PB001: Úvod do informačních technologií

Luděk Matyska

Fakulta informatiky Masarykovy univerzity

podzim 2015

Obsah přednášky

1. Návrh OS
2. Klasifikace OS
3. Kernel operačního systému

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

Návrh OS – principy

- efektivita
- robustnost
- flexibilita
- přenositelnost
- kompatibilita
- bezpečnost

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. řídící počítače atomových ponorek USA)
- V současné době zájem o *self-healing* programy
  - rozpoznání problému (výpadek nějaké komponenty)
  - návrh reakce a její implementace
  - nemusí garantovat plnou funkčnost/kapacitu
  - garantuje nezhroucení celého systému

Flexibilita

- Možnost úpravy (adaptace) podle změněných potřeb – *Adaptabilita*
  - Příklad: výměna hw komponenty bez efektu na systém
    - změna parametrů (rychlejší disk s větší kapacitou), ale nikoliv změna principu
- Často používána i 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)
  - Příklad: přidání dalšího procesoru, disku či grafické karty

Přenositelnost

- Velmi významná pro operační systémy
  - V minulosti se OS dělal přím ona konkrétní hardware, nebylo jej kam přenášet
  - ,,Revoluce“:
    - OS/360 firmy IBM v 60. letech
    - UNIX (a jazyk C) v sedmdesátých letech (Multics)
    - IBM PC a MS Windows
    - Linux
- Dostatečná abstrakce detailů
  - Virtuální „disk“ namísto konkrétního zařízení
  - Programy psány bez odkazů na speciální vlastnosti
- Využití standardů
- Opět možný rozpor s požadavkem efektivity

Kompatibilita

- Úzce souvisí s přenositelností
- Odstínění specifických detailů
  - usnadňuje práci uživatelům OS (včetně programátorů)
- Využití standardů
- Efektivita?
  - Skrytí výkonových specifik
  - Je možné kompenzovat
  - Tlak na výrobce, ať „nekomplikují“ situaci
    - sjednocení architektury CPU
    - minimalizace variant GPU

Bezpečnost

Cíl:

- Ochrana dat a majetku před krádeží, zneužitím, či poškozením při současném zachování přístupu vybraných uživatelů.

Problémy:

- Větší nároky na správu systému
- Snižuje snadnost použití
  - musíte se přihlásit
- Klade dodatečná omezení na uživatele (disciplina)
  - bezpečnost není jen technický, ale především organizační úkol
- Srovnání: MS Windows 95 versus MS Windows NT

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,

Řešit různými operačními systémy nebo jedním dostatečně variabilním?

Klasifikace OS

- Monolitický
- Vrstvený
- Modulární
- Koncept kernelu a mikro-kernelu

Monolický OS

- Původní operační systémy (proprietární)
- Abstrakce nepoužívána příliš *dovnitř*
  - jedna skupina "opravdových programátorů" po celou dobu životnosti OS
- Nejasné rozlišení funkcí uvnitř operačního systému
- „Velké", špatně rozšiřitelné, špatně udržovatelné
- Poplatné době pomalejšího vývoje hardware a jeho vysoké ceny

Vrstvený OS

- Vrstvy odpovídají procesům správy:
  - Správa CPU
  - Správa paměti
  - Správa periferií
  - Správa systému souborů
- Lepší abstrakce
- Komunikace mezi vrstvami
  - Komplikuje strukturu
  - Riziko obcházení (shortcuts)
  - Jistá penalizace ve výkonu

Modulární OS

- Moduly namísto vrstev
- Zapouzdření (enkapsulace) funkcí
- Komunikace mezi moduly
  - Složitější na obejití
  - Může mít vyšší režii (vyšší penalizace ve výkonu)
- Příbuzný objektovému přístupu
- Lepší údržba
  - Moduly menší, snáze se vyměňují než celé vrstvy
- Riziko vzniku „fatware“
  - Příliš mnoho příliš malých modulů

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
- Sestává se z definicí funkcí, datových struktur a tříd
- 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

- 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

- Ovladače na nejnižší úrovni („nejblíže“ hardware)
  - Specifické „jazyky“ ovládání periferií na této úrovni
  - Práce se *signály* (např. změna stavu periferie)

Periferie z pohledu (modulárního) OS

Začlenění ovladače do jádra

- kooperativní vs. hierarchické (možnost preempce)
- efektivita vs. stabilita
- formální verifikace ovladačů: Microsoft Static Driver Verifier

Příklady

- Práce s diskem
- Ovládání klávesnice a myši (čtení signálů)
- Grafika a ovládání grafických rozhraní
- Síťové karty