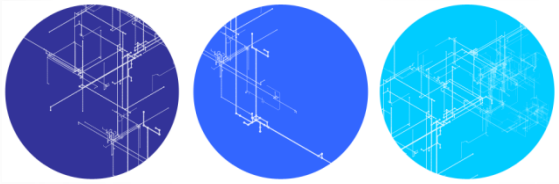
PB153
OPERAČNÍ SYSTÉMY A JEJICH ROZHRANÍ



Vlákna

06

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

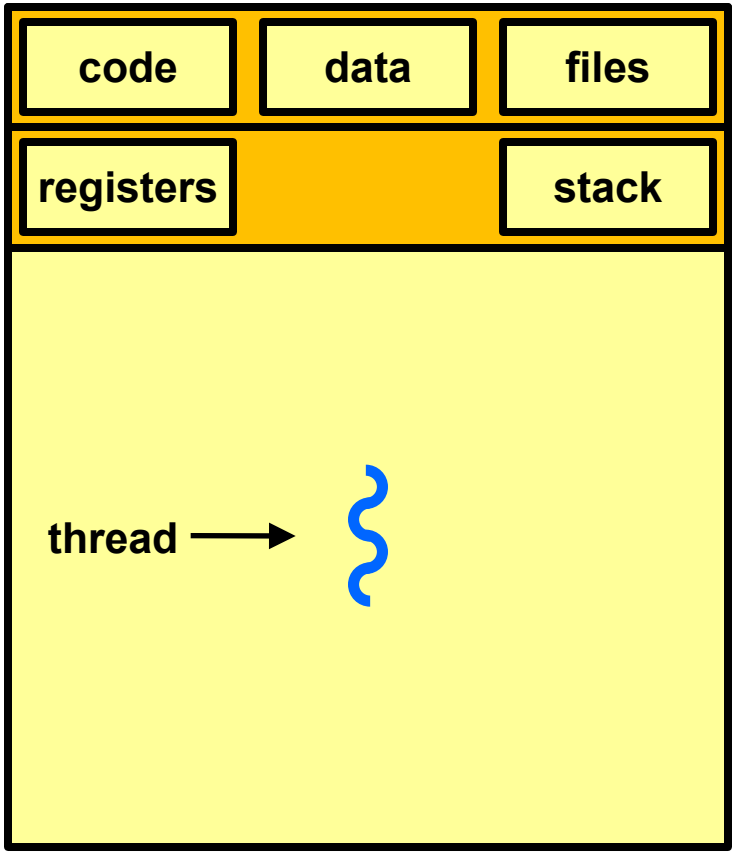
PROCESY A VLÁKNA

- 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 všechny 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)

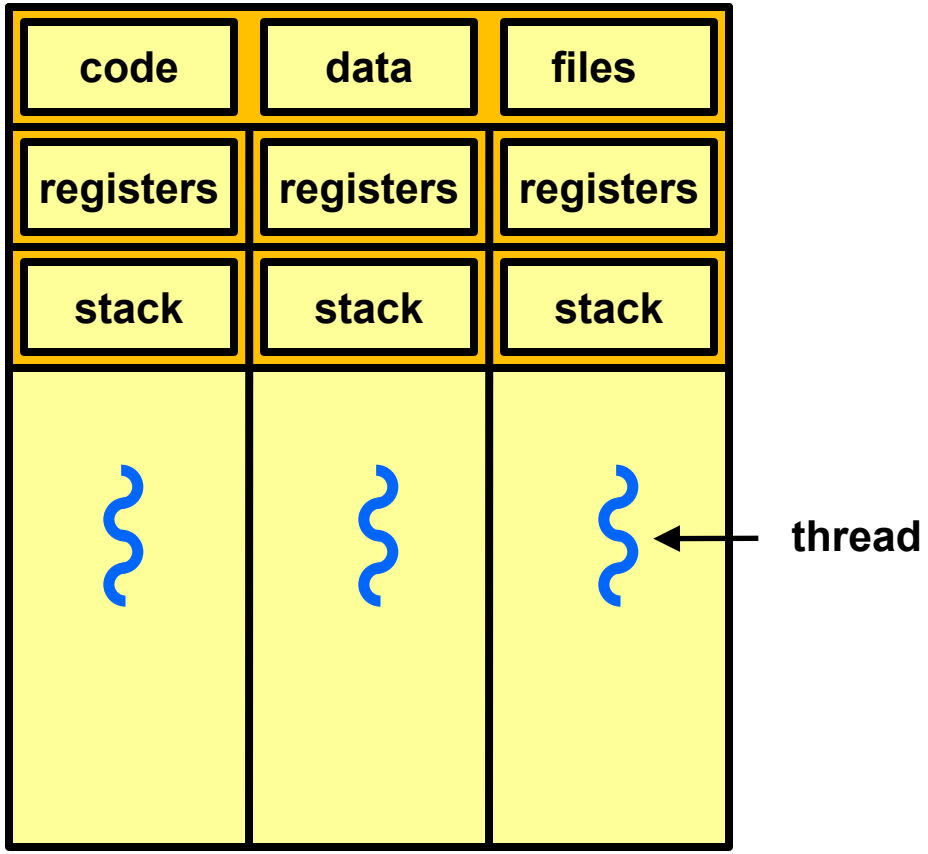
PROCESY A VLÁKNA

- Proč využít vlákna
  - využití multiprocesorových strojů (vlákna jednoho procesu mohou běžet na různých CPU)
  - jednodušší programovaní
  - 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



single-threaded process



multithreaded process

JEDNO/MULTIVLÁKNOVÝ OS

- Jednovláknový OS:
  - nepodporuje koncept vláken (nezná pojem vlákno)
  - MS-DOS: 1 proces, 1 vlákno
  - UNIX: n procesů, 1 vlákno / 1 proces
- Multivláknový OS:
  - podporuje koncept více vláken v rámci procesů
  - Windows XP, Solaris, …

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, která 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 modifikaci stejných dat dvěmi 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

VLÁKNA NA UŽIVATELSKÉ ÚROVNI

- User-Level Threads (ULT)
  - Správa vláken se provádí prostřednictvím vláknové knihovny („thread library“) na úrovni uživatelského / aplikačního programu
  - Jádro o jejich existenci neví
    - přepojování mezi vlákny nepožaduje provádění funkcí jádra
    - nepřepíná se ani kontext procesu ani režim procesoru
  - Plánování přepínání vláken je specifické pro konkrétní aplikaci
    - aplikace si volí pro sebe nejvhodnější (např. plánovací) algoritmus

VLÁKNA NA UŽIVATELSKÉ ÚROVNI

- „Threads library“ obsahuje funkce pro
  - vytváření a rušení vláken
  - předávání zpráv a dat mezi vlákny
  - plánování běhů vláken
  - uchovávání a obnova kontextů vláken
- Co dělá jádro pro vlákna na uživatelské úrovni
  - jádro neví o aktivitě vláken, proto manipuluje s celými procesy
  - když některé vlákno zavolá službu jádra, je blokován celý proces dokud se služba nesplní
  - pro „thread library“ je takové vlákno ale stále ve stavu „běží“
  - stavy vláken jsou na stavech procesu nezávislé

VLÁKNA NA UŽIVATELSKÉ ÚROVNI

- **Výhody**
  - přepojování mezi vlákny nepožaduje provádění jádra (tj.vyšší rychlost)
    - nepřepíná se ani kontext ani režim procesoru
  - plánování je specifické pro konkrétní aplikaci
    - aplikace volí si pro sebe nejvhodnější algoritmus
  - ULT mohou běžet pod kterýmkoliv OS
    - není vyžadována podpora na úrovní jádra OS
  - ULT potřebují uživatelskou knihovnu (ke slinkovaní s aplikací)
- **Nevýhody**
  - většina volání služeb OS způsobí blokování celého procesu (tj. všech vláken procesu)
  - jádro může přidělovat procesor pouze procesům, dvě vlákna stejného procesu nemohou běžet na dvou procesorech

VLÁKNA NA ÚROVNI JÁDRA

- Kernel-Level Threads (KLT)
- Správu vláken podporuje jádro, nepoužívá se „thread library“
  - používá se API pro vláknové služby jádra
  - informaci o kontextu procesů a vláken udržuje jádro
  - přepojování mezi vlákny aktivuje jádro
  - plánování na bázi vláken již v jádře OS
- Příklady
  - OS/2
  - Windows 95/98/NT/2000/XP
  - Solaris
  - Tru64 UNIX
  - BeOS
  - Linux

VLÁKNA NA ÚROVNI JÁDRA

- Výhody
  - jádro může současně plánovat běh více váken stejného procesu na více procesorech
  - k blokování dochází na úrovni vlákna (není blokován celý proces)
  - I programy jádra mohou mít multivláknový charakter
- Nevýhody
  - přepojování mezi vlákny stejného procesu zprostředkovává jádro (tj. pomaleji)
  - při přepnutí vlákna se 2x přepíná režim procesoru (tj. režie navíc)

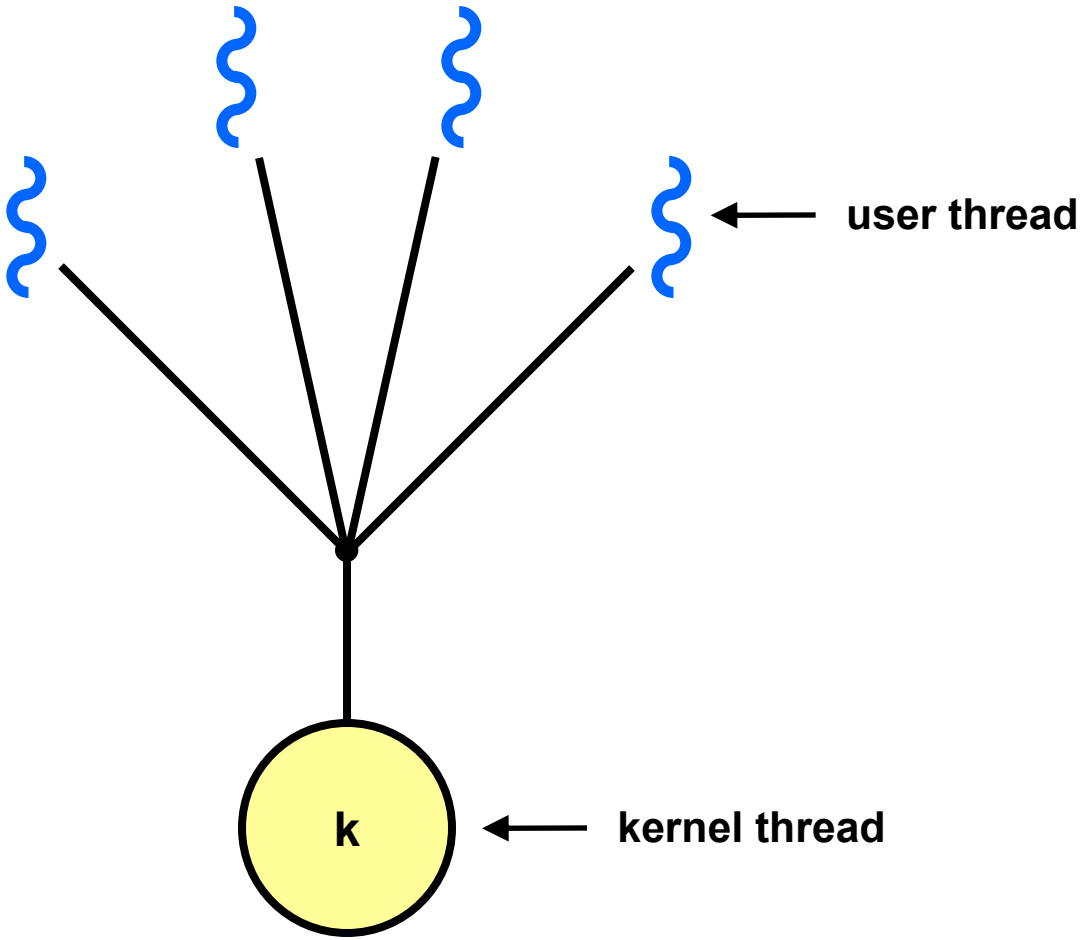
KOMBINACE VLÁKEN ULT/KLT

- Vlákna se vytvářejí v uživatelském prostoru
- Většina plánování a synchronizace se dělá v uživatelském prostoru
- Programátor může nastavit počet vláken na úrovni jádra
- Lze kombinovat přínosy oboru přístupů
- Např. OS Solaris <=8

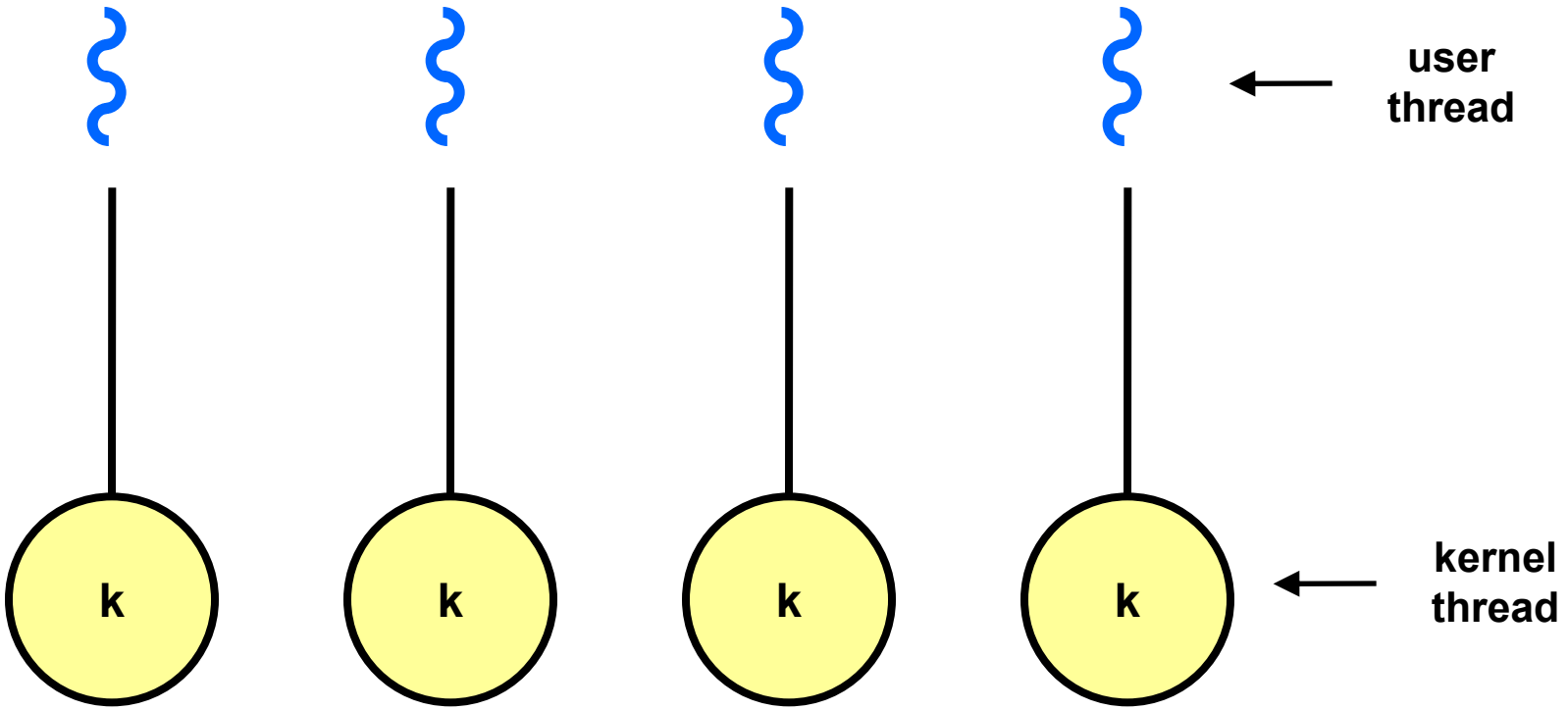
MULTIVLÁKNOVÉ MODELY

- n : 1
  - více ULT se zobrazuje do 1 KLT
  - používá se na systémech, které nepodporují KLT
- 1 : 1
  - každý ULT se zobrazuje do 1 KLT
  - Windows 95/98/NT/2000/XP, OS/2
- n : m
  - více ULT se může zobrazovat do více KLT
  - OS může vytvořit dostatečný počet KLT
  - Solaris 2, Windows NT/2000 s *ThreadFiber* package

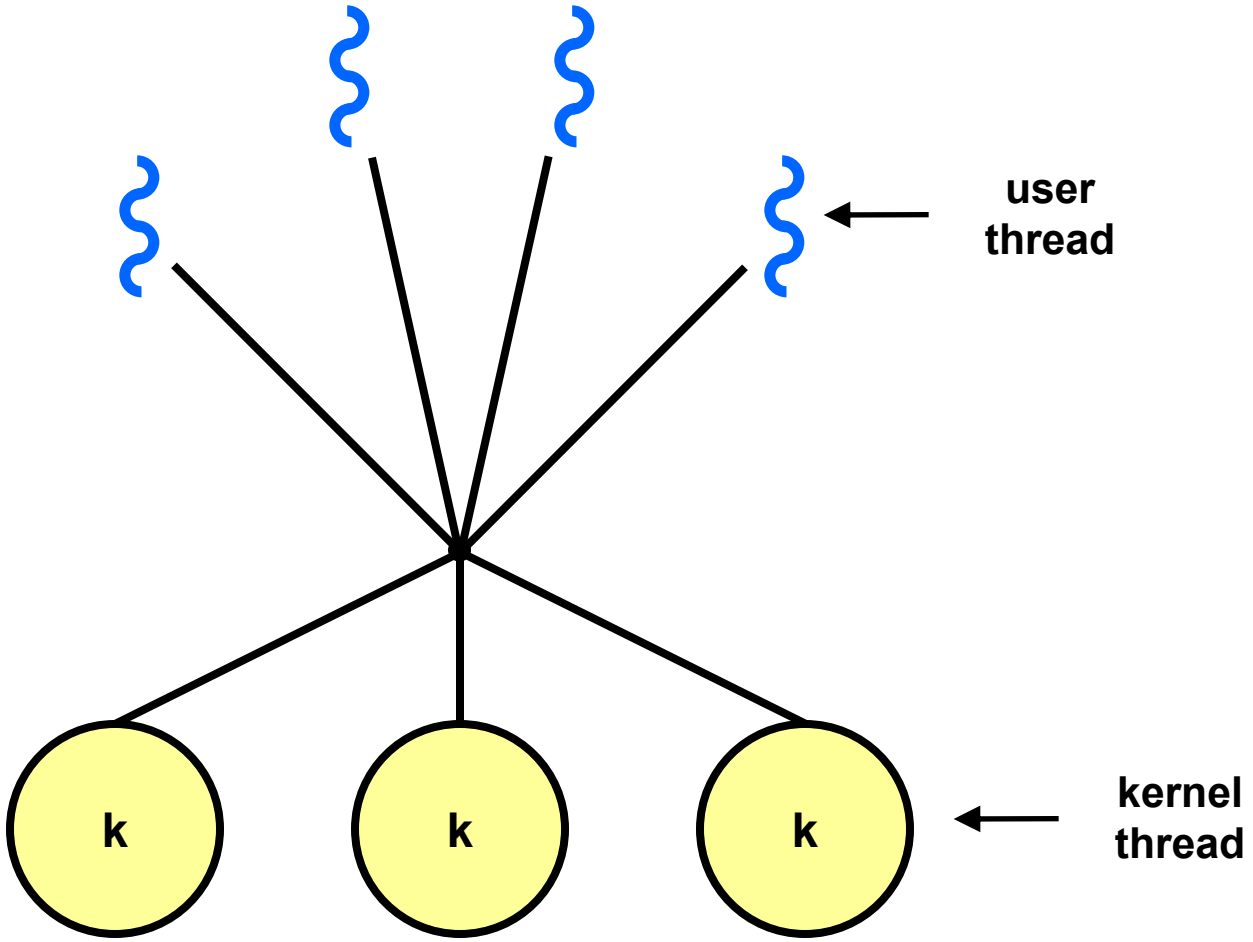
MODEL n:1



MODEL 1:1



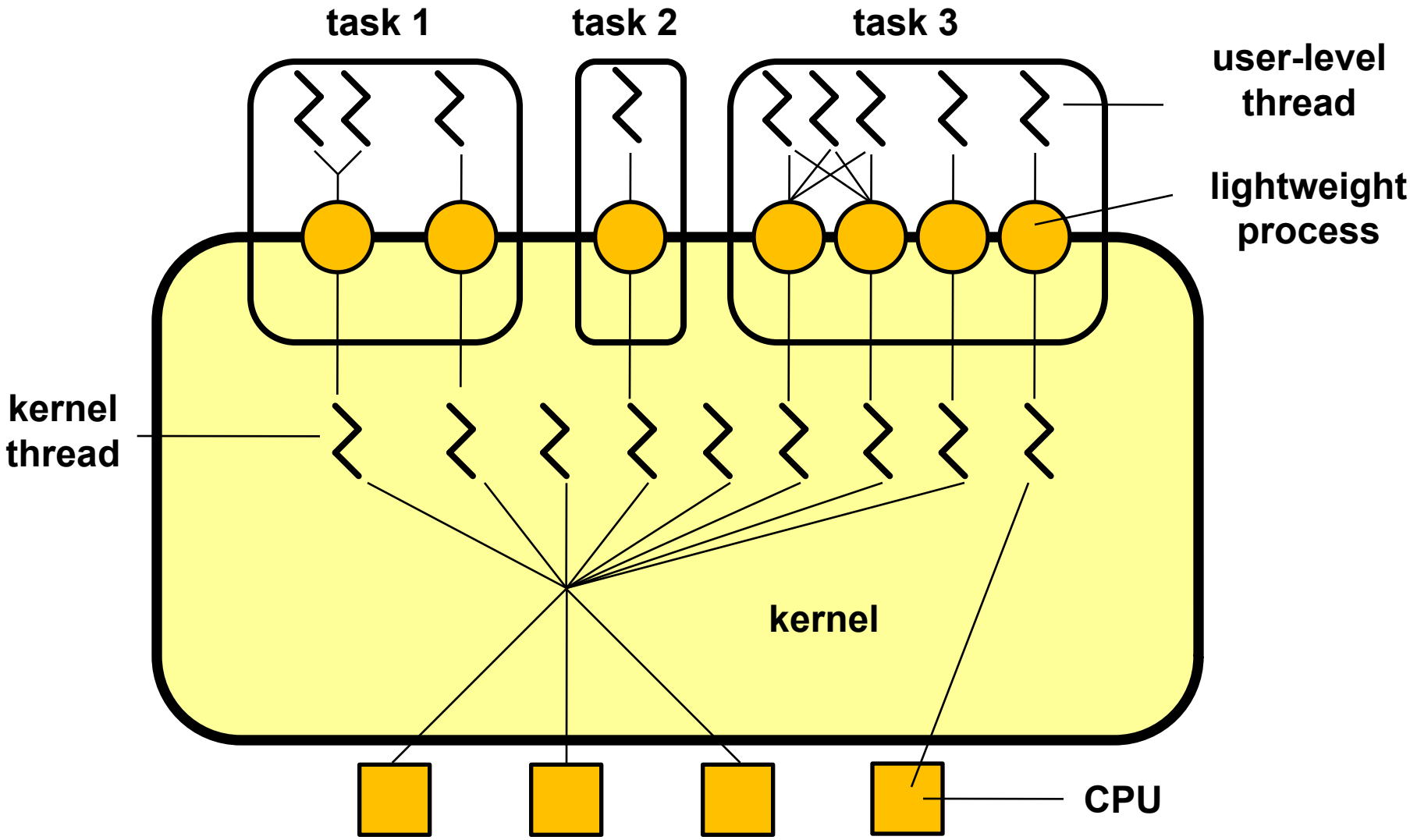
MODEL m:n



PŘÍKLAD: SOLARIS 2

- Proces
  - uživatelský adresový prostor
  - zásobník
  - PCB (process control block)
- ULT
  - OS je nevidí
- KLT
  - jednotka pro přidělování času procesoru
- Lightweight processes (LWP)
  - LWP podporuje 1 nebo více ULT a zobrazuje je do 1 KLT
  - LWP – rozhraní pro paralelismus pro aplikace

PŘÍKLAD: SOLARIS 2 (2)



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 (spusť jiný thread – je-li připraven)
  - TlsAlloc, TlsFree, TlsSetValue, TlsGetValue (thread local storage)

PŘÍKLAD: WIN32 (2)

- „A Win32®-based application consists of one or more processes. A *process*, in the simplest terms, is an executing program. One or more threads run in the context of the process. A *thread* is the basic unit to which the operating system allocates processor time. A thread can execute any part of the process code, including parts currently being executed by another thread. A *fiber* is a unit of execution that must be manually scheduled by the application. Fibers run in the context of the threads that schedule them“
- Služby OS
  - ConvertThreadToFiber
  - CreateFiber
  - DeleteFiber
  - GetFiberData
  - SwitchToFiber

PŘÍKLAD: LINUX – UNIX - POSIX

- Knihovna „pthreads“
- Služby knihovny
  - pthread_create
  - pthread_exit
  - pthread_join
  - pthread_detach
  - pthread_attr_init

PŘÍKLAD: LINUX

- Implementace POSIX threads
  - LinuxThreads
    - Odpovídá POSIX standardu IEEE 1003.1c až na ovladače signálů.
    - Vlákna mají různá PID (Process Identifier)
    - Není nadále vyvíjeno
  - NTPL (Native POSIX Threads Library for Linux)
    - Nahradilo LinuxThreads
    - Lepší výkon
    - Vyžaduje jádro řady 2.6
    - Dnes součást knihovny GNU C
    - Model 1:1
  - NGPT (Next Generation POSIX Threads)
    - Alternativa k NTPL, které se neprosadila

PŘÍKLAD: LINUX (2)

- Služby jádra OS

```
#include <sched.h>

int clone(int (*fn)(void *), void *child_stack,int flags,
    void *arg);

_syscall2(int, clone, int, flags, void *, child_stack);
```

- služba jádra sys_clone a knihovní funkce clone
  - vytvoří vlákno, které sdílí (v rámci procesu) adresový prostor, tabulku deskriptorů souborů, tabulku ovladačů signálů, trasovací informace, process ID

Výukovou pomůcku zpracovalo

Servisní středisko pro e-learning na MU

CZ.1.07/2.2.00/28.0041
Centrum interaktivních a multimediálních studijních opor pro inovaci výuky a efektivní učení

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ