



PB 169

Počítačové sítě a operační systémy

Procesy a vlákna

Vnitřní struktura OS

- Existuje řada přístupů a implementací
 - jedno velké monolitické jádro
 - modulární, hierarchický přístup
 - malé jádro a samostatné procesy
- Struktura mnoha OS je poznamenána historií OS a původními záměry, které se mohou od současného stavu radikálně lišit

Struktura s mikrojádroem

- Microkernel System Structure
- Malé jádro OS plní pouze několik málo nezbytných funkcí
 - primitivní správa paměti (adresový prostor)
 - komunikace mezi procesy – Interprocess communication (IPC)
- Většina funkcí z jádra se přesouvá do „uživatelské“ oblasti
 - ovladače HW zařízení, služby systému souborů, virtualizace paměti ...
 - mezi uživatelskými procesy se komunikuje předáváním zpráv



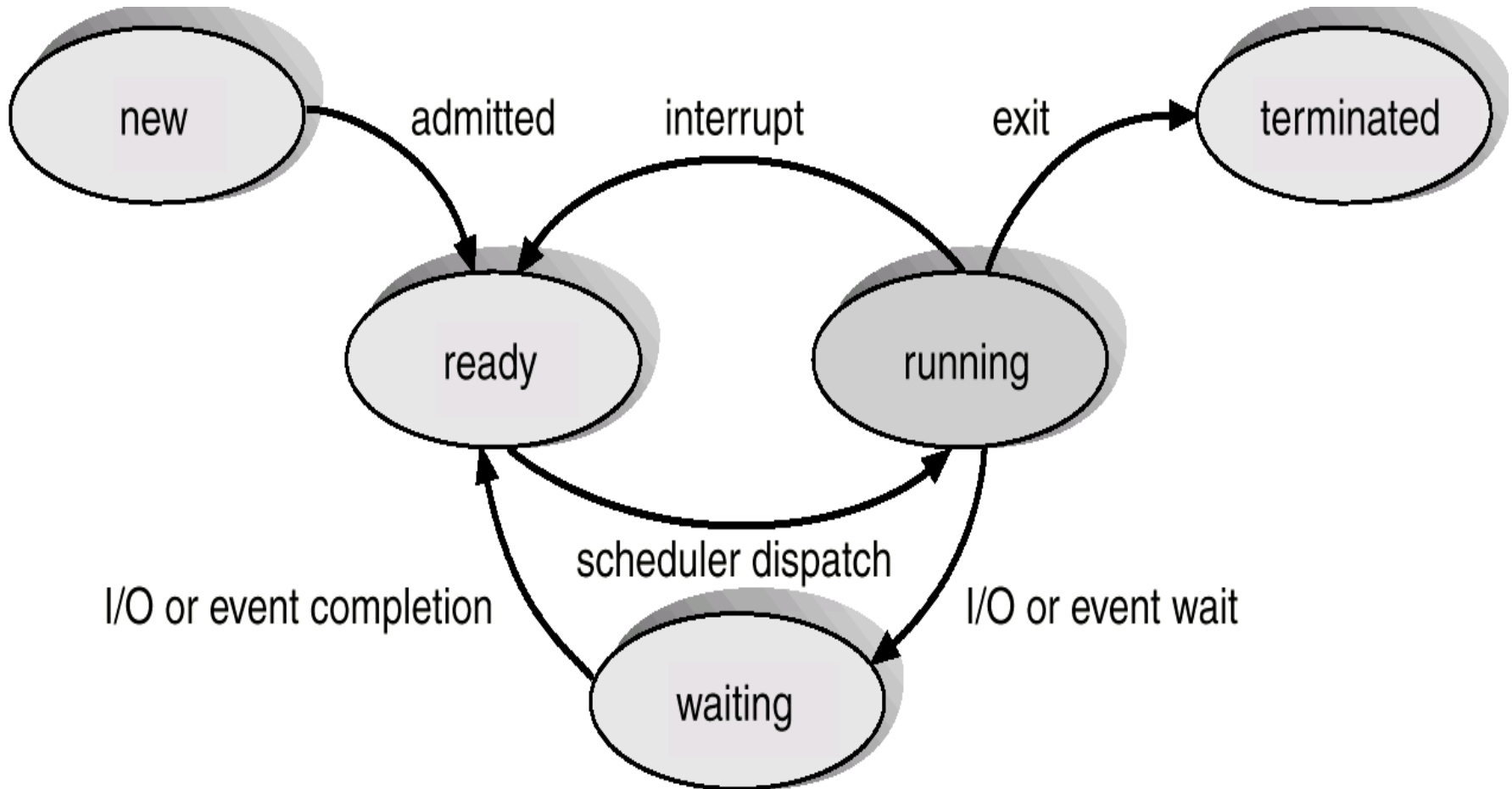
Co je to proces

- Pro spuštěný program máme řadu pojmenování
 - dávkové systémy: úlohy, dávky, jobs
 - multiprogramové systémy: procesy (processes, tasks), vlákna (threads)
- Společné pojmenování pro spuštěný program je proces (někdy používáme synonymum task)
- Dále zavádíme pojem vlákno pro „dílčí“ proces v rámci „procesu“
- Proces obsahuje
 - čítač instrukcí
 - zásobník
 - datovou sekci
 - program

Stavy procesu

- Proces se může nacházet v jednom ze stavů:
 - nový (new): právě vytvořený proces
 - běžící (running): některý procesor právě vykonává instrukce procesu
 - čekající (waiting): čeká na určitou událost
 - připravený (ready): čeká na přidělení času procesoru
 - ukončený (terminated): ukončil své provádění

Stavy procesu

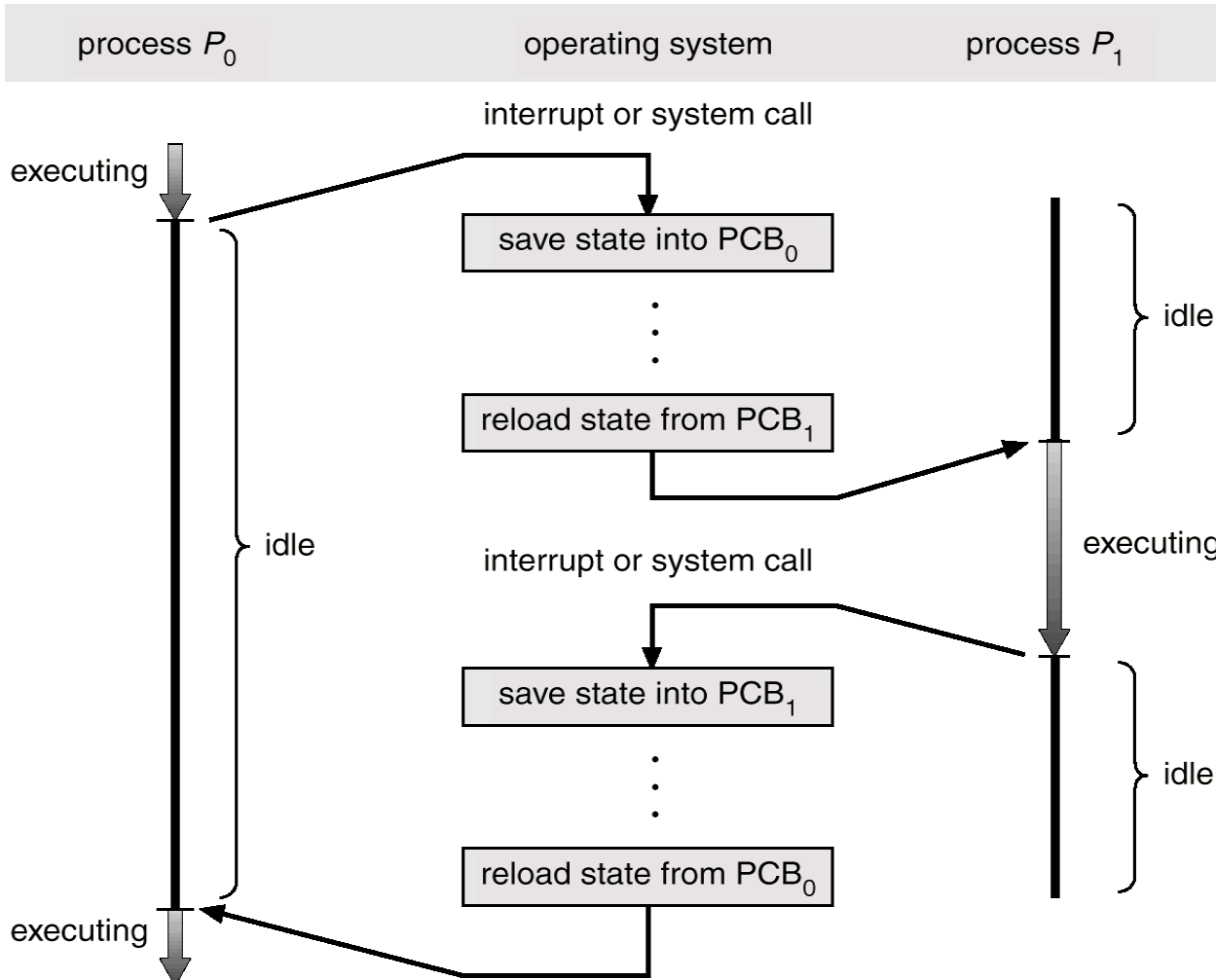


Informace OS o procesu

- Process Control Block -- tabulka obsahující informace potřebné pro definici a správu procesu
 - stav procesu (běžící, připravený, ...)
 - čítač instrukcí
 - registry procesoru
 - informace potřebné pro správu paměti
 - informace potřebné pro správu I/O
 - účtovací informace

pointer	process state
process number	
program counter	
registers	
memory limits	
list of open files	
⋮	

Přepnutí procesu





Přepnutí kontextu

- Vyžádá se služba, akceptuje se některé asynchronní přerušení, obslouží se a nově se vybere jiný běžící proces
- Když OS přepojuje CPU z procesu X na proces Y, musí:
 - uchovat (uložit v PCB procesu X) stav původně běžícího procesu
 - zavést stav nově běžícího procesu (z PCB procesu Y)
- Přepnutí kontextu představuje režijní ztrátu (zátěž)
 - během přepínání systém nedělá nic efektivního
- Doba přepnutí závisí na konkrétní HW platformě
 - Počet registrů procesoru, speciální instrukce pro uložení/načtení všech registrů procesoru apod.
- Při přerušení musí procesor
 - uchovat čítač instrukcí
 - zavést do čítače instrukcí hodnotou adresy vstupního bodu ovladače přerušení z vektoru přerušení



Vytvoření procesu

- Rodič vytváří potomky (další procesy)
- Potomci mohou vytvářet další potomky ...
- Vzniká strom procesů
- Sdílení zdrojů – varianty při vytváření potomků
 - rodič a potomek sdílejí zdroje původně vlastněné rodičem
 - potomek sdílí rodičem vyčleněnou podmnožinu zdrojů s rodičem
 - potomek a rodič jsou plně samostatné procesy, nesdílí žádný zdroj
- Běh
 - rodič a potomek mohou běžet souběžně
 - rodič čeká na ukončení potomka



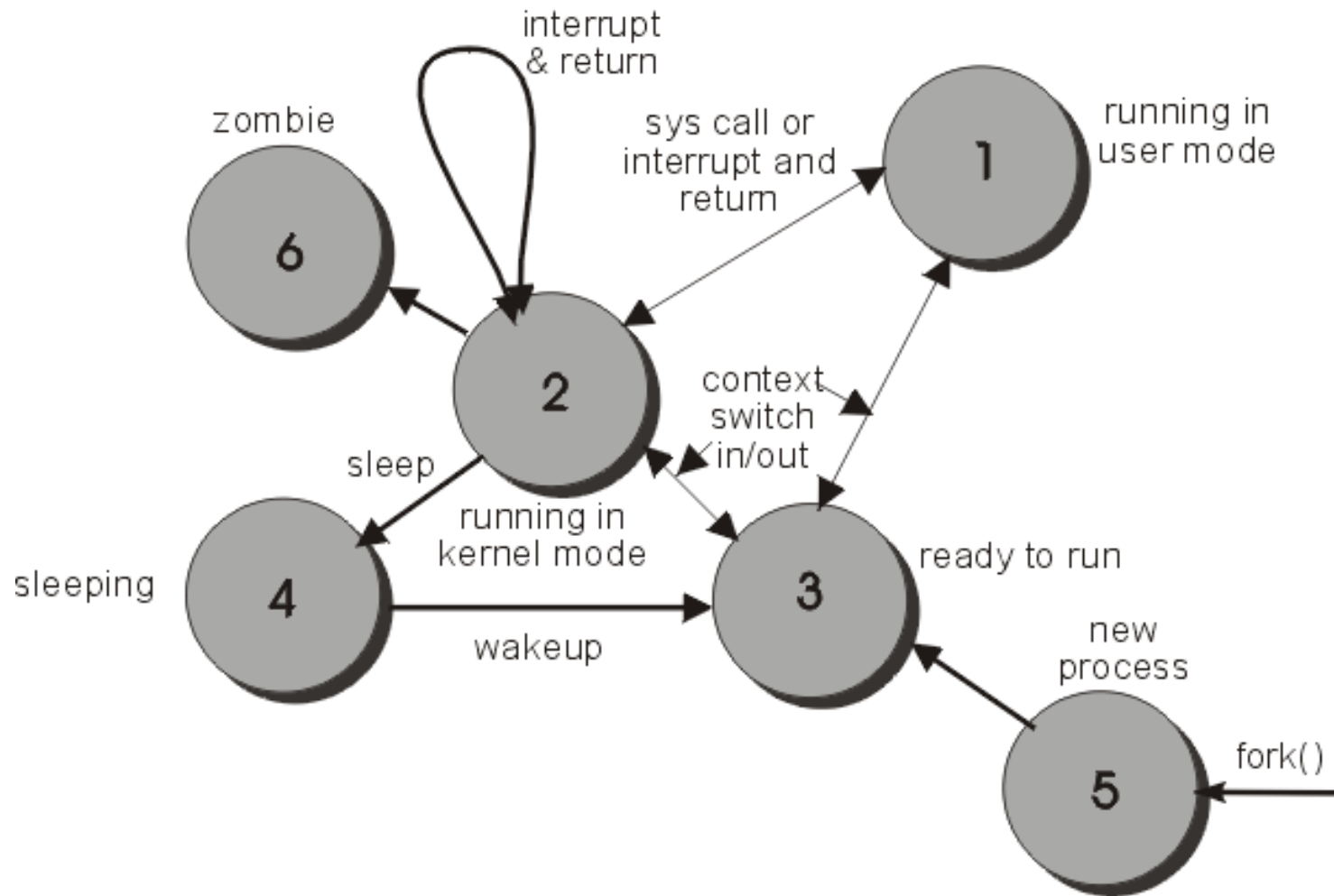
Ukončení procesu

- Proces provede poslední příkaz a sám požádá OS o ukončení
 - výstupní data procesu se předají rodiči (pokud o to má zájem – např. čeká na ukončení potomka voláním wait)
 - zdroje končícího procesu se uvolňují operačním systémem
- O ukončení procesu žádá jeho rodič (nebo jiný proces s dostatečnými právy), protože např.:
 - potomek překročil stanovenou kvótu přidělených zdrojů
 - úkol přidělený potomkovi rodič již dále nepotřebuje
 - rodič končí svoji existenci a nebylo povoleno, aby potomek přežil svého rodiče
 - může docházet ke kaskádnímu ukončování (ukončí se celá větev stromu procesů)

● ● ● | Příklad: Linux

- volání `fork()` implementováno jako copy-on-write (tj. dokud paměť není měněna je sdílena a až při pokusu o modifikaci je vytvořena kopie)
- `vfork` – upravené `fork`, které nekopíruje stránky paměti rodičovského procesu
 - rychlejší
 - vhodné pro okamžité spuštění `execve`
- `clone` – upravené `fork`, které umožňuje sdílet některé zdroje (například paměť, deskriptory souborů, ovladače signálů) mezi rodičovským a nově vytvořeným procesem.
- Informace o procesu jsou uloženy ve struktuře `task_struct` (viz `usr/include/sched.h`)

Příklad: Linux (2)





Procesy a vlákna

- Program
 - soubor definovaného formátu obsahující instrukce, data a další informace potřebné k provedení daného úkolu
- Proces
 - systémový objekt charakterizovaný svým paměťovým prostorem a kontextem (paměť i některé další zdroje jsou přidělovány procesům)
- Vlákno, také „sled“
 - objekt, který vzniká v rámci procesu, je viditelný pouze uvnitř procesu a je charakterizován svým stavem (CPU se přidělují vláknům)
- Model – jen procesy (ne vlákna)
 - proces: jednotka plánování činnosti i jednotka vlastnící prostředky
- Model – procesy a vlákna
 - proces: jednotka vlastnící zdroje
 - vlákno: jednotka plánování činnosti

Procesy a vlákna

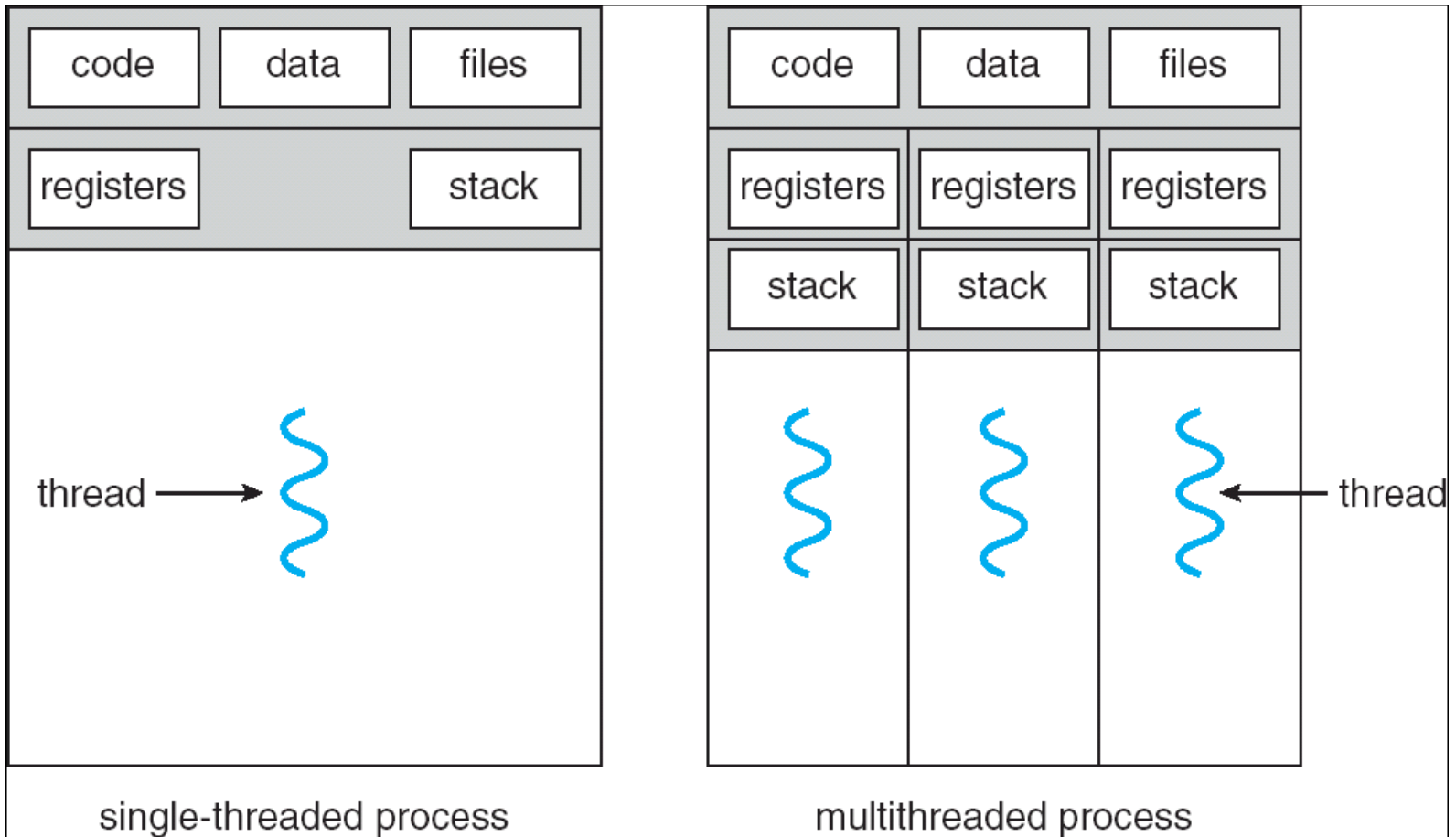
- Každé vlákno si udržuje svůj vlastní
 - zásobník
 - PC (program counter)
 - registry
 - TCB (Thread Context Block)
- Vlákno může přistupovat k paměti a ostatním zdrojům svého procesu
 - zdroje procesu sdílí všechna vlákna jednoho procesu
 - jakmile jedno vlákno změní obsah (nelokální – mimo zásobník) buňky, všechny ostatní vlákna (téhož procesu) to vidí
 - soubor otevřený jedním vláknem mají k dispozici všechny ostatní vlákna (téhož procesu)



Procesy a vlákna

- Proč využít vlákna
 - využití multiprocessorových strojů (vlákna jednoho procesu mohou běžet na různých CPU)
 - jednodušší programování
 - typický příklad: jedno vlákno provádí uživatelem požadovaný úkol a druhé vlákno překresluje obrazovku
- 1:1
 - UNIX Systém V, (MS-DOS)
 - pojem vlákno neznámý, každé „vlákno“ je procesem s vlastním adresovým prostorem a s vlastními prostředky
- 1:M
 - OS/2, Windows XP, Mach, ...
 - v rámci 1 procesu lze vytvořit více vláken
 - proces je vlastníkem zdrojů (vlákna sdílejí zdroje procesu)

Procesy vs. vlákna





Výhody využití vláken

- Výhody
 - vlákno se vytvoří rychleji než proces
 - vlákno se ukončí rychleji než proces
 - mezi vlákny se rychleji přepíná než mezi procesy
 - jednodušší programování (jednodušší struktura programu)
 - u multiprocesorových systémů může na různých procesorech běžet více vláken jednoho procesu současně
- Příklady
 - síťový souborový (nebo i jiný) server
 - musí vyřizovat řadu požadavků klientů
 - pro vyřízení každého požadavku vytváří samostatné vlákno (efektivnější než samostatný proces)
 - 1 vlákno zobrazuje menu a čte vstup od uživatele a současně 1 vlákno provádí příkazy uživatele
 - překreslování obrazovky souběžně se zpracováním dat

● ● ● | Problém konzistence

- Program se skládá z několika vláken které běží paralelně
- Výhody
 - když vlákno čeká na ukončení I/O operace, může běžet jiné vlákno téhož procesu, aniž by se přepínalo mezi procesy (což je časově náročné)
 - vlákna jednoho procesu sdílí paměť a deskriptory otevřených souborů a mohou mezi sebou komunikovat, aniž by k tomu potřebovaly služby jádra (což by bylo pomalejší)
- Konzistence
 - vlákna jedné aplikace se proto musí mezi sebou synchronizovat, aby se zachovala konzistentnost dat (musíme zabránit současné modifikace stejných dat dvěma vlákny apod.)

Příklad (problém konzistence)

- Situace:
 - 3 proměnné: A, B, C
 - 2 vlákna: T1, T2
 - vlákno T1 počítá $C = A+B$
 - vlákno T2 přesouvá hodnotu X z A do B (jakoby z účtu na účet)
- Představa o chování
 - T2 dělá $A = A-X$ a $B = B+X$
 - T1 počítá konstantní C, tj. $A + B$ se nezmění
- Ale jestliže
 - T1 spočítá $A+B$
 - po té co T2 udělá $A = A-X$
 - ale dříve než co T2 udělá $B = B+X$
 - pak T1 nezíská správný výsledek $C = A+B$



Stavy vláken

- Tři klíčové stavy vláken:
 - běží
 - připravený
 - čekající
- Vlákna se (samostatně) neodkládají
 - všechny vlákna jednoho procesu sdílejí stejný adresový prostor
- Ukončení procesu ukončuje všechny vlákna existující v rámci tohoto procesu

Příklad: Win32

- Implementuje vlákna na úrovni jádra OS (implementace je zdařilá, umožňuje mimo jiné paralelní běh vláken jednoho procesu na různých procesorech)
- Služby OS
 - CreateThread
 - ExitThread
 - GetExitCodeThread
 - CreateRemoteThread (vytváří vlákno jiného procesu)
 - SuspendThread
 - ResumeThread
 - GetProcessAffinityMask (běh vlákna na procesorech)
 - SetProcessAffinityMask
 - SetThreadIdealProcessor
 - SwitchToThread (spust' jiný thread – je-li připraven)
 - TlsAlloc, TlsFree, TlsSetValue, TlsGetValue (thread local storage)