OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Vlákna

Ing. Pavel Smolka, Ph.D.

Program, proces a vlákno

- Program:
  - soubor přesně definovaného formátu obsahující
    - instrukce,
    - data a služební údaje potřebné k provedení stanoveného úkolu
- Proces:
  - systémový objekt – entita realizující výpočet podle programu charakterizovaná svým paměťovým prostorem a kontextem
  - prostor ve FAP se přiděluje procesům (nikoli programům)
  - patří mu obraz jeho adresního prostoru na vnější paměti
  - může vlastnit soubory, I/O zařízení
  - může vlastnit komunikační kanály k jiným procesům
  - přiděluje se mu čas procesoru
- Vlákno:
  - objekt vytvářený v rámci procesu

Vztah procesu a vlákna

- Vlákno (*thread*)
  - Objekt vytvářený v rámci procesu a viditelný uvnitř procesu
  - Tradiční proces je proces tvořený jediným vláknem
  - Vlákna podléhají plánování a přiděluje se jim strojový čas i procesory
  - Vlákno se nachází ve stavech: běží, připravené, čekající, …
    - Podobně jako při přidělování času procesům
  - Když vlákno neběží, je kontext vlákna uložený v TCB (*Thread Control Block*):
    - analogie PCB
    - prováděcí zásobník vlákna, obraz PC, obraz registrů, ...
  - Vlákno může přistupovat k LAP a k ostatním zdrojům svého procesu a ty jsou sdíleny všemi vlákny tohoto procesu
    - Změnu obsahu některé buňky LAP procesu vidí všechna ostatní vlákna téhož procesu
    - Soubor otevřený jedním vláknem je viditelný pro všechna ostatní vlákna téhož procesu
    - Vlákna patřící k jednomu procesu sdílí proměnné a systémové zdroje přidělené tomuto procesu

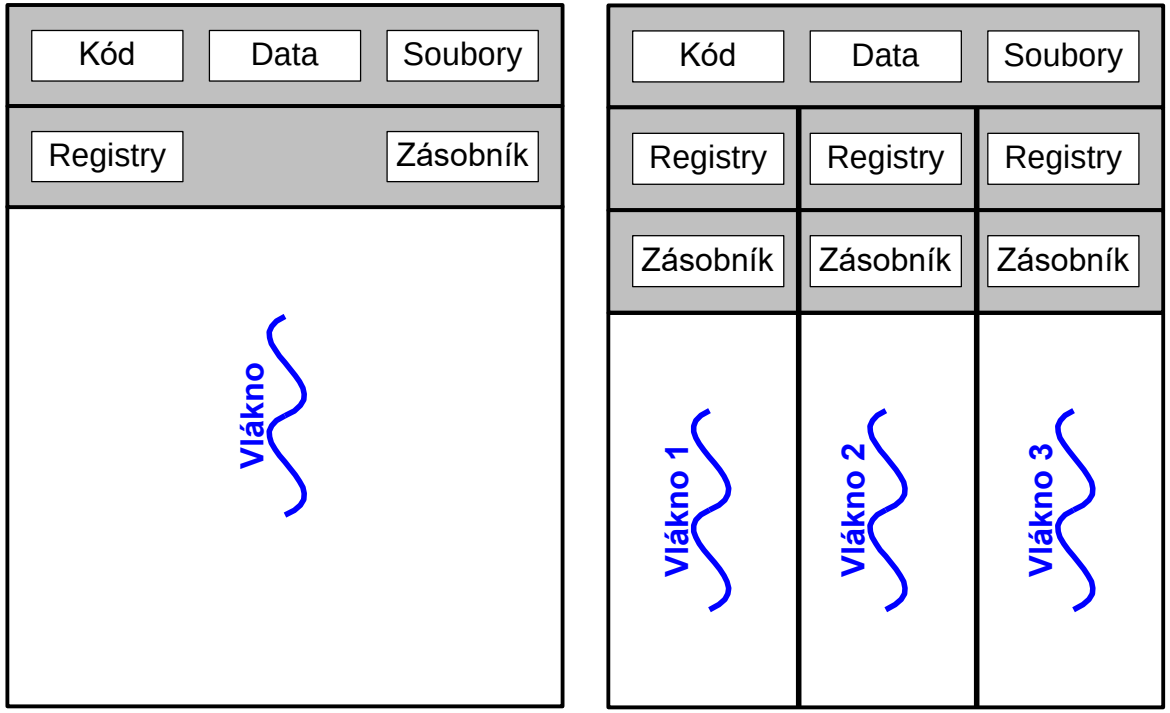
OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Proces a jeho vlákna

- Jednovláknové (tradiční) procesy
  - proces: jednotka plánování činnosti a jednotka vlastnící přidělené prostředky
  - každé vlákno je současně procesem s vlastním adresovým prostorem a s vlastními prostředky
  - tradiční UNIXy

- Procesy a vlákna (Windows, Solaris, …)
  - proces: jednotka vlastnící prostředky
  - vlákno: jednotka plánování činnosti
  - v rámci jednoho procesu lze vytvořit více vláken
  - proces definuje adresový prostor a dynamicky vlastní prostředky

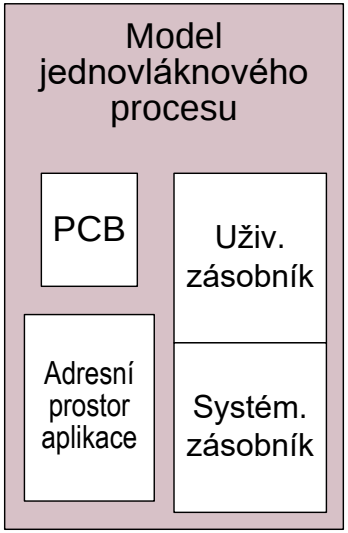
Procesy a vlákna

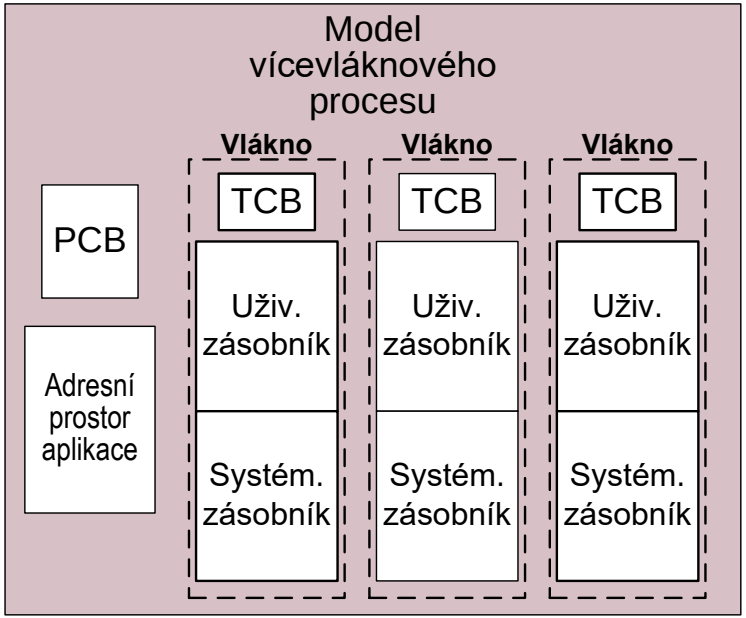


Jednovláknový proces

Vícevláknový proces

Procesy a vlákna – řídicí struktury





Procesy, vlákna a jejich komponenty

Co patří procesu a co vláknu?

kód programu:	proces
lokální a pracovní data:	vlákno
globální data:	proces
alokované systémové zdroje:	proces
zásobník:	vlákno
data pro správu paměti:	proces
čítač instrukcí:	vlákno
registry procesoru:	vlákno
plánovací stav:	vlákno
uživatelská práva a identifikace:	proces

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Význam vláken

- Přednosti
  - Vlákno se vytvoří i ukončí rychleji než proces
  - Přepínání mezi vlákny je rychlejší než mezi procesy
  - Dosáhne se lepší strukturalizace programu
- Příklady
  - Souborový server v LAN
    - Musí vyřizovat během krátké doby několik požadavků na soubory
    - Pro vyřízení každého požadavku se zřídí samostatné vlákno
  - Symetrický multiprocesor
    - na různých procesorech mohou běžet vlákna souběžně
  - Menu vypisované souběžně se zpracováním
  - Překreslování obrazovky souběžně se zpracováním dat
  - Paralelizace algoritmu v multiprocesoru
- Lepší a přehlednější strukturalizace programu

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Problém konzistence sdílených dat

- Mějme aplikaci která sestává z více nezávislých částí, z nichž každá je implementována jako samostatné vlákno
- Vlákna nemusí běžet v sekvenci
  - Když vlákno čeká na nějakou událost, může běžet jiné vlákno téhož procesu, aniž by se přepínalo mezi procesy
- Vlastnosti takové implementace
  - Vlákna jednoho procesu sdílí paměť, 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 konzistenci zpracovávaných dat

Problém konzistence – příklad

- Scénář:
  - Proces vytvořil vlákna $\boldsymbol{T}_1$ a $\boldsymbol{T}_2$
  - $\boldsymbol{T}_1$ počítá $C = A + B$,
  - $\boldsymbol{T}_2$ používá hodnotu X: $A = A - X$; $B = B + X$;
  - $\boldsymbol{T}_1$ a $\boldsymbol{T}_2$ pracují souběžně, mohou se však prokládat
- Úmysl programátora
  - Nechť $A = 2$, $B = 3$, $X = 10$
  - $\boldsymbol{T}_2$ udělá $A = A - X$ a $B = B + X$ [$A = -8$, $B = 13$]
  - $\boldsymbol{T}_1$ spočítá $C = A + B$, hodnota C nezávisí na X [$C = 5$]
- Možná realita
  - $\boldsymbol{T}_2$ udělá $A = A - X$ a pak je mu odňat procesor [$A = -8$]
  - $\boldsymbol{T}_1$ spočítá $C = A + B = A - X + B$ [$C = -5$]
  - $\boldsymbol{T}_2$ udělá $B = B + X$ a to už hodnotu C neovlivní [$B = 13$]
  - Máme dva různé výsledky v proměnné C
- Poznámka
  - Kdyby nedošlo k preempci vlákna $\boldsymbol{T}_2$, žádný problém by nebyl nenastal!

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Stavy a odkládání vláken

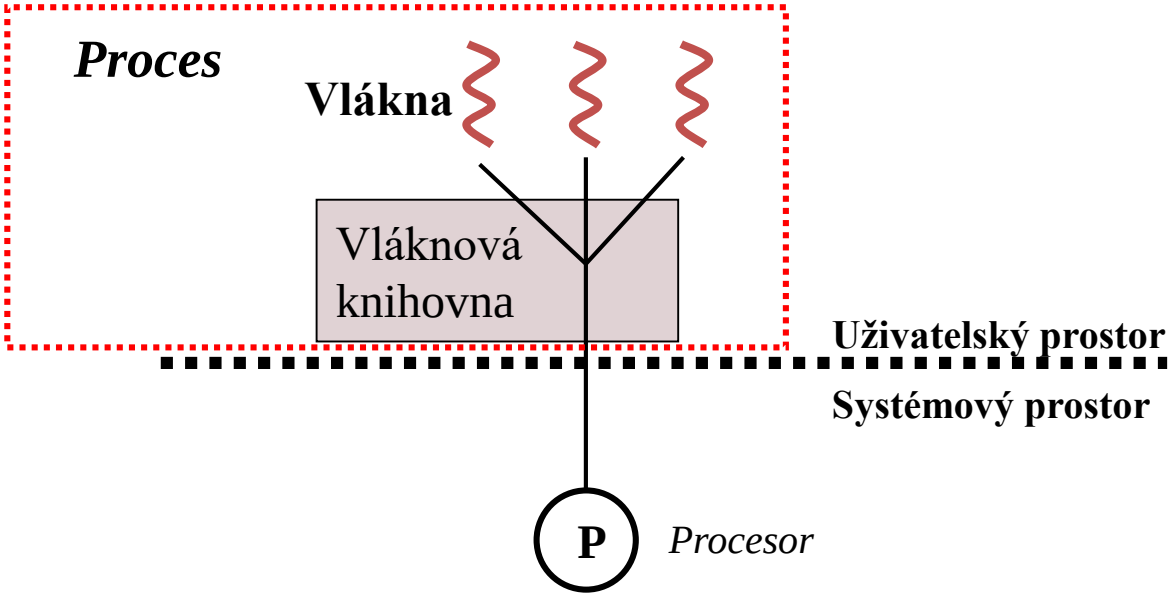
- Vlákna podléhají plánování a mají své stavy podobně jako procesy
- Základní stavy
  - běžící
  - připravené
  - čekající
- Všechna vlákna jednoho procesu sdílejí společný adresní prostor
  - => vlákna se samostatně neodkládají, odkládá je jen proces
- Ukončení procesu ukončuje všechna vlákna existující v tomto procesu

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Vlákna na uživatelské úrovni, ULT (1)

- *User-Level Threads* (**ULT**)
- Vlastnosti
  - Správu vláken provádí tzv. vláknová knihovna (*thread library*) na úrovni aplikačního procesu, JOS o jejich existenci neví
  - Přepojování mezi vlákny nepožaduje provádění funkcí jádra
  - Nepřepíná se ani kontext procesu ani režim procesoru
  - Aplikace má možnost zvolit si nejvhodnější strategii a algoritmus pro plánování vláken
- Příklady
  - POSIX – některé implementace knihovny Pthreads
  - Solaris1 – knihovna Threads
- Výhoda
  - Lze používat i v OS, který neobsahuje žádnou podporu vláken, stačí speciální knihovna (model 1 : *M*)

Vlákna na uživatelské úrovni, ULT (2)



Vlákna na uživatelské úrovni, ULT (3)

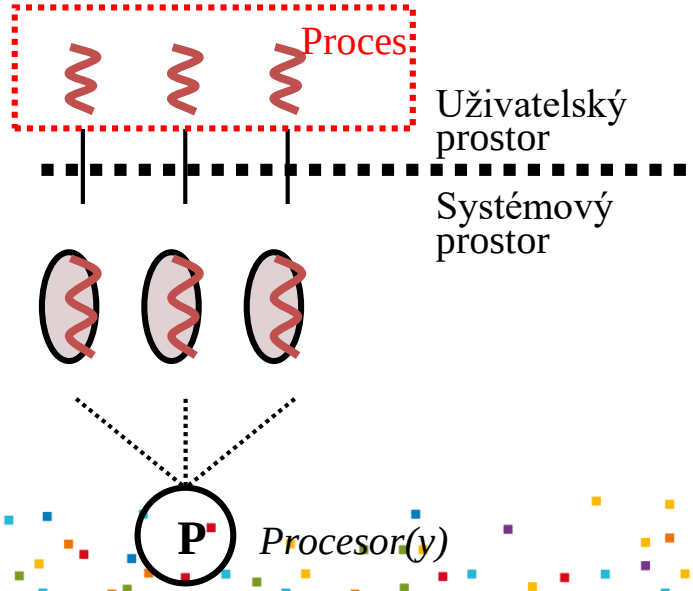
- Vláknová knihovna obsahuje funkce pro
  - vytváření a rušení vláken
  - předávání zpráv a dat mezi vlákny
  - plánování běhů vláken
  - uchovávání a obnova kontextů vláken
- Problém stavu vláken: Co když se proces nebo vlákno zablokuje?
  - Nechť proces *A* má dvě vlákna T_1 a T_2, přičemž T_1 právě běží
  - Mohou nastat následující situace:
    - T_1 požádá JOS o I/O operaci nebo jinou službu:
      - Jádro zablokuje proces *A* jako celek.
      - TCB_1 však indikuje, že vlákno T_1 běží, což není pravda. (Celý proces čeká, neboť žádost o systémovou službu nešla přes vláknovou knihovnu, takže ta o ní neví.)
    - Proces *A* vyčerpá časové kvantum:
      - JOS přeřadí proces *A* mezi připravené
      - TCB_1 indikuje, že T_1 je stále ve stavu „běžící“ (ve skutečnosti *neběží*!)
    - T_1 potřebuje akci realizovanou vláknem T_2:
      - Vlákno T_1 se zablokuje. Vlákno T_2 se rozběhne
      - Proces *A* zůstane ve stavu „běžící“ (což je správně)

Výhody a nevýhody uživatelských vláken

- Výhody:
  - Rychlé přepínání mezi vlákny (bez účasti JOS)
  - Lze použít i v OS, který vlákna nepodporuje; je nutná pouze vláknová knihovna
  - Rychlá tvorba a zánik vláken
  - Uživatelský proces má plnou kontrolu nad vlákny (např. může zadávat priority či volit plánovací algoritmus)
- Nevýhody:
  - Volání systémové služby jedním vláknem zablokuje všechna vlákna procesu
  - Dodatečná práce programátora pro řízení vláken
    - Lze však ponechat knihovnou definovaný implicitní algoritmus plánování vláken
  - Jádro přiděluje procesor pouze procesům, takže dvě vlákna téhož procesu nemohou běžet současně, i když je k dispozici více procesorů

Vlákna na úrovni jádra, KLT

- *Kernel-Level Threads* (**KLT**)
- Veškerá správa vláken je realizována OS
- Každé vlákno v uživatelském prostoru je zobrazeno na vlákno v jádře (model 1:1)
- JOS vytváří, plánuje a ruší vlákna
- Jádro může plánovat vlákna na různé CPU
  - Skutečný multiprocessing
- Příklady
  - Windows NT/2000/XP
  - Linux
  - 4.4BSD UNIXy
  - Tru64 UNIX



OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Výhody a nevýhody KLT

- Výhody:
  - Volání systému neblokuje ostatní vlákna téhož procesu
  - **Jeden proces může využít více procesorů** (skutečný paralelismus uvnitř jednoho procesu – každé vlákno běží na jiném procesoru)
  - Tvorba, rušení a přepínání mezi vlákny je levnější než mezi procesy
  - I moduly jádra mohou mít vícevláknový charakter
- Nevýhody:
  - Systémová správa je režijně nákladnější než u čistě uživatelských vláken
  - Klasické plánování není spravedlivé: Dostává-li vlákno své kvantum, pak procesy s více vlákny dostávají více času

Knihovna Pthreads

- Pthreads je POSIX-ový standard definující API pro vytváření a synchronizaci vláken a specifikace chování těchto vláken
- Knihovna Pthreads poskytuje unifikované API:
  - Nepodporuje-li JOS vlákna, knihovna Pthreads bude pracovat čistě s ULT
  - Implementuje-li příslušné jádro KLT, pak knihovna Pthreads toho bude využívat
  - Pthreads je tedy systémově závislá knihovna
- **Příklad:** Samostatné vlákno, které počítá součet prvních *n* celých čísel

Příklad volání API Pthreads

Příklad: Samostatné vlákno, které počítá součet prvních *n* celých čísel; *n* se zadává jako parametr programu na příkazové řádce

```
#include <pthread.h>
#include <stdio.h>

int sum;                                        /* sdílená data */
void *runner(void *param);                      /* rutina realizující vlákno */

main(int argc, char *argv[]) {
        pthread_t tid;                          /* identifikátor vlákna*/
        pthread_attr_t attr;                    /* atributy vlákna */
        pthread_attr_init(&attr);               /* inicializuj implicitní atributy */
        pthread_create(&tid, &attr, runner, argv[1]);   /* vytvoř vlákno */
        pthread_join(tid,NULL);                 /* čekej až vlákno skončí */
        printf("sum = %d\n",sum);
}

void *runner(void *param) {
    int upper = atoi(param);  int i;  sum = 0;
    if (upper > 0) {
        for (i = 1; i <= upper; i++)
            sum += i;
     }
    pthread_exit(0);
}
```

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Implementace vláken ve Windows

- Aplikace ve Windows běží jako proces tvořený jedním nebo více vlákny
- Windows implementují mapování 1:1
- Někteří autoři dokonce tvrdí, že *Proces se nemůže vykonávat, neboť je jen kontejnerem pro vlákna a jen ta jsou schopná běhu*
- Každému vláknu patří
  - identifikátor vlákna
  - sada registrů
  - samostatný uživatelský a systémový zásobník
  - privátní datovou oblast

Vlákna v Linuxu a Javě

- Vlákna Linux:
  - Linux nazývá vlákna *tasks*
  - Vytváření vláken je realizováno službou **`clone()`**
  - **`clone()`** umožňuje vláknu (task) sdílet adresní prostor s rodičem
    - **`fork()`** vytvoří zcela samostatný proces s kopií prostoru rodičovského procesu
    - **`clone()`** vytvoří vlákno, které dostane odkaz (pointer) na adresní prostor rodiče

- Vlákna v Javě:
  - Java má třídu „Thread“ a instancí je vlákno
    - Samozřejmě lze z třídy Thread odvodit podtřídu a některé metody přepsat
  - Vlákna jsou spravována přímo JVM
    - JVM spolu se základními Java třídami vlastně vytváří virtuální stroj obsahující jak „hardware“ (vlastní JVM) tak i na něm běžící OS podporující vlákna

OSTRAVSKÁ UNIVERZITA
PŘÍRODOVĚDECKÁ FAKULTA

Děkuji za pozornost