PB002 – Základy informačních technologií

Operační systémy II

12. října 2016

Správa paměti

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

Správa paměti OS

1. Virtualizace paměti – nutno uvolnit fyzickou paměť
2. Swapping
   1. Celých procesů
   2. „Děr“ v paměti
3. Stránkování
4. Segmentace

Ochrana a bezpečnost

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

Principy návrhu bezpečných systémů

1. Zveřejnění algoritmů
2. Standardní nastavení = žádný přístup
3. Pravidelné kontroly
4. Minimální oprávnění
5. Jednoduchý a uniformní mechanismus
6. Úrovně oprávnění

Client-server model

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

Vlastnosti modelu client-server

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

Společné vlastnosti

1. Interoperabilita
2. Portabilita
3. Integrace
4. Transparence
5. Bezpečnost

Příklady

1. `telnet`
2. X Window systém na Unixu
3. Světová pavučina (World Wide Web)

Třívrstevný model

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

„Tlustý“ a „tenký“

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

Middleware

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

Middleware – příklady

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

Mobilní systémy

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

Návrh – principy

1. efektivita
2. robustnost
3. flexibilita
4. přenositelnost
5. kompatibilita

Efektivita

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

Robustnost

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

Flexibilita

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

Přenositelnost

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

Kompatibilita

1. Odstínění specifických detailů
2. Využití standardů
3. Efektivita?
    1. Nemusí být negativně ovlivněna

Externí požadavky (na funkcionalitu OS)

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

Bezpečnost

1. Snižuje snadnost použití
2. Klade dodatečná omezení na uživatele (disciplina)
3. Větší nároky na správu systému
4. Srovnání: MS Windows 95 versus MS Windows NT

Klasifikace OS

1. Monolitický
2. Vrstvený
3. Modulární
4. mikro-kernel

Monolický OS

1. Původní operační systémy (proprietární)
2. Abstrakce nepoužívána příliš *dovnitř*
3. Nejasné rozlišení funkcí uvnitř operačního systému
4. „Velké“, špatně rozšiřitelné, špatně udržovatelné
5. Poplatné době pomalejšího vývoje hardware a jeho vysoké ceny

Vrstvený OS

1. Vrstvy odpovídají procesům správy:
    1. Správa CPU
    2. Správa paměti
    3. Správa periferií
    4. Správa systému souborů
2. Lepší abstrakce
3. Komunikace mezi vrstvami

Modulární OS

1. Moduly namísto vrstev
2. Zapouzdření (enkapsulace) funkcí
3. Komunikace mezi moduly
4. Příbuzný objektovému přístupu
5. Lepší údržba
6. Riziko vzniku „fatware“

Kernel operačního systému

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

Aplikační programová rozhraní (API)

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

API – příklady

1. Práce se soubory:
   1. Otevření: `int open(char *path, int oflag, ...)`
   2. Čtení: `int read(int fildes, char *buf, unsigned nbytes)`
   3. Zápis: `int write(int fildes, char *buf, unsigned nbytes)`
   4. Zavření: `int close(int fildes)`
2. Práce s pamětí:
   1. Alokace paměti: `void *malloc(size_t size)`
   2. Uvolnění paměti: `void free(void *ptr)`
   3. Změna alokace: `void *realloc(void *ptr, size_t size)`

Periferie z pohledu (modulárního) OS

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

Periferie – pohled „shora“

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

Systém souborů

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

Struktura systému souborů

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

Struktura souborů

1. Posloupnost bytů – vnitřní struktura pro OS neznáma
2. Posloupnost záznamů (records)
3. Strom – každý uzel má vlastní klíč

Struktura souborů

1. Posloupnost bytů – vnitřní struktura pro OS neznáma
2. Posloupnost záznamů (records)
3. Strom – každý uzel má vlastní klíč

Typ a přístup

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

Struktura na disku

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

Struktura – inodes

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

Volné bloky

1. V tabulce
2. Bitový vektor
3. Provázaný seznam
4. Většinou zpracovávány podle FCFS (First Come First Served)

Vyrovnávací paměť

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

Ochrana souborů

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

Řízení přístupu k souborům

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

Ochrana přístupu uvnitř OS

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

Přístup k paměti

1. Příslušnost virtuálních stránek k procesu
2. Výpadek stránky: nepovolený přístup
3. Ochrana
    1. Mezi procesem a jádrem
    2. Mezi procesy
    3. Uvnitř procesu

Přerušení

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

Význam přerušení

1. Výpadek proudu
2. Výpadek hardware
3. Problém v programovém vybavení
   1. Neautorizovaný přístup
   2. Nelegální instrukce nebo operandy
4. Zásah operátora
5. Podpora I/O
6. Požadavek počítačem řízeného systému

Příklady

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

Principy přerušení

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

Obsluha přerušení

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

Další vlastnosti

1. Maskování přerušení
   1. dočasné a trvalé
   2. možná ztráta přerušení/události
2. Priorita přerušení/obsluhy
   1. Základní tři úrovně:
      1. Nemaskovaná přerušení: vyšší priorita
      2. Aktuálně zpracovávané přerušení
      3. Maskovaná přerušení: nižší priorita

Polling

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