Typologie, funkční skladby a architektury OS

PB 152 ◇ Operační systémy

Jan Staudek
http://www.fi.muni.cz/usr/staudek/vyuka/

Verze: jaro 2017

Obsah přednášky

- Typologie operačních systémů
- Generické komponenty operačních systémů
- Trendy vývoje operačních systémů
- Typologie výpočetních prostředí
- Koncept služeb poskytovaných OS

Typologie OS

- *Mainframe operating systems*, OS střediskových počítačů
- *Server operating systems*, OS serverů
- *Multiprocessor operating systems*, OS multiprocesorů
- *Personal computer operating systems*, OS osobních počítačů
- *Handheld operating systems*, OS tabletů, mobilů, ...
- *Embedded operating systems*, OS vestavěných počítačů
- *Sensor node operating systems*, OS uzlů senzorových sítí
- *Real-time operating systems*, OS pro řízení v reálném čase
- *Smart card operating systems*, OS chipových karet

Studujeme obecné, společné, generické rysy OS těchto typů OS

Typologie OS

- Mainframe operating systems
  - ✓ OS systémů datových center
  - ✓ spousta periferií (tisíce disků, spousty terabajtů dat)
  - ✓ spousta procesů se řeší souběžně s obrovským objemem IO
  - ✓ dávkové zpracování + transakční zpracování + interaktivní zpracování
    dávkové – generování zpráv o produkci, ...
    transakční – rezervační systémy, ...
    interaktivní – kladení dotazů do velké databáze
  - ✓ V současnosti orientace na LINUX
- Server operating systems
  - ✓ OS velmi velkého PC (výkonem, paměťovou a komunikační kapacitou)
  - ✓ obsluha mnoha vzdálených uživatelů (klientů)
  - ✓ klientům poskytuje tiskové, souborové, webovské ... služby
  - ✓ typičtí reprezentanti: Solaris, FreeBSD, Linux, Windows Server 200x

Typologie OS

- Multiprocessor operating systems
  - ✓ typicky variace na *server/mainframe OS*
  - ✓ v současnosti implementace i na PC a mobilech
  - ✓ speciální plánování činnosti více CPU
  - ✓ problém souběhu více funkcí OS
  - ✓ v současnosti i na noteboocích s vícejádrovými CPU
  - ✓ Rys implementovaný jak v Linuxech, tak i ve Windows
- Personal computer operating systems
  - ✓ V současnosti vždy podpora multitaskingu
  - ✓ Cíl – dobrá podpora jednomu uživateli – monouživatelské OS
  - ✓ zpracování dokumentů, tabulkové kalkulátory, přístup na Internet, ...
  - ✓ Příklady – Linux, FreeBSD, Windows 10, Masintosh OS

Typologie OS

- Handheld operating systems
  - ✓ OS pro tablety, chytré mobily, ...
  - ✓ Nepočítají s vnější pamětí
  - ✓ Jsou propracované z hlediska ovládání telefonie, digifota, ...
  - ✓ Běžně se provozují (ne vždy důvěryhodné) aplikace třetích stran
  - ✓ Příklady OS: iOS* 5.0, Microsoft Windows* Phone 7.0, Android

Typologie OS

- Embedded operating systems
  - ✓ Řídí zařízení, která nevypadají jako počítač
  - ✓ uživatel nemá možnost nic do systému instalovat
  - ✓ OS mikrovlnek, TV, v autech, v DVD recorderech ...
  - ✓ častá orientace na real-time
  - ✓ nelze instalovat žádné nové aplikace, vše je typicky v ROM
  - ✓ QNX/www.qnx.com, VxWorks/www.windriver.com/products/vxworks, oba kompatibilní s POSIX
- Sensor node operating systems
  - ✓ OS uzlů senzorových sítí, senzorový uzel – počítač+senzor(y)+komunikace
  - ✓ Dlouhodobá činnost v bezdrátové síti, malá paměť, bateriový provoz
  - ✓ Veškeré programy bývají instalované předem
  - ✓ Příklad: TinyOS, www.tinyos.net/

Typologie OS

- Smart card operating systems
  - ✓ omezenost výkonem, pamětí, extrémně jedoduché OS
  - ✓ obvykle v ROM je interpretr Java Virtual Machine
    aplikace – javovské applety (malé programy)
    někdy i v režimu multitasking
- Real-time operating systems
  - ✓ klíčový problém – faktor času a plnění úloh v čase
- Modelové prostředí našeho studia vymezují rodiny operačních systémů Unix a Windows
  - ✓ tam kde to bude vhodné zmíníme specifika derivátů systémů Unix (Linux, Mac OS X, Android apod.)

Studované bázové generické problémy řešené v OS

- Architektura, skladba OS
- Procesy, – interpretace programů, sdílení CPU procesy, kooperace procesů
- Adresové prostory, kooexistence mnoha dějů v mnoha různých pamětech
- Input/Output, ovládání periferií
- Ochrany, bezpečnost
- Rozhraní služeb pro procesy, pro uživatele

- *Soubory dat,*
  dlouhodobé uchovávání dat na vnějších pamětech (PV062)

Bázová idea OS

- OS poskytuje uživateli / aplikacím jednoduché a přitom mocnější rozhraní než hardware
- Uživatelé / aplikace volají provádění služeb vysoké úrovně, dostupných na rozhraních OS a vykonávaných OS
- Uživatelé / aplikace nemohou přistupovat k privilegovaným rysům hardware přímo
- Sestava služeb poskytovaných OS je to, co si myslí uživatelé / aplikace, že je OS
- uživatelé / aplikace nic jiného než služby OS nevidí
- Všechny soudobé OS považují za generické předměty správy PROCESY – děje řízené programy uloženými v paměti a realizované CPU a periferiemi

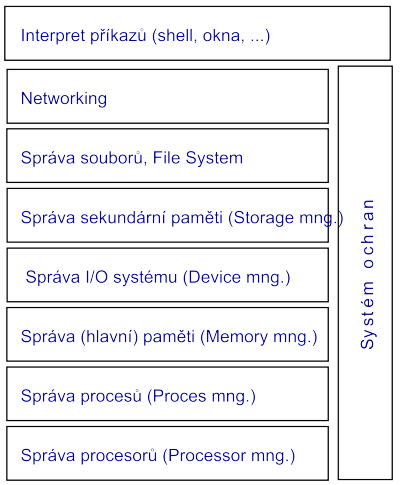
Generické funkční komponenty OS

- Správa procesorů – kdy ten který proces ,,poběží"
- Správa procesů a vláken – podpora souběžnosti dějů
- Správa (hlavní, operační) paměti – řízení jejího využívání
- Správa souborů – dat uchovávaných na vnější paměti
- Správa I/O systému – správa činnosti periferních zařízení
- Správa vnější (sekundární) paměti – řízení jejího využívání
- Networking (síťování) – podpora distribuovaných systémů
- Systém ochran – zajištění bezpečnosti
- Interpret příkazů – uživatelů u terminálů
- Systémové programy – kompilátory, editory, . . .
  - ✓ stavové informace, podpora jazyků, podpora komunikace, manipulace se soubory, aplikační systémy (databáze, . . .)

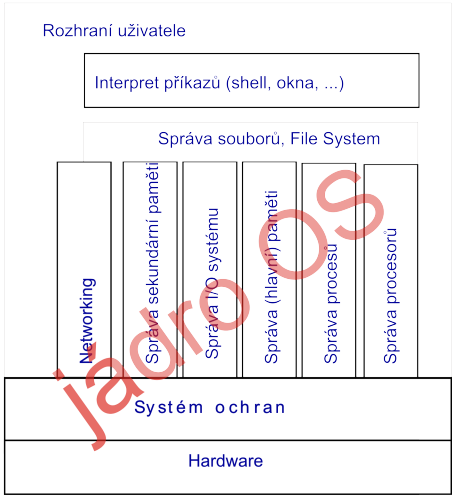
Tou či onou formou jsou implementované v každém OS

Generické funkční komponenty OS

Zidealizovaný nástin hierarchie komponent OS



Častá reálná implementace:



Generické funkční komponenty OS, širší popis

- Správa procesorů
  - ✓ dispečer, krátkodobý plánovač
- Správa procesů a vláken
  - ✓ vytváření a rušení procesů a vláken
  - ✓ pozastavování a obnova běhu procesů a vláken
  - ✓ mechanismy synchronizace procesů a vláken
  - ✓ mechanismy komunikace mezi procesy a vláken
- Správa (hlavní, operační) paměti
  - ✓ zobrazování LAP do FAP
  - ✓ virtualizace paměti
  - ✓ sledování které části FAP jsou používány a kým
  - ✓ mechanismy přidělování a uvolňování paměti (FAP) na žádost
  - ✓ střednědobé plánování

Generické komponenty OS, širší popis, 2

- Správa I/O systému
  - ✓ správa vyrovnávacích paměti
  - ✓ univerzální rozhraní ovladačů
  - ✓ ovladače
- Správa vnější (sekundární) paměti
  - ✓ správa volné paměti
  - ✓ přidělování paměti
  - ✓ plánování optimálního pořadí (diskových) operací
- Správa souborů (systém souborů, *File System*)
  - ✓ manipulace s kolekcemi dat na vnějších pamětech
  - ✓ vytváření, rušení, katalogizace, archivace, obnova, . . . souborů

Generické komponenty OS, širší popis, 3

- Networking (síťování), distribuované systémy
  - ✓ kooperace procesorů nesdílejících ani paměť ani hodiny
  - ✓ každý procesor má svou lokální paměť a hodiny
  - ✓ propojení komunikační sítí
  - ✓ nástroje pro sdílení zdrojů (distribuovaný systém souborů, . . .)
- Interpret příkazů
  - ✓ rozhraní uživatele na služby operačního systému
- Systém ochran
  - ✓ mechanismy pro řízení přístupu procesů a uživatelů ke zdrojům
  - ✓ rozlišování autorizovaných a neautorizovanyých používání
  - ✓ specifikace vnucovaných ochranných opatření
  - ✓ nástroje pro prosazování ochranných opatření

Správa procesů

- provedení programu – proces (*process*, *task*)
  - ✓ program – pasivní entita
  - ✓ proces – aktivní entita,
    více procesů může být řízeno tímtéž programem souběžně
  - ✓ proces – jednotka plánování činností definovaných programem
- proces potřebuje pro svoji realizaci jisté zdroje:
  - ✓ CPU (procesor), paměť, I/O zařízení, soubory . . .
  - ✓ inicializační data
  - ✓ proces – entita schopná vlastnit zdroje
- Omezená varianta pojmu proces – vlákno
  - ✓ jednotka POUZE plánování činností definovaná v programu
  - ✓ vlákna nic nevlastní, využívají zdroje přidělené JEJICH procesu

Správa procesů

- 1-vláknový proces
  - ✓ proces vlastnící jediný čítač instrukcí určující příště prováděnou instrukci
  - ✓ proces provádí instrukce sekvenčně, po jedné instrukci v čase, dokud se neukončí nebo dokud není jeho běh přerušený
  - ✓ běh procesu nemusí být v čase kontinuální – multitasking
- více-vláknový proces
  - ✓ proces vlastnící jeden čítač instrukcí pro každé v něm definované vlákno
  - ✓ proces provádí instrukce vláken sekvenčně, po jedné v čase, dokud se vlákno neukončí nebo dokud není jeho běh přerušený
  - ✓ vlákna jsou řešena v režimu multiprogramování/multitasking
  - ✓ proces definující vlákna je rovněž řešený v režimu multiprogramování/multitasking

Správa procesů

- Studujeme principy univerzálních OS
  - ✓ OS umožňuje soběžné provádění mnoha procesů a tyto procesy náleží mnoha uživatelům
  - ✓ pro řešení procesů má OS k dispozici alespoň jeden procesor
  - ✓ souběžnost se dosahuje přepínáním procesoru(ů) mezi procesy (vlákny)
- OS je z hlediska správy procesů odpovědný za
  - ✓ Vytváření a rušení uživatelských a systémových procesů
  - ✓ Potlačení a obnovování běhu procesů
  - ✓ Poskytnutí mechanismů pro
    - – synchronizaci procesů, pro
    - – komunikaci mezi procesy a pro
    - – zvládání uváznutí procesů (vesměs na úrovni middleware)

Správa procesorů

- OS (správa procesorů) je z hlediska správy procesorů odpovědný za
  - ✓ výběr procesu běžícího na (dostupném) procesoru
  - ✓ výběr se řídí podle definované plánovací politiky
    - – cyklické plánování, prioritní plánování, . . .
  - ✓ rovněž dispečer, plánovač CPU, . . .
- Plánování vláken řeší podle typu OS
  - ✓ jádro OS (správa procesorů) jak pro procesy tak i pro vlákna
  - ✓ ,,run-time support'', tj. moduly na úrovni knihoven, řešící plánování vláken v rámci procesu

Správa (hlavní, operační, primární) paměti

- adresový prostor hlavní (operační, primární) paměť
  - ✓ Fyzický Adresový Prostor, FAP
  - ✓ pole samostatně adresovatelných slov nebo bytů
  - ✓ repositář elektronicky dostupných dat CPU a I/O zařízením
  - ✓ repozitář instrukcí interpretovaných procesorem
- Hlavní pamět je energeticky závislé zařízení
  - ✓ pamatovaná data se ztrácí po výpadku energie

Správa (hlavní, operační, primární) paměti

- Správa (hlavní) paměti je odpovědná za
  - ✓ vedení přehledu, který proces kterou část paměti v daném okamžiku využívá
  - ✓ rozhodování, kterému procesu uspokojit jeho požadavek na prostor paměti po uvolnění prostoru paměti.
  - ✓ přidělování a uvolňování paměti podle potřeby
  - ✓ rozhodování, který proces nebo která část procesu uvolní hlavní paměť, aby bylo možno uspokojit (oprávněné, prioritnější, ...) požadavky jiných procesů na prostor v hlavní paměti

Správa (hlavní) paměti, virtualizace paměti

- pohled programátora na paměť v počítači
- Logický adresový prostor, LAP
  - ✓ formát LAP je vymezen formátem adresy v instrukci
  - ✓ kapacita LAP je daná bitovou šířkou adresy v instrukci
  - ✓ OS zavádí do FAP části programů a dat podle potřeby
  - ✓ v současnosti se programy uchovávají ve formě přeložení do LAP
  - ✓ tranformace adres LAP na adresy FAP se provádějí až při provádění instrukce v CPU
- Struktury LAP
  - ✓ lineární (pole stránek) – virtualizace stránkováním na žádost
  - ✓ dvoudimensionální – kolekce samostatných lineárních segmentů (proměnné délky), virtualizace segmentováním na žádost, segmentovaný LAP bývá často navíc i stránkovaný

Správa (hlavní) paměti, virtualizace paměti

- Lineární LAP může být zobrazovaný do FAP *identitou*
- běžně se používají propracovanější způsoby zobrazování
- Zobrazování LAP do dostupného FAP se děje pomocí spolupráce hardware a funkcionality jádra OS
  - ✓ DAT, *Dynamic Address Translation*
  - ✓ také MMU, *Memory Management Unit*
- Při odkázání místa s adresou LAP, které není zobrazeno ve FAP
  - ✓ správa paměti aktivovaná přerušením nalezne (vytvoří) ve FAP volný blok
  - ✓ na toto místo zavede blok z obrazu LAP s požadovanou informací
- nutná úzká spolupráce se specializovaným systémem souborů
  - ✓ na vnější paměti se udržuje kopie LAP procesu

Správa I/O systému

- skrývá před uživatelem specifika konkrétních I/O zařízení
- organizuje repositář vyrovnávacích pamětí a cache pamětí
- organizuje *spooling*, překrývání výstupů jednoho procesu se vstupy dalšího procesu, resp. dalších procesů
- podporuje univerzální rozhraní driverů (ovladačů) I/O zařízení
- obsahuje drivery (ovladače) jednotlivých hardwarových I/O zařízení

Správa informačních skladů, správa souborů dat

- OS poskytuje jednotný, logický pohled na sklad informací
  - ✓ abstrahuje fyzické vlastnosti skladů do logických jednotek – souborů
  - ✓ každý druh paměťového média je ovládaný relevantním druhem zařízení (páskový stojan, diskový stojan, . . .)
  - ✓ jednotlivé druhy zařízení se vzájemně liší rychlostí přístupu, kapacitou, rychlosti přenosu dat, přístupovou metodou (sekvenčně, libovolně, . . .), . . .
- Soubor
  - ✓ identifikovatelná kolekce souvisejících informací definovaná svým tvůrcem
  - ✓ vnitřně se člení na samostatně zpřístupnitelné záznamy
  - ✓ záznamy bývají vnitřně strukturovány do položek
  - ✓ reprezentace jak programů, tak i dat ve vnější paměti

Správa informačních skladů, správa souborů dat

- Systém správy souborů odpovědný za:
  - ✓ vytváření a rušení adresářů (katalogů)
  - ✓ organizování souborů do katalogů – adresářů
  - ✓ poskytnutí nástrojů pro přidělování a kontrolování přístupových práv
  - ✓ vytváření a rušení souborů
  - ✓ podporu primitivních operací pro manipulaci se soubory a s adresáři
  - ✓ zobrazování souborů do konkrétně použité sekundární paměti
  - ✓ archivování souborů na stabilní energeticky nezávislá média
  - ✓ zpřístupňování, doplňování záznamů souborů

Správa vnější (sekundární) paměti

- Hlavní (primární, operační) paměť
  - ✓ je energeticky závislá, neschopná udržet informaci trvale
  - ✓ má malou kapacitu na to, aby v ní bylo možné uchovávat všechna data a programy
- počítačový systém musí obsahovat pro zálohování hlavní (primární) paměti energeticky nezávislou, dostatečně kapacitní sekundární paměť
  - ✓ i za cenu nemožnosti přímé dostupnosti jejích obsahů procesorem
- Většina současných počítačů používá pro roli vnější (sekundární) paměti pro uchovávání programů i dat disky
- OS co správce vnější (sekundární) paměti je odpovědný za
  - ✓ Správu volné paměti na disku
  - ✓ Přidělování paměti disku souborům
  - ✓ Plánování činnosti disku

Správa vnější (sekundární) paměti

- mnohé typy vnějších pamětí nemusí být ,,pohotové'', rychlé
  - ✓ terciální paměti — optické paměti, magnetické pásky, . . .
  - ✓ dále pak paměti typu WORM (write-once, read-many-times)
  - ✓ stále se musí ale vykonávat jejich správa

Systém ochran, bezpečnost

- Ochrana
  - ✓ mechanismus pro řízení přístupu k systémovým a k uživatelským zdrojům
- Bezpečnost
  - ✓ obrana systému proti vnitřním i vnějším útokům, odmítnutí služby, červi, viry, zcizení identity, zcizení služby, ...
- Systém ochran je součástí všech vrstev OS
- Systém ochran musí
  - ✓ rozlišovat mezi autorizovaným a neautorizovaným použitím
  - ✓ poskytnout prostředky pro své prosazení

Systém ochran, bezpečnost

- OS obvykle primárně rozlišuje uživatele, aby mohl určit co kdo může dělat
  - ✓ identita uživatele se obvykle reprezentuje jménem a asociovaným číslem (*user ID*, *uid*, *security ID*, ...), po jednom na jednoho uživatele
  - ✓ uid se spojuje se soubory, procesy, ..., které uživatel vlastní a odvozují se z ní přístupová/manipulační práva
  - ✓ obvykle lze nějakou formou pracovat se skupinami uživatelů, ty pak mají své gid (group ID)
  - ✓ pokud uživatel používá bezpečný nástroj, původně vlastněný uživatelem s vyššími přístupovými právy, může takový nástroj být uživatelem provozovaný s právy jeho tvůrce, efektivní uid (effective ID) téhož programu se může měnit

Interpret příkazů

- Většina zadání je předávána operačnímu systému řídícími příkazy, které zadávají požadavky na
  - ✓ správu a vytváření procesů
  - ✓ ovládání I/O
  - ✓ správu sekundárních pamětí
  - ✓ správu hlavní paměti
  - ✓ zpřístupňování souborů
  - ✓ ochranu
  - ✓ práci v síti, ...
- program, který čte a interpretuje řídicí příkazy se nazývá v různých OS různými názvy
  - ✓ command-line interpreter, shell, command.com, ...
- základní funkcí interpretu příkazů je získávat řídicí příkazy a zajišťovat jejich provedení

Hlavní přístupy k rozvoji architektu OS

- Mikrojádrová architektura, *Microkernel architecture*
  - ✓ mikrojádro OS: pouze správa paměti, správa procesorů, komunikace mezi procesy
  - ✓ ostatní služby OS plní procesy (tzv. servery) běží v uživatelském režimu, pro mikrojádro mají charakter aplikací
  - ✓ koncept zjednodušuje implementaci OS, pružnost, je vhodný pro distribuované prostředí

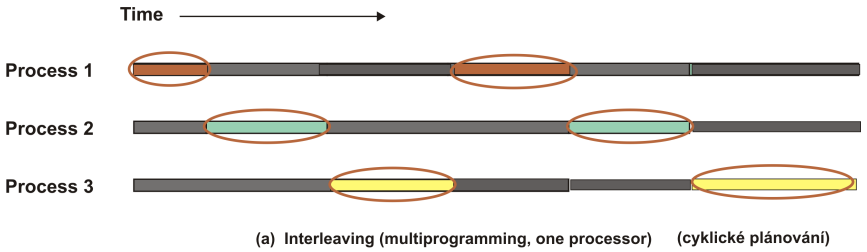
Hlavní přístupy k rozvoji architektu OS

- Multi-vlákna, *Multithreading*
  - ✓ Proces lze řešit více souběžnými sekvenčními toky operací – vlákny
  - ✓ vlákno je jednotkou plánování, nikoliv subjekt vlastnící zdroje, vlákno je částí procesu, proces je subjekt vlastnící zdroje
  - ✓ vlákno má svůj kontext a svoji datovou oblast umožňující volání podprogramů
  - ✓ programátorovi dává silnější nástroje pro modularitu aplikace a časového řízení událostí souvisejících s aplikací

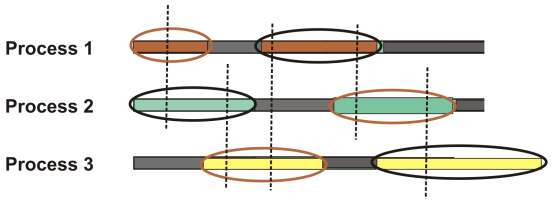
Hlavní přístupy k rozvoji architektu OS

- Symetrický multiprocesing, *Symetric multiprocessing*
  - ✓ počítač vybavený n shodnými procesory se společnou paměti a se společnými IO
  - ✓ velký výkon – OS plánuje provádění vláken / procesů, n procesů / vláken může běžet paralelně
  - ✓ vysoká dostupnost výkonu – porucha 1 procesoru nezastaví systém
  - ✓ inkrementální zvyšování výkonu – doplňováním procesorů
  - ✓ snadná úměrnost dostupného výkonu aplikaci

Symetrický multiprocesing



(a) Interleaving (multiprogramming, one processor) (cyklické plánování)



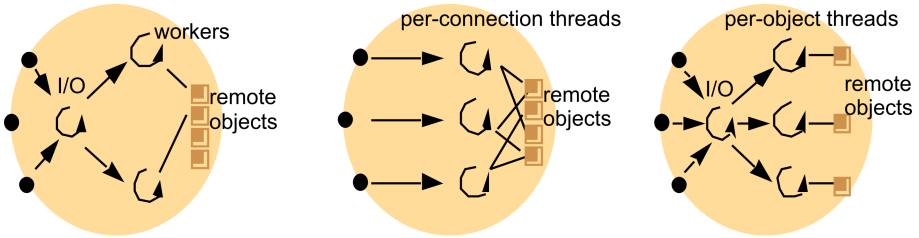
(b) Interleaving and overlapping (multiprocessing; two processors)

Výpočetní prostředí

- tradiční počítače
  - ✓ pojem tradice se v průběhu času dost mlží
  - ✓ kancelářskáprostředí –
    terminály připojené ke střediskovému počítači, . . .
    PC připojené k síti, . . . ,
    webovská interní / externí dostupnost protálů, . . .
  - ✓ domácí sítě –
    izolovaný PC, . . . , síť, s firewallovou ochranou, . . .
- klient-server model
  - ✓ jednoduché terminály byly nahrazeny propracovanými PC – klienti
  - ✓ centrální výpočetní zdroj (server) reaguje na požadavky klientů
  - ✓ server a klienti typicky propojeni sítí
  - ✓ server – databáze, katalog souborů, tiskový systém, . . .

Příklady architektur klient-server s více vlákny

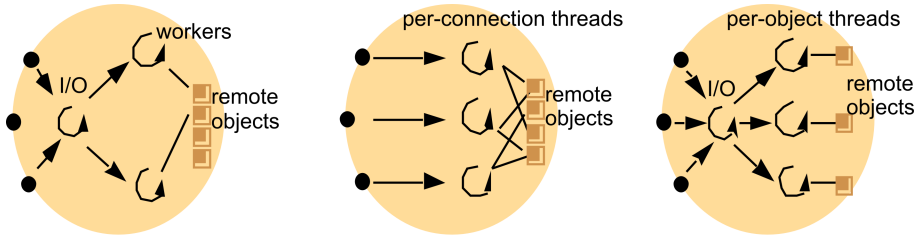
- Thread-per-request Architecture
  - ✓ I/O vlákno serveru vytvoří pro řešení každého nového požadavku klienta na zpřístupnění vzdáleného objektu nové vlákno (*worker*)
  - ✓ po splnění služby se vlákno *worker* samo zruší
  - ✓ vlákna nesdílí žádnou frontu – maximální propustnost
  - ✓ časté vytváření / rušení vláken – vyšší režie



a. Thread-per-request b. Thread-per-connection c. Thread-per-object

Příklady architektur klient-server s více vlákny

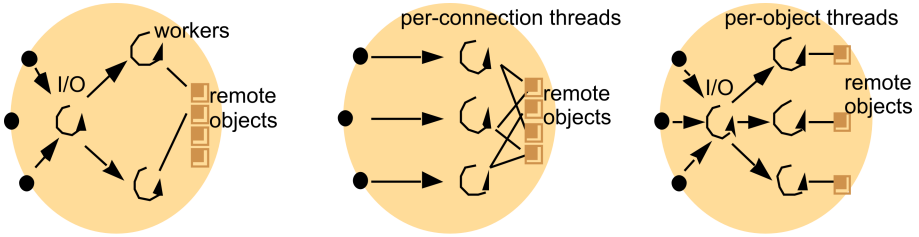
- Thread-per-connection Architecture
  - ✓ Server vytvoří nové vlákno pro každé spojení s jedním klientem a požadavky jednotlivých klientů řeší sekvenčně
  - ✓ po uzavření spojení s klientem se vlákno zruší
  - ✓ menší režie než v případě *Thread-per-request Architecture*
  - ✓ potenciálně niší propustnost díky frontování požadavků



a. Thread-per-request b. Thread-per-connection c. Thread-per-object

Příklady architektur klient-server s více vlákny

- Thread-per-object Architekture
  - ✓ Každý zpřístupňovaný objekt serveru je obsluhovaný samostatným vláknem
  - ✓ I/O vlákno přijímá požadavky klientů na zpřístupnění objektů
  - ✓ požadavky na týž objekt se řadí do fronty na objekt
  - ✓ vlákno se zruší při zrušení objektu



a. Thread-per-request b. Thread-per-connection c. Thread-per-object

Síťový OS vs. distribuovaný OS

- Síťový OS
  - ✓ Unix, Windows
  - ✓ OS řídicí 1 uzel sítě s vestavěnými schopnostmi pracovat se vzdálenými zdroji v síti
  - ✓ některé zdroje lze zpřístupňovat se síťovou transparencí (NFS zpřístupňující soubory v síti, . . .)
  - ✓ mnohé zdroje si zachovávají uzlovou autonomii (OS řídí procesy ve svém uzlu, plánovat procesy v jiném uzlu nelze, uživatel se musí otevírat relace v jednotlivých uzlech explicitně, . . .
- Distribuovaný OS
  - ✓ zatím v komerční, ekonomicky efektivní rovině neexistuje
  - ✓ celá síť se uživateli jeví jako jediný systém

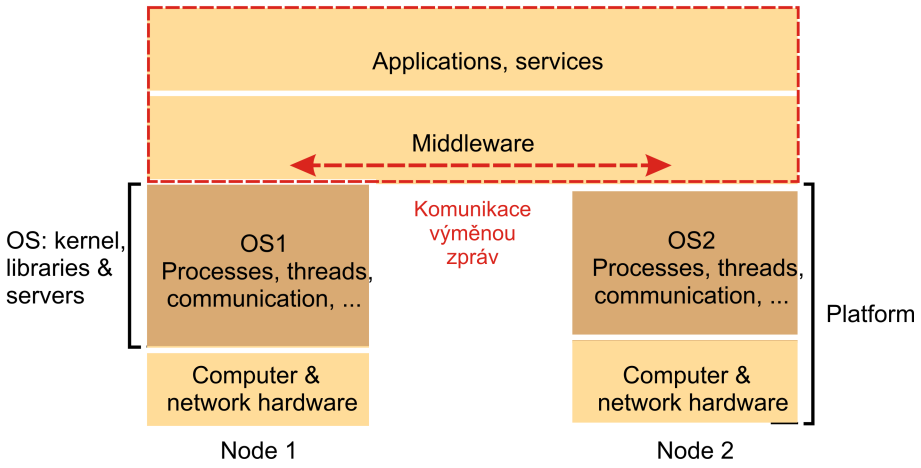
Middleware

- Softwarová vrstva ležící mezi aplikacemi a OS poskytující aplikacím programovací abstrakci a maskování heterogenity podpůrných sítí, počítačů, operačních systémů, programovacích jazyků, . . .
- Poskytuje aplikačním programátorům jednotný výpočetní model vesměs na bázi paradigmat server-klient, příp. dalších forem pro podporu distribuovaných aplikací (např. RPC) v prostředí podporovaném síťovým OS (protože distribuované OS jsou chiméra)
- middleware = procesy a objekty v počítačích propojených sítí + systém výměn zpráv
- CORBA, Java RMI, WEB Services, DCOM, . . .

Middleware

- Nadstavba síťového OS řešící neexistenci distribuovaných OS
- OS běžící v uzlu (jádro OS + služby na uživatelské úrovni) poskytuje lokální abstrakce a ty využívá middleware pro implementaci mechanismů pro vzdálené manipulace s objekty a procesy v uzlech
  (řeší se uváznutí, transakce, obnova po výpadku, vzájemné vyloučení kritických sekcí procesů, shoda, . . . , . . .)
- Kombinace middleware a síťového OS je akceptovatelné kompromisní řešení vyváženosti mezi požadavky na autonomii na jedné straně a síťovou transparentostí na druhé straně

Typové hierarchické uspořádání distribuovaného systému



Volání služeb systému, System Calls

- Volání služeb systému podporuje rozhraní mezi běžícími procesy a operačním systémem
- programátorské rozhraní na služby OS
  - ✓ genericky dostupné na úrovni symbolického strojové jazyka (assembly-language)
  - ✓ Jazyky určené jako náhrada symbolického strojové jazyka pro systémové programování umožňují volat *system calls* přímo, (např. knihovny C, C++)
- Aplikační programy si služby OS zpřístupňují spíše přes API (*Application Program Interface*) vysoké úrovně než přímým voláním systému
  - ✓ snadnější přenositelnost programů
  - ✓ srozumitelnější vyjádření

Typy / kategorie poskytovaných služeb, System Calls

- Řízení procesů, *Process Control*
  - ✓ zavedení programu do hlavní paměti a start jeho řešení – procesu, ukončení procesu (normálně, nestandardně – s indikací chyby)
  - ✓ fork(), exec(), wait(), abort(), ...

Ilustrativní výčet typů

- ✓ load, execute, create process, fork, ...
- ✓ end, abort, terminate process, ...
- ✓ get process attributes, set process attributes
- ✓ wait for time
- ✓ wait event, signal event
- ✓ allocate memory, free memory
- ✓ ...

Typy / kategorie poskytovaných služeb, System Calls

- správa souborů, *File management*
  - ✓ Manipulace s daty ve správě systému souborů
  - ✓ schopnost číst, zapisovat, vytvářet a rušit soubory dat na vnějších pamětech a data v souborech seskupená
  - ✓ open(), close(), chmod(), link(), stat(), creat(), get(), put(), ...

Ilustrativní výčet typů

- ✓ create file, delete file
- ✓ open, close
- ✓ read, write, reposition
- ✓ get file attributes, set file attributes, ...

Typy / kategorie poskytovaných služeb, System Calls

- Správa IO zařízení, *Device Management*
  - ✓ Provedení I/O operace, IO Operation
  - ✓ uživatelský program nesmí provádět I/O operace přímo, OS musí proto poskytovat prostředky vykonávající I/O
  - ✓ ioctl(), select(), read(), write(), ...

Ilustrativní výčet typů

- ✓ request device, release device
- ✓ read, write, reposition
- ✓ get device attributes, set device attributes
- ✓ logically attach, logically detach devices
- ✓ ...

Typy / kategorie poskytovaných služeb, System Calls

- Údržba informací, *Information Maintenance*
  - ✓ time(), acct(), gettimeofday(), ...

Ilustrativní výčet typů

- ✓ get time, get date, set time, set date
- ✓ get system data, set system data
- ✓ get process attributes, file attributes, device attributes
- ✓ set process attributes, file attributes, device attributes
- ✓ ...

- Detekce chyb a chybové řízení, Error Control
  - ✓ záruka za správnost výpočtu detekcí chyb v CPU, v paměťovém hardware, v I/O zařízeních a v programech

Typy / kategorie poskytovaných služeb, System Calls

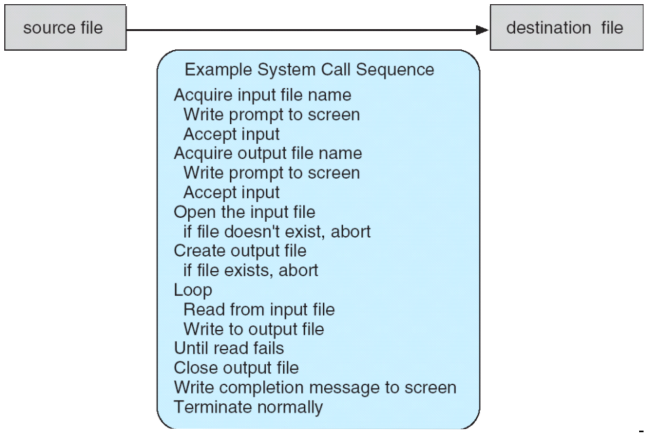
- Komunikace, komunikace mezi procesy
  Communications, Interproces Communications (IPC)
  - ✓ výměna informací mezi procesy realizovaná
    – buď to v rámci jednoho počítače
    – nebo mezi různými počítači pomocí sítě
  - ✓ implementace buď to sdílenou pamětí nebo předáváním zpráv
  - ✓ socket(), accept(), send(), recv(), wait(), signal(), . . .
- Komunikace, *Communication*
  - ✓ create communication connection, delete communication connection
  - ✓ send message, receive message
  - ✓ transfer status information
  - ✓ attach remote devices, detach remote devices

Vnitřní služby OS

- nejsou určeny k tomu, aby pomáhaly přímo uživateli,
- slouží pro zabezpeční efektního provozu systému
- Přidělování prostředků (zdrojů), Resource Allocation
  - ✓ mezi více souběžně operujících uživatelů resp. jejich procesů
- účtování, resp. protokolování, Accounting
  - ✓ udržování přehledu o tom, kolik kterých zdrojů systému který uživatel používá
  - ✓ cíl – účtování za služby, sběr statistik pro plánování, . . .
- ochrana a bezpečnost, Protection / Security
  - ✓ péče o to, aby veškeré přístupy k systémovým zdrojům ,,byly pod kontrolou''

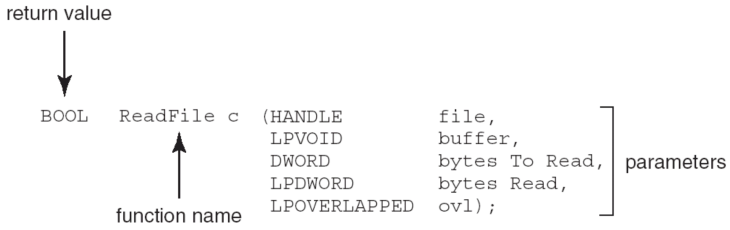
Volání služeb systému, System Calls

- Ilustrační příklad – kopie jednoho souboru do jiného souboru



Ilustrace standardního API – Win32

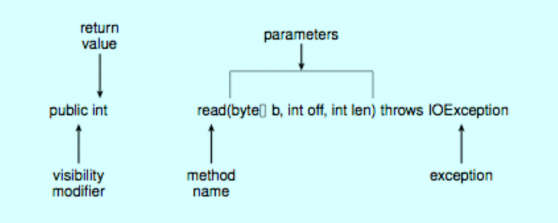
- ✓ Funkce ReadFile()
- ✓ čtení ze souboru dat



- ✓ HANDLE file – jméno souboru, ze kterého se čte
- ✓ LPVOID buffer – cílová vyrovnávací paměť
- ✓ DWORD bytesToRead – délka vyrovnávací paměti
- ✓ LPDWORD bytesRead – délka přečtených dat
- ✓ LPOVERLAPPED ovl – čekat / nečekat na konec operace

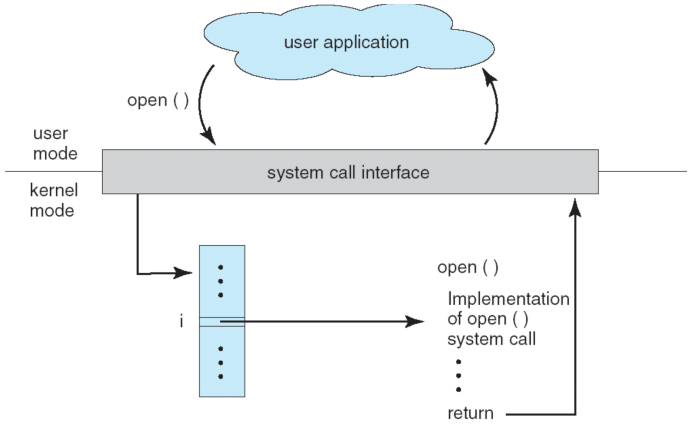
Ilustrace standardního Java API

- ✓ metoda read() z třídy java.io.InputStream
- ✓ metoda vrací int reprezentující počet přečtených bytů

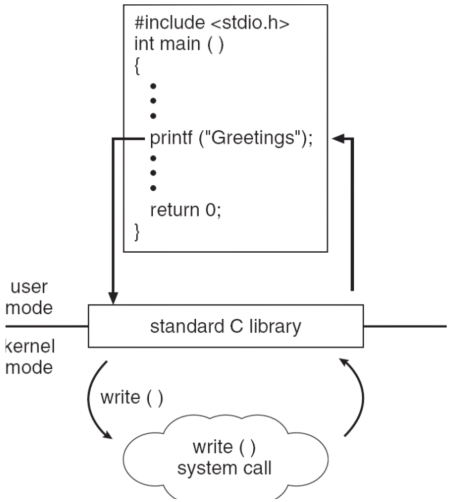


- ✓ IOException – odbočka pro řešení IO chyby
- ✓ byte [] b – cílový buffer
- ✓ int off – počáteční offset v b, kam se zapisují data
- ✓ int len maximum čtených bytů

Vztah API – volání systému – OS



Volání systému, příklad standardní knihovny C



Volání systému, příklad, program pro tisk adresáře

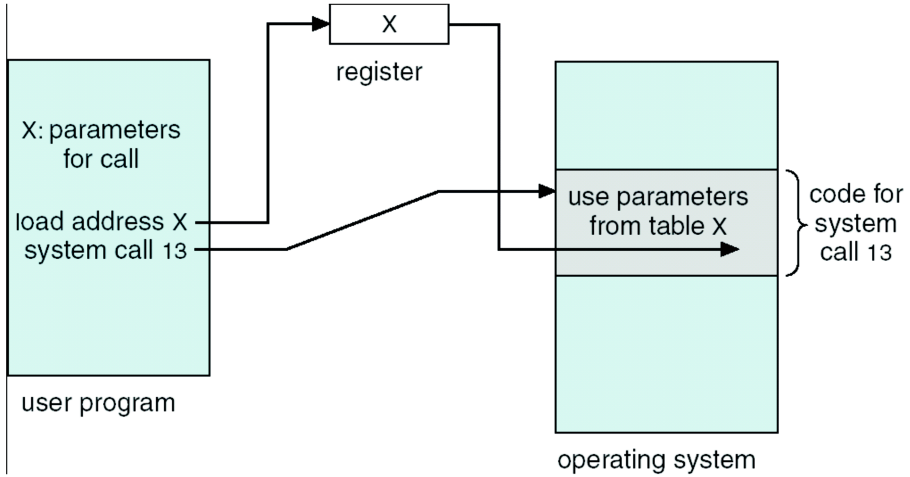
```
#include    <sys/types.h>
#include    <dirent.h>
#include    "ourhdr.h"

int main(int argc, char *argv[])
{
    DIR                 *dp;
    struct dirent   *dirp;

    if (argc != 2)
      err_quit("a single argument (the directory name) is required");
    if ( (dp = opendir(argv[1])) == NULL)
      err_sys("can't open %s", argv[1]);

    while ( (dirp = readdir(dp)) != NULL)
      printf("%s\n", dirp->d_name);

    closedir(dp);
    exit(0);
}
```

Knihovní funkce obsahují instrukci vyvolávající synchronní přerušení

Metody předávání parametrů mezi běžícím procesem a OS

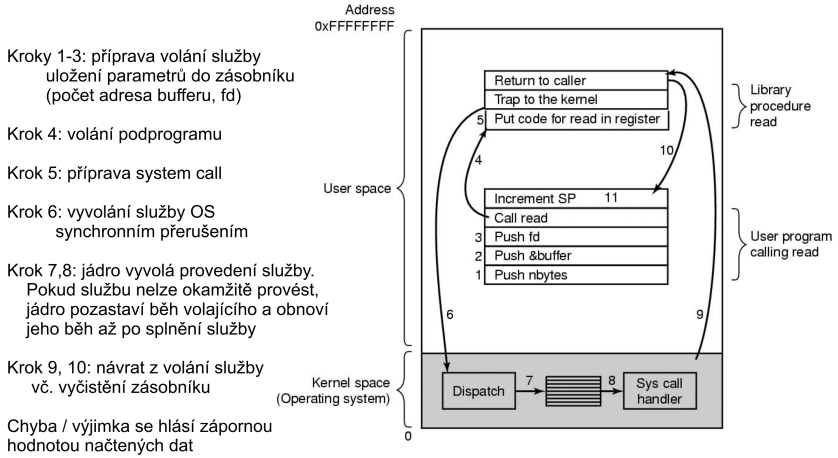
- v registrech – registry jsou dostupné procesu i OS
- v tabulce uložené v hlavní paměti –
  adresa tabulky se umístí v registru, (Linux, Solaris)
- v zásobníku –
  - zásobník je dostupný procesu i OS
  - program provede ,,push''(store),
    OS provede ,,pull''(load)

Volání systému, předávání parametrů tabulkou



Volání systému, typový příklad zásobníkem

11 kroků řešení volání služby *read(fd, buffer, nbytes)* (knihovna C)



Volání systému, příklad, maximálně jednoduchý *shell*

```
while (TRUE) {                                  /* repeat forever */
   type_prompt( );                               /* display prompt */
   read_command (command, parameters)            /* input from terminal */

if (fork() != 0) {                              /* fork off child process */
   /* Parent code */
   waitpid( -1, &status, 0);                     /* wait for child to exit */
} else {
   /* Child code */
   execve (command, parameters, 0);             /* execute command */
 }
}
```