Architektury OS

přehled a dělení

Struktury operačních systémů

Monolitická struktura

- **Charakteristika**
  - Celý operační systém běží jako jeden velký program v režimu jádra (kernel mode). Všechny systémové služby (správa procesů, souborů, paměti, ovladače zařízení) jsou součástí jednoho modulu
  - **Výhody**
    - Vysoký výkon díky minimálním režijním nákladům na přepínání kontextu mezi moduly
  - **Nevýhody**
    - Velmi obtížné ladění a údržba; chyba v jedné části může shodit celý systém; nízká modularita
- **Příklady**
    - Původní UNIX, Linux

Struktury operačních systémů

Vrstvená struktura

- **Charakteristika**
  - OS je rozdělen do hierarchických vrstev, kde každá vrstva poskytuje služby vrstvě nad ní a využívá služby vrstvy pod ní. Nejnižší vrstva je hardware, nejvyšší uživatelské aplikace.
  - **Výhody**
    - Snadnější ladění a vývoj díky modularitě; změny v jedné vrstvě neovlivňují ostatní, pokud je rozhraní zachováno.
  - **Nevýhody**
    - Nižší výkon kvůli režijním nákladům na komunikaci mezi vrstvami.
- **Příklady**
    - THE OS (první vrstvený systém), Multics.

Struktury operačních systémů

Mikrokernel (mikrojádro)

- **Charakteristika**
  - Jádro OS je minimální a obsahuje pouze základní funkce (správa procesů, správa paměti a meziprocesová komunikace). Všechny ostatní služby (souborový systém, ovladače zařízení, síťové protokoly) běží jako samostatné procesy v uživatelském režimu.
  - **Výhody**
    - Vyšší spolehlivost (chyba v jednom ovladači neshodí celý systém); snadnější rozšiřitelnost a modularita.
  - **Nevýhody**
    - Potenciálně nižší výkon kvůli častému přepínání kontextu mezi uživatelským a jádrovým režimem a meziprocesové komunikaci.
- **Příklady**
  - Mach (základ pro macOS X), Minix, QNX.

Struktury operačních systémů

Modulární struktura

- **Charakteristika**
  - Moderní přístup, který kombinuje výhody monolitického a mikrokernelového designu. Jádro je monolitické, ale je navrženo tak, aby bylo možné dynamicky načítat a odstraňovat moduly (např. ovladače zařízení) za běhu.
  - **Výhody**
    - Dobrý kompromis mezi výkonem a flexibilitou; snadná rozšiřitelnost.
- **Příklady**
  - Většina moderních operačních systémů, včetně Linuxu (kde je jádro monolitické, ale modulární), Windows.

Struktury operačních systémů

Hybridní jádro

- **Charakteristika**
  - Snaží se kombinovat nejlepší aspekty monolitických a mikrokernelových jader. Části OS, které jsou kritické pro výkon (např. I/O operace), zůstávají v jádře, zatímco ostatní komponenty mohou běžet v uživatelském režimu
  - **Výhody**
    - Dobrý výkon a zároveň určitá modularita a spolehlivost
- **Příklady:** Microsoft Windows NT (a jeho nástupci), macOS

Architektury OS

Základní režimy provozu

- **Režim jádra (Kernel Mode / Supervisor Mode):**
  - **Charakteristika**
    - Nejprivilegovanější režim. Kód běžící v tomto režimu má plný a neomezený přístup ke všem hardwarovým zdrojům a instrukcím procesoru.
  - **Kdo zde běží**
    - Pouze operační systém (jádro OS), ovladače zařízení.
  - **Účel**
    - Zajištění stability a bezpečnosti systému; brání aplikacím v přímém poškození hardwaru nebo dat jiných aplikací.

Architektury OS

Uživatelský režim (User Mode)

- **Charakteristika**
  - Méně privilegovaný režim. Kód běžící v tomto režimu má omezený přístup k hardwaru a nemůže přímo provádět citlivé operace.
  - **Kdo zde běží**
    - Všechny uživatelské aplikace (webové prohlížeče, textové editory, hry atd.).
  - **Účel**
    - Izolace aplikací; chyba v jedné aplikaci neshodí celý systém.

Architektury OS

Přepínání mezi režimy

Přepínání mezi režimem jádra a uživatelským režimem je kritické pro bezpečnost a funkčnost OS. Aplikace v uživatelském režimu nemohou přímo přistupovat k hardwaru. Pokud aplikace potřebuje provést I/O operaci (např. čtení z disku nebo tisk), musí použít **systémové volání (system call)**.

- **Systémové volání**
  - speciální instrukce, která způsobí přerušení (interrupt) procesoru. Procesor pak přepne z uživatelského režimu do režimu jádra a předá řízení kódu operačního systému. OS provede požadovanou operaci a po dokončení vrátí řízení zpět aplikaci v uživatelském režimu.

Architektury OS

Virtuální paměť

Moderní OS využívají **virtuální paměť**, což je technika, kteráumožňuje programům přistupovat k více paměti, než je fyzicky dostupné v RAM. OS mapuje virtuální adresy, které programy používají, na fyzické adresy v RAM nebo na disku (tzv. stránkování nebo segmentace). Tím se také zajišťuje izolace paměti mezi procesy.