EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Operační systémy

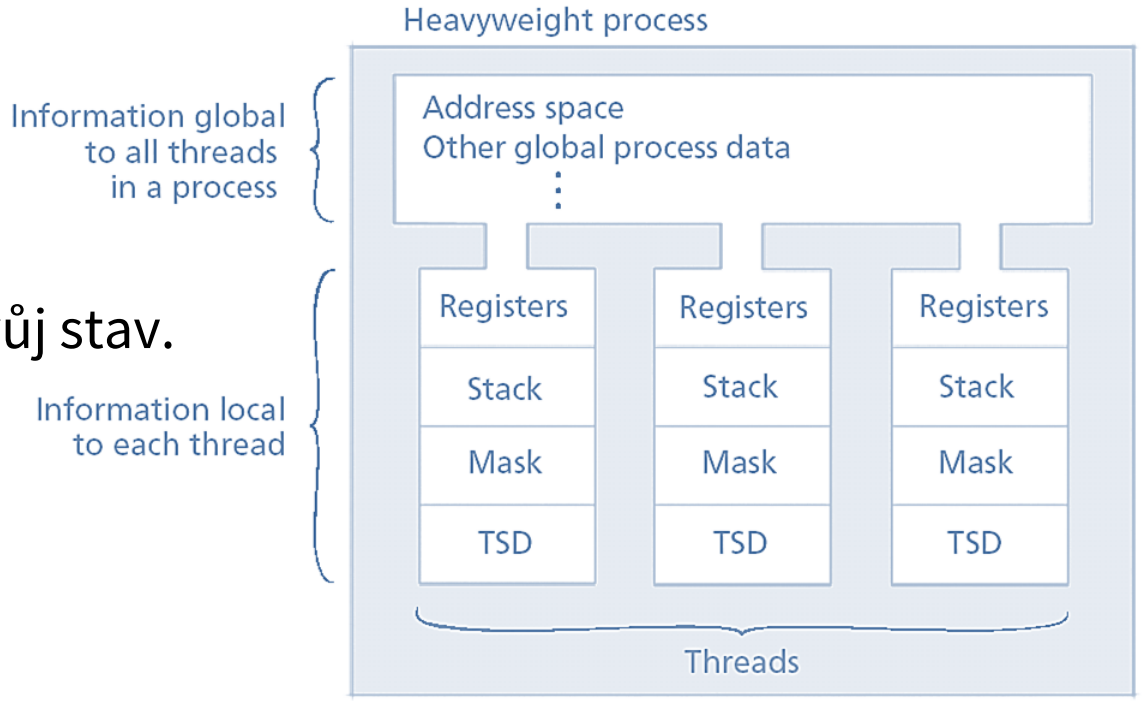
Vlákna (Threads)

Strategický projekt UTB ve Zlíně, reg. č. CZ.02.2.69/0.0/0.0/16_015/0002204

Martin Sysel
Fakulta aplikované informatiky
Univerzita Tomáše Bati ve Zlíně

Definice vlákna

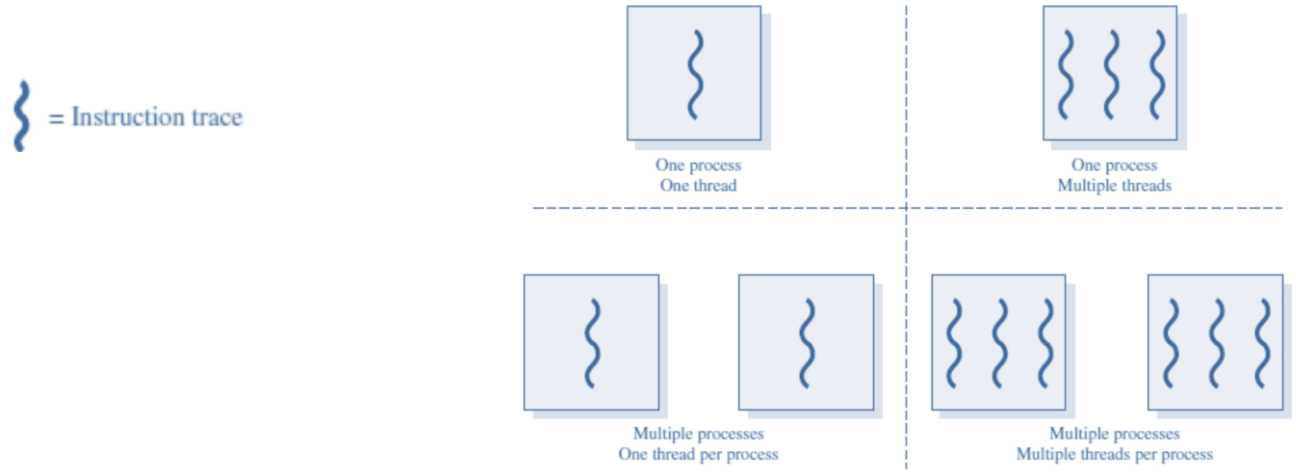
- Odlehčený proces (Lightweight process)
- Vlákna jsou plánována na CPU.
  - Jednotka plánování je vlákno. Vlákno má svůj stav.
  - Bod běhu procesu. Funkce *int main ()*
    - Každý proces má minimálně jedno vlákno.
  - Každé vlákno provádí vlastní sadu instrukcí.
- Neexistuje samo, stále patří do procesu.
  - Sdílí mnoho zdrojů v rámci procesu.
  - Registry, zásobník, lokální proměnné, … má každé vlákno vlastní.
- Vlákna mohou být spravována OS (KLT) nebo uživatelskou aplikací (ULT).
- Jako vlastník zdrojů je stále označován proces.
- Kontext vlákna je uložený v TCB (Thread Control Block) – obdoba PCB



(Deitel, Deitel & Choffness, 2004)

Multithreading

- Více vláknové zpracování odkazuje na schopnost operačního systému podporovat v rámci jednoho procesu více vláken.
  - Možnost souběžného vykonávání
- Tradiční přístup byl proces s jediným vláknem.



(Silberschatz, Galvin & Gagne, 2013)

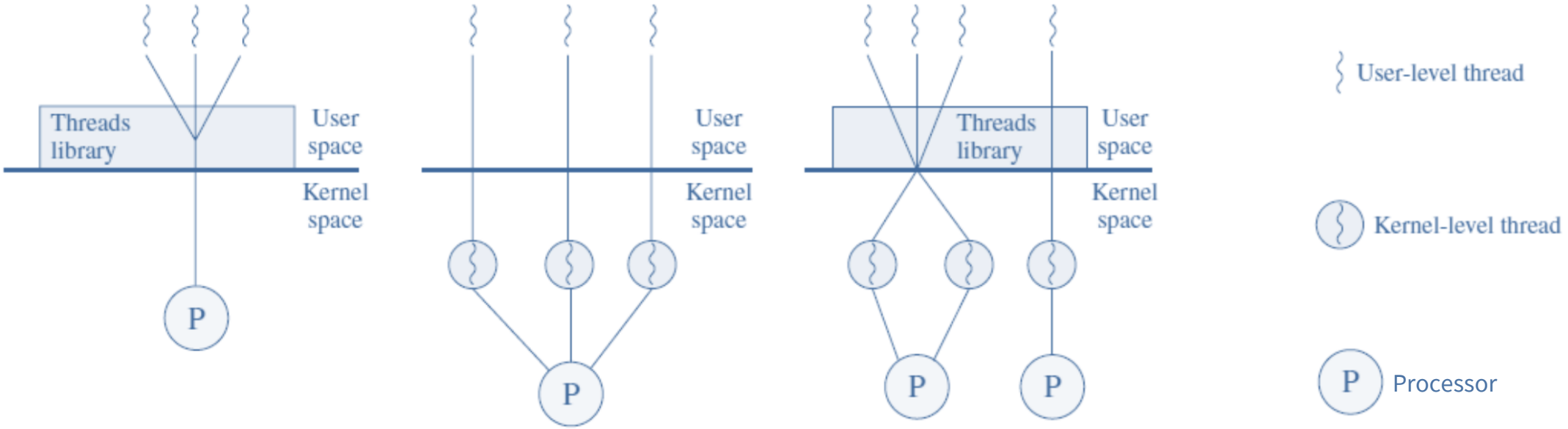
Výhody vláken

- Vlákna jsou výhodná v důsledku trendů v návrhu softwaru, výkonu hardwaru. Umožňují lepší spolupráci.
  - Lepší škálování pro systémy s více procesory.
  - Sdílený adresní prostor má menší režii než IPC.
  - Méně času na vytvoření a ukončení vlákna v procesu.
    - Vytváření procesu je asi desetkrát pomalejší.
  - Méně času na přepínání mezi dvěma vlákny v rámci stejného procesu než na přepínání mezi procesy.
  - Existuje šance, že nějaká data procesu zůstanou v cache.
  - Vlákna zvyšují efektivitu komunikace v rámci procesu.

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

Vláknové modely

- Tři nejoblíbenější modely závitů
  - Vlákna na uživatelské úrovni (ULT – User Level Threads)
  - Vlákna na úrovni jádra (KLT – Kernel Level Threads)
  - Kombinace vláken na uživatelské úrovni a na úrovni jádra.



(Silberschatz, Galvin & Gagne, 2013)

Vlákna na uživatelské úrovni (ULT)

- Vlákna na uživatelské úrovni provádějící operace v uživatelském prostoru
  - Vlákna vznikají za běhu pomocí knihovny, která nemůže spouštět privilegované instrukce a nemůže přistupovat k primitivům jádra. Správa vláken probíhá v uživatelském prostoru.
    - Mapování vláken many-to-one - OS mapuje všechna vlákna v procesu do kontextu jednoho vlákna, které vidí.
- Výhody
  - Knihovna na uživatelské úrovni, může optimalizovat plánování svých vláken
    - optimalizace specifická pro aplikaci
  - Přepínání vláken nevyžaduje oprávnění k privilegovaným instrukcím jádra (kernel mód) - rychlejší
  - Synchronizace je prováděná mimo jádro, vyhneme se tak přepínání kontextu
  - Lehce přenositelné na jiné architektury a OS (portable)
- Nevýhoda
  - Jádro vnímá proces s více uživatelskými vlákny jako proces s jedním vláknem
    - Pokud vlákno provádí I/O, je dané vlákno, a tedy tím i celý proces zablokován – snížení výkonu
    - Nelze plánovat na více procesorů najednou (Systémová volání vláken blokují celý proces)
    - Nelze věrohodně sledovat stav vláken.

(Deitel, Deitel & Choffness, 2004)

Vlákna na úrovni jádra (KLT)

- Vlákna na úrovni jádra se pokouší řešit omezení vláken na uživatelské úrovni mapováním každého vlákna do svého vlastního spouštěcího kontextu
  - Vlákna na úrovni jádra poskytují mapování vláken one-to-one
  - Přepínání vláken vyžaduje jádro
- Výhody
  - Zvýšená škálovatelnost, interaktivita a propustnost
- Nevýhody
  - Vyšší režie v důsledku přepínání kontextu
  - Horší přenositelnost díky rozdílným rozhraním API specifickým pro OS
  - Plánování vláken na úrovni jádra není vždy optimálním řešením pro více vláknové aplikace, protože rozhoduje jádro.

(Deitel, Deitel & Choffness, 2004)

Kombinované ULT a KLT

- Mapování mnoha uživatelských vláken do mnoha vláken na úrovni jádra
  - mapování vláken many-to-many (*m-to-n*).
  - Počet uživatelských vláken a vláken na úrovni jádra nemusí být stejný.
  - Implementací sdružování vláken na uživatelské úrovni lze někdy snížit režii ve srovnání s mapováním vláken one-to-one.
- Pracovní vlákna (Worker threads)
  - Trvalá vlákna jádra, která zabírají fond vláken.
  - Zvyšuje výkon v prostředích, kde jsou vlákna často vytvářena a ukončována.
  - Každé nové vlákno je prováděno pracovním vláknem.
- Aktivace plánovače
  - Technika, která umožňuje vláknové knihovně na uživatelské úrovni naplánovat vlákna.
  - Vyskytuje se, když operační systém volá ULT knihovnu, která určuje, zda některé z jejích vláken potřebuje přeplánování.

(Deitel, Deitel & Choffness, 2004)

Operace s vlákny

- Vlákna a procesy mají společné operace
  - Vytvoření (Create)
  - Ukončení (Exit, Terminate)
  - Pozastavení (Suspend)
  - Obnovení (Resume)
  - Uspání (Sleep)
  - Probuzení (Wake)
  - Zrušení (Cancel)
    - Označuje, že vlákno by mělo být ukončeno, ale nezaručuje to. Vlákna mohou signál maskovat.
  - Sloučení (Join)
    - Primární vlákno může čekat (je blokováno) na ukončení všech ostatních vláken.

Vláknové operace neodpovídají přesně procesním operacím

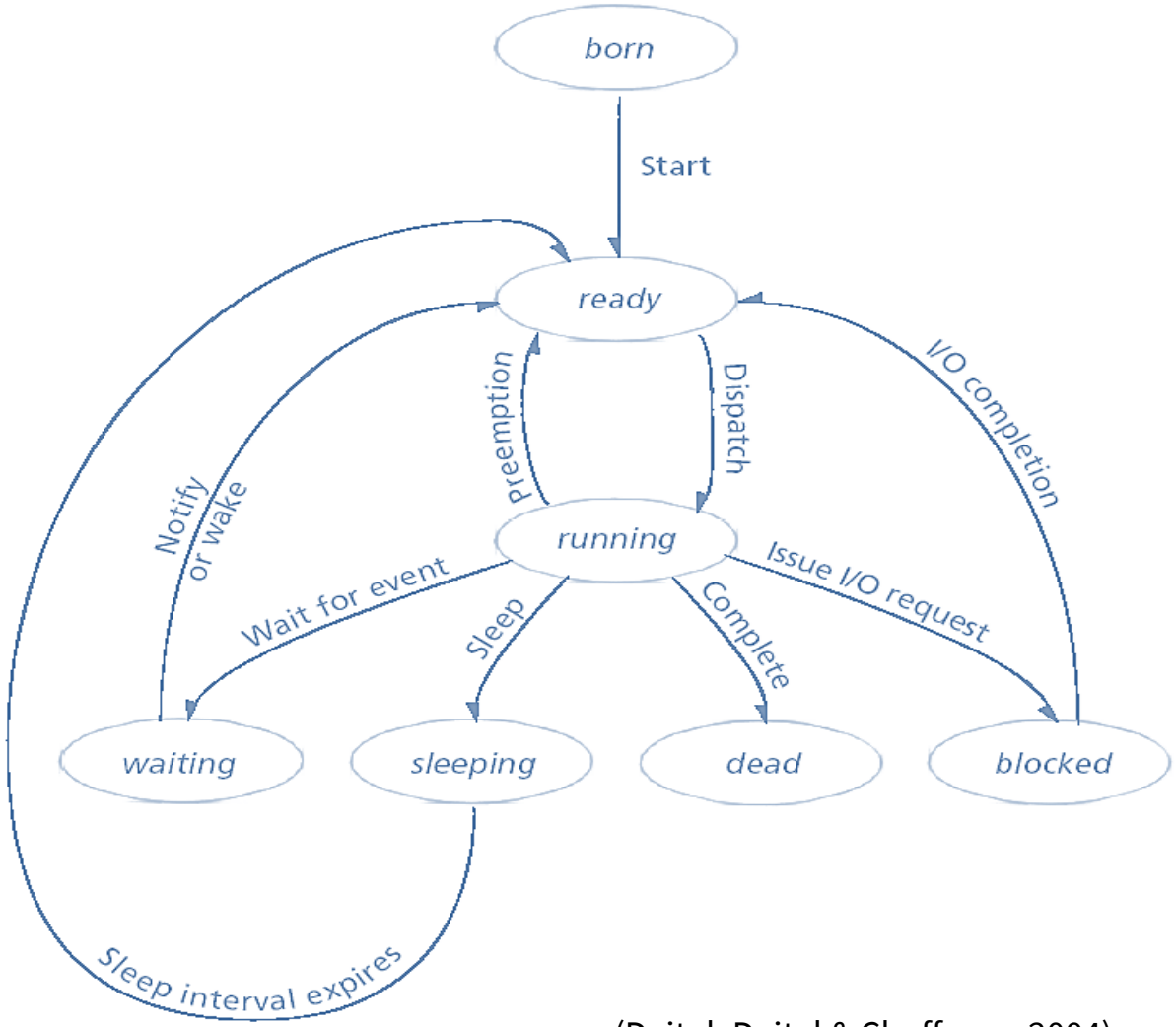
(Deitel, Deitel & Choffness, 2004)

Stavy vláken

- *Born* – Initicializace vlákna.
- *Ready* – Vlákno je připravené k vykonání, čeká na CPU.
- *Running* – Vykonávané vlákno.
- *Waiting* – Vlákno čeká na událost.
- *Sleeping* – Vlákno je uspáno po definovaný čas.
- *Blocked* – Vlákno čeká na provedení I/O požadavku.
- *Dead* – Vlákno bylo dokončeno.

(Deitel, Deitel & Choffness, 2004)

Obecný životní cyklus vláken



(Deitel, Deitel & Choffness, 2004)

POSIX a Pthreads

POSIX (Portable Operating System Interface) Standard unixových OS. Jednotné rozhraní, které zajišťuje přenositelnost programů.

- Pthreads – vlákna využívající POSIX vláknové API
  - Unifikované API, není definované zda ULT nebo KLT
    - Jedná se o systémově závislou knihovnu
    - Je-li KLT definováno, tak je využito, jinak se použije ULT
  - V POSIX jsou registry procesoru, zásobník a maskování signálů udržovány jednotlivě pro každé vlákno.
    - Obslužné programy signálů jsou globální pro všechna vlákna procesu.
  - POSIX specifikuje, jak by OS měl předávat signály do Pthreads, určuje několik režimů rušení vlákna.

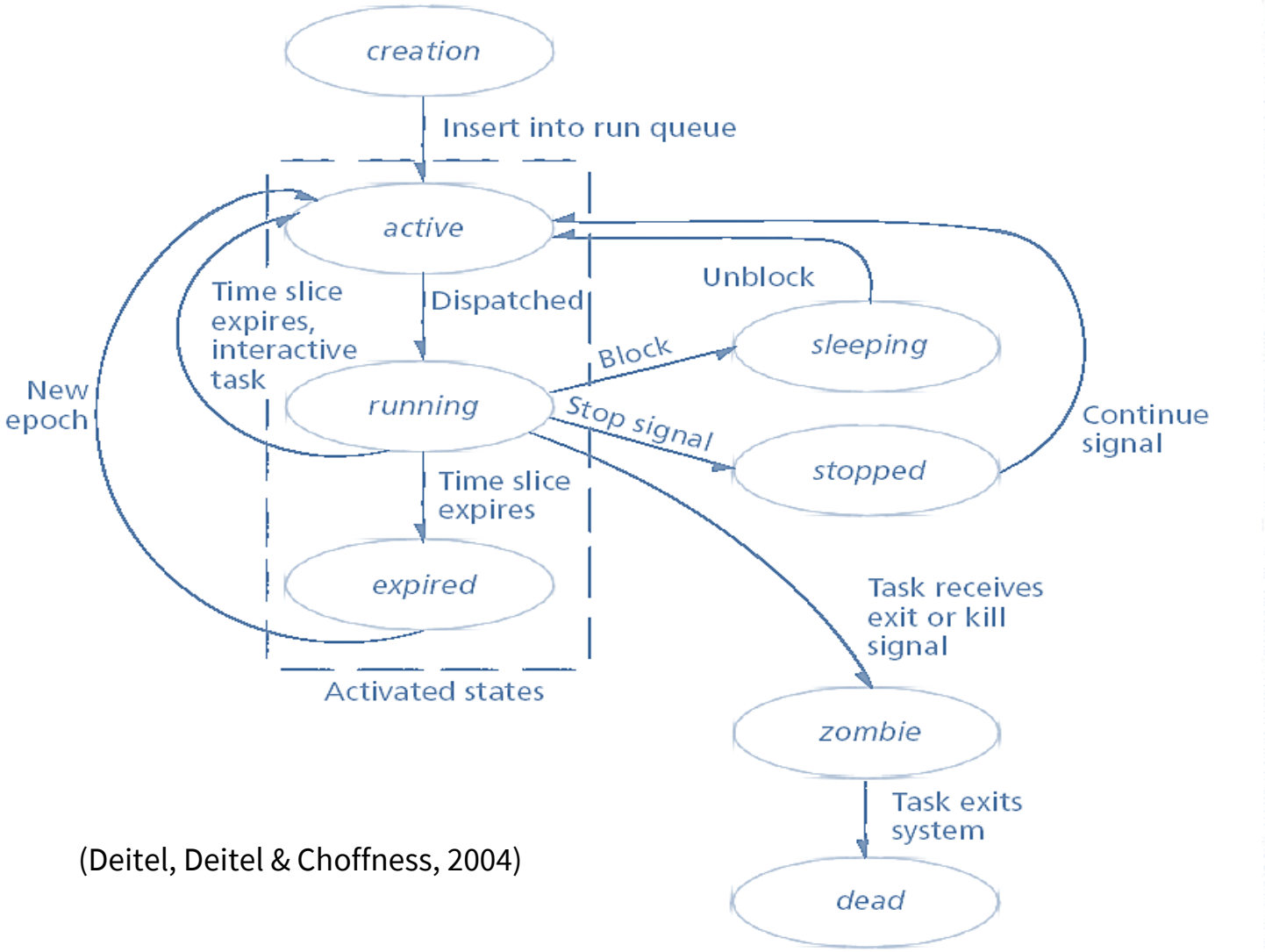
Vlákna v Linuxu

kernel <= 2.4

- Linux přiděluje stejný typ deskriptoru procesům a vláknům (***tasks***)
  - Linux používá k vytvoření procesů systémové volání *fork* (UNIX-based)
    - vytvoří zcela samostatný proces s kopií prostoru rodičovského procesu (viz dříve)
  - Současné systémy používají upravenou verzi *clone*
    - *Clone* přijímá argumenty, které určují, které zdroje se budou sdílet s potomkem.
      - dostane odkaz (pointer) na adresní prostor rodiče
        - » Copy on Write

- Nativní knihovna POSIX vláken v jádře > = 2.6
  - plná shoda s normou POSIX.
  - Model 1: 1, vylepšená škálovatelnost a výkon

Vlákna v Linuxu – životní cyklus



(Deitel, Deitel & Choffness, 2004)

Stavy vláken v Linuxu

- *Běžící (Running)* – Běžící vlákno.
- *Aktivní (Active)* – Vlákna připravená k běhu, která ještě nevyčerpala svá časová kvanta. (viz algoritmy plánování)
- *Expirovaná (Expired)* - Vlákna připravená k běhu, která vyčerpala svá časová kvanta a čekají než bude fronta aktivních vláken prázdná.
  - Následně dojde k záměně aktivní a expirované fronty.
  - Plánovač se snaží zvýhodňovat interaktivní vlákna.
- *Spící, čekající (Sleeping)* – Vlákna čekající na událost, zdroje, nebo jsou uspaná na definovaný čas.
  - Linux rozlišuje mezi třemi typy čekajících vláken; (před kernelem 2.6.25 pouze dva typy)
    - *Interruptible* - můžou být přerušena signálem
    - *Uninterruptible* - nemůžou být přerušena, protože čekají na provedení určitého systémového volání.
    - *Killable* – (TASK_KILLABLE) nemůžou být přerušena, ale můžou být zabita (kill)

(Yu-Hsin Hung, 2016; Kumar, 2008)

Stavy vláken v Linuxu

- *Stopped* – Vlákna byla zastavena (stopnuta pomocí signálu) [^Z].
  - Čekají na signál pro pokračování.
- *Zombie* – Vlákno, které uvolnilo své zdroje, ale popisovač stále existuje. Rodičovský proces čeká pomocí systémového volání *wait()* na návratovou hodnotu a další info od potomka
  - *waitpid(), waitid()*.
- *Dead* – Zombie je kompletně odstraněno z paměti
  - Po volání wait().

(Hoffman, 2016)

POSIX vlákna

Thread call	Description
pthread_create	Vytvoření nového vlákna
pthread_exit	Ukončení vlákna
pthread_join	Čekání na určité vlákno
pthread_yield	Uvolnění CPU pro jiné vlákno
pthread_attr_init	Vytvoření a inicializace struktury atributů vlákna
pthread_attr_destroy	Odstranění struktury atributů vlákna

Nejdůležitější POSIX thread volání.
(TANENBAUM, 3e., 2008; Hendler, D., Meisels, A., 2011)

Vytvoření vlákna – Linux, Pthread

```
#include <iostream>
#include <stdlib.h>
#include <pthread.h>

using namespace std;
#define NUM_THREADS 5

void *PrintHello(void *threadID) {
    long tid = *(long *)threadID;
    cout << "THREAD: Hello from thread ID: " << tid << endl;
    pthread_exit(EXIT_SUCCESS);
}

int main() {
    pthread_t threads[NUM_THREADS];
    int i, rc;                          //thread number, result code
    //create all threads
    for (i = 0;i < NUM_THREADS;i++) {
        cout << "MAIN: Creating thread ID: " << i << endl;
        if (rc = pthread_create(&threads[i], NULL, PrintHello, (void *)i)) {
            cout << "Error: unable create thread ID: " << i << ", " << rc << endl;
            exit(-1);
        }
    }
    //wait for each thread to complete
    for (i = 0;i < NUM_THREADS; i++) {
        rc = pthread_join(threads[i], NULL);
        cout << "MAIN: Thread ID: " << i << " has been ended" << endl;
    }
    pthread_exit(EXIT_SUCCESS);
}
```

```
MAIN: Creating thread ID: 0
MAIN: Creating thread ID: 1
THREAD: Hello from thread ID: 1
MAIN: Creating thread ID: 2
THREAD: Hello from thread ID: 2
MAIN: Creating thread ID: 3
THREAD: Hello from thread ID: 3
MAIN: Creating thread ID: 4
THREAD: Hello from thread ID: 4
MAIN: Thread ID: 0 has been ended
MAIN: Thread ID: 1 has been ended
MAIN: Thread ID: 2 has been ended
MAIN: Thread ID: 3 has been ended
THREAD: Hello from thread ID: 0
MAIN: Thread ID: 4 has been ended
```

Vlákna ve Windows

- Jednotka plánování je vlákno, mapují se 1:1, KLT
  - Proces je tvořen jedním nebo více vlákny
- Každé vlákno obsahuje
  - Vlastní identifikátor
  - Zásobník (uživatelský a systémový)
  - Sada registrů
  - Vlastní datovou oblast

Použití vláken (The CreateThread function example)

- https://docs.microsoft.com/en-us/windows/desktop/procthread/creating-threads

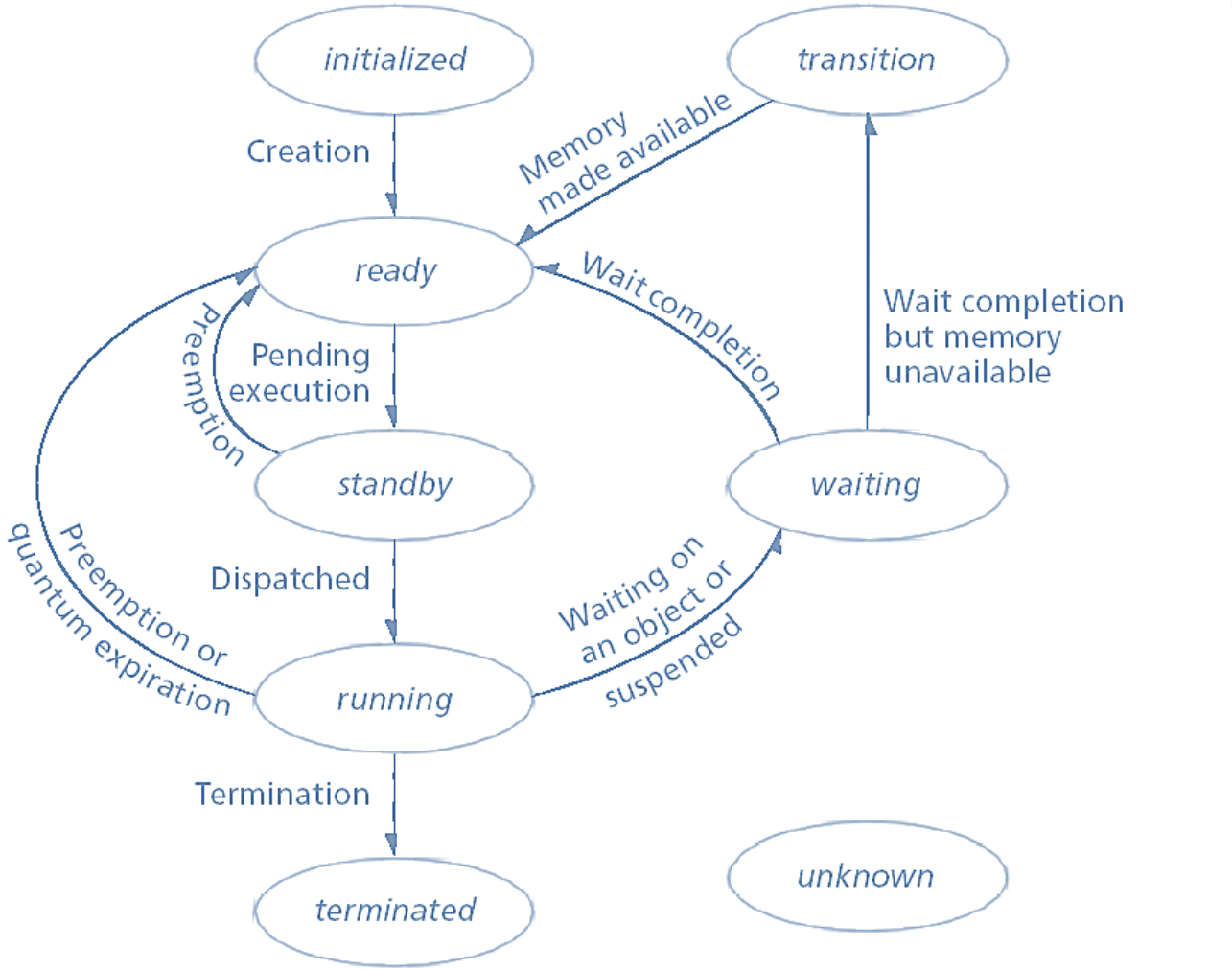
(Lažanský, 2018)

Windows Fibers

- Vlákna systému Windows mohou vytvářet *fibers.*
  - Fiber je ULT, Vlákna se spouští v kontextu vláken, která je plánují.
  - Každé vlákno (Thread) může spravovat více fibers.
    - Fibers neposkytují výhody oproti dobře navržené více vláknové aplikaci.
- Windows poskytují každému procesu fond *pracovních vláken*
  - Jedná se o vlákna jádra, která vykonávají funkce určené uživatelskými vlákny
    - Mapování many-to-many (kombinace ULT a KLT)
- Použití Fibers (The CreateFiber function example)
  - https://docs.microsoft.com/en-us/windows/desktop/procthread/using-fibers

(Yosifovich, 2017)

Vlákna ve Windows – životní cyklus



(Deitel, Deitel & Choffness, 2004)

Stavy vláken ve Windows

- *Ready* – Vlákno čekající na provedení na CPU.
- *Deferred ready* – vlákno bylo vybráno tak, aby běželo na konkrétním procesoru, ale tam se ve skutečnosti nespustilo.
- *Standby* – Vybrané vlákno ke zpracování na konkrétním CPU.
  - Dispatcher provádí kontext switch mezi tímto a běžícím vláknem.
  - Pro každý procesor v systému může být v pohotovostním stavu pouze jedno vlákno.
  - Vlákno lze před spuštěním navrátit do *Ready*. (např. přijde proces s vyšší prioritou)
- *Running* – Vlákno, které se provádí.
  - Provádí se dokud
    - Neskončí časové kvantum (a další vlákno se stejnou prioritou je připraveno ke spuštění)
    - Je přerušeno
    - Je dokončeno
    - Neuvolní CPU pro jiné vlákno
    - Dobrovolně nevstoupí do stavu čekající

(YOSIFOVICH, 2017)

Stavy vláken ve Windows

- *Waiting* – Vlákno čeká na I/O nebo na nějakou událost:
  - Dobrovolně čeká na synchronizaci (např. semafor),
  - Řeší I/O (např. stránkování)
- *Transition* – Vlákno je připraveno pro vykonávání, ale bylo odstránkováno. Jakmile bude načteno v hlavní paměti, tak bude přesunuto do připraveného stavu.
- *Terminated* – Vlákno dokončilo svou úlohu.
- *Initialized* – Interní stav zatímco je vlákno vytvářeno.
- *Unknown* – Stav vlákna je kvůli chybě neznámý.

(YOSIFOVICH, 2017)

Win32 volání pro správu procesů a vláken

Win32 API Function	Description
CreateProcess	Vytvoří nový proces a jeho primární vlákno.
CreateThread	Vytvočí vlákno pro vykonání v rámci virtuálního adresního prostoru procesu
CreateFiber	Vytvoří Fiber, přidělí mu zásobník a připraví ho ke spuštění na počáteční adrese
ExitProcess	Ukončí proces a všechny jeho vlákna
ExitThread	Ukončí volající vlákno
TerminateThread	Ukončí vlákno
Sleep	Odloží vykonávané vlákno na definovanou dobu
SuspendThread	Odloží definované vlákno

Vybrané Win32 funkce
(source: https://docs.microsoft.com/en-us/windows/desktop/procthread/process-and-thread-functions)

Použitá a doporučená literatura

- DUARTE, Gustavo. Anatomy of a Program in Memory. *Many But Finite: Tech and science for curious people.* [online]. 2009, Jan 27th, 2009 [cit. 2018-02-28]. Dostupné z: https://manybutfinite.com/post/anatomy-of-a-program-in-memory/
- DEITEL H. M., DEITEL P. J. & CHOFFNES D. R.: *Operating systems*. 3rd ed., Pearson/Prentice Hall, 2004. ISBN 0131246968.
- TANENBAUM A. S.: *Modern operating systems*. 4th ed. Boston: Pearson, 2015. ISBN 0-13-359162-x.
- SILBERSCHATZ A., GALVIN P. B. & GAGNE G.: *Operating system concepts*. 9th ed. Hoboken, NJ: Wiley, 2013. ISBN 978-1-118-06333-0.
- STALLINGS W.: *Operating Systems: Internals and Design Principles.* 8th ed., Pearson Education Limited, 2014.

Použitá a doporučená literatura

- YOSIFOVICH, P., IONESCU, A., RUSSINOVICH, M.E., SOLOMON, D. A.: Windows Internals, Part 1: System architecture, processes, threads, memory management, and more (7th Edition). Microsoft Press, 2017.
- Yu-Hsin Hung. *Linux Kernel: Process Scheduling* [online]. Mar 25, 2016 [cit. 2019-02-07]. Dostupné z: https://medium.com/hungys-blog/linux-kernel-process-scheduling-8ce05939fabd
- Kumar Avinesh. *TASK_KILLABLE: New process state in Linux*. [online] 2008 [cit. 2019-02-07]. Dostupné z: https://www.ibm.com/developerworks/linux/library/l-task-killable/
- HOFFMAN, Chris. What Is a “Zombie Process” on Linux?. *How-To Geek* [online]. September 28, 2016 [cit. 2019-02-07]. Dostupné z: https://www.howtogeek.com/119815/htg-explains-what-is-a-zombie-process-on-linux/
- HENDLER, D., MEISELS, A.: Operating Systems. 2011.