ÚVOD DO OPERAČNÍCH SYSTÉMŮ

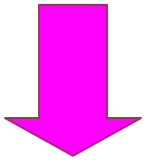
Ver.1.00

Procesy a vlákna (*Processes and Threads*)

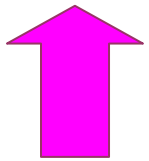
Správa procesů a vláken

České vysoké učení technické Fakulta elektrotechnická

Studijní materiály a informace o předmětu



- http://measure.feld.cvut.cz/vyuka/predmety/bakalarske/navody
- http://aldebaran.feld.cvut.cz/vyuka/uvod_do_os



Použitá literatura

[1] Stallings, W.: Operating Systems. Internals and Design Principles. 4th Edition. Prentice Hall, New Jersey, 2001.

[2] Silberschatz, A. – Galvin, P. B. - Gagne, G. : Operating System Concepts. 6th Edition. John Wiley & Sons, 2003.

[3] Tanenbaum, A.: Modern Operating Systems. Modern Operating Systems. Prentice Hall, New Jersey, 2008.

Proces (*Process*) *

Proces je definován jako:

- Spuštěný (tj. běžící) počítačový program
- „Instance programu“ spuštěného v počítači
- „Entita“ přidělená procesoru a vykonávaná procesorem
- „Jednotka aktivity“ charakterizovaná sekvencí prováděných instrukcí, okamžitým stavem a přiřazenou množinou systémových instrukcí [1]

Proces obsahuje **kód programu** a **data**, se kterými pracuje.

Proces vlastní:

- privátní adresový prostor
- systémové prostředky (soubory, dynamicky alokovanou paměť, synchronizační prostředky apod.)
- nejméně jedno vlákno

Viz [3]

PCB (*Process Controll Block*)

PCB =(„tabulka popisu procesů“) datová struktura spravovaná jádrem OS obsahující informace potřebné pro správu procesů:

- PID
- Identifikace stavu
- Priorita
- Obsah registrů procesoru (*Program Counter, Memory Pointers* =ukazatele na kód a data procesu a dále paměťové bloky sdílené s ostatními procesy, *Context Data* = data v registrech procesoru)
- Informace o I/O zařízeních (*I/O requests*, I/O zařízeních, souborech) přidělených procesu
- *Accounting* („účtovací informace“, čas spuštění, kolik procesorového času spotřeboval, aj.

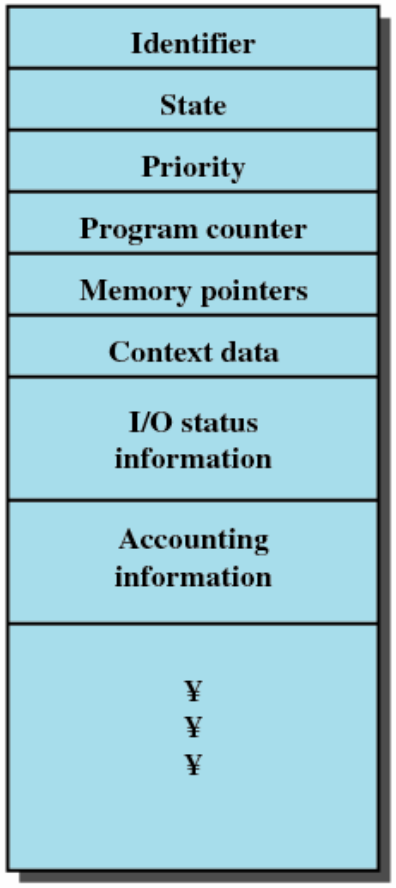


Figure 3.1 Simplified Process Control Block

Viz [1]

Přepínání kontextu (*Context Switch*)

V OS systému (víceúlohovém) se **PCB** používá při přepínání procesoru z jednoho procesu na další tzv. **přepínání kontextu** (*context switch*):

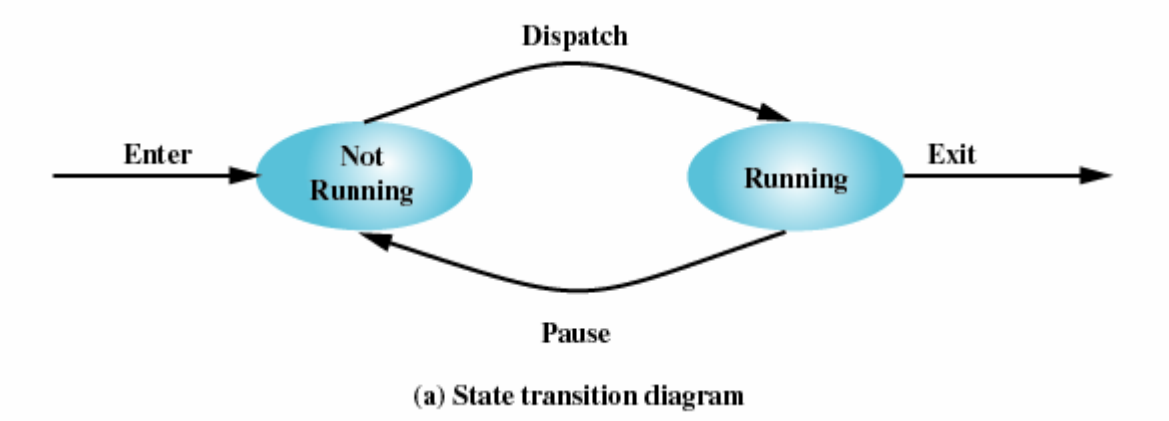
- kontext procesu je reprezentován PCB
- při přepínání kontextu je uložen PCB starého procesu a nahrán PCB nově spuštěného
- přepínání procesů řídí **plánovač** (*scheduller*)
- **doba přepnutí kontextu** je v řádu jednotek až stovek mikrosekund (závisí na HW počítače)
- **doba přidělení procesoru** procesu je v řádu jednotek až desítek milisekund

Stavový model procesu

Dvoustavový model

Proces může mít pouze 2 stavy:

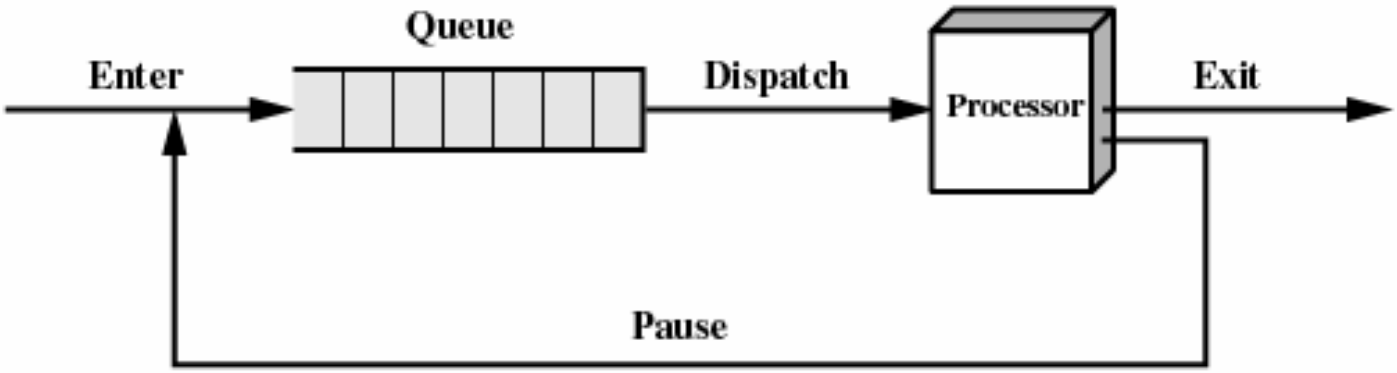
- *Running*
- *Not-running*



(a) State transition diagram

Viz [1]

Fronta procesů připravených ke spuštění procesorem



(b) Queuing diagram

Viz [1]

Složitější model procesu

Pětistavový model: přidány další 3 stavy

Stav „blokován“ (také „čekající“ nebo “spící“) významně zlepšuje výsledné využití procesoru.

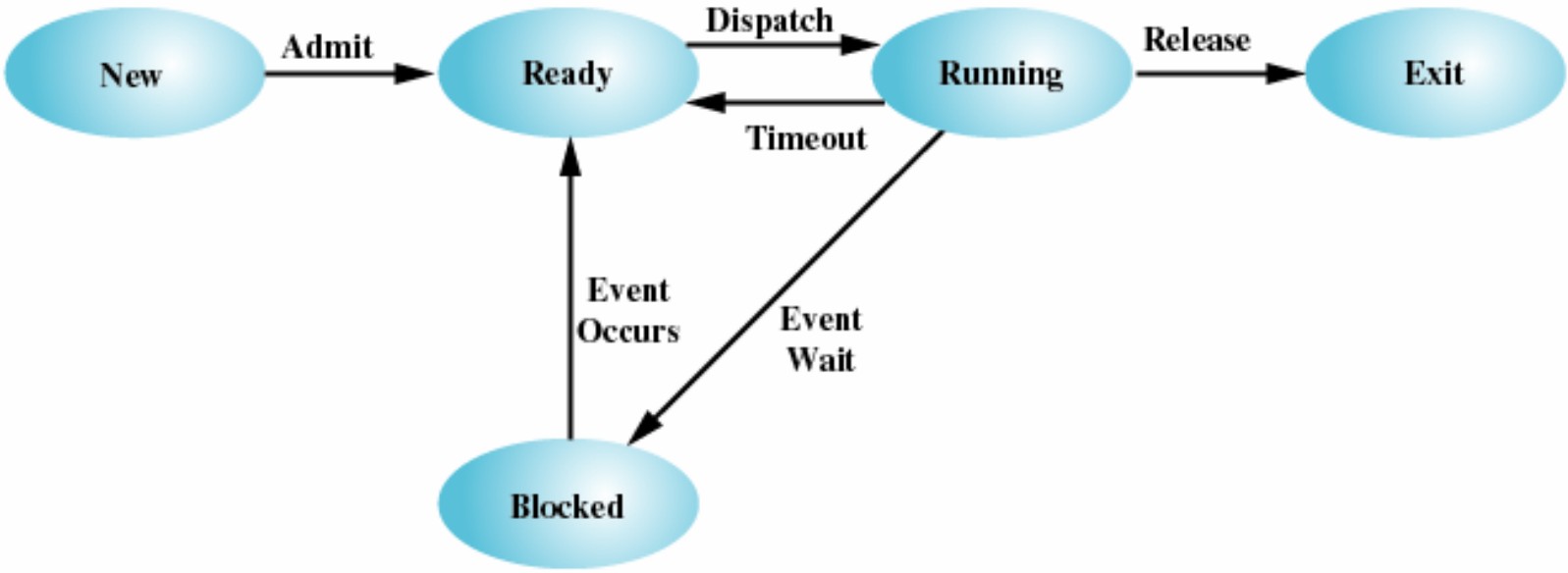
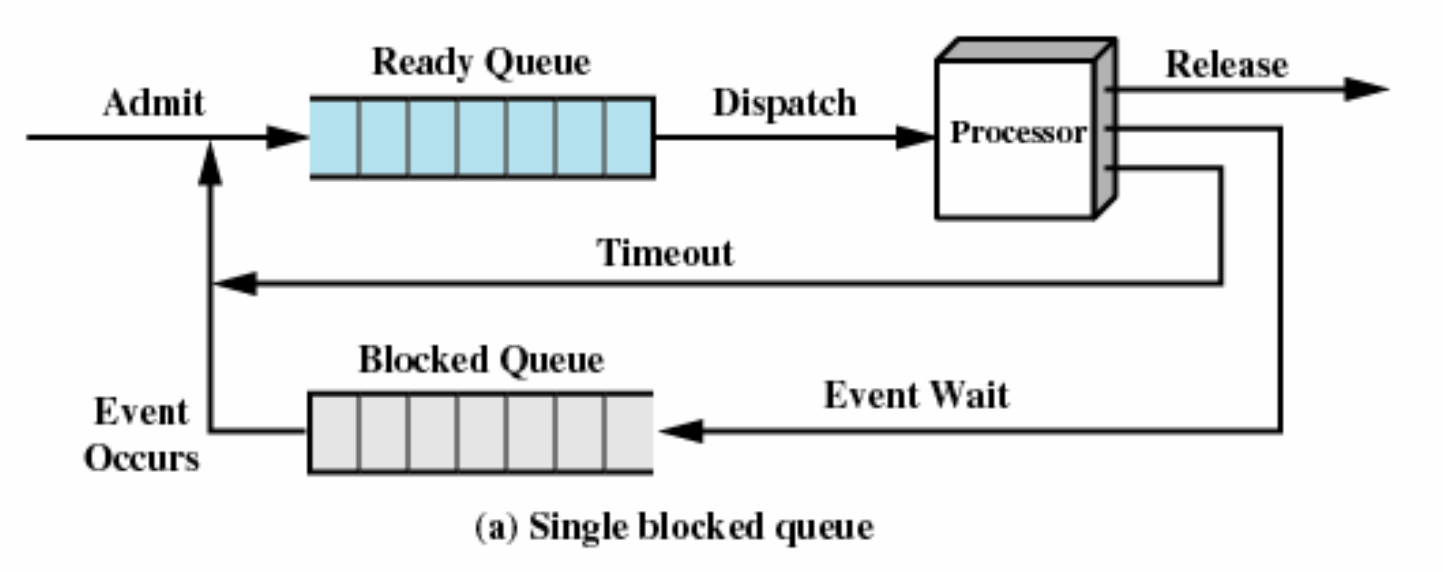


Figure 3.6 Five-State Process Model

Viz [1]

Použití dvou front procesů

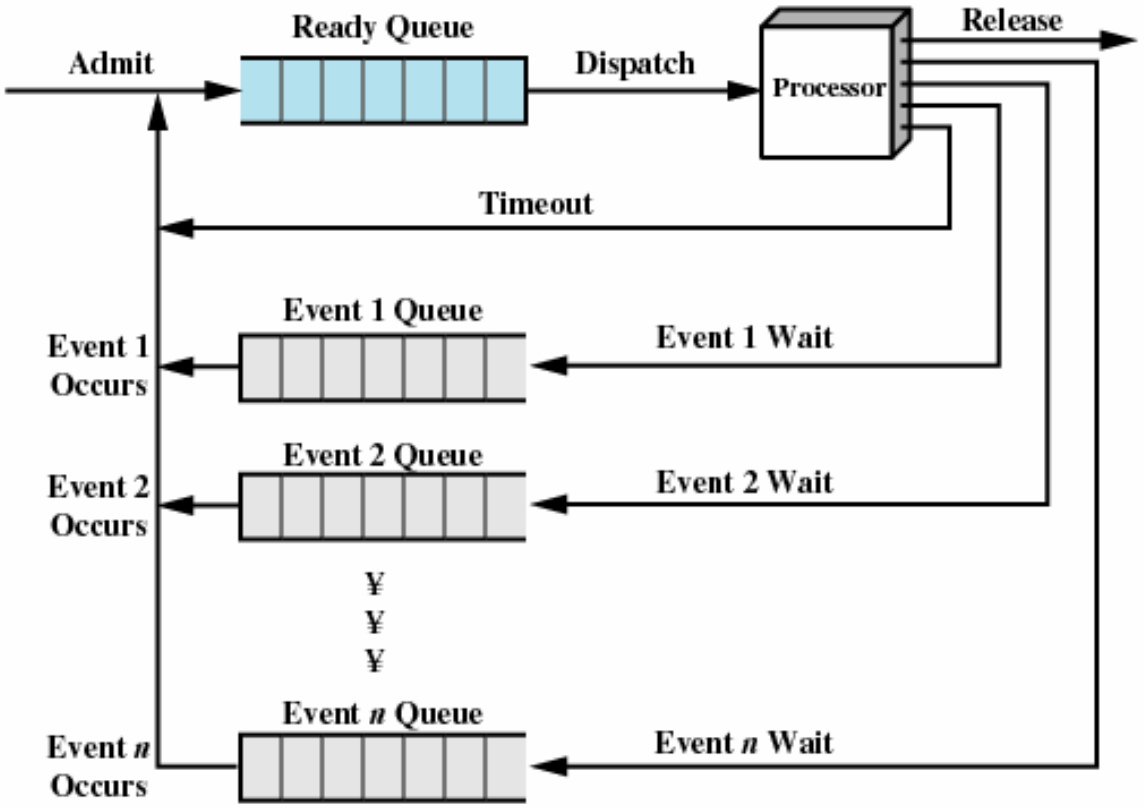
Pětistavový model vyžaduje použití minimálně dvou front



(a) Single blocked queue

Viz [1]

Použití více front procesů



(b) Multiple blocked queues

Figure 3.8 Queuing Model for Figure 3.6

Viz [1]

Příklad pseudoparalelního běhu tří procesů A, B, C

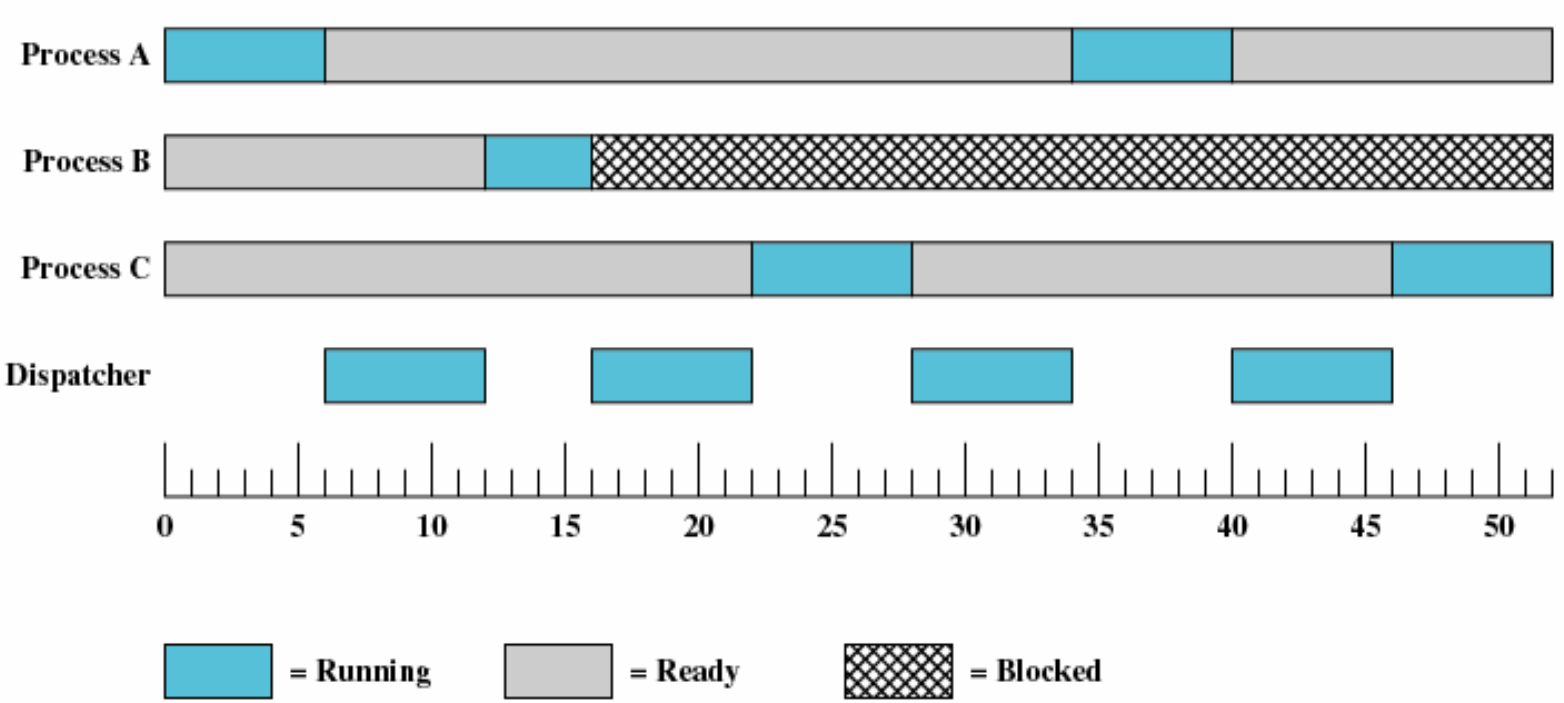


Figure 3.7 Process States for Trace of Figure 3.4

Viz [1]

Příklad

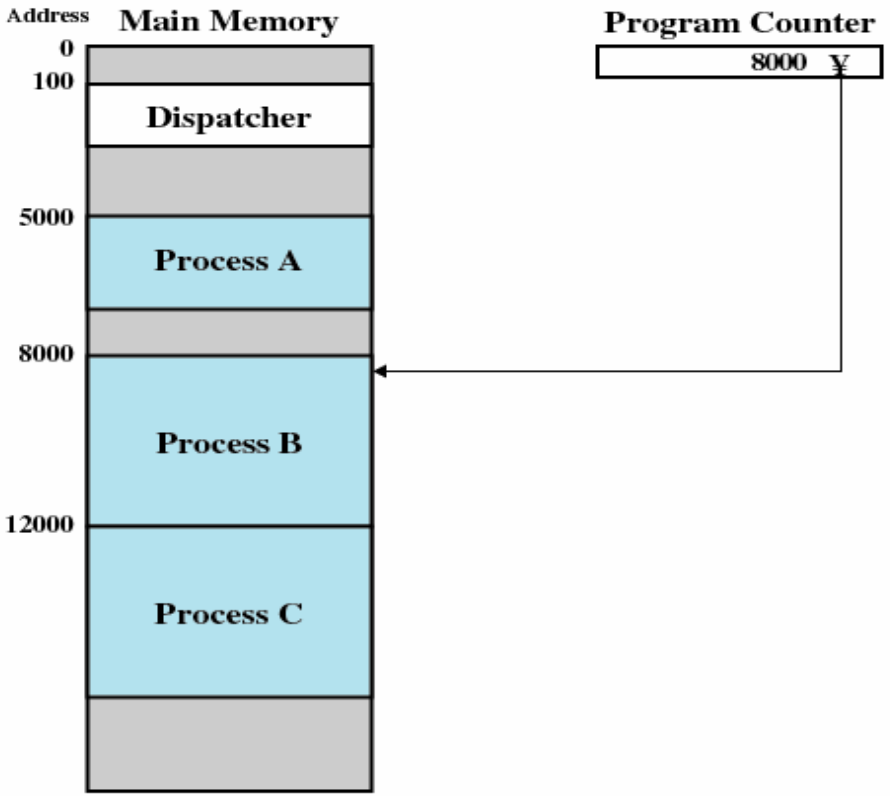


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

Viz [1]

Příklad

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.3 Traces of Processes of Figure 3.2

Viz [1]

Příklad

1	5000
2	5001
3	5002
4	5003
5	5004
6	5005
----------	Time out
7	100
8	101
9	102
10	103
11	104
12	105
13	8000
14	8001
15	8002
16	8003
----------	I/O request
17	100
18	101
19	102
20	103
21	104
22	105
23	12000
24	12001
25	12002
26	12003
27	12004
28	12005
----------	Time out
29	100
30	101
31	102
32	103
33	104
34	105
35	5006
36	5007
37	5008
38	5009
39	5010
40	5011
----------	Time out
41	100
42	101
43	102
44	103
45	104
46	105
47	12006
48	12007
49	12008
50	12009
51	12010
52	12011
----------	Time out

100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.4 Combined Trace of Processes of Figure 3.2

Viz [1]

Vlákna (*Threads*)

Vlákno (*thread*) – abstrakce „toku programu“

Tradiční OS (UNIX, MS DOS):

jeden proces obsahuje jedno vlákno = jednovláknový proces (*single-threaded process*)

Moderní OS umožňují:

- spuštění více současných úloh v rámci jednoho procesu = vícevláknový proces (*multi-threaded process*).
- po startu procesu běží jediné tzv. **primární vlákno**
- primární vlákno může vytvářet vlákna sekundární.

Vlákna (*Threads*) - pokračování

Vlákna:

- sdílejí privátní adresový prostor procesu (= mají společný kontext paměti), tzn. mohou přistupovat ke globálním proměnným
- sdílejí systémové prostředky procesu (kontext prostředí)
- každé vlákno má vlastní kontext procesoru a zásobník (včetně lokálních proměnných)

Procesy a vlákna (*Threads*)

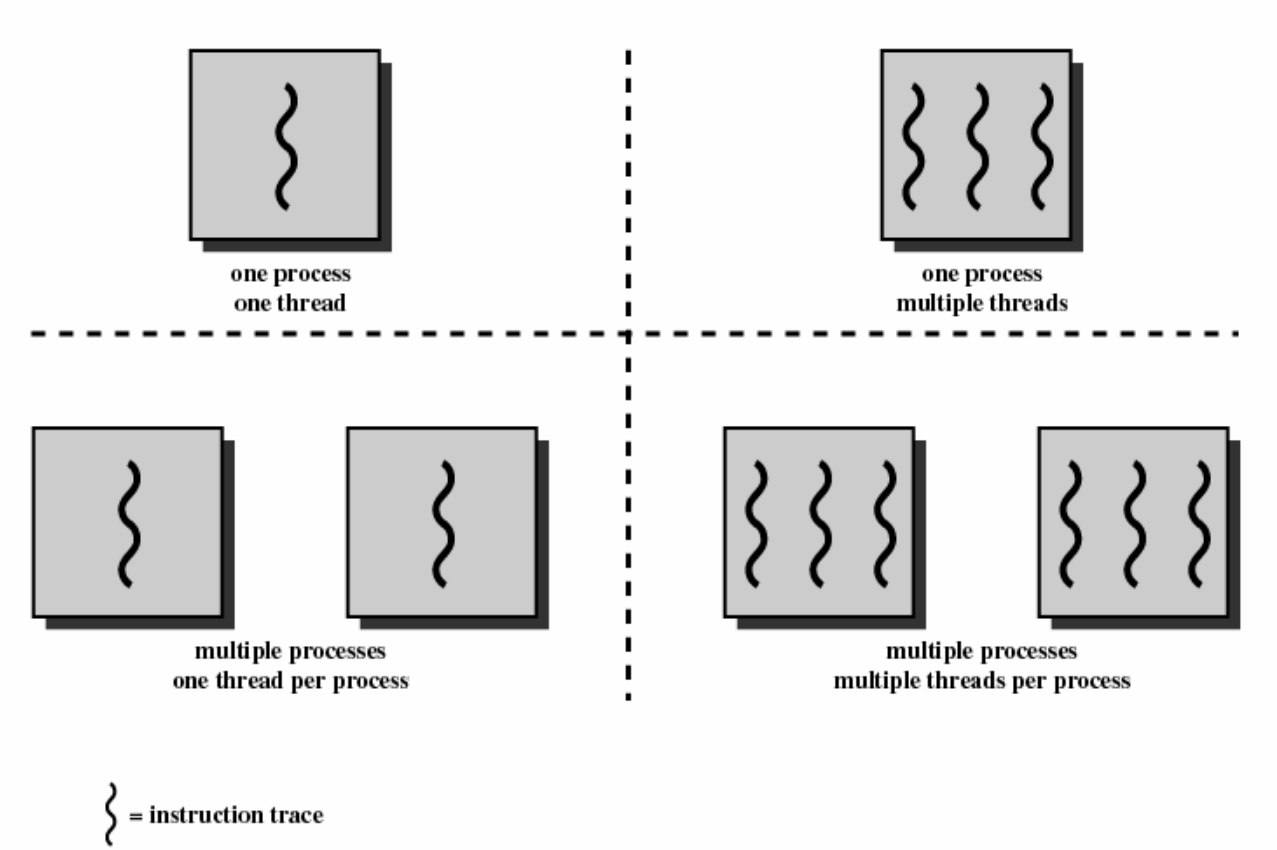


Figure 4.1 Threads and Processes [ANDE97]

Viz [1]

Procesy a vlákna (*Threads*)

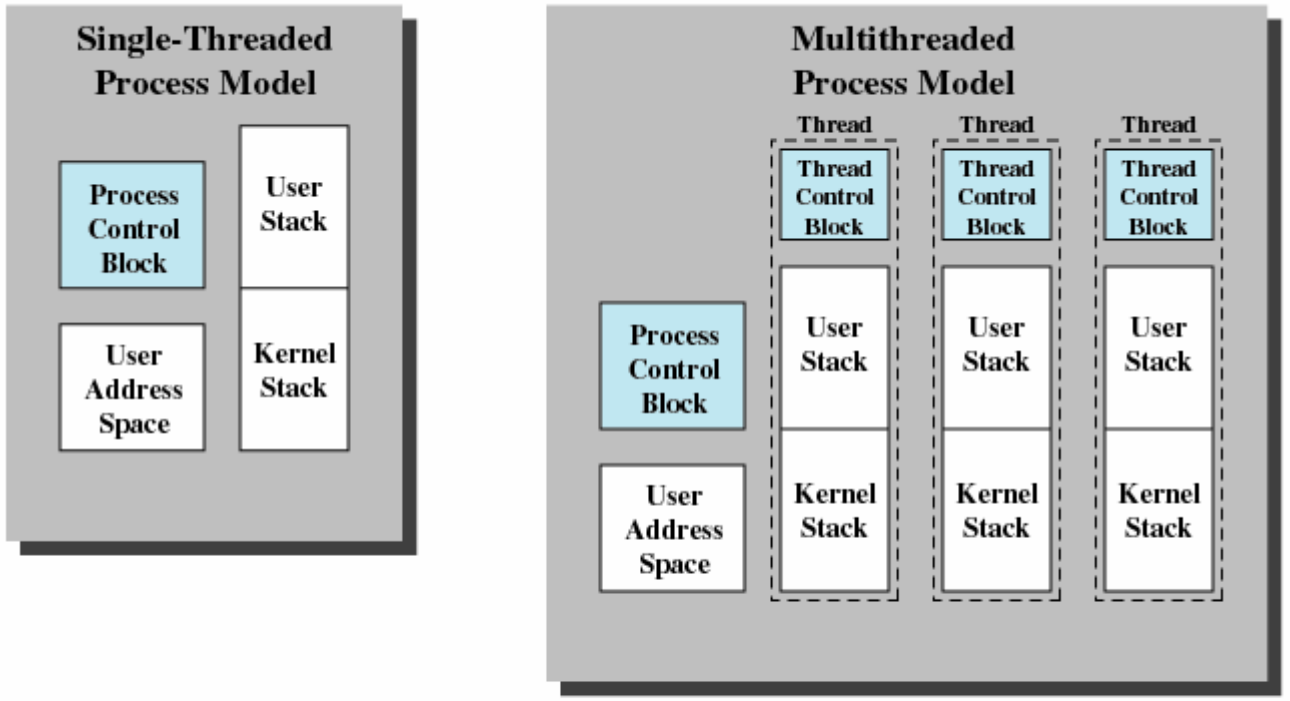


Figure 4.2 Single Threaded and Multithreaded Process Models

Viz [1]

Výhod vláken

- Vícevláknový proces není blokován při zablokování jednoho z vláken (např. při čekání na událost)
- Paměť a systémové prostředky jsou sdíleny (na rozdíl od procesů), zvyšuje se efektivita jejich využití, zjednodušuje se přenos dat mezi vlákny (např. použití globálních proměnných ale POZOR na kritické sekce!!! v programu)
- Přepínání kontextu vláken je výrazně rychlejší než přepínání kontextu u procesů
- Efektivní provádění pomalých I/O operací (I/O operace se mohou překrývat s výpočetními operacemi)
- Výrazné zrychlení programu v případě multiprocesorové architektury (= počítačový systém obsahuje několik procesorů nebo vícejádrové procesory); vlákna běží skutečně paralelně (NE pseudoparalelně!!!)
- Lepší strukturování programu (Ale s rozumem!!! Nepřehánět počet vláken, zvyšuje časovou režii při přepínání kontextu)

Implementace vláken - ULT

Přepínání kontextu vláken: na úrovni uživatele (a), jádra (b) nebo (c) kombinované.

(a) Na uživatelské úrovni (*ULT User-level Thread*)

Na úrovni aplikačního procesu (*User Mode*) musí být implementována speciální knihovna pro správu vláken (*thread library*)

Výhody:

- Nezávislost na podpoře jádra
- Rychlejší přepínání kontextu a vytváření nových vláken
- Plná kontrola procesu nad správou vláken ?!

Neýhody:

- volání služby (blokujícího volání jádra) jedním vláknem blokuje všechna vlákna procesu
- nutnost dodatečného programování (řízení vláken programátorem)
- Vlákna jednoho procesu nemohou běžet na více procesorech

Implementace vláken - KLT

(b) Na úrovni jádra (*KLT Kernel-level Thread*)

Výhody:

- Systémová volání neblokují ostatní vlákna téhož procesu
- Vlákna jednoho procesu mohou běžet na více procesorech
- Programy jádra mohou být vícevláknová

Neýhody:

- Správa vláken je nákladnější (časově náročnější) než u čistě uživatelských vláken

Příklady:

- MS Windows NT (včetně XP, Vista, 2007)
- Linux.4 BSD Unix

Implementace vláken – ULT + KLT

(c) Kombinace ULT+KLT

Některé OS podporují oba způsoby běhu vláken. Uživatelská vlákna se k systémovým přiřazují automaticky nebo programátor specifikuje uživatelská vlákna jako ULT nebo KLT.

Příklady:

- WAX
- Windows 2000/XP s nadstavbou ThreadFiber
- FreeBSD 5.x

ÚVOD DO OPERAČNÍCH SYSTÉMŮ

KONEC 3. přednášky

České vysoké učení technické Fakulta elektrotechnická