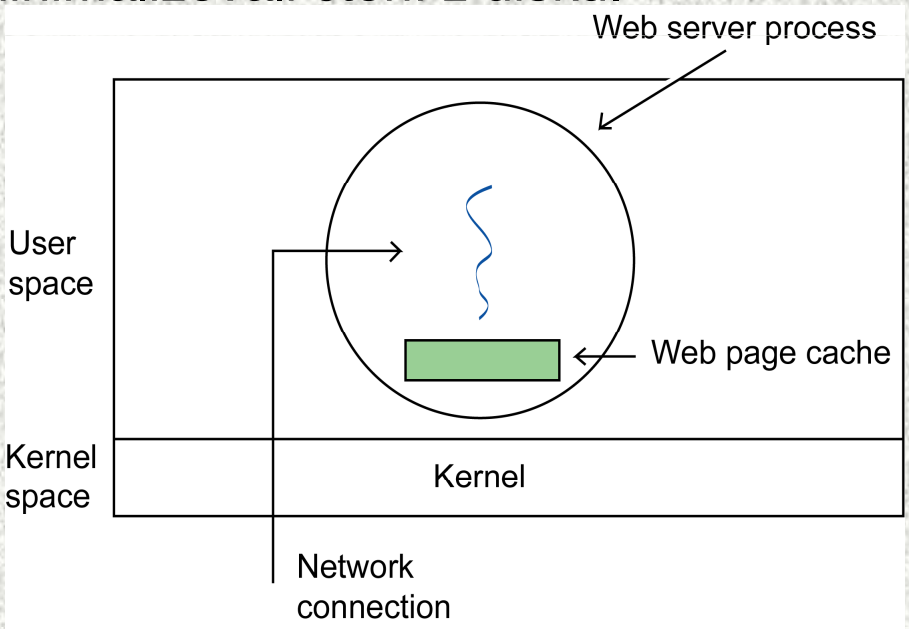


Aplikace vláken

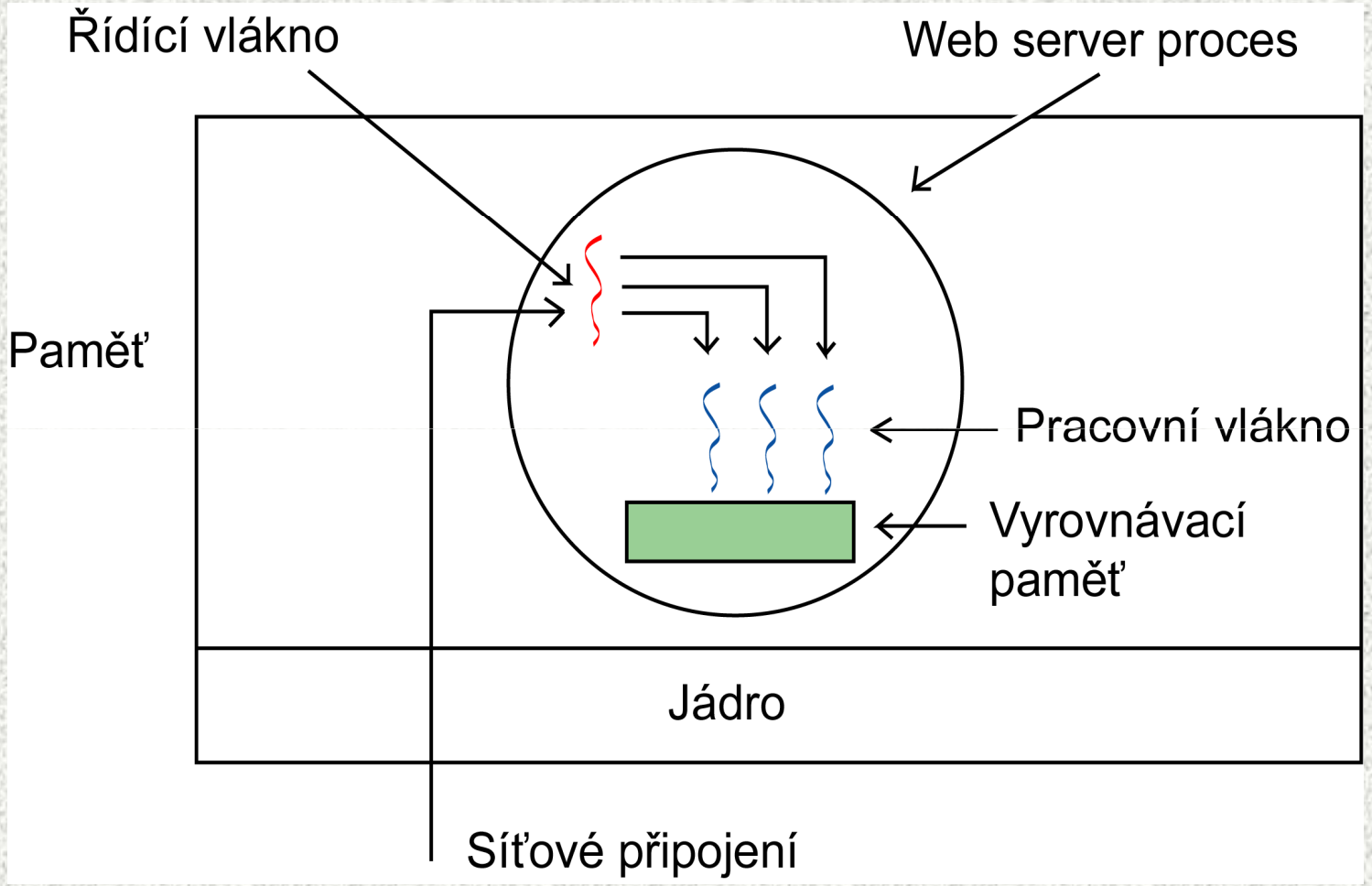
- Časově náročné akce
  - Delší než 1 vteřina
    - Obrázky
- Čekání na vstup od uživatele
  - na jednotlivé klávesy, zamyšlení, náročnější vstup
    - Kontrola pravopisu
- Opakující se výpočty
  - Simulace, počítačové hry
    - „paralelní spuštění činnosti“
- Úlohy typu producent-konzument
  - „paralelní příprava dat“

Příklad: jednovláknový Web Server

- Klient
  - **pošle požadavek na konkrétní web. stránku**
- Server
  - **ověří zda klient může přistupovat k dané www stránce**
  - **načte stránku a pošle obsah stránky klientovi**
- Pozn.: Často používané stránky zůstávají uloženy v hlavní paměti, abychom minimalizovali čtení z disku.



Příklad: vícevláknový Web Server



Příklad: vícevláknový Web Server (2)

- Řídící vlákno
  - **čte příchozí požadavky,**
  - **zkoumá požadavek,**
  - **vybere nevyužité pracovní vlákno a předá mu tento požadavek.**

- Pracovní vlákno
  - **načte požadovanou stránku z hlavní paměti nebo disku a pošle ji klientovi.**

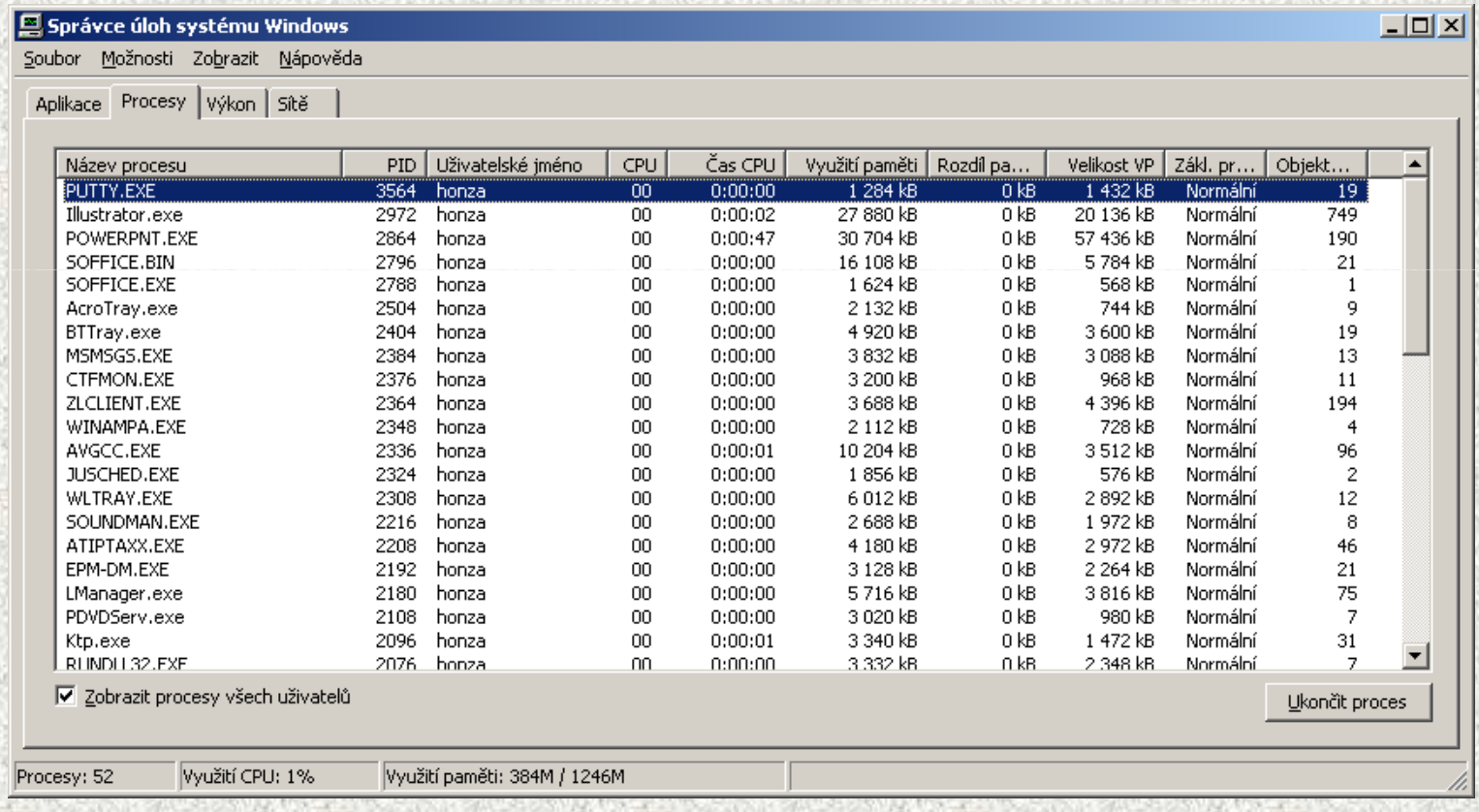
Výhody vláken

- Snadné sdílení informací
  - **data uvnitř daného procesu jsou přístupná pro všechny vlána daného procesu.**

- Rychlé přepínání kontextu mezi vlákny daného procesu.

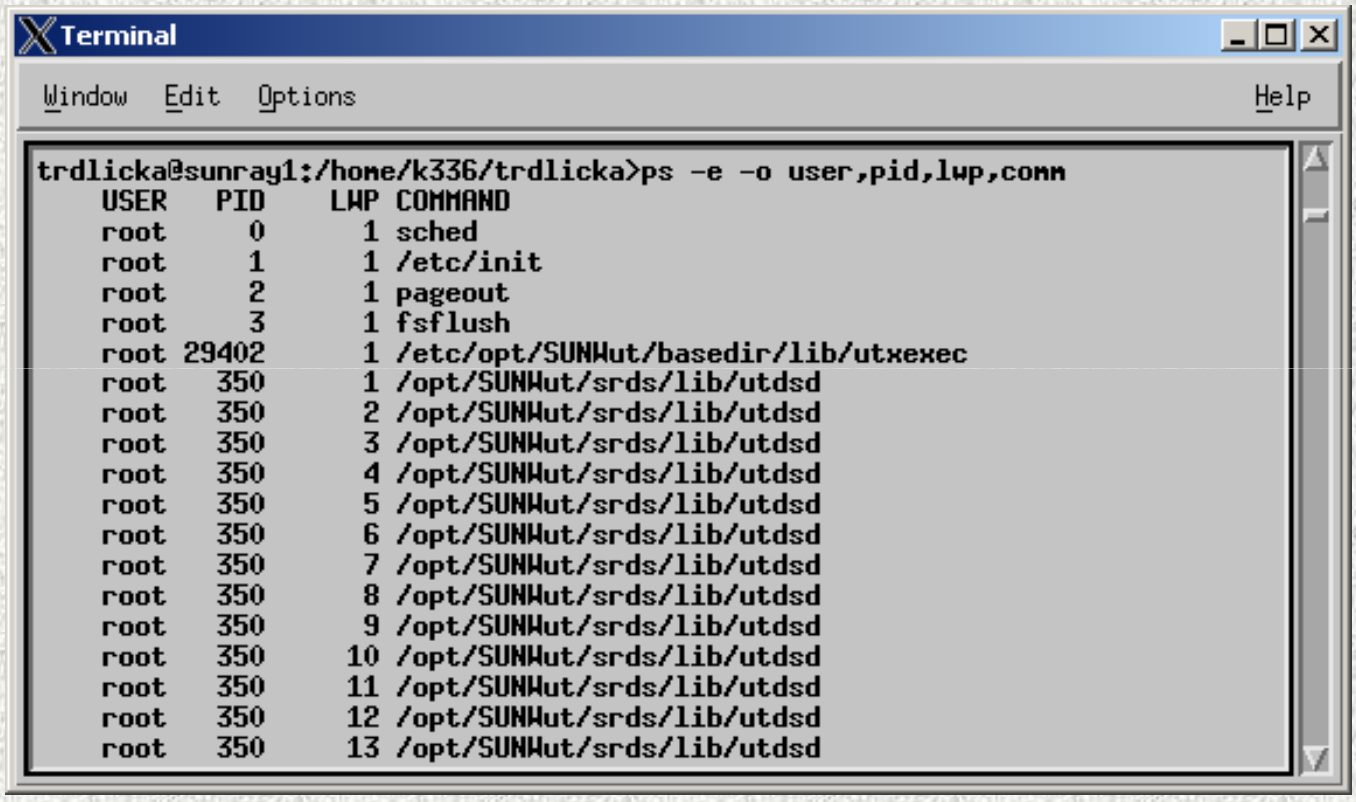
Zobrazení informací o vláknech

- MS Windows XP
  - CTRL ALT DEL → Správce úloh



Zobrazení informací o vláknech (2)

- Solaris
  - **např.pomocí příkazu ps**



  - **V uvedeném výpisu je proces utdsd reprezentován několika vlákny.**

Problémy při komunikaci mezi procesy/vlákny

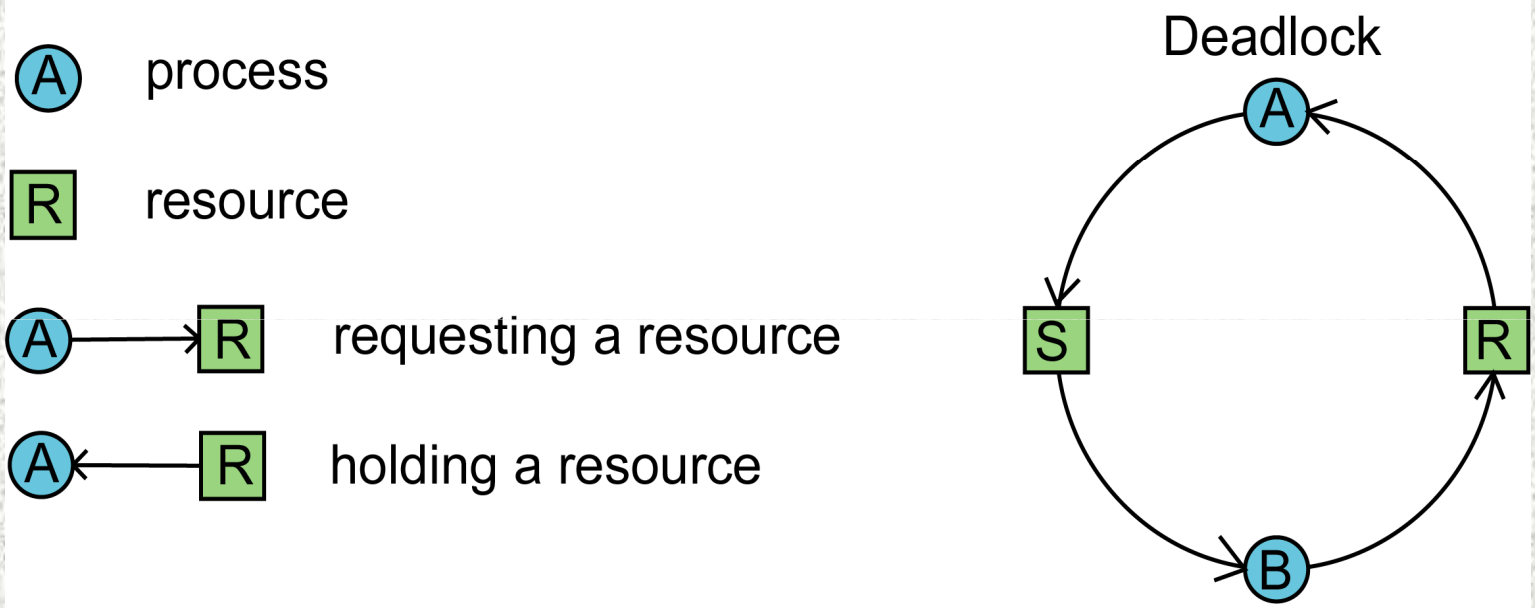
- **Časově závislé chyby**
  - **Pokud dva nebo více procesů čte/zapisuje na sdílené prostředky (např. sdílená paměť, proměnné, soubory,...), pak může dojít k** časově závislé chybě.
  - **Výsledek výpočtu závisí na náhodném průběhu prokládání procesů.**
- **Uváznutí**
  - **Stav, kdy množina procesů nemůže pokračovat v činnosti, protože každý proces z množiny čeká na uvolnění prostředku, přiděleného jinému procesu z množiny.**
- **Stárnutí**
  - **Stav, kdy proces může teoreticky pokračovat v činnosti, ale ve skutečnosti se neustále nachází ve stejném stavu (např. je neustále předbíhán jinými procesy).**
- Tyto chyby se vyskytují náhodně a proto se velmi špatně odhalují!

Řešení časově závislých chyb

- Definujeme kritickou sekci
  - **část programu, kde procesy používají sdílené prostředky (např. sdílenou paměť, proměnnou,...).**
- Časově závislým chybám předcházíme pomocí vzájemného vyloučení.
- Vzájemné vyloučení:
  - **mechanismus, který znemožňuje, aby se dva nebo více procesů nacházelo v související kritické sekci současně.**
- Vzájemného vyloučení dosahujeme pomocí synchronizace procesů.

Modelování uváznutí

- Uváznutí můžeme modelovat pomocí alokačního grafu.



- Každá smyčka v grafu představuje uváznutí (procesy ve smyčce čekají a nemohou pokračovat).

Pro zájemce

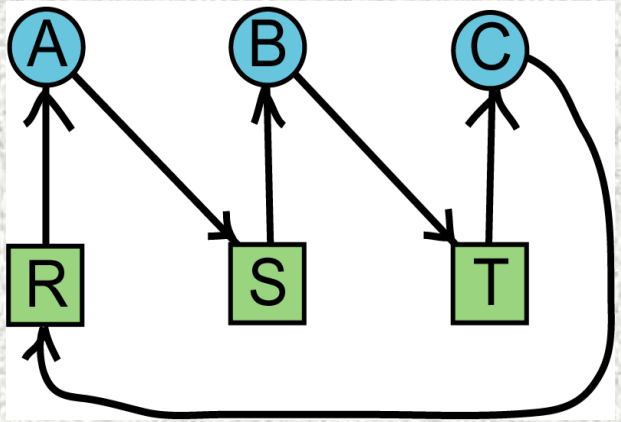
Příklad: Uváznutí procesů

proces A
Žádost o R
Žádost o S
Použití R a S
Uvolnění R
Uvolnění S

proces B
Žádost o S
Žádost o T
Použití S a T
Uvolnění S
Uvolnění T

proces C
Žádost o T
Žádost o R
Použití T a R
Uvolnění T
Uvolnění R

Alokace s uváznutím:
A: Žádost o R
B: Žádost o S
C: Žádost o T
A: Žádost o S ⇒ proces je uspán
B: Žádost o T ⇒ proces je uspán
C: Žádost o R ⇒ **uváznutí !!!**



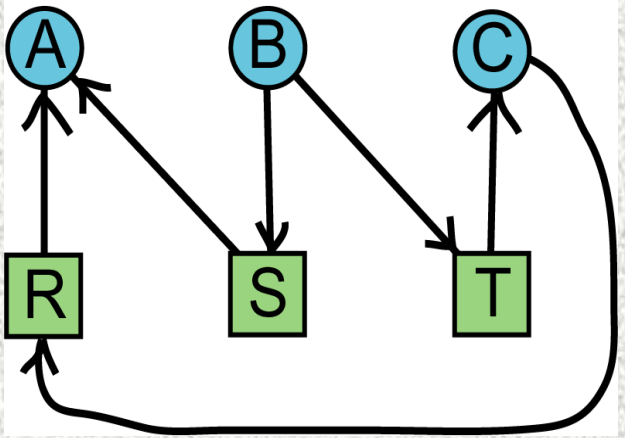
Příklad: Uváznutí procesů (2)

proces A
Žádost o R
Žádost o S
Použití R a S
Uvolnění R
Uvolnění S

proces B
Žádost o S
Žádost o T
Použití S a T
Uvolnění S
Uvolnění T

proces C
Žádost o T
Žádost o R
Použití T a R
Uvolnění T
Uvolnění R

Alokace bez uváznutí:
A: Žádost o R
C: Žádost o T
A: Žádost o S
C: Žádost o R ⇒ proces je uspán
A: uvolnění R . . . **bez uváznutí!!!**



Pro zájemce