Vlákna

PB 152 ◇ Operační systémy

Jan Staudek
http://www.fi.muni.cz/usr/staudek/vyuka/

Verze: jaro 2017

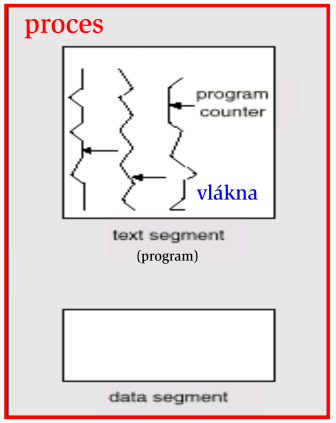
Proces a vlákna

Proces, resp. také task

– držitel zdrojů, vč. prostoru ve virtuální paměti pro uchování obrazu procesu

– jednotka plánování

cíl dynamického přidělování procesoru pro běh procesu



Vlákno

Sekvenční děj definovatelný v procesu.

Pro OS jednotka plánování,

nikoli vlastnictví zdrojů

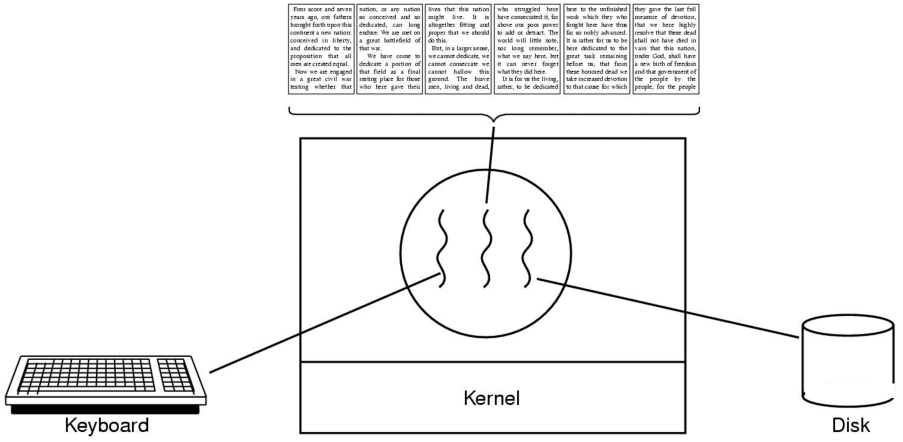
Všechna vlákna definovaná

v procesu se řeší souběžně, multitasking.

Koncept sekvenčního procesu může být neefektivní

- Textový editor
  - ✓ čte z klávesnice příkazy k editaci
  - ✓ fomátuje text podle příkazů
  - ✓ z důvodu spolehlivosti se požaduje periodicky kopírovat editovaný text do diskového souboru
- Řešení editoru čistě sekvenčním procesem
  - ✓ během archivace bude ignorovat klávesnici a myš dokud se archivace neukončí,
  - ✓ výkon editování bude nízký
- Naprogramování do editoru obsluhy přerušení od myši a klávesnice během archivace
  - ✓ výkon se vylepší, program bude složitý

Řešení editoru pomocí vláken



✓ Řešení problému třemi procesy problém neřeší, všechny tři procesy by musely pracovat s jediným dokumentem

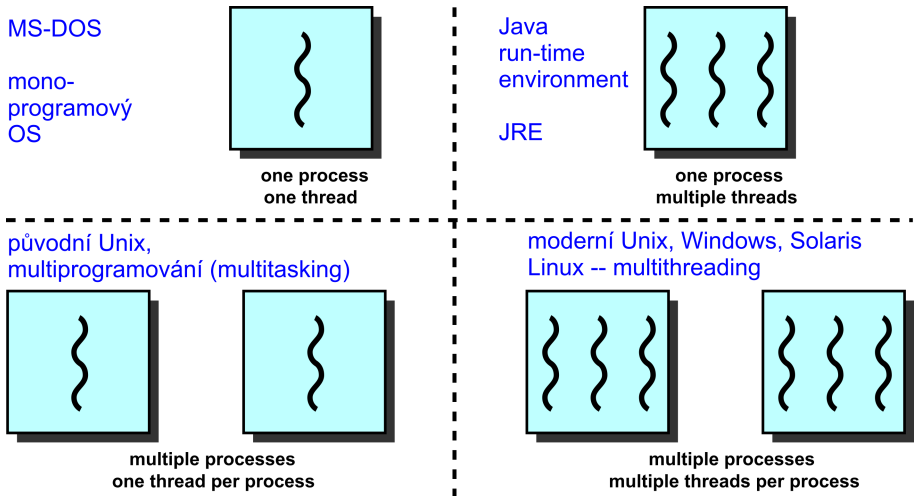
Proces a vlákno

- vlákno (*thread*)
  - ✓ systémový objekt, který se vytváří v rámci procesu
  - ✓ je viditelný pouze uvnitř procesu
  - ✓ je charakterizován svým stavem (procesory se přidělují vláknům)
  - ✓ „klasický process" – proces s jedním vláknem
  - ✓ vlákno se nachází ve stavech: běží, připravené, ...
  - ✓ když vlákno neběží, kontext vlákna je uložený v TCB (*Thread Control Block*), v deskriptoru vlákna
  - ✓ vlákno může přistupovat k LAP a k ostatním zdrojům svého procesu
  - ✓ tyto zdroje sdílejí všechny vlákna jednoho procesu
    - jakmile 1 vlákno změní obsah některé buňky LAP, všechny ostatní vlákna (téhož procesu) nový obsah vidí
    - soubor otevřený jedním vláknem vidí všechny ostatní vlákna (téhož procesu)
  - ✓ skupina vláken jednoho procesu sdílí proměnné (LAP, přidělený FAP), otevřené soubory, ...

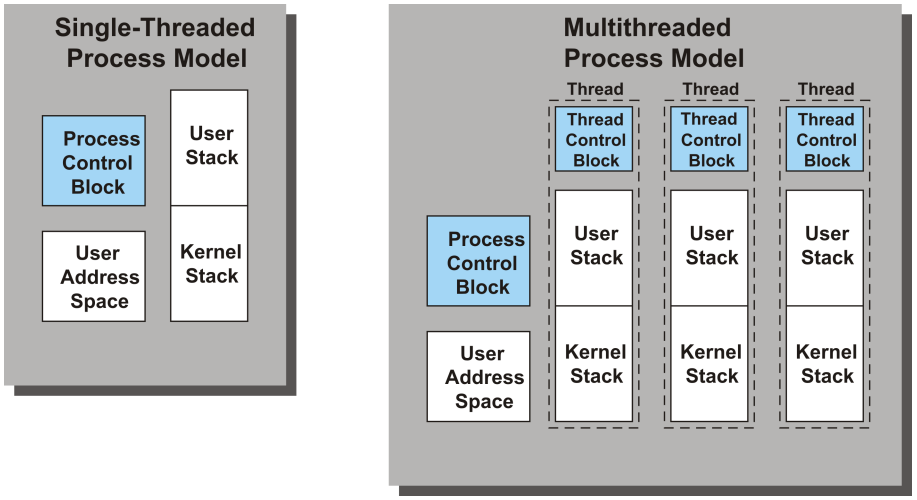
Generický obsah TCB, deskriptoru vlákna

- id vlákna
- odkaz na jeho rodičovský proces umožňující vláknu přístup ke zdrojům přiděleným procesu (na vlastníka vlákna)
- priorita vlákna
- stav vlákna
- stav čítače instrukcí a dalších registrů procesoru po přerušení
- odkaz na zásobník

Multitasking / multithreading

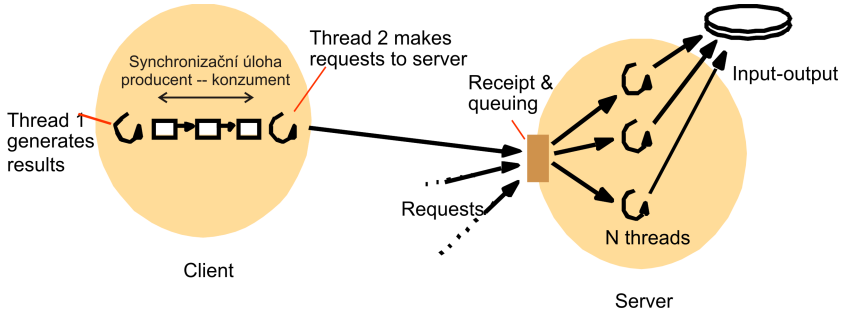


Vlákna vs. Procesy

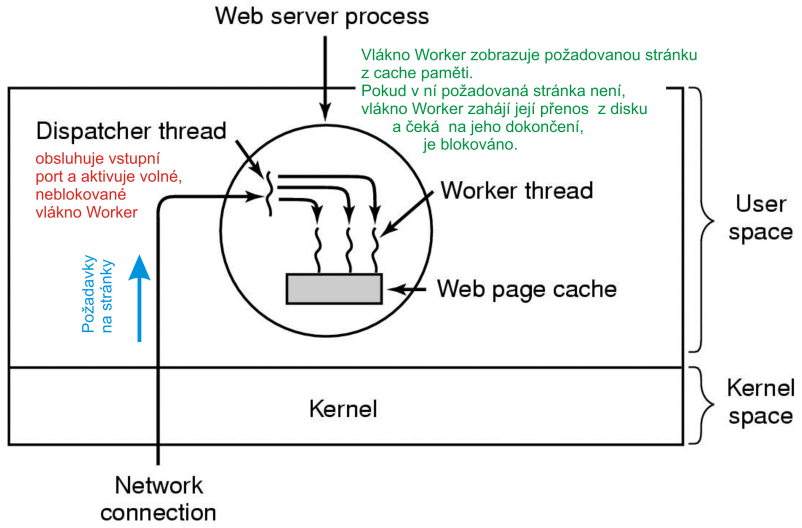


Klient-server řešený pomocí vláken

- Proces může definovat více aktivit proveditelných souběžně
  - ✓ Např. server může obsluhovat více požadavků klientů souběžně



Web server řešený pomocí vláken



Použití vláken

- Univerzální nástroj pro všechny aplikace
  - ✓ od interaktivního kreslení po hry
  - ✓ např. během čtení klávesnice jedním vláknem jiné vlákno vykresluje obrázek
- Efektivní využití multiprocesorových počítačů
  - ✓ možnost skutečně paralelního běhu vláken na různých procesorech
  - ✓ místo multitaskingu sdílejícího jediný procesor
- Přednosti
  - ✓ Vlákno se vytvoří rychleji než proces
  - ✓ Vlákno se ukončí rychleji než proces
  - ✓ Mezi vlákny se rychleji přepíná než mezi procesy
  - ✓ Dosáhne se lepší strukturalizace programu

Přínosy použití vláken, příklady aplikací

- proč se zavádí vlákna v rámci procesu, příklady přínosů
  - ✓ menu vypisované souběžně se zpracováním
  - ✓ periodicky provádění automatické kopie souboru
  - ✓ překreslování obrazovky souběžně se zpracováním dat
  - ✓ paralelizace algoritmu v multiprocesoru
  - ✓ dosažení lepší strukturalizace programu
- Příklady
  - ✓ Souborový server LAN
    – musí vyřizovat během krátké doby několik požadavků na soubory
    – pro vyřízení každého požadavku se zřídí samostatné vlákno
  - ✓ Symetrický multiprocesor
    – na různých procesorech může běžet více vláken současně
  - ✓ jedno vlákno zobrazuje menu a čte vstup od uživatele a současně jiné vlákno provádí příkazy uživatele
  - ✓ periodické provádění automatické kopie souboru
  - ✓ překreslování obrazovky souběžně se zpracováním

Co se získá použitím vláken

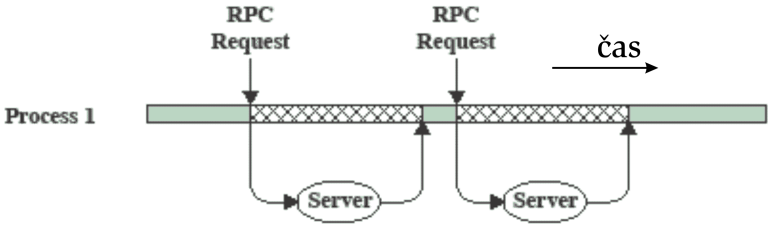
- schopnost lépe reagovat na řešený problém
  - ✓ multivláknová interaktivní aplikace může stále běžet, i když její část je blokovaná / se dlouho řeší / . . .
  - ✓ interakce s prohlížečem v jednom vláknu prohlížeče, zavádění obrázku jiným vláknem
- snažší sdílení zdrojů
  - ✓ aplikace, které se liší pouze několika odlišnými aktivitami řešenými samostatně vlákny, mohou sdílet stejný adresový prostor, hlavní program, . . .

Co se získá použitím vláken

- snížení systémové režie
  - ✓ přepínání kontextu mezi vlákny je jednodušší než přepínání kontextu mezi procesy
  - ✓ např. Solaris
    - vytvoření procesu je 30x pomalejší než vytvoření vlákna
    - přepínání kontextu mezi procesy je 5x pomalejší než mezi vlákny
- účinnější využití multiprocesorových architektur
  - ✓ souběžnost řešení vláken může nabýt formy paralelismu

Příklad použití vláken: RPC, *Remote Procedure Call*

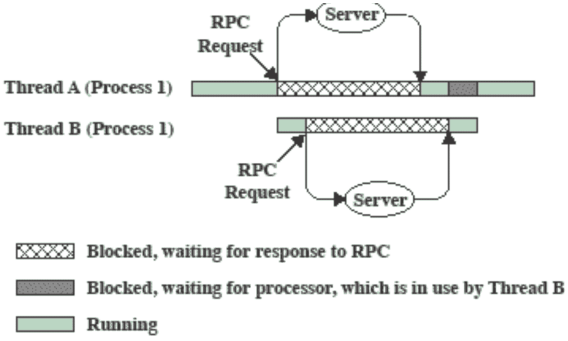
- Program provádí dvě RPC ke 2 různým serverům a počítá výsledek z obou hodnot
- Implementace pomocí 1-vláknového procesu



Výsledky se získávají sekvenčně, proces čeká na odpověďi obou serverů

Příklad použití vláken: RPC, *Remote Procedure Call*

- Program provádí dvě RPC ke 2 různým serverům a počítá výsledek z obou hodnot
- Implementace pomocí více-vláknového procesu na monoprocesoru – 1 vlákno na 1 server



Problém konzistence mezi vlákny

- Mějme aplikaci která sestává z více nezávislých částí
- Části nemusí běžet v sekvenci
- Každá část se implementuje jako vlákno
- Vlastnosti takové implementace
  - ✓ Když vlákno čeká konec I/O operace, může běžet jiné vlákno téhož procesu, aniž by se přepojovalo mezi procesy
  - ✓ vlákna jednoho procesu sdílí paměť a soubory a tudíž mohou mezi sebou komunikovat, aniž by k tomu potřebovaly služby jádra
  - ✓ vlákna jedné aplikace se proto musí mezi sebou synchronizovat, aby se zachovala konzistentnost zpracovávaných dat

Příklad problému zachování konzistence dat

- Scénář
  - ✓ v procesu jsou vytvořena vlákna $T1$ a $T2$
  - ✓ $T1$ počítá $C = A + B$,
    $T2$ přenáší hodnotu X z A do B: $A = A - X; B = B + X$
  - ✓ $T1$ a $T2$ běží souběžně, nevíme jak rychle každé z nich
- Formální představa o chování systému vláken $T1$ a $T2$
  - ✓ $T2$ udělá $A = A - X$ a $B = B + X$
  - ✓ $T1$ počítá $C = A + B$, hodnota C se tudíž provedení $T2$ nezmění
- Možná realita
  - ✓ $T2$ udělá $A = A - X$
  - ✓ $T1$ spočítá $C = A + B = A - X + B$
  - ✓ $T2$ udělá $B = B + X$, což C už neovlivní

Stavy vláken

- tři klíčové stavy
  - ✓ běží
  - ✓ připravené
  - ✓ čekající
- všechny vlákna jednoho procesu sdílejí stejný adresový prostor
  - ✓ vlákna se samostatně neodkládají
- ukončení procesu ukončuje všechny vlákna existující v rámci tohoto procesu

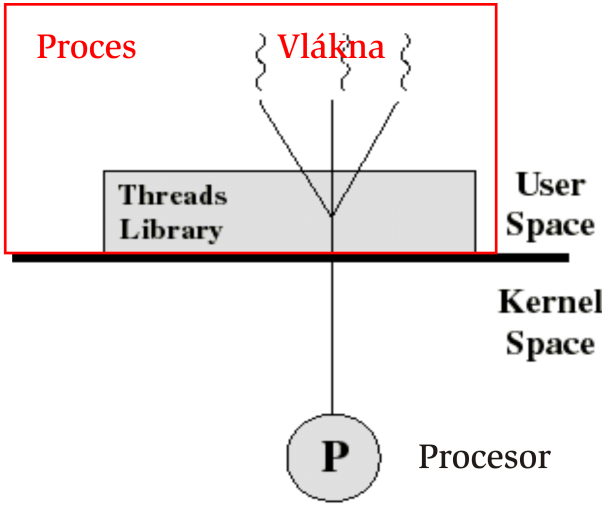
Typy vláken

- *User-Level Threads* (ULT)
  - ✓ jádro OS podporuje procesy
  - ✓ vlákna podporuje vláknová knihovna sestavovaná do programu uživatele
- *Kernel-Level Threads* (KLT)
  - ✓ podporuje je přímo jádro OS
  - ✓ také *lightweight processes* nebo *kernel-supported threads*

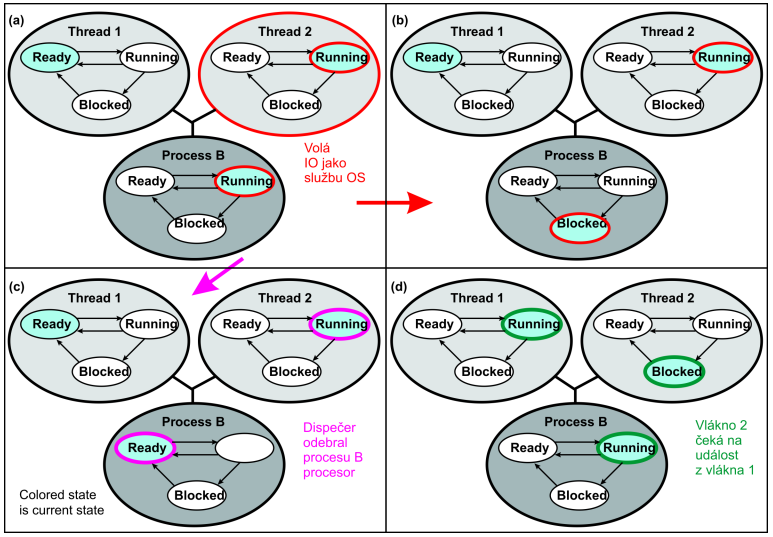
Vlákna na uživatelské úrovni, ULT

- *User-Level Threads* (ULT)
- vlastnosti
  - ✓ Správa vláken se provádí prostřednictvím vláknové knihovny (*thread library*) na úrovni uživatelského / aplikačního procesu,
  - ✓ 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 algoritmus
- příklady
  - ✓ POSIX – Pthreads
  - ✓ Mach – C-threads
  - ✓ Solaris – threads

Vlákna na uživatelské úrovni, ULT



Vztahy mezi ULT vlákny a stavy procesu



Vlákna na uživatelské úrovni, ULT

- vláknová knihovna 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, manipuluje pouze s procesy
  - ✓ Když některé vlákno zavolá službu jádra a čeká dokud se služba nesplní je blokován celý proces, viz předchozí bod
  - ✓ Stavy vláken jsou na stavech procesu nezávislé pro vláknovou knihovnu může vlákno být stále ve stavu ,,běžící'' i když je proces vrácený mezi ,,připravené''procesy

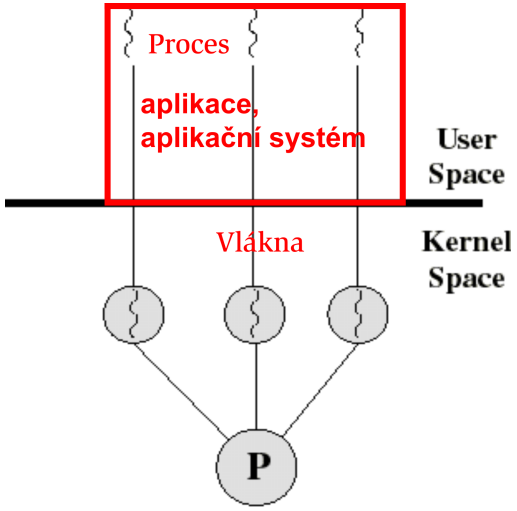
Vlákna na uživatelské úrovni, ULT

- Přednosti
  - ✓ Přepojování mezi vlákny nepožaduje provádění jádra
  - ✓ nepřepíná se ani kontext ani režim procesoru
  - ✓ Plánování je specifické pro konkrétní aplikaci
    – volí si pro sebe nejvhodnější algoritmus
    – ULT mohou běžet pod kterýmkoliv OS
    – ULT potřebují pouze adekvátní vláknovou knihovnu
- Nedostatky
  - ✓ Většina volání služeb OS způsobuje blokování procesu
  - ✓ Jádro blokuje procesy a ne vlákna
  - ✓ Žádné vlákno existující v rámci blokovaného procesu nemůže běžet
  - ✓ Jádro může přidělovat procesor pouze procesům,
    takže dvě vlákna jednoho procesu nemohou běžet paralelně,
    i když proces běží v multiprocesoru

Vlákna na úrovni jádra, KLT

- *Kernel-Level Threads* (KLT)
- vlastnosti
  - ✓ Celou správu vláken podporuje jádro,
  - ✓ vláknová knihovna se nepoužívá
  - ✓ používá se API na vláknové služby jádra
  - ✓ Informaci o kontextu procesu a vláken udržuje jádro
  - ✓ Přepojování mezi vlákny aktivuje jádro
  - ✓ Plánování se řeší na bázi vláken
- příklady
  - ✓ Windows
  - ✓ Solaris, Tru64 UNIX, Linux
  - ✓ Mac OS X

Vlákna na úrovni jádra, KLT



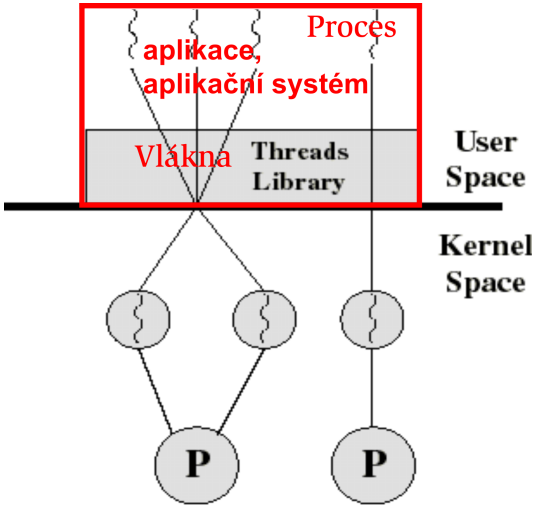
Vlákna na úrovni jádra, KLT

- Přednosti
  - ✓ jádro může současně plánovat běh více vláken stejného procesu
    na více procesorech
  - ✓ k blokování dochází na úrovni vláken
  - ✓ i programy jádra mohou mít multi–vláknový charakter
- Nedostatky
  - ✓ přepojování mezi vlákny téhož procesu zprostředkovává jádro
  - ✓ při přepnutí vlákna se 2x se přepíná režim procesoru
  - ✓ výsledkem je zpomalení, snížení dostupného aplikačního výkonu

Kombinace ULT/KLT

- vlákna se vytvářejí v uživatelském prostoru
- programátor může nastavit počet vláken na úrovni jádra
- lze kombinovat přínosy obou přístupů
- podporuje Solaris

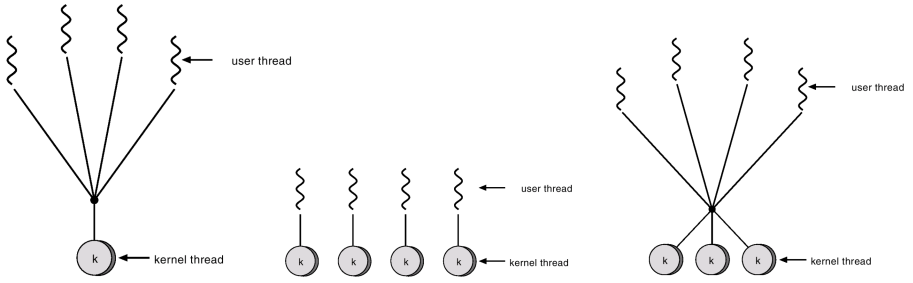
Kombinace ULT/KLT



Multivláknové modely

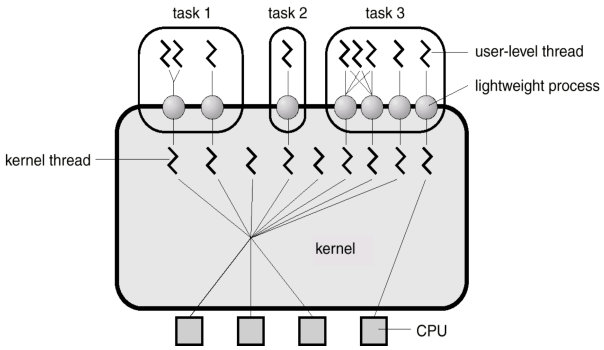
- 1 : 1
  - ✓ Každý ULT se zobrazuje do jednoho KLT
  - ✓ tradiční Unixy verze
- n : 1
  - ✓ Více ULT se zobrazuje do jednoho KLT
  - ✓ používá se na systémech, které nepodporují KLT
  - ✓ Windows 95/98/NT/2000/XP, Linux, Solaris 9 a vyšší verze. Solaris Green Threads, GNU Portable Threads
- n : m
  - ✓ více ULT se může zobrazovat do více KLT
  - ✓ OS může vytvořit dostatečný počet KLT
  - ✓ Solaris do verze 9, Windows NT/2000 with the ThreadFiber package
  - ✓ two-level model – model n : m s možností vázat konkrétní ULT na konkrétní KLT (IRIX, HP=UX, True64 Unix)

Multivláknové modely



N : 1 1 : 1 n : m

Vlákna v systému Solaris

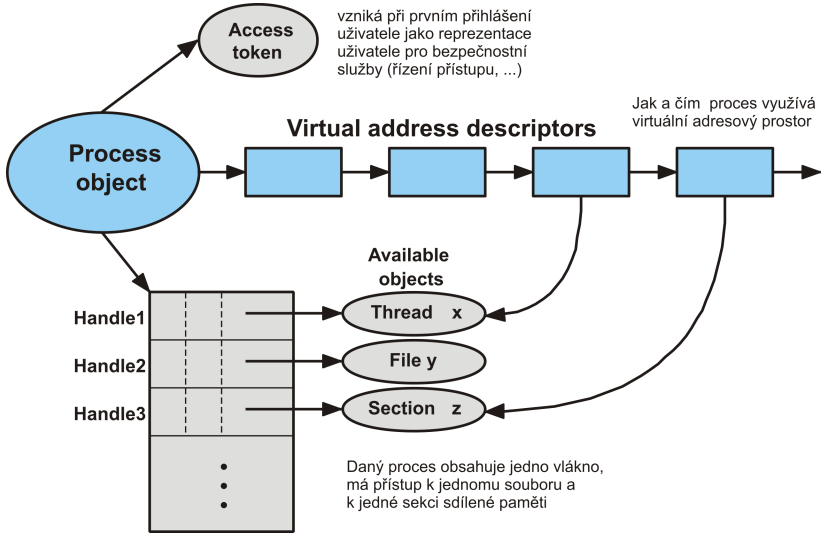


- OS nevidí ULT
- OS přiděluje procesor KLT
- *Lightweight process* (LWP)
  - ✓ rozhraní pro paralelismus pro aplikace
  - ✓ každý LWP podporuje 1 nebo více ULT a zobrazuje je do jednoho KLT

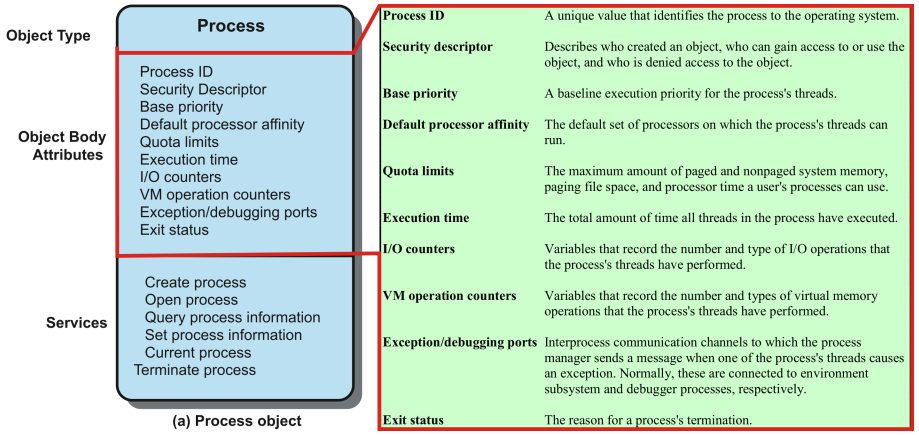
Proces ve Windows a jeho zdroje

- Jsou implementované jako objekty
- Proces lze vytvořit jako nový proces nebo
  jako kopii existujícího procesu
- V procesu lze definovat jedno nebo více vláken
- Procesy i vlákna mají vestavěné synchronizační vlastnosti
- Proces
  - ✓ je definovaný jistým počtem akcí /služeb, které může vykonávat
  - ✓ provádí službu,
    když je volaný některou z publikovaných metod rozhraní
  - ✓ vzniká jako nový proces generováním instance objektu ze šablony
  - ✓ vlastnostem objektu se přiřadí hodnoty při generování objektu

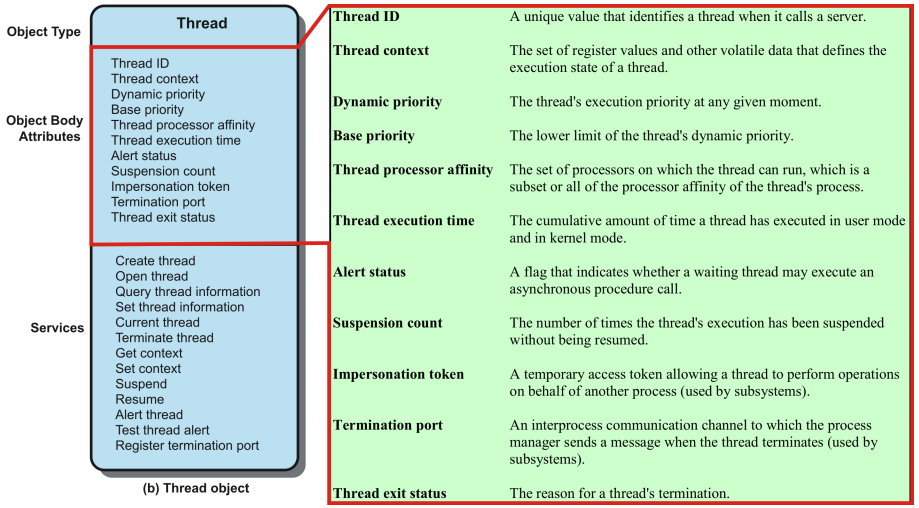
Vztah procesů a zdrojů ve Windows



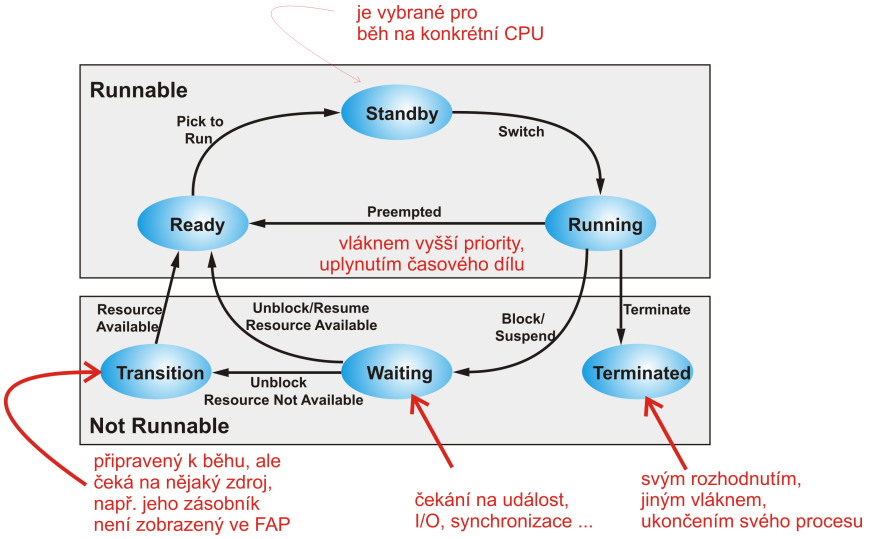
Vlastnosti objektu proces ve Windows



Vlastnosti objektu vlákno ve Windows



Stavy vlákna ve Windows

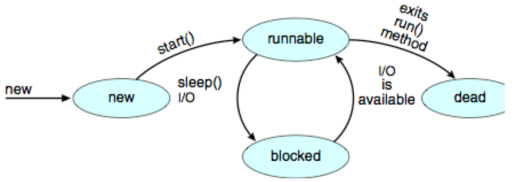


Windows 7 Architecture, vlákna a SMP

- vlákna všech procesů vč. systémových (Executive) mohou běžet na kterémkoliv dostupném procesoru
- vlákna mohou běžet paralelně
- vlákna jednoho procesu mohou běžet paralelně na různých procesorech v rámci jednoho procesu
- server může použít více vláken pro současnou obsluhu více klientů současně
- data a zdroje mohou procesy sdílet
- procesy mohou mezi sebou komunikovat

Vlákna Java

- vlákna jsou fundamentálním model řešení javovského programu



✓ jak implementovat JVM do hostitelského OS žádný standard neřeší – např. XP: model 1:1

✓ vláknová knihovna Java bývá mapovaná na vláknovou knihovnu hostitelského OS

– JVM ve XP: na API Win32
– JVM v Linux/Solaris: na API Pthreads

Vlákna Java, příklad

```
class MutableInteger
{
  private int value;
  public int getValue() {
   return value;
  }
  public void setValue(int value) {
   this.value = value;
  }
}

class Summation implements Runnable
{
  private int upper;
  private MutableInteger sumValue;
  public Summation(int upper, MutableInteger sumValue) {
   this.upper = upper;
   this.sumValue = sumValue;
  }
  public void run() {
   int sum = 0;
   for (int i = 0; i <= upper; i++)
      sum += i;
   sumValue.setValue(sum);
  }
}
```


Vlákna Java, příklad

```
public class Driver
{
  public static void main(String[] args) {
   if (args.length > 0) {
    if (Integer.parseInt(args[0]) < 0)
     System.err.println(args[0] + " must be >= 0.");
    else {
     // create the object to be shared
     MutableInteger sum = new MutableInteger();
     int upper = Integer.parseInt(args[0]);
     Thread thrd = new Thread(new Summation(upper, sum));
     thrd.start();
     try {
       thrd.join();
       System.out.println
             ("The sum of "+upper+" is "+sum.getValue());
     } catch (InterruptedException ie) { }
    }
   }
   else
    System.err.println("Usage: Summation <integer value>");
  }
}
```


Poznámky k práci s vlákny

- sémantika fork() a exec()
  - ✓ fork duplikuje proces v jednovláknovém procesu
  - ✓ má duplikovat celý proces nebo jenom vlákno, které fork vyvolalo ?
- jak ukončit běh (cílového) vlákna, který dosud neskončil ?
  - ✓ asynchronní zrušení – jedno vlákno bezprostředně ukončí cílové vlákno
  - ✓ odložené zrušení – cílové vlákno se periodicky dotazuje, zda má skončit

Oznámení přerušení cílového vlákna

```
Thread thrd = new Thread(new InterruptibleThread());
thrd.start();
. . .
thrd.interrupt();
```


Poznámky k práci s vlákny

- ✓ odložené zrušení – kontrolování stavu „přerušení"

```
class InterruptibleThread implements Runnable
{
   /**
    * This thread will continue to run as long
    * as it is not interrupted.
    */
   public void run() {
      while (true) {
         /**
          * do some work for awhile
          * . . .
          */

         if (Thread.currentThread().isInterrupted()) {
            System.out.println("I'm interrupted!");
            break;
         }
      }
      // clean up and terminate
   }
}
```

Poznámky k práci s vlákny

- zvládání signálů
  - ✓ signál – Unixovská softwarová notifikace procesu, že se stala (oznamovaná) událost
  - ✓ signály spravuje správce signálů
  - ✓ signál se procesu dopravuje operačním systémem (programově)
  - ✓ volby:
    - má se signál doručit 1 konkrétnímu vláknu v cílovém procesu?
    - má se signál doručit všem vláknům v cílovém procesu?
    - má se signál doručit jen jistým vláknům v cílovém procesu?
    - má jedno vlákno spravovat všechny signály doručené procesu ?

Poznámky k práci s vlákny

- práce s bankem vláken (*Thread Pool*)
  - ✓ bank vláken – skupina vláken připravených na spuštění
  - ✓ při žádosti o násobnou obsluhu serverem se nemusí generovat nové vlákno
  - ✓ neomezené generování nových vláken by mohlo vyčerpat dostupné zdroje
  - ✓ připravené vlákno se rychleji spustí než vytvářené vlákno
  - ✓ tři javovské architektury banku vláken:
    1. *Single thread executor* – bank o rozměru 1
    2. *Fixed thread executor* – bank o rozměru > 1
    3. *Cached thread pool* - bank neomezeného rozsahu
- práce s vlastními daty vlákna (*Thread Specific Data*)
  - ✓ vlákno může mít vlastní kopii dat, nesdílenou s ostatními vlákny procesu
  - ✓ např. při používání banku vláken

Příklady vláknových knihoven

- Pthreads
  - ✓ standard API, IEEE 1003.1c, pro vytváření a synchronicazi vláken
  - ✓ API určuje chování, implementaci standard nepředpisuje
  - ✓ používají unixově orientované systémy (Solaris, Linux, Mac OS X)
- Windows XP Threads
  - ✓ implementuje mapování 1:1
  - ✓ každé vlákno sestává z:

    id vlákna, sestava registrů, samostatný uživatelský zásobník a zásobník jádra a privátní paměťová oblast – kontext vlákna
- vlákna v Linuxu
  - ✓ Linux používá pojem *tasks* místo *threads* (vlákno)
  - ✓ vlákno se vytváří voláním systému *clone()*
  - ✓ vlákno je potomek a sdílí s rodičovským procesem adresový prostor