



# PB001: Úvod do informačních technologií

**Luděk Matyska (Eva Hladká)**

podzim 2020



# Obsah přednášky

## Číselné soustavy

## Operační systémy

- Historie

- Účel

- Základní složky

  - Procesy

  - Paměť

  - Systém souborů



# Číselné soustavy

- Dle základu: dvojková, osmičková, šestnáctková, desítková
- Volně mezi sebou převoditelné (celá čísla bez ztráty přesnosti)
- Celá čísla a zlomky
- Reálná čísla
- První počítač v desítkové soustavě
- Konečné reprezentace

# Dvojková soustava

- Základem je číslo dvě
  - pouze dvě číslice – dva stavy
  - vhodná pro reprezentaci v číslicových systémech

## Dvojková soustava v počítači

- Celá čísla
- Záporná čísla
  - Přímý kód
  - Inverzní kód (binární negace)
  - Dvojkový doplňkový kód (binární negace + 1)
- Pro reálná čísla
  - Rozlišitelnost (nejmenší zobrazitelné číslo):  $X + \epsilon > X$   
a  $X + \epsilon/2 = X$
  - Přesnost (rozsah)
  - Zobrazení: mantisa  $m$  a exponent  $e$   
 $0 \leq m \leq 1 \wedge x = m \cdot 2^e$

## Záporná čísla – zobrazení

- Přímý kód
  - přidáno znaménko – znaménkový bit
  - Dvě nuly:  $+0$  a  $-0$  (**10** . . . **00**)
  - Rozsah:  $\langle -MAX, -0 \rangle$  a  $\langle +0, +MAX \rangle$
- Inverzní kód:
  - Přidáme znaménko
  - Dvě nuly:  $+0$  a  $-0$  (**11** . . . **11**)
  - Rozsah:  $\langle -MAX, -0 \rangle$  a  $\langle +0, +MAX \rangle$

## Záporná čísla – zobrazení

- Dvojkový doplňkový kód
  - Inverze bitu a přičtení jedničky
  - Pouze jedna nula (**11** . . . **11** je **-1**)
  - Nesymetrický rozsah:  $\langle -MAX - 1, -1 \rangle$  a  $\langle +0, +MAX \rangle$ 
    - skutečně používán v počítačích

## Rozsahy čísel

- Podle počtu bitů
  - Byte: 8 bitů, tj.  $\langle 0, 255 \rangle$  nebo  $\langle -128, 127 \rangle$
  - Půl slovo, 2 byte: 16 bitů, tj.  $\langle 0, 65535 \rangle$  nebo  $\langle -32768, 32767 \rangle$
  - Slovo, 4 byte: 32 bitů, tj. přibližně  $\langle -2 \cdot 10^9, 2 \cdot 10^9 \rangle$
  - Dvojslovo (nebo dlouhé slovo), 8 byte: 64 bitů, tj. přibližně  $\langle -9 \cdot 10^{18}, 9 \cdot 10^{18} \rangle$
  - Dnes již i „čtyřslovo“, 16 byte: 128 bitů, tj. přibližně  $\langle -1, 7 \cdot 10^{38}, 1, 7 \cdot 10^{38} \rangle$





# Racionální čísla

- Formát dle IEEE 754
- Součásti
  - Znaménko
  - Mantisa (přímý kód, normalizace)
  - Exponent (v kódu posunutě nuly)

## Racionální čísla

- Normalizace mantisy:
  - Nejvyšší bit vždy jedna: **1.aaaaaa**; **1** nezapisujeme
  - Nejmenší kladné číslo:  $1.0 \times 2^{-2^{n-1}+1}$  ( $2^{-127}$  pro  $n = 8$ )
  - Největší číslo:  $1.0 \times 2^{2^{n-1}}$  ( $2^{128}$ )
- Exponent (n bitů, dvojková soustava)
  - Přičteme  $2^{n-1} - 1$  (=127 pro  $n = 8$ ), abychom získali správnou hodnotu pro uložení
  - **00000000** je **-127**
  - **11111111** je **128**
- Zvláštní a nenormalizovaná čísla

## Racionální čísla II

- Rozsah zobrazení:  $\langle$ největší záporné, největší kladné $\rangle$
- Přesnost zobrazení: počet bitů mantisy + 1
- Rozlišitelnost: nejmenší nenulové číslo
  - Normalizované vs. nenormalizované ( $2^m$ krát menší,  $m$  počet bitů mantisy)



## Jiné soustavy

- Osmičková
  - $001\ 101\ 101\ 111_2 = 1557_8 = 879_{10}$
- Šestnáctková
  - $0011\ 0110\ 1111_2 = 36F_{16} = 879_{10}$
- Používány především pro hutný zápis binárních čísel

## OS jako příklad

- Operační systém je velmi komplikovaný programový produkt
- Vývoj odráží změny v informačních technologiích
  - programovací jazyky
  - softwarové inženýrství
  - vývoj hardware (kvalita, kapacita, ...)
  - vývoj periferií
- (Prakticky) každý se s ním potká
- Principy návrhu proto demonstrovány na operačních systémech a jejich komponentách

## Operační systémy – trocha historie

- Bootstrap loader
- Spooling
  - Nezávislé zavádění programu a jeho vykonávání
  - Vyžaduje DMA (Direct Memory Access)
  - Zavedlo *multiprogramování*
  - Stále zpracování *dávek* (batch processing)
- Timesharing
  - Virtualizace počítače/CPU
  - Zpracování *interaktivních* úloh
  - Souvisí se zavedení *disků* (Direct Access Storage Device, DASD od IBM, 60tá léta)

# Operační systémy: účel

- *Zkrásnění:*
  - Zjednodušení práce s počítačem
    - Práce s pamětí
    - Práce se soubory
    - Přístup k periferiím

# Operační systémy: účel

## *Sdílení a správa:*

- Zajistit sdílení (a správu) vzácných zdrojů
- Musí zajistit:
  - Aby to vůbec fungovalo
  - Aby to fungovalo účinně (využití, propustnost, rychlost odezvy)
  - Aby to fungovalo správně
    - Omezení následků chyb (avšak pozor na chyby v samotném operačním systému)
    - Oprávnění k přístupu (autentizace a autorizace)



## OS: základní složky

- Procesy a jejich správa
- Paměť a její správa
- Systém souborů
- Periferie a jejich správa
- Ochrana a bezpečnost

# Procesy

- Proces je abstrakce průchodu programem
  - Sekvenční model: program = 1 proces
  - Paralelní model: program  $>$  1 proces
- Proces má *interní stav*, charakterizovaný
  - programovým čítačem (program counter)
  - zásobníkem (volání funkcí a procedur)
  - vlastní paměť pro data

## Typy procesů

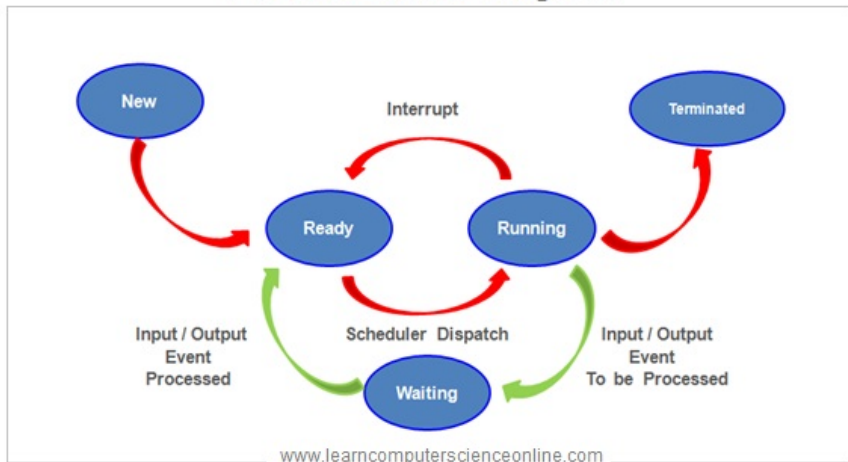
- Klasické (heavy-weight) procesy (např. UNIX)
  - Všechna data privátní
  - Sdílen pouze program (read-only)
- *Lehké* (light-weight) procesy či Vlákna (threads)
  - Minimum vlastní paměti
  - Většina dat sdílena

## Procesy detailněji

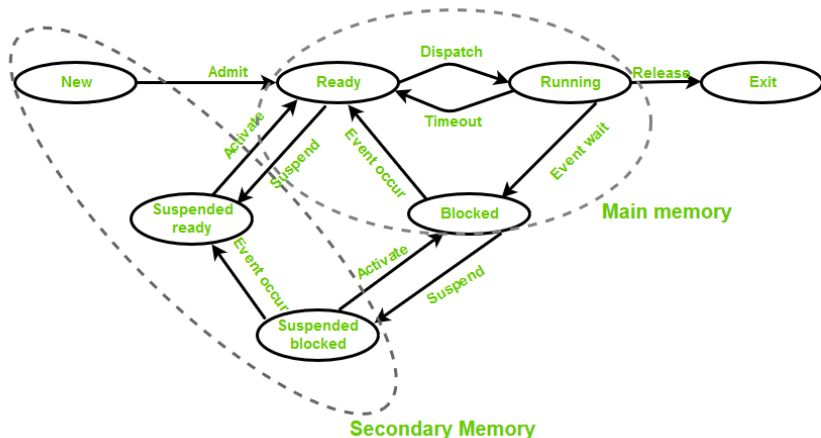
- Vytvoření procesu
  - `fork()` a jeho varianty
  - *Přesná* kopie původního procesu
  - *Rodič* a *potomek*
  - První proces v OS vytvářen jinak (`init` v Unixu)
- Stav
  - Start/vytvoření, připraven (`ready`), běží (`running`), je blokován (`čeká`), skončil

# OS: Stavy procesu

## Process Status Diagram



## OS: Stavy procesu (s externí pamětí)



## OS: problém časování

- Periferie výrazně pomalejší než procesor
- Příklad
  - 1 GHz Pentium IV:  **$1 \cdot 10^9$**  operací za sekundu
  - Běžný disk: 10 ms pro přečtení 1 byte
  - Poměr **1 : 10 000 000**
  - Stejně zpomalení člověka: 1 úhoz na klávesnici cca 20 dní.
- Možné řešení: prokládání I/O a výpočtu
  - Spust' diskovou operaci  
  Prováděj instrukce nad jinými daty  
  (alespoň 1~milión instrukcí)
  - Počkej na dokončení
  - Příliš těžkopádné a složité

## OS časování: jiné řešení

```
Proces 1 {  
Spust' diskovou operaci  
Počkej na dokončení  
Zpracuj získaná data  
}
```

```
Proces 2 {  
Nějaká jiná aplikace  
}
```

- Přehlednější
- OS musí „přepínat“ mezi procesy (*priorita*)



## OS: paměť

- Většina paměti nevyužita
  - Zpracování cyklu (zbytek programu)
  - Zpracování konkrétních dat (ostatní neaktivní)
  - Čekání na I/O
- *Virtualizace* paměti
  - Data a programy na disku
  - Do paměti *na žádost*
  - Umožňuje
    - Každý program má „celou“ paměť
    - Program může adresovat více jak rozsah fyzické paměti
- Ochrana paměti

# OS: Systém souborů

- Základní funkce:
  - Vytvoření souboru
  - Čtení a psaní z/do souboru
  - Odstranění (smazání) souboru
  - Spuštění souboru (soubor=program)
- Podpora na úrovni operačního systému

## Struktura systému souborů

- Hierarchické systémy:
  - Kořen (root)
  - Adresáře jako speciální typ (meta)souboru: drží informace o souborech, nikoliv jejich vlastní data
- Databázové systémy:
  - Soubory (jejich části resp. jejich metadata) jako položka v databázi
  - Bohatší množina operací
  - Složitější implementace

Apache Hadoop má prvky databázového systému souborů

- Hadoop Distributed File System
- Soubory jsou rozděleny na části, které jsou distribuovány na prvky clusteru
- Primární přístup k souborům přes nativní rozhraní HDFS

## Struktura souborů

- Posloupnost bytů – vnitřní struktura pro OS neznáma
- Posloupnost záznamů (records)
- Strom – každý uzel má vlastní klíč
  
- Výše uvedeny **příklady** struktury, ne všechny varianty

## Typ a přístup

- Typy souborů (v UNIXovém OS)
  - Řádné: běžné soubory
  - Adresáře: udržení hierarchické struktury
  - Speciální: přístup ke konkrétnímu zařízení (/dev/mouse, /dev/audio, /dev/lp); speciální /proc systém
  - Blokové: náhodný přístup na základní úrovni (/dev/hd, /dev/kmem)
- Přístupové metody; příklady:
  - Sekvenční
  - Náhodný (random)
  - Indexsekvenční (není v běžném UNIXu)

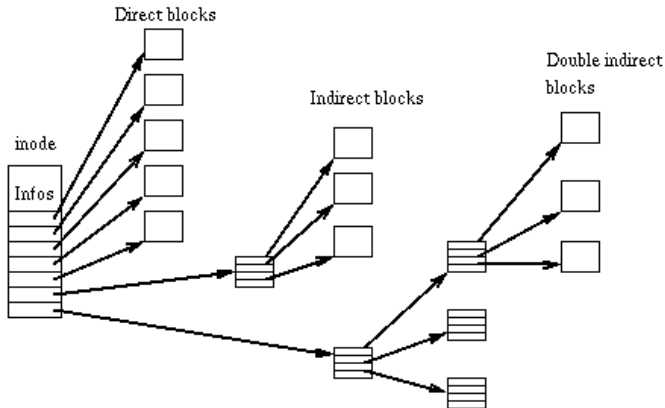
## Struktura na disku

- Možné typy
  - Souvislé
    - souvislé posloupnost bloků (složitá alokace, plýtvání místem)
  - Provázaný seznam:
    - každý blok odkazuje na další (může růst, vyšší režie – pro ukazatel, složitý náhodný přístup)
  - Indexové:
    - Např. FAT (File Allocation Table) v MS DOSu
    - Tabulka pro všechny bloky na disku
    - Provázaný odkazem na další blok daného souboru
  - inodes

## Struktura – inodes

- Podobné indexové organizaci
- Pevná délka tabulky pro každý soubor
  - Kratší soubory adresovány přímo
  - Pro delší soubory alokována další tabulka
  - Tabulky provázány hierarchicky (1., 2. a 3. úroveň)
- Flexibilní, malá režie

# inodes



Ext2-inodes



## Systémy souborů se žurnálováním

- Problém interní konzistence informací uvnitř systému souborů
  - co se stane při výpadku proudu či nečekaném zhroucení operačního systému
- Riziko nekonzistentních dat při postupném zápisu
  - část dat nebo metadat není ještě zapsána
- Klasické řešení: **fsck**
  - procházení všech datových struktur v systému souborů
  - nalezení a oprava nekonzistencí (zpravidla několik průchodů)
  - velmi pomalé pro velké systémy souborů
- Alternativa: zapsat zvlášť prováděné změny a teprve poté je skutečně realizovat
  - úspěšný zápis všech dat a metadat vede ke smazání údajů
- **journaled file system**
- V případě výpadku se použijí tato data na zajištění interní konzistence systému souborů

## Volné bloky

- V tabulce
- Bitový vektor
- Provázaný seznam
- Většinou zpracovávají podle FCFS (First Come First Served)

## Vyrovnávací paměť

- Obecně přístup pro skrytí *zpoždění* (latence)
- Nejčastěji používané bloky/soubory uloženy v paměti
- Pouze pro čtení (snazší) nebo i pro zápis
- Problém: konzistence při přístupech/zápisech z více míst
- Základní typy
  - Write-through: okamžitě po zápisu i na disk
  - Write-back: až po určité době (30 s)