



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

**MŠMT**  
MINISTERSTVO ŠKOLSTVÍ,  
MLÁDEŽE A TĚLOVÝCHOVY

# Operační systémy

## Úvod

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

# Struktura počítačového systému

## ❑ Uživatel

- Lidé, stroje, jiné počítače

## ❑ Aplikace (SW)

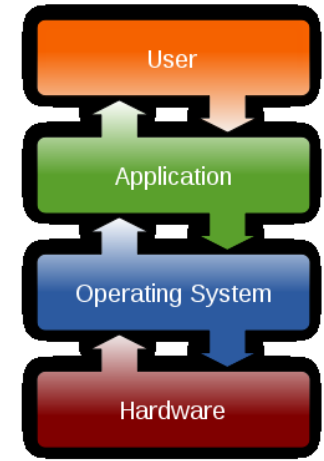
- Programy, které využívají systémové zdroje počítače k vyřešení zadání uživatelů
  - Vývojové nástroje, kancelářské aplikace, webový prohlížeč, hry, ...

## ❑ Operační systém (OS)

- Řídí a koordinuje efektivní využití HW a poskytuje rozhraní pro SW.

## ❑ Hardware (HW)

- Systémové zdroje
  - CPU, operační paměť, úložiště, další vstupní a výstupní zařízení



(Silberschatz, Galvin & Gagne, 2013)

(Silberschatz, Galvin & Gagne, 2013; Staudek, 2013)

# Co znamená operační systém?

- V šedesátých letech byl definován jako SW, který řídí HW
  - Dnes je nutno použít obsáhlejší definici
  - Neexistuje žádná obecně platná
  
- Odděluje aplikace od hardwaru
- Jedná se o softwarovou vrstvu
  - Spravuje hardware
  - Poskytuje služby softwaru

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

# Definice operačního systému

- ❑ Operační systém je **správce zdrojů**
  - Spravuje všechny systémové zdroje (CPU, paměť, I/O zařízení, ...)
  - Řeší konflikty požadavků
    - cílem je efektivní a spravedlivé využívání systémových zdrojů
- ❑ Poskytuje **abstrakci systémových zdrojů**
  - OS skrývá HW před SW (izolace)– vytváří virtuální HW (Virtual Machine)
- ❑ Operační systém řídí provádění programů
  - Zabraňuje chybám a nesprávnému využití

(Deitel, Deitel & Choffness, 2004; Tanenbaum, 2015; Silberschatz, Galvin & Gagne, 2013)

# Proč studujeme operační systém?

- ❑ Jedná se o nejsložitější SW systémy
- ❑ Uplatňují se zde různé oblasti vědění
  - Algoritmizace, programování, správa zdrojů, souběžnost, krizové rozhodování
    - Vývoj lze následně uplatnit i v jiné oblasti.
- ❑ Příliš jednoduchý operační systém je nákladný
  - Nevyužívá systémové zdroje efektivně, neposkytuje vše dostupné

# Náš cíl v operačních systémech

- Hlavním cílem je porozumět
  - Pochopit roli operačního systému
    - Co operační systém vlastně dělá
  - Pochopit jak operační systém pracuje
    - Design operačního systému
    - Principy funkce, služby, rozhraní
  - Naučit se používat operační systém
    - Microsoft Windows, Linux

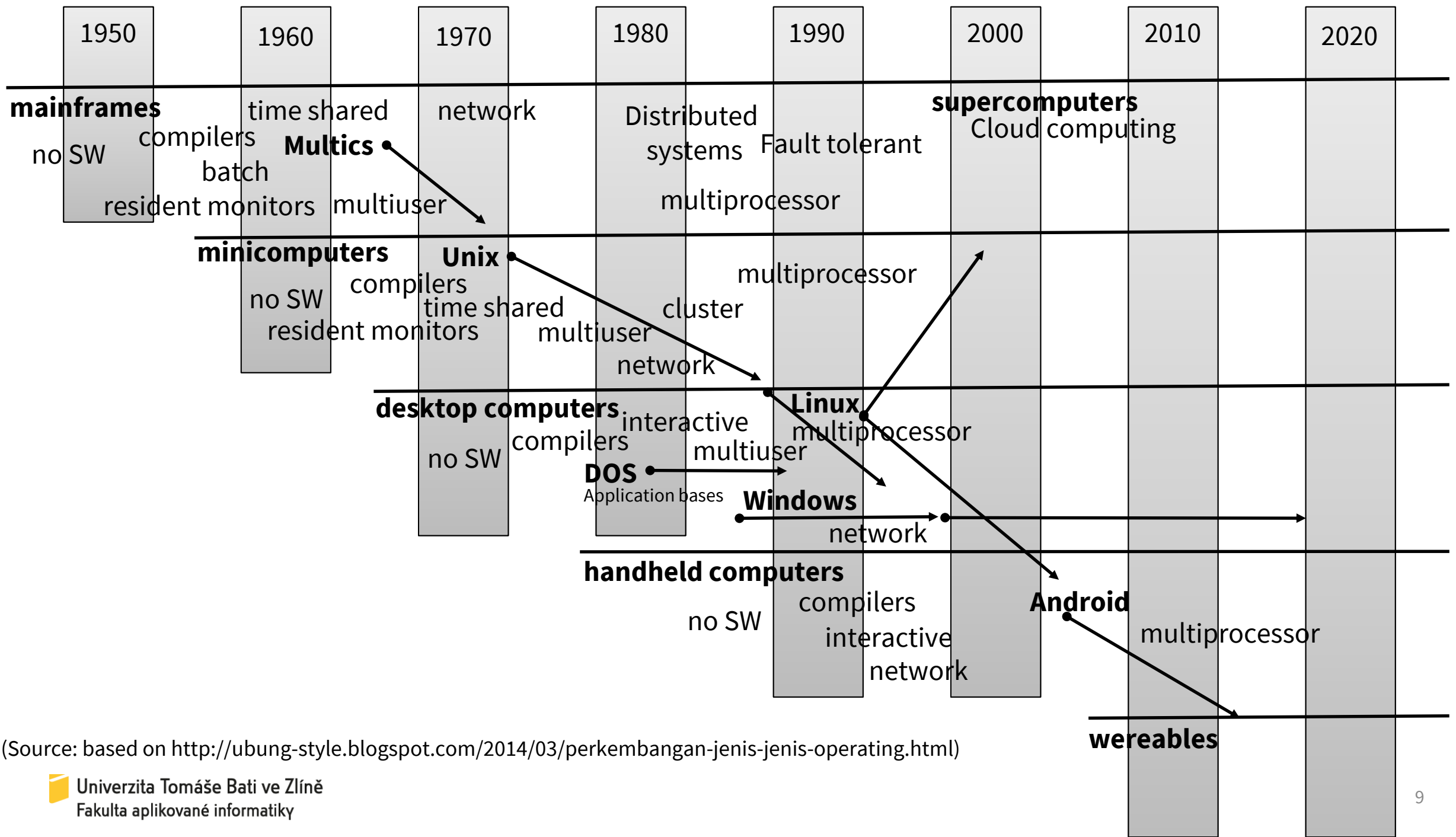
# Co všechno se naučíte

- ❑ Struktura operačního systému
  - Vnitřní návrh, procesy, vlákna, systémová volání
- ❑ Správa procesů
  - Plánování procesů a vláken
- ❑ Komunikace mezi procesy
  - Problémy synchronizace, mutexy, semaforey, ...
- ❑ Správa paměti
  - Adresní prostor, virtuální paměť
- ❑ Úložiště
  - Souborové systémy

# Historie

- ❑ 40. a 50. léta 20. století – první generace
  - První počítače neměly operační systém, vykonávala se vždy jedna úloha. V 50. letech se začalo využívat dávkové zpracování.
- ❑ 60. léta – druhá generace
  - Stále se používá dávkové zpracování, začíná multiprogramming a zpracování více úloh v čase, více uživatelů, interaktivita, virtuální paměť.
- ❑ 70. léta – třetí generace
  - Rozvoj operačních systémů, multitasking, zpracování v reálném čase,
- ❑ 80. léta – čtvrtá generace
  - Osobní počítače, GUI
- ❑ Pátá generace
  - Stále ještě čekáme





(Source: based on <http://ubung-style.blogspot.com/2014/03/perkembangan-jenis-jenis-operating.html>)

# Dělení operačních systémů

- ❑ Podle počtu procesorů
- ❑ Podle úrovně sdílení CPU
- ❑ Podle počtu uživatelů
- ❑ Podle způsobu nasazení/požadavků na odezvu
- ❑ Podle velikosti hardwaru
- ❑ Podle míry distribuovanosti
- ❑ Podle funkcí
- ❑ ...

# Cíle OS

- ❑ Efektivita
- ❑ Robustnost
- ❑ Škálovatelnost (nové zdroje)
- ❑ Rozšiřitelnost (nové technologie)
- ❑ Přenositelnost
- ❑ Bezpečnost
- ❑ Interaktivita
- ❑ Použitelnost

(Deitel, Deitel & Choffness, 2004)



# Části OS

- ❑ **Správa procesorů a procesů**
- ❑ **Správa hlavní paměti**
  - Alokace a dealokace paměti podle potřeby
  - Udržuje informaci, která část paměti je používána a kým
- ❑ **I/O podsystém, vstupy a výstupy**
  - Společné rozhraní ovladačů zařízení, ovladače pro specifická zařízení
  - Rozvrh diskových operací
  - Souborový systém – soubory a adresáře
- ❑ **Uživatelské rozhraní**
  - CLI (Command Line Interface)
  - GUI (Graphical User Interface)
- ❑ **Síť (Network)**

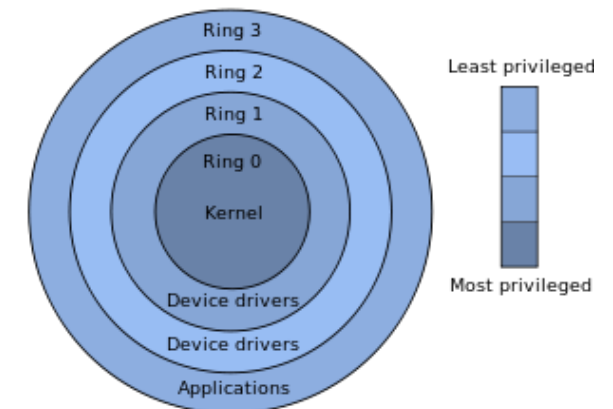
(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013)

**Ochrana a bezpečnost**  
– přístup ke zdrojům pouze pro autorizované procesy, Specifikace přístupu, Mechanismus ochrany (paměti, souborů, ...)



# Architektura OS

- ❑ Obecná architektura operačních systémů je budovaná s využitím dříve zmíněných základních komponent.
  - Základní rozdíly spočívají v tom, kde běží dané subsystémy a jak spolu navzájem komunikují, jaká mají oprávnění.
  - Pomáhá to zvládnout složitost OS.
- ❑ Systém běží ve dvou základních režimech
  - Uživatelský režim
  - Privilegovaný režim (kernel mode)



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

# Architektura OS

- ❑ Monolitická architektura
- ❑ Mikrokernel architektura
- ❑ Hybridní architektura
- ❑ Vrstvená architektura
- ❑ Virtuální stroj
- ❑ Model klient-server

(Deitel, Deitel & Choffness, 2004; Silberschatz, Galvin & Gagne, 2013, Lažanský, 2014)

# Architektura OS

## ❑ Monolitický operační systém (jádro)

- Všechny části OS jsou obsaženy v kernelu
  - přímo komunikují s ostatními.
  - Mají stejná oprávnění.
  - Nízká režie, důraz na efektivitu.
  - Možnost použití modulárního designu. Vše běží v paměťovém prostoru jádra.
- Kernel spuštěn s neomezeným přístupem k celému systému.
- Uživatelské programy mají minimální oprávnění.
- Většina starších operačních systémů (OS/360, VMS, Linux).

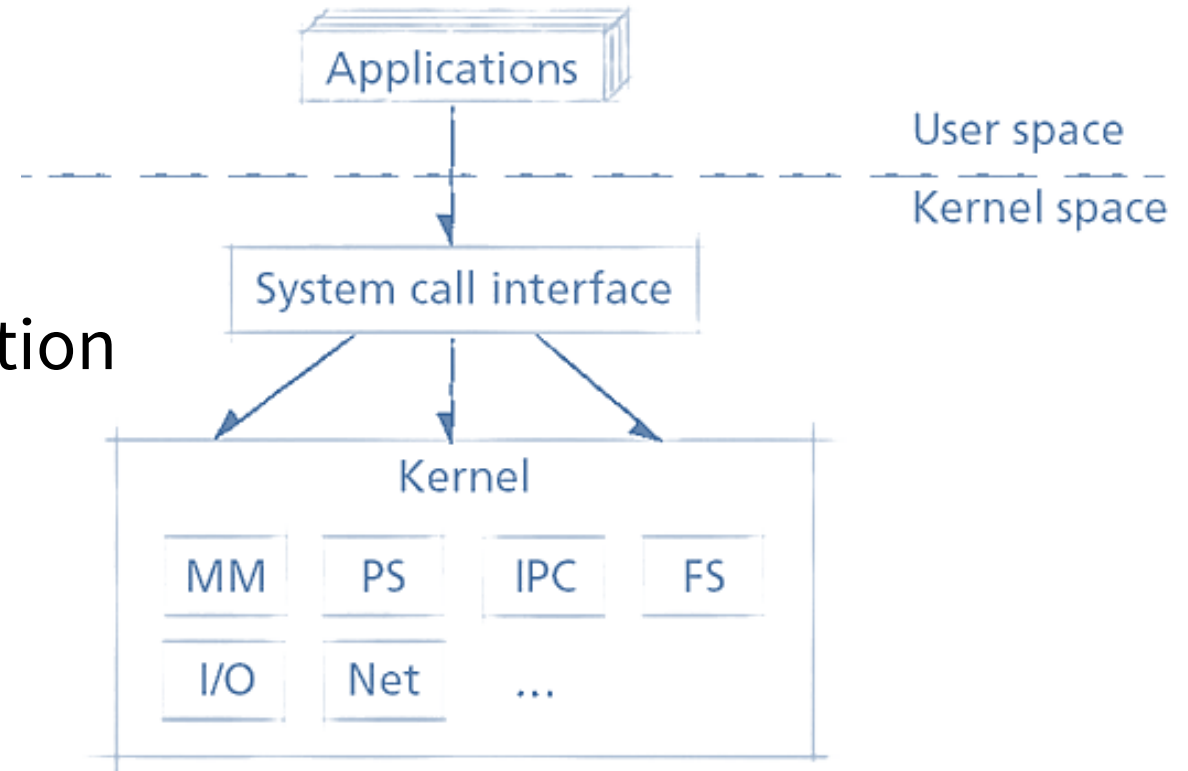
(Deitel, Deitel & Choffness, 2004)



# Architektura OS

## ❑ Monolitický operační systém

- MM      Memory management
- PS      Process scheduler
- IPC      Interprocess communication
- FS      File systém
- I/O      Input/output manager
- Net      Network manager



(Deitel, Deitel & Choffness, 2004)

# Architektura OS

## ❑ Mikrokernel operační systém

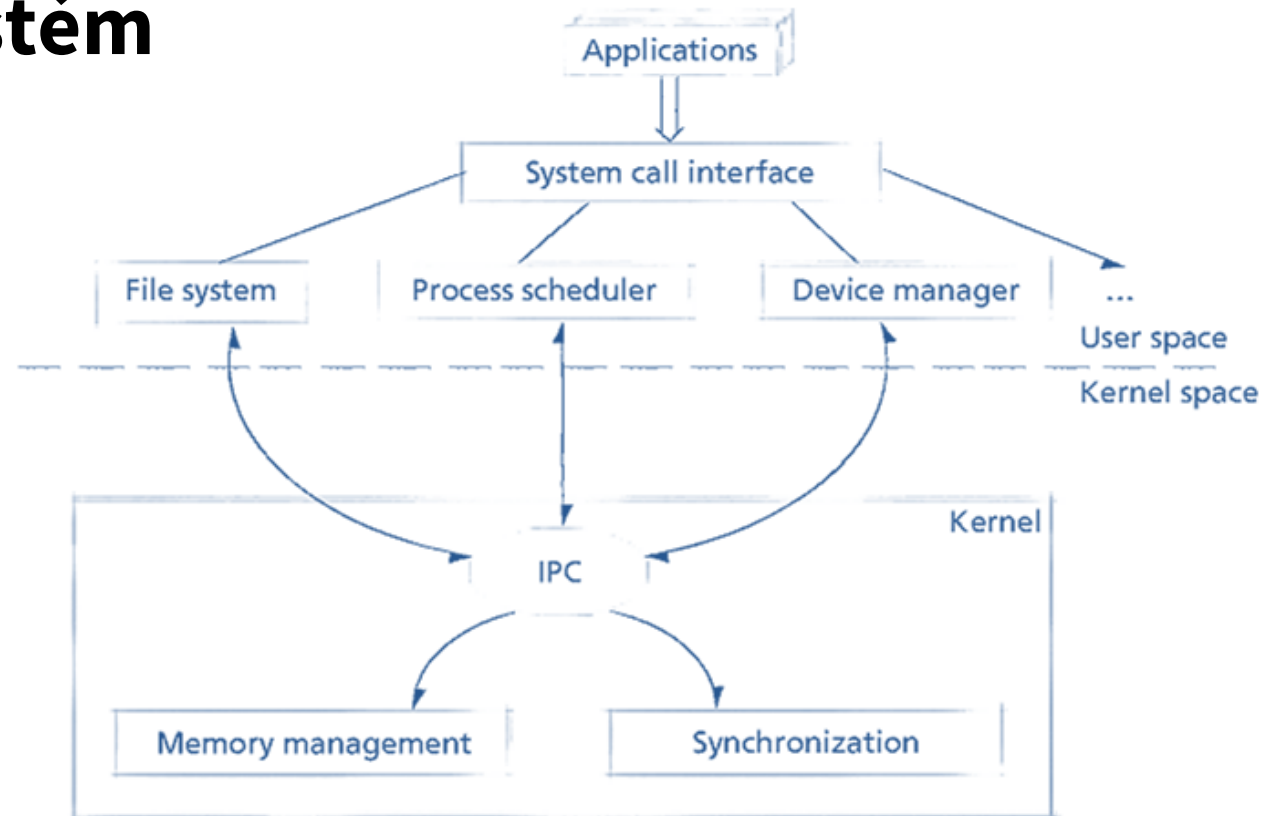
- Poskytuje základní služby v pokusu udržet jádro malé a škálovatelné.
  - Základní synchronizace procesů potřebnou pro spolupráci mezi procesy
  - Nízko úrovněvý paměťový management
  - Obsluha přerušení.
  - Komunikace mezi procesy
    - Při komunikaci je zajištěna autorizace.
- Další části OS spuštěny mimo jádro OS s menší úrovní práv
- OS Mach, GNU Hurd, MS Windows, ...

(Deitel, Deitel & Choffness, 2004)

# Architektura OS

## ❑ Mikrokernel operační systém

- Velká modularita
- Vyšší úroveň komunikace
  - Může degradovat výkon



(Deitel, Deitel & Choffness, 2004)

# Architektura OS

## ❑ Hybridní operační systém

- Hybridní jádra jsou něco mezi monolitickým jádrem a mikrojádretem.
  - Hybridní jádro se snaží zkombinovat rychlost a jednoduchost designu monolitického jádra s bezpečnostními výhodami mikrojader.
  - Některé služby (jako souborový systém nebo implementace síťového protokolu) běží v jádře ke zredukování režie proti mikrojádrům, ale jiné části monolitického jádra (ovladače zařízení) běží jako server v uživatelském prostoru.
- MS WinNT

# Architektura OS

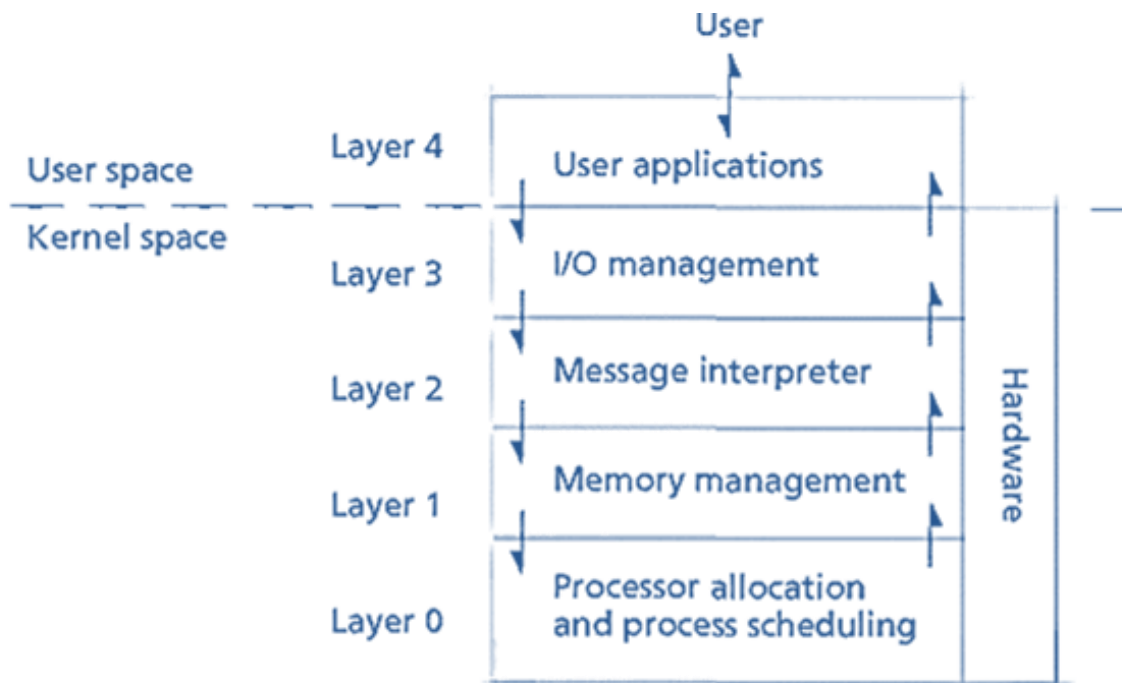
## □ Vrstvený operační systém

- Vytvořena hierarchie procesů.
  - Nejnižše položené vrstvy komunikují s hardwarem, každá další vyšší vrstva poskytuje abstraktnější virtuální stroj.
  - Jednotlivé vrstvy nelze obcházet, každá vrstva komunikuje výhradně se sousední vrstvou.
  - Výhodou je možnost výstavby systému od nejnižších vrstev, modularita.
  - OS/2, OS „THE“ (Technische Hogeschool Eindhoven)
  - Mnoho dnešních systémů obsahuje prvky vrstvené architektury

(Deitel, Deitel & Choffness, 2004)

# Architektura OS

## □ Vrstvený operační systém

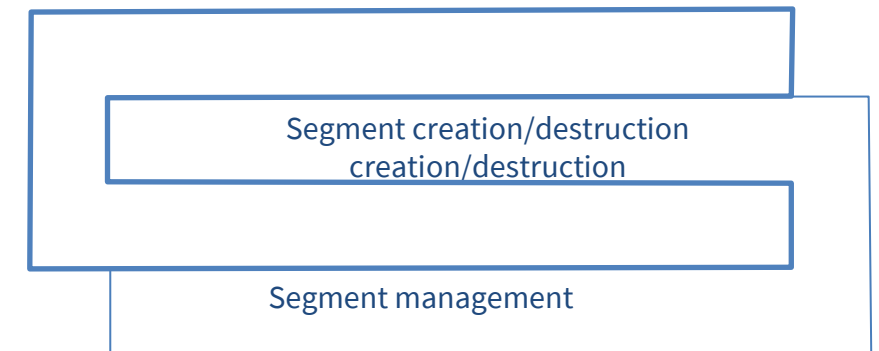


(Deitel, Deitel & Choffness, 2004)

## □ Funkční hierarchie

- obtížné rozčlenit OS do striktní hierarchie
  - vznikají závislosti

### Process management



### Memory management

# Architektura OS

## ❑ Virtuální stroj

- Logická struktura OS s principem vrstvení dotaženým do extrému
- Virtuální stroj je softwarový produkt, který chápe hardware a jádro operačního systému jako jednu společnou (hardwarovou) vrstvu
- Virtuální stroj vytváří duplikát původního hardware
- Současný běh více OS s vlastnostmi původního fyzického počítače
  - Každý uživatel na sdíleném stroji může tak užívat jiný OS
- Virtuální stroj zajišťuje úplnou ochranu systémových zdrojů
  - Každý virtuální stroj je izolován od ostatních

(Lažanský, 2014)

# Architektura OS

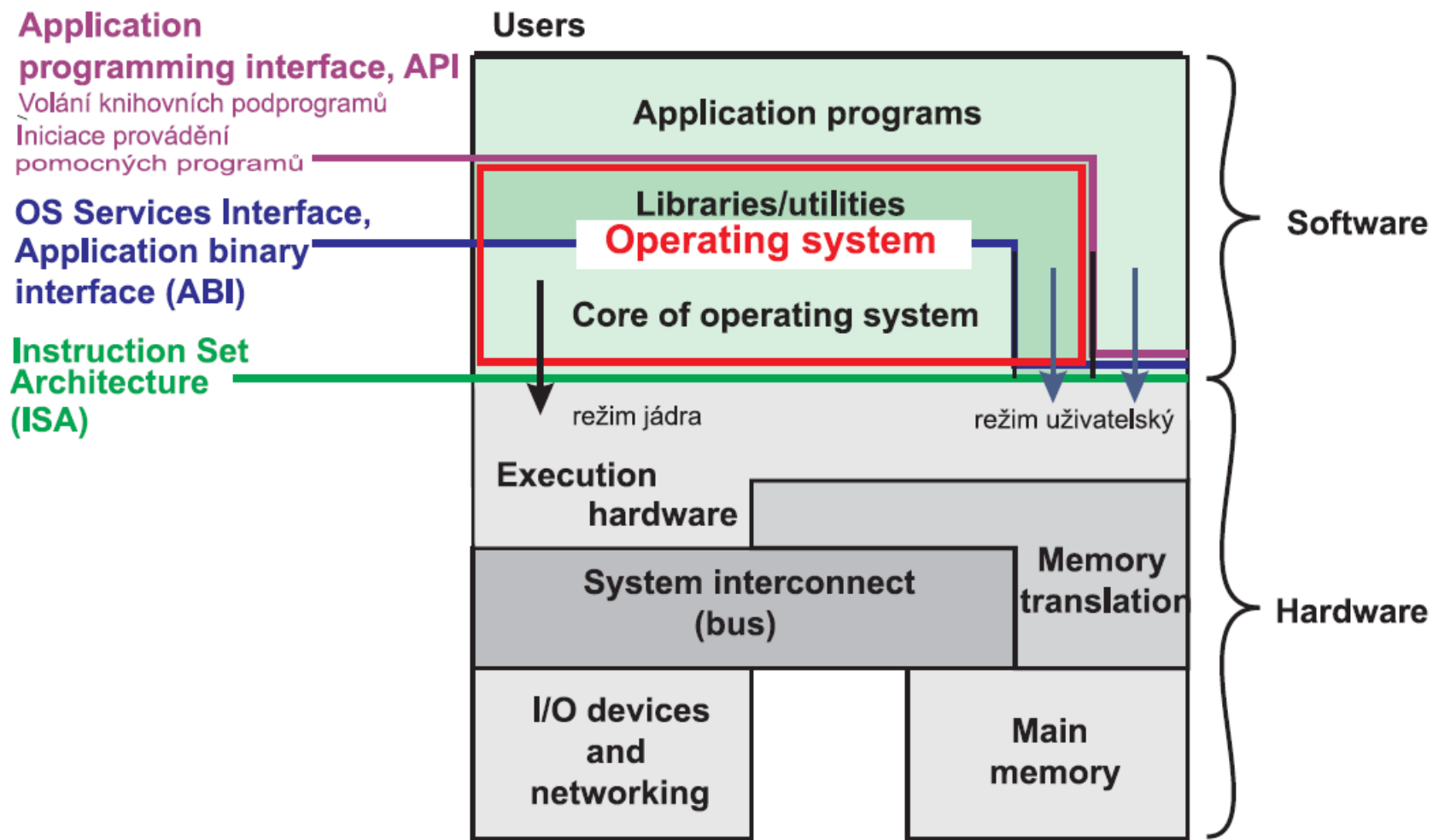
## ❑ Model klient-server

- Minimalizace rozsahu jádra (mikrojádru)
- Implementace funkcí OS jako uživatelský proces
- Požadavek na službu (např. čtení ze souboru) zasílá uživatelský proces (klient) na obsluhu souborů (file server).
  - Mikrojádru zajišťuje zejména komunikaci
- Vhodné pro distribuované OS

(Lažanský, 2014)



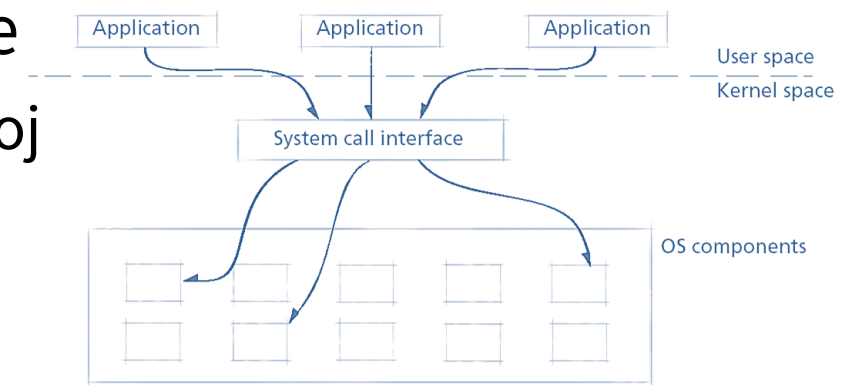
# Struktura HW a SW



(Staudek, 2013)

# Základ běhu aplikací

- ❑ Operační systém poskytuje API
  - Application Programming Interface
  - Specifikuje sadu funkcí, které jsou dostupné aplikačnímu programátorovi.
    - Programátor využívá knihovny pro přístup k API
  - Zjednodušuje přístup k HW pomocí abstrakce
    - Zabraňuje přístupu k HW a poskytuje virtuální stroj
  - API poskytuje **systemová volání**
    - Aplikace žádá operační systém o provedení akce.






(Deitel, Deitel & Choffness, 2004)

# Příklad API

- ❑ `read()` funkce popsaná v manuálových stránkách (`man read`)

```
#include <unistd.h>
```

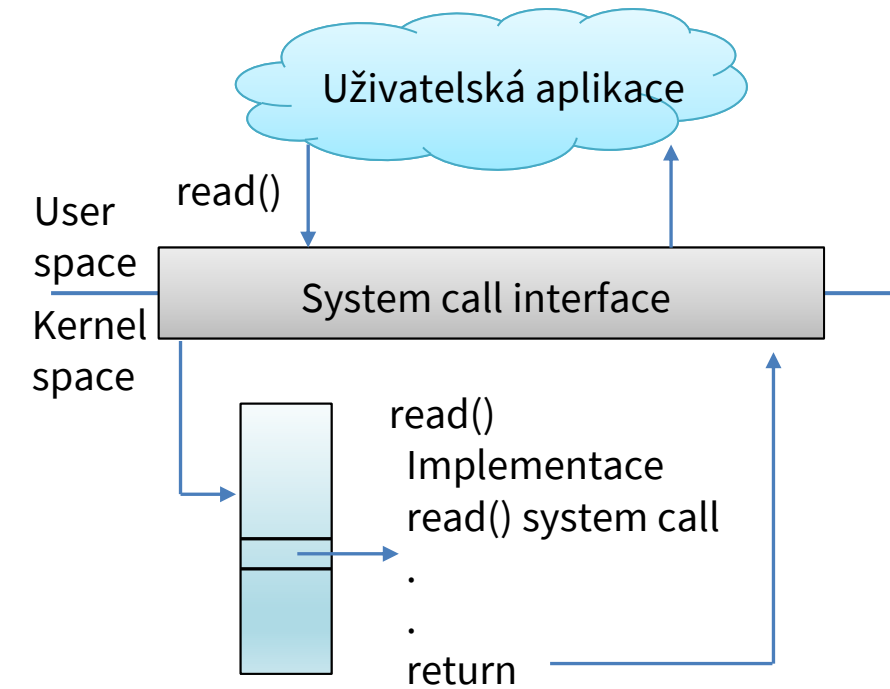
<code>ssize_t</code>	<code>read</code>	<code>(int fd, void *buf, size_t count)</code>
		
return value	function name	parameters

- Funkce je deklarována v hlavičkovém souboru *unistd.h*
  - Vstupní parametry
    - *int fd* – deskriptor souboru; *void \*buf* – buffer pro načtení dat; *size\_t count* – max. velikost
  - Návratová hodnota
    - Nula znamená konec souboru, při chybě čtení vrací -1.

(Silberschatz, Galvin & Gagne, 2013)

# Systémové volání (System Call)

- ❑ Metody předání vstupních parametrů operačnímu systému
  - Nejjednodušší přístup je předat parametry pomocí registrů
    - Problém s počtem a případně velikostí registrů
  - Parametry jsou uloženy v bloku a tento blok paměti je předán pomocí registru
  - Parametry jsou umístěny do zásobníku (stack), kde si je operační systém vyzvedne.



(Silberschatz, Galvin & Gagne, 2013)

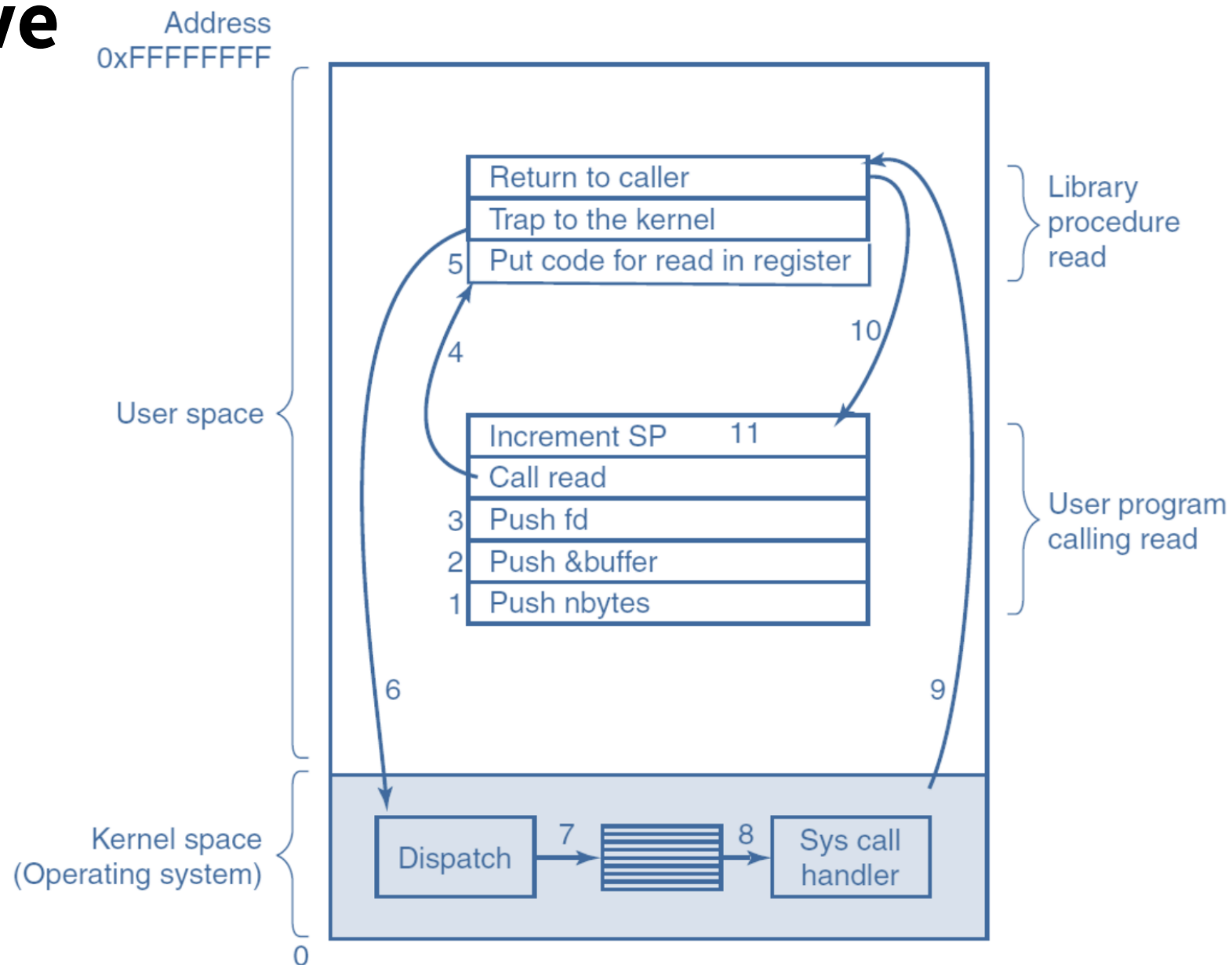
(Silberschatz, Galvin & Gagne, 2013)

# Zpracování systémového volání *read()*

- ❑ Vložení parametrů do zásobníku (1-3)
- ❑ Volání knihovní funkce (4)
- ❑ Vložení čísla systémového volání do registru (5)
- ❑ Provedení TRAP instrukce na přepnutí z uživatelského módu do kernel módu a začátek provádění kódu na fixní adrese v kernelu (6)
- ❑ Odeslání do odpovídající obsluhy systémového volání (7)
- ❑ Běh obsluhy systémového volání (8)
- ❑ Po dokončení je řízení předáno knihovně v uživatelském prostoru (9)
- ❑ Návrat do uživatelského programu, pomocí return (10)
- ❑ Vyčištění zásobníku po volání (11)

(Silberschatz, Galvin & Gagne, 2013)

# Systemové volání



# Application Binary Interface – ABI

- ❑ Definuje rozhraní na úrovni strojového kódu
  - V jakých registrech se předávají parametry
  - V jakém stavu je zásobník
  - Zarovnání vícebytových hodnot v paměti
- ❑ ABI se liší nejen mezi OS, ale i mezi procesorovými architekturami stejného OS.
  - Např: Linux i386, amd64, arm,...
  - Možnost podpory více ABI: int0x80, sysenter, 32/64bit

(Štěpán, 2018)

# Použitá a doporučená literatura

- ❑ DEITEL H. M., DEITEL P. J. & CHOFFNES D. R.: *Operating systems*. 3<sup>rd</sup> ed., Pearson/Prentice Hall, 2004. ISBN 0131246968.
- ❑ TANENBAUM A. S.: *Modern operating systems*. 4<sup>th</sup> ed. Boston: Pearson, 2015. ISBN 0-13-359162-x.
- ❑ SILBERSCHATZ A., GALVIN P. B. & GAGNE G.: *Operating system concepts*. 9<sup>th</sup> ed. Hoboken, NJ: Wiley, 2013. ISBN 978-1-118-06333-0.
- ❑ STALLINGS W.: *Operating Systems: Internals and Design Principles*. 8<sup>th</sup> ed., Pearson Education Limited, 2014.



# Použitá a doporučená literatura

- ❑ Štěpán Petr. *Operační systémy*. Přednášky FEL ČVUT v Praze, 2018.
- ❑ Lažanský Jiří. *Operační systémy a databáze*. Přednášky FELK ČVUT v Praze, 2014.
- ❑ Staudek Jan. *Operační systémy*. Přednášky FI MUNI, 2013.
- ❑ Vojnar Tomáš. *Operační systémy*. Přednášky FIT VUT v Brně, 2011.
- ❑ Klimeš Cyril. *Principy výstavby počítačů a operačních systémů*. On-line: <https://publi.cz/books/11/Cover.html>