

Úvod do informačních technologií

Luděk Matyska

FI MU

Základní podmínky

- Přednášky nejsou povinné
- Zhruba od listopadu budou na webu k dispozici slidy z přednášky
- Zkouška je pouze písemná
 - Termíny oznámím do konce října (2 řádné, jeden kombinovaný, jeden pouze opravný)
 - Jeden náhradní termín
 - Vzhledem ke kapacitním omezením doporučuji rané termíny
 - Prototypové zadání písemky dám k dispozici koncem listopadu
- Studijní literatura uvedena u sylabu
- Přednáška je nahrávána na video

Cíle přednášky

- Základní orientace v oblasti informačních technologií.
- Úvod do technického a programového vybavení současných počítačů a propojovacích sítí.
- Etické a sociální rozměry informačních technologií.
- Systémový (konceptuální) pohled na celou oblast.
- „Průvodce“ následujícím studiem

Dualita Informatiky

- Unikátní kombinace reálných a abstraktních (virtuálních) systémů
 - Technické komponenty (*hardware*): podléhají fyzikálním zákonům
 - Programy (*software*): „ztělesnění“ abstraktních konstrukcí

Dualita Informatiky

- Unikátní kombinace reálných a abstraktních (virtuálních) systémů
 - Technické komponenty (*hardware*): podléhají fyzikálním zákonům
 - Programy (*software*): „ztělesnění“ abstraktních konstrukcí
- Důsledky:
 - Virtuální prostředí
 - Pocit, že IT stojí mimo „realitu“
 - Fenomémem *vaporware*

Dekompozice

- Dva rozměry:
 - Od fyzické po programovou vrstvu
 - Různé komponenty na téže vrstvě

Dekompozice

- Dva rozměry:
 - Od fyzické po programovou vrstvu
 - Různé komponenty na téže vrstvě

- Příklady:
 - ISO OSI síťový model: např. linková, transportní a aplikační vrstva

Dekompozice

- Dva rozměry:
 - Od fyzické po programovou vrstvu
 - Různé komponenty na téže vrstvě
- Příklady:
 - ISO OSI síťový model: např. linková, transportní a aplikační vrstva
 - Vrstvy operačního systému: např. kernel, ovladače, překladače, aplikace

Dekompozice

- Dva rozměry:
 - Od fyzické po programovou vrstvu
 - Různé komponenty na téže vrstvě
- Příklady:
 - ISO OSI síťový model: např. linková, transportní a aplikační vrstva
 - Vrstvy operačního systému: např. kernel, ovladače, překladače, aplikace
 - Různé typy procesorů

Dekompozice

- Dva rozměry:
 - Od fyzické po programovou vrstvu
 - Různé komponenty na téže vrstvě
- Příklady:
 - ISO OSI síťový model: např. linková, transportní a aplikační vrstva
 - Vrstvy operačního systému: např. kernel, ovladače, překladače, aplikace
 - Různé typy procesorů
 - Různé programovací jazyky

Disciplíny

- Technické prostředky
 - Architektura počítačů a sítí

Disciplíny

- Technické prostředky
 - Architektura počítačů a sítí
- Programové prostředky
 - Operační systémy
 - Programovací jazyky
 - Aplikace

Technické prostředky – základní pojmy

- Procesor(–řadič)–paměť–periferie: von Neumannova architektura

Technické prostředky – základní pojmy

- Procesor(–řadič)–paměť–periferie: von Neumannova architektura
- Řízené zpracování *dat*

Technické prostředky – základní pojmy

- Procesor(–řadič)–paměť–periferie: von Neumannova architektura
- Řízené zpracování *dat*
- Jiné modely architektury:
 - Turingovy stroje
 - Dataflow přístup
 - Objektově-orientovaná
 - Deklarativní (funkcionální či logická)

Procesor

- Stroj vykonávající *instrukce*
- Vnitřní hodiny: takt procesoru
- Základní jednotka sekvenční (ALU)
- Může obsahovat více jednotek: vnitřní paralelismus
- Instrukční cyklus: výběr a provedení instrukce jednou jednotkou

Typy procesorů

- Univerzální
 - CISC: Complex Instruction Set Computer
 - RISC: Reduced Instruction Set Computer
 - ...
- Specializované
 - Vektorové
 - Embedded
 - ...

Paměť (vnitřní)

- Uchovává data
- Přímá adresovatelná: sloupec a řádek
 - Rozsah adres: 16, 32, 64, . . . bitů
- Cyklus paměti: doba nezbytná pro vystavení nebo zápis dat
- Vzpomínání se po provedené operaci, prokládání paměti
- Statická vs. dynamická paměť, volatilita
- Hierarchie paměti
 - Rychlá—pomalá
 - Drahá—levná

Paměť (vnitřní)

- Uchovává data
- Přímá adresovatelná: sloupec a řádek
 - Rozsah adres: 16, 32, 64, . . . bitů
- Cyklus paměti: doba nezbytná pro vystavení nebo zápis dat
- Vzpomínání se po provedené operaci, prokládání paměti
- Statická vs. dynamická paměť, volatilita
- Hierarchie paměti
 - Rychlá—pomalá
 - Drahá—levná
 - Ilustrace *ekonomického imperativu* v IT

Periferie

- Zajišťují vstup/výstup informací:
 - komunikace s uživatelem
 - permanentní ukládání dat
 - komunikace s jinými systémy

Komunikace s uživatelem

- Interaktivní
 - Klávesnice: vstup
 - Myš, tablet, stylus, . . . : vstup
 - Obrazovka: výstup i vstup
 - Zvuk: výstup i vstup
- Dávkové: prostřednictvím jiných zařízení

Permanentní ukládání dat

- Paměti (ROM, PROM, EPROM, NVRAM)
- Disky
 - Magnetické
 - Magnetooptické
 - Optické
- Pásy
- Síť

Permanentní ukládání dat

- Paměti (ROM, PROM, EPROM, NVRAM)
- Disky
 - Magnetické
 - Magnetooptické
 - Optické
- Pásky
- Sítě
- Papír: *trvanlivost!*

Komunikace

- Počítačové sítě
 - Drátové
 - * Elektrické
 - * Optické

Komunikace

- Počítačové sítě
 - Drátové
 - * Elektrické
 - * Optické
 - Bezdrátové
 - * Radiové vlny
 - * Optické

Komunikace

- Počítačové sítě
 - Drátové
 - * Elektrické
 - * Optické
 - Bezdrátové
 - * Radiové vlny
 - * Optické
 - * Akustické

Komunikace

- Počítačové sítě

- Drátové

- * Elektrické

- * Optické

- Bezdrátové

- * Radiové vlny

- * Optické

- * Akustické

- Mechanické

Speciální periferie

- Virtuální realita
 - Brýle a helmy
 - 3D projekce a prostorový zvuk
 - Haptika (rukavice, ...)
 - Detekce polohy a pohybu

Speciální periferie

- Virtuální realita
 - Brýle a helmy
 - 3D projekce a prostorový zvuk
 - Haptika (rukavice, ...)
 - Detekce polohy a pohybu
- Wearable computers

Co je to *počítač*?

- Standardní pohled:
 - Procesor(y)
 - Paměť
 - Periferie
- Možné i jiné pohledy
 - Buněčné automaty
 - Neuronové počítače
 - ...

Paralelní systémy

- Úzce propojené (tightly coupled)
- Volně propojené (loosely coupled)
- Distribuované
- Gridy

Úzce propojené systémy

- Často společná paměť
- Minimální vliv vzdálenosti procesorů
- Speciální propojení procesorů a pamětí
- Vhodné pro tzv. *jemný* paralelismus
- Typický výpočetní model: sdílená paměť (i kdyby byla pouze virtuální)

Volně propojené systémy

- Převážně distribuovaná paměť (každý procesor zvlášť)
- Vzdálenost procesorů může hrát roli
- Speciální propojení procesorů
- Výrazně vyšší latence (zpoždění) v meziprocessorové komunikaci ($\approx \mu\text{s}$ a méně)
- Existence operací `remote put` a `remote get` pro přístup do paměti vzdáleného procesoru
- Typický výpočetní model: zasílání zpráv

Distribuované systémy

- Rozšíření předchozího modelu
- Vždy distribuovaná paměť
- Vzdálenost procesorů hraje významnou roli
- Propojení procesorů často formou běžné LAN sítě
- Vysoká latence v meziprocessorové komunikaci (1 – 100 μ s)
- Typický výpočetní model: zasílání zpráv

Gridy

- Systém distribuovaný po geograficky rozsáhlých prostorech (země, kontinent, ...)
- Propojeny samostatné počítače (včetně paralelních)
- Propojení počítačů WAN sítí
- Extrémně vysoká latence v meziprocesorové komunikaci (desítky ms)
- Prakticky jediný výpočetní model: zasílání zpráv

Programové vybavení

- Nadstavba technických prostředků
- Vrstvy operačního systému:
 - Technické vybavení
 - Správa paměti
 - Správa procesů
 - Správa periferií (I/O)
 - Správa souborů dat
 - Uživatelský prostor (nepřesné)

Programové vybavení – jiný pohled

- Operační systém
 - UNIX, Linux, OS/370, MS Windows, . . .
- Programovací jazyky
 - C, Pascal, Ada, Occam, ML, Prolog, perl, python, Java, . . .
- Podpůrné nástroje
 - debuggery, profilery, . . .
- Aplikační programy

Programovací jazyky

- Rozlišujeme
 - Styl
 - Míru abstrakce
 - „Dialekt“

Programovací jazyky – styl

- Imperativní/Procedurální: C, Fortran
- Objektově orientované: Java, C++
- Deklarativní/Funkcionální: ML, Lisp, MIRANDA
- Deklarativní/Logické: Prolog, GHC
- S jediným přiřazením: SISAL
- Produkční systémy: OPS5
- Semantické sítě: NETL
- Neuronové sítě: SAIC ANSpec

Procedurální vs. deklarativní styl

```
fac := 1;
if n > 0 then
  for i:=1 to n do
    fac := i*fac;
```

```
|   fac(0)    := 1;
|   fac(n>0) := n*fac(n-1);
|
|-----
|
|   fac(0,1).
|   fac(N,F1*N) :- fac(N-1,F1).
```

Programovací jazyky – míra abstrakce

- Strojový jazyk: přímo kódy jednotlivých instrukcí
- Assembler: jména instrukcí, operandy, pojmenované cílové adresy skoků
- Vyšší jazyky: obecné konstrukty, tvoří „kontinuum“
 - Agregované datové typy
 - Cykly namísto skoků
 - Procedury a funkce
 - Procesy a vlákna

Programovací jazyky – implementace

- Překladače
 - Zdrojový kód–mezijazyk–strojový jazyk
 - Překlad a sestavení

Programovací jazyky – implementace

- Překladače

- Zdrojový kód–mezijazyk–strojový jazyk
- Překlad a sestavení

- Interprety

- Abstraktní počítač
- Vhodné pro složité operace (např. práce s texty, s maticemi a algebraickými objekty)

Programovací jazyky – implementace

■ Překladače

- Zdrojový kód–mezijazyk–strojový jazyk
- Překlad a sestavení

■ Interprety

- Abstraktní počítač
- Vhodné pro složité operace (např. práce s texty, s maticemi a algebraickými objekty)

■ Just-in-time překladače (nejen Java)

- Známy již od osmdesátých let (řešil se tak nedostatek paměti)

Výpočetní model

- Souvislost mezi *architekturou a jazyky*
 - von Neumannova architektura a imperativní jazyky
 - objektová architektura a objektově orientované jazyky
 - redukční architektura a funkcionální programování

Výpočetní model – varianty

- Datově orientovaný (nejpoužívanější)
- Objektový
- Funkcionální
- Logický

Aplikace

Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)

Aplikace

Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)
- Zpracování informací (Hollerith/IBM: census USA)

Aplikace

Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)
- Zpracování informací (Hollerith/IBM: census USA)
- Zábava (počítačové hry, video-on-demand)

Aplikace

Dávají počítačům smysl

- Vědecko-technické výpočty (vojenství: atomová bomba)
- Zpracování informací (Hollerith/IBM: census USA)
- Zábava (počítačové hry, video-on-demand)
- Řízení (management, control)

Společenské aspekty

- Nástroj vědy
- Komunikace
- Zábava

Společenské aspekty

- Nástroj vědy
- Komunikace
- Zábava
- Kriminální činnost

Nástroj vědy

- Původní použití počítačů
- Trvale klíčový směr využití
- Ovlivňuje způsob vědecké práce
 - Experimenty versus simulace
 - Statistické zpracování velkých souborů
 - * Astronomie
 - * Bio-informatika
 - Virtuální vědecké týmy (spolupráce)
- Formule 1 výpočetní techniky

Komunikace

- Komunikace mezi počítači
- komunikace mezi lidmi (případně člověk–automat) – opět roste význam
 - Telefony
 - Faxy
 - Mobilní komunikace
- Změna forem spolupráce (B2B, B2C, C2C)
- Zvýšení fragility společnosti

Zábava

- Televize
- Počítačové hry
- Pasivní versus aktivní přístup
- Peer to peer sítě (Napster, Gnutella, ...)
- Virtuální realita

Kriminální činnost

- Kriminalita bílých límečků
- Zneužívání zdrojů na síti (účty, výpočetní výkon, kapacita sítě, poštovní služby, ...)
- Krádeže informací (čísla kreditních karet, telefonní linky, špionážní činnost)
- Viry
- Záměrně špatné informace

Právo a etika v IT

- V podstatě inženýrská disciplína avšak neinženýrské přístupy (shrink wrap licence, minimální odpovědnost za chyby, . . .)
- Kódy/normy správného chování/přístupu
- Faktická a právní odpovědnost
- IPR (Intellectual Property Rights), autorská ochrana, softwarové patenty

Číselné soustavy

- Definovány základem: desítková, dvojková, osmičková, šestnáctková
- Volně mezi sebou převoditelné (celá čísla bez ztráty přesnosti)
- Celá čísla a zlomky
- Reálná čísla
- Konečná reprezentace

Číselné soustavy

- Definovány základem: desítková, dvojková, osmičková, šestnáctková
- Volně mezi sebou převoditelné (celá čísla bez ztráty přesnosti)
- Celá čísla a zlomky
- Reálná čísla
- Konečná reprezentace
- První počítače v desítkové soustavě

Dvojková soustava

- Základ číslo dvě:
 - pouze dvě číslice/dva stavy
 - vhodná pro reprezentaci v elektrických systémech

Dvojková soustava v počítači

- Konečná reprezentace: interval hodnot
- 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:
 - Přímý kód
 - Inverzní kód
 - Dvojkový doplňkový kód

Záporná čísla – zobrazení

- **Přímý kód:**
 - Přidáme znaménko
 - 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: $\langle -MAX, -0$ a $\langle +0, +MAX \rangle$

Záporná čísla – zobrazení

- Dvojkový doplňkový kód:
 - Inverze bitu a přičtení jedničky
 - Pouze jedna nula ($11 \dots 11$ 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 $\langle -9 \cdot 10^{18}, 9 \cdot 10^{18} \rangle$

Racionální čísla

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

Racionální čísla

- Normalizace mantisy:
 - Nejvyšší bit vždy jedna: $1.aaaaaa$; 1 nezapíšeme
 - Nejmenší číslo: $1.0 \times 2^{2^{n-1}+1}$
- Exponent (n bitů, dvojková soustava)
 - Přičteme $2^{n-1} - 1$, abychom získali správnou hodnotu pro uložení
 - 000000001 je -126
 - 111111111 je 128
- Zvláštní a nenormalizovaná čísla

Racionální čísla II

- Rozsah zobrazení: ⟨Největší záporné, Největší kladné⟩
- 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

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 zavedením *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:*
 - Zajistit sdílení 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: 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~milion 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: základní složky

- Procesy a jejich správa
- Paměť a její správa
- Periferie a jejich správa
- Systém souborů
- 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)
- Stavy
 - Start/vytvoření, připraven (`ready`), běží (`running`), je blokován (čeká), skončil

Synchronizace – problém

- Race condition: soupeření v čase

- Proces P {
Load RegistrA, X
Load RegistrB, Y
Add RegistrA, RegistrB
Store RegistrA, X # X+=Y

- Dvě instance procesu P

- Nedefinovatelné výsledky

Synchronizace – řešení

- Semaforey
- Monitory
- Smrtelné objekty (deadlock)
- Odstranění sdílených zdrojů: zasílání zpráv
 - Synchronizace na úrovni zasílání a přijímání zpráv
 - Buffery

Procesy – plánování

- Sdílení (timesharing)
 - časové kvantum
 - přerušení
- Prioritní
 - Statistické
 - Real-time
- Plánovač (scheduler)

Správa paměti

- Dvě základní operace:
 - alokuj/přiděl paměť (velikost, vrací počáteční adresu)
 - dealokuj/uvolni paměť (velikost a počáteční adresa)
 - Většinou závislé (lze uvolnit jen přesně totéž, co jsme alokovali dříve)
 - Doplnková operace: změň rozsah alokované paměti (reallocate)
- Organizace paměti
- Čištění paměti (garbage collection)

Správa paměti OS

- Virtualizace paměti – nutno uvolnit fyzickou paměť
- Swapping
 - Celých procesů
 - „Děř“ v paměti
- Stránkování
- Segmentace

Ochrana a bezpečnost

- Ohrožení:
 - Přístup (čtení)
 - Zápis (modifikace)
 - Znepřístupnění služby (denial of service)
- Trojský kůň
 - Vydává se za něco, co není
- Viry

Principy návrhu bezpečných systémů

- Zveřejnění algoritmů
- Standardní nastavení = žádný přístup
- Pravidelné kontroly
- Minimální oprávnění
- Jednoduchý a uniformní mechanismus
- Úrovně oprávnění

Client-server model

- Distribuované počítání: rozložení úkolů na více prvků
- Client-server model
 - Speciální případ distribuovaného počítání
 - Více strukturované
 - Asymetrické: *klient* posílá požadavek na zpracování *serveru*
 - Server pro jednoho klienta může být klientem pro jiný server.

Vlastnosti modelu client-server

- Klient a server samostatné procesy
- Na stejném nebo různých počítačích
- Interní informace je „soukromá“ pro každý proces
- Komunikují tzv. peer-to-peer protokolem

Společné vlastnosti

- Interoperabilita
- Portabilita
- Integrace
- Transparence
- Bezpečnost

Příklady

- telnet
- X Window systém na Unixu
- Světová pavučina (World Wide Web)

Třívrstevný model

- Základní rozčlenění
 - Data
 - Logika
 - Prezentace
- Sousední možno kombinovat/rozdělit (tj. např. Logika může být součástí datové i prezentační vrstvy, a to i současně)

„Tlustý“ a „tenký“

- Platí pro server i klient, podstatné zejména v souvislosti s klienty
- „Tlustý“ (fat) klient:
 - Značná spotřeba lokálních zdrojů (CPU, paměť, disk)
 - Komplexní provedení i instalace
 - Příklad: Mozilla
- „Tenký“ (thin) klient:
 - Jednodušší
 - Snadná správa
 - Menší škálovatelnost (příliš mnoho práce dělá server)
 - Zpravidla vyšší nároky na propustnost sítě

Middleware

- „Zkratka“ v rámci protokolů
- Komunikace přímo na vyšší abstraktní úrovni
- Realizuje jednu (RPC) nebo více (DCE) funkcí

Middleware – příklady

- Primitivní: přenos souborů
- Základní: RPC (Remote Procedure Call)
- Integrované: DCE (Distributed Computing Environment)
- Distribuované objektové služby: CORBA, OGSA (Open Grid Service Architecture)

CORBA

- Common Object Request Broker Architecture
- Základem ORB: vrstva, která zprostředkovává komunikaci (middleware pro middleware)
- Komponenty:
 - Rozhraní (řetězce)
 - Pojmenování (naming service)
 - „Obchodní“ služba (trader)

Mobilní systémy

- Inherentně distribuované
- Často klient-server model
- Tencí i tlustí klienti
 - Kompromis mezi výkonem a propustností sítě/připojení
- Konvergence
 - Od notebooků po mobilní telefony

Návrh – principy

- efektivita
- robustnost
- flexibilita
- přenositelnost
- kompatibilita

Efektivita

- Maximální využití dostupných zdrojů
- Použití jednoduchých a jasných principů
- Dekompozice návrhu
 - Objektově orientovaný návrh (pozor na přílišnou fragmentaci)
 - Agenti
 - Komponentní programování

Robustnost

- Schopnost úspěšně se vzpamatovat po výpadku
- Řešeno redundancí (standardní inženýrské řešení): snižuje ovšem pozorovanou efektivitu
 - První výzkum v ČR koncem 50. a začátkem 60. let (Ing. Svoboda)
 - Běžné trojnásobné jištění (např. řídicí počítače atomových ponorek USA)
- V současné době zájem o *self-healing* programy

Flexibilita

- Možnost úpravy (adaptace] podle změněných potřeb
- Často používána ve významu *rozšiřitelnost* (extenzibilita)
 - Definuje a fixuje se rámec (framework)
 - Přidání nové složky bez změny rámce snadné
 - Případně hierarchie rámců (přidání či modifikace nového rámce)

Přenositelnost

- Úzce souvisí s operačními systémy
- Dostatečná abstrakce detailů
 - Virtuální „disk“ namísto konkrétního zařízení
 - Programy psány bez odkazů na speciální vlastnosti
- Využití standardů
- Opět možný rozpor s požadavkem efektivity

Kompatibilita

- Odstínění specifických detailů
- Využití standardů
- Efektivita?

Externí požadavky (na funkcionalitu OS)

- Stejný (podobný) hw a různé priority
 - Server: např. stabilita, bezpečnost, propustnost
 - Pracovní stanice: např. snadnost ovládání, rozumný výkon ve všech oblastech
 - Specializovaná grafická stanice: maximalizace grafického výkonu
 - Řídící systém: požadavky real-time, robustnost,

Bezpečnost

- Snižuje snadnost použití
- Klade dodatečná omezení na uživatele (disciplina)
- Větší nároky na správu systému
- Srovnání: MS Windows 95 versus MS Windows NT

Klasifikace OS

- Monolitický
- Vrstvený
- Modulární
- mikro-kernel

Monolický OS

- Původní operační systémy (proprietární)
- Abstrakce nepoužívána příliš *dovnitř*
- Nejasné rozlišení funkcí uvnitř operačního systému
- uvVelké, špatně rozšiřitelné, špatně udržovatelné
- Poplatné době pomalejšího vývoje hardware a jeho vysoké ceny

Vrstvený OS

- Vrstvy odpovídají procesům správy:
 - Správa CPU
 - Správa paměti
 - Správa periférií
 - Správa systému souborů
- Lepší abstrakce
- Komunikace mezi vrstvami

Modulární OS

- Moduly namísto vrstev
- Zapouzdření (enkapsulace) funkcí
- Komunikace mezi moduly
- Příbuzný objektovému přístupu
- Lepší údržba
- Riziko vzniku „fatware“

Kernel operačního systému

- Kernel, též *jádro* operačního systému:
 - Základní složka operačního systému
 - Odpovídá za:
 - * Alokaci a správu zdrojů
 - * Přímé ovládání hardware (nízkoúrovňové interfaces)
 - * Bezpečnost
- Mikrokernel:
 - *Malé je pěkné*
 - Modulární přístup, malé moduly odpovídající za konkrétní operace
 - Řada funkcí až v uživatelském prostoru
 - Vysoce flexibilní, upravení operačního systému podle potřeby

Aplikační programová rozhraní (API)

- Definují způsob („calling conventions“) přístupu k operačnímu systému a dalším službám
- Definováno na úrovni zdrojového kódu
- Představuje *abstrakci* volané služby
- Účel:
 - Přenositelnost
 - Snadná správa kódu
- Další použití
 - Překlad mezi službami vysoké a nízké úrovně
 - * Převod typů/struktury parametrů
 - * Převod mezi způsoby předávání parametrů (by-value a by-reference)

API – příklady

- Práce se soubory:

- Otevření: `int open(char *path, int oflag, ...)`
- Čtení: `int read(int fildes, char *buf, unsigned nbytes)`
- Zápis: `int write(int fildes, char *buf, unsigned nbytes)`
- Zavření: `int close(int fildes)`

- Práce s pamětí:

- Alokace paměti: `void *malloc(size_t size)`
- Uvolnění paměti: `void free(void *ptr)`
- Změna alokace: `void *realloc(void *ptr, size_t size)`

Periferie z pohledu (modulárního) OS

- Ovladače na nejnižší úrovni („nejblíže“ hardware)
 - Specifické „jazyky“ ovládání periferií na této úrovni
 - Práce se *signály* (např. změna stavu periferie)
 - Příklady
 - * Práce s diskem
 - * Ovládání klávesnice a myši (čtení signálů)
 - * Grafika a ovládání grafických rozhraní
 - * Síťové karty

Periferie – pohled „shora“

- Zpřístupněny prostřednictvím příslušného API
- Abstrakce: možnost výměny konkrétního zařízení (disk, síťová karta) bez vlivu na způsob použití
- Příznaky a klíče pro ovládání specifických vlastností: přenositelnost versus efektivita

System 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 (nebo jejich části) jako položka v databázi
 - Bohatší množina operací
 - Složitější implementace

Struktura souborů

- Posloupnost bytů – vnitřní struktura pro OS neznáma
- Posloupnost záznamů (records)
- Strom – každý uzel má vlastní klíč

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
 - * Provázány odkazem na další blok daného souboru
 - inodes

Struktura – inodes

- Podobné indexovému
- 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

Volné bloky

- V tabulce
- Bitový vektor
- Provázaný seznam
- Většinou zpracovávány 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)

Ochrana souborů

- Základní operace:
 - čtení, zápis (včetně vytvoření), smazání, prodloužení a spuštění souboru
- Ochranné domény:
 - Skupina, která má stejná práva
 - Např.: Já, moji přátelé, ostatní
 - Statické versus dynamické
 - UNIX: user—group—other

Řízení přístupu k souborům

- Access Control List, ACL (seznamy přístupových oprávnění); připojen ke každému souboru
 - Základní (z UNIXových systémů):
 - * r: čtení souboru (čtení obsahu adresáře)
 - * w: zápis souboru (včetně vytvoření)
 - * x: spuštění (sestoupení do podadresáře)
 - Plné ACL: více práv, dynamická práce se skupinami
- Capability List, CL
 - Uspořádání podle domén, nikoliv podle souborů
 - Vhodné pro distribuované systémy
 - Schopnost (capability) tj. práva přístupu patří procesu a ten je může předávat dalším procesům

Ochrana přístupu uvnitř OS

- Kernel a uživatelský prostor
- Oddělení na hw úrovni
- Každá stránka někomu patří
- Pouze kernel má přístup k hardware
 - Kontroluje práva přístupu
 - Obsluhuje zařízení (pro všechny)
 - Garantuje serializaci přístupu
- Uživatelské procesy používají *volání* kernelu (jádra)

Přístup k paměti

- Příslušnost virtuálních stránek k procesu
- Výpadek stránky: nepovolený přístup
- Ochrana
 - Mezi procesem a jádrem
 - Mezi procesy
 - Uvnitř procesu

Přerušení

- Operační systémy obecně reagují na události (events)
- *Přerušení*: mechanismus, jak přerušit vykonávanou práci na základě externí příčiny (nějaké události)

Význam přerušení

- Výpadek proudu
- Výpadek hardware
- Problém v programovém vybavení
 - Neautorizovaný přístup
 - Nelegální instrukce nebo operandy
- Zásah operátora
- Podpora I/O
- Požadavek počítačem řízeného systému

Příklady

- Přerušení od časovače (přeplánování procesů, timeout, ...)
- Přerušení od periferie (klávesnice, myš, síťová karta, ...)
- Přerušení z procesoru (dělení nulou, chybná operace, ...)

Principy přerušení

- *Přeruší* běh aktuálního programu
 - Nutno schovat stav
 - a zapamatovat místo návratu
- Více zdrojů a příčin přerušení
 - Nutno rozlišit typy (příčinu) přerušení
 - Nutno zapamatovat zdroj přerušení

Obsluha přerušení

- Obsluha přerušení realizována v kernelu
 - Zajištění serializace
 - Bezpečnost
- Vyvolá tzv. přepnutí kontextu

Další vlastnosti

- Maskování přerušení
 - dočasné a trvalé
 - možná ztráta přerušení/události
- Priorita přerušení/obsluhy
 - Základní tři úrovně:
 - * Nemaskovaná přerušení: vyšší priorita
 - * Aktuálně zpracovávané přerušení
 - * Maskovaná přerušení: nižší priorita

Polling

- Polling = opakované dotazování (na stav/událost)
- Možná alternativa pro některá přerušení
 - Zaměstnává procesor
 - Může zůstat v uživatelském prostoru