KIV/ZOS 2003/2004
Přednáška 1

Úvod
====

!! Pokud v textu přednášek najdete něco nesrozumitelného apod., !!
!! dejte mi to prosím vědět! !!

Věci, které jsou uvedeny s poznámkou typu ”pro zajímavost” nebudu požadovat u zkoušky, ale bylo by dobré jim alespoň rozumět.

Nejčastější zkratky, použité v tomto textu:

* OS (Operating System) = operační systém
* CPU (Central Processing Unit) = procesor
* I/O (Input/Output) = vstup/výstup
* fs (file system) = systém souborů
* SW = software, softwarový
* HW = hardware, hardwarový

Moderní počítačový systém se skládá z těchto komponent:

* 1 nebo více CPU
* hlavní paměť (dále jen ”paměť”)
* disky
* klávesnice, myš, obrazovka, tiskárna, síťové rozhraní a ostatní I/O zařízení.

Celkem poměrně složitý systém, napsat program který se všemi těmito komponentami zachází správně a efektivně je pracné. Proto jsou počítače vybaveny SW vrstvou nazývanou {operační systém}, jehož úlohou je spravovat HW a poskytovat k němu programům jednodušší rozhraní.

Poznámka:

Neexistuje všeobecně přijímaná definice, co je a co není součástí OS. Dva extrémní pohledy:

* OS je vše, co dodavatel poskytuje jako OS. (Potíž - velká různorodost; v některých systémech i grafické programy, v jiných není ani textový editor.)

* OS je program, běžící po celou dobu běhu výpočetního systému; všechny ostatní programy jsou aplikace. (Potíž - některé nízkoúrovňové součásti OS mohou být zaváděny na žádost, např. ovladače v systému Linux.)

[]

My ovšem potřebujeme vědět, čím se v tomto předmětu budeme zabývat, takže se na problém musíme podívat podrobněji.

Výpočetní systém se často hierarchicky popisuje takto:

* nejníže fyzická zařízení
* nejvýše uživatel
* mezi nimi vrstvy abstrakce:

uživatelé (lidé, stroje, jiné počítače) - využívají aplikace.
/---
| aplikační programy - textové procesory, tabulkové programy, překladače...
| (využívají zdroje pro řešení problémů, do značné míry nezávislé na
| vnitřní práci OS)
+---
| systémové programy - příkazový interpret, window system, správa systému...
| (z hlediska ZOS se nepovažují za součást operačního systému)
+---

```
| operační systém - částečné zakrytí HW, vysokoúrovňové rozhraní -
| např. čtení bloku ze souboru
========holý stroj================================================
| strojový jazyk - cca 50-300 instrukcí viditelných pro programátora ve
| strojovém jazyce (používají převážně registry a hl. paměť; přesuny dat
| mezi registry a pamětí, aritmetické operace, porovnání hodnot...)
+-------------------------------------------------------------------
| mikroarchitektura - funkční jednotky vzniklé spojením fyzických
| zařízení, např. datová cesta obsahující ALU (přenést operandy z reg.
| do ALU, výpočet, ...), může být řízená HW nebo mikroprogramem
+-------------------------------------------------------------------
| fyzická zařízení - zdroje, dráty, integrované obvody...
\-------------------------------------------------------------------
```

* pro účely předmětu ZOS budeme za OS považovat tu část SW, která
  zprostředkovává aplikacím přístup k HW
* většinou běží v tzv. privilegovaném režimu procesoru, takže je chráněn před
  zasahováním uživatele resp. jeho aplikací
* aplikace běží v uživatelském režimu
* operační systém může zasahovat do běhu aplikací, aplikace nemohou zasahovat
  do běhu OS; aplikace mohou pouze OS požádat o nějakou službu

Poznámka:

Z výše uvedeného vybočují jednoduché systémy jako MS DOS a zapouzdřené systémy, kde často neplatí, že OS běží v privilegovaném režimu procesoru.

V některých moderních systémech (budeme o nich mluvit dále) běží části OS (např. systém souborů) v uživatelském režimu.

Podobně vybočují interpretované systémy (např. založené na jazyku Java).

[]

OS koordinuje a poskytuje služby aplikacím. OS můžeme přirovnat k dopravnímu systému; sám o sobě by nebyl užitečný, ale poskytuje prostředí, ve kterém jiné programy mohou vykonávat svou činnost.

Co je operační systém

Pokus o definici: OS je program, který slouží jako prostředník mezi aplikacemi a HW počítače.

* primární cíl - pohodlné používání,
* sekundární cíl - efektivní využití zdrojů.

Z toho plynou dva základní pohledy na OS - jako rozšířený stroj (pohled shora dolů) a jako správce zdrojů (pohled zdola nahoru).

OS jako rozšířený stroj

Holý stroj je primitivní a obtížně programovatelný, zejména I/O.

Např. disky:

* ovládání na úrovní HW - různý HW se liší, ovládání na úrovni HW
  obtížné (např FD: roztočit, přesunout hlavičky, zahájit čtení,
  zkontrolovat CRC/zkusit znovu, zastavit aby se neošoupala)
* nutné řešit problémy alokace a dealokace bloků dat
* pokud chce více programů sdílet stejné médium, musí se dohodnout na
  společném způsobu řešení problémů alokace, dealokace, pojmenování -
  raději jednoduchý pohled - pojmenované soubory, které mohou být čteny
  a zapisovány = řešení společného problému všech aplikací.

OS skrývá před aplikacemi pravdu o HW a poskytuje jim jednoduché

souborově orientované rozhraní (systém souborů), stejně tak skrývá přerušení (I/O, správa procesů), správu paměti a další nízkoúrovňové vlastnosti.

V tomto pohledu OS poskytuje aplikacím ekvivalent ”rozšířeného stroje”, tj. kromě strojových instrukcí jsou programům dostupné vysokoúrovňové služby (”rozšířené instrukce”) pomocí tzv. systémových volání.

My se v ZOS budeme zabývat převážně tím, jak jsou služby OS implementovány.

OS jako správce zdrojů
.....

Příklad problému: 3 programy by chtěly použít stejnou tiskárnu, což není možné současně; OS musí nějak zajistit, aby tisknout mohl vždy jen jeden.

Pohled zdola nahoru - od HW - OS jako řídící program, poskytovatel/správce zdrojů (resource allocator/manager):

Ve výpočetním systému jsou k dipozici různé zdroje (čas procesorů, paměť, I/O zařízení), úlohou OS je správná a řízená alokace programům, které je požadují.

OS zprostředkovává požadavky na zdroje a zabraňuje nesprávnému použití počítače a chybám (přístupová práva, vzájemné vyloučení). Mohou být konfliktní požadavky na zdroje - musí rozhodnout v jakém pořadí budou požadavky vyřízeny, aby to bylo efektivní a spravedlivé.

Ukážeme si, jak jednotlivé komponenty OS vznikaly jako přirozené řešení problémů pro jednotlivé generace počítačů (= historie).

Historická perspektiva
==========

Počítače - už ve 30 letech pokusy

1. počítač - ENIAC - spuštěn 15. 2. 1946
plocha tělocvičny, 18 tis. elektronek
prvky v regálech, chlazeno leteckými motory
5000 operací za sekundu

generace počítačů podle technologie:

1. gen. elektronky
2. tranzistory
3. integrované obvody
4. LSI, VLSI (mikroprocesory apod.)

z hlediska OS není tak důležité jako vývoj HW a SW architektury; časové úseky jsou zhruba stejné (vývoj OS spolu s architekturami počítačů, na kterých byly provozovány).

I. generace (1945-1955) - elektronky a propojovací desky
---
* v průběhu a po 2. sv. válce výpočetní stroje s tisíci elektronek
* programování v absolutním jazyce propojováním zdířek na desce
* od 1949 programy v paměti
* téměř všechny řešené problémy přímočaré numerické kalkulace

Stejní lidé stroj navrhli, postavili a programovali. Obvyklý způsob práce - zatrhnout blok času na rozvrhu, zastrčit desku do počítače a doufat, že žádná z 20000 elektronek neshoří. :-)

* OS ještě neexistují, ke konci programování pomocí děrných štítků (místo zapojování na desce), první assemblery, knihovny, FORTRAN
* centrální řídící program automatizace základní sekvence zaveď/přelož/zaveď/spusť

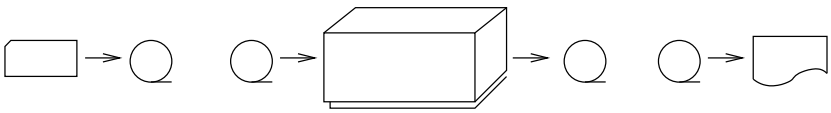
II. generace (1955-1965) - tranzistory a dávkové OS

* řádově vyšší spolehlivost => začalo mít smysl vyrábět a prodávat
* radikální změna!
* poprvé oddělení návrhářů, výroby, operátorů, programátorů, údržby
* cena několik mil. $ => pouze velké firmy, vlády, některé univerzity

Stroje v klimatizovaných sálech, týmy profesionálních operátorů. Programátor napsal program, vyděroval na štítky, krabici se štítky přinesl na výpočetní středisko a dal operátorovi; výsledek běhu vytisknut na tiskárně.

Operátor nosil věci sem a tam => ztráta drahého poč. času => rychlý vznik dávkových systémů.

Sběr úloh, přečíst ze štítků na magnetickou pásku (na levném počítači), výpočet a zápis na magnetickou pásku, tisk výsledků opět na levném stroji. Mladší dávkové systémy spooling na vstupu i na výstupu.



Často VT výpočty, FORTRAN a assembler. Typická úloha:

```
$JOB, maximální čas v minutách, číslo a jméno účtu
$FORTRAN
C    PROGRAM P1 ... program v jazyce FORTRAN
...
$LOAD      ... načíst přeložený program
$RUN       ... spustit program
  1 100.5  ... data pro program
$END       ... konec úlohy
```

Snaha o efektivitu, ale sekvenční vykonávání dávek. Ochrana systému nevyřešený problém - kdokoli dokázal systém shodit.

Typický OS je IBSYS = ”IBM SYSTEM FOR 7094”.

III. generace (1965-1980) - integrované obvody a multiprogramování

* většina výrobců 2 nekompatibilní řady počítačů:
  - slovně orientované počítače s velkým rozsahem pro VT výpočty
  - znakově orientované komerční stroje - banky a pojišťovny pro zpracování záznamů na magnetických páskách a tisk

Vývoj dvou řad počítačů drahý => IBM360 - řada kompatibilních strojů pro VT i komerci, odlišnost pouze v ceně a výkonu. Později kompatibilní následníci s využitím moderní technologie.

Síla i slabost: SW včetně OS provozován na všech typech od malých strojů (pro kopírování dat ze štítků na pásku) po velké stroje, komerční účely (zpracování výplat) i VT výpočty (předpovídání počasí).

Cit. [Tan92]:

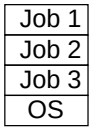
> ”Nebylo možné, aby IBM (nebo kdokoli jiný) vytvořil SW, který by splňoval všechny tyto konfliktní požadavky. Výsledkem byl enormně a výjimečně složitý OS... Skládal se z milionů řádků v assembleru, napsaných tisíci programátorů, a obsahoval tisíc na tisícátou chyb, které si vynutily vydávání nových verzí. Každá nová verze některé chyby opravila a přidala nové, takže počet chyb pravděpodobně zůstával v průběhu času konstantní.”

(Potíže při vývoji OS/360 inspirovaly knihu [Brooks1975].)

Přes výše uvedené problémy OS/360 nejspíše splňoval základní požadavky

zákazníků, navíc zpopularizoval některé techniky, které se v OS druhé generace nevyskytovaly:

* multiprogramování. V OS II. generace pokud úloha prováděla I/O, CPU čekala. Pro vědecké výpočty byla doba strávená I/O zanedbatelná; v komerčních úlohách I/O často 80-90% => nutno využít drahý čas CPU => více úloh v paměti (zpočátku konstantní počet), zavedení HW pro ochranu paměti



- každá úloha běží ve vlastní oblasti paměti; zatímco jedna úloha I/O, druhá počítá => CPU může být využita téměř na 100%

* spooling (Simultaneous Peripherial Operation On Line)
  načtení štítků na disk, po skončení některé úlohy OS zavede novou úlohu z disku; používalo se také pro výstup

OS III. generace stále dávkové systémy, čas mezi dodáním úlohy a výsledky často několik hodin. Programátoři nostalgicky vzpomínali na dobu I. generace, kdy měli celý počítač pro sebe.

Systémy se sdílením času (time-shared systems) - varianta multiprogramování, kde každý uživatel má on-line terminál, CPU střídavě vykonává jednotlivé úlohy.

CTSS (MIT 1962) - první opravdový systém se sdílením času, ale úspěch až po příchodu HW pro ochranu paměti.

Po úspěchu CTSS - MIT, Bell Labs a GE (General Electric) myšlenka: 1 velký stroj poskytující výpočetní sílu pro stovky uživatelů (model podle elektr. sítě - poskytuje tolik výkonu, kolik je třeba), naz. MULTICS (Multiplexed Information and Computing Service). Spousta dobrých myšlenek, ale... Bell Labs se odpojily, GE opustila počítačový průmysl úplně.

Příchod minipočítačů - DEC PDP1 v 1961, pam. 4k x 18 bitů, pouze $120,000!, (3.5mKč) ”prodávaly se jako housky”. Následovaly další PDP až po PDP 11 (na rozdíl od strojů IBM vzájemně nekompatibilní).

Jeden z výzkumníků Bell Labs pracujících původně na MULTICSu, Ken Thompson, našel nepoužívanou PDP-7, napsal omezenou jednouživatelskou verzi MULTICSu, vznik UNIXu a jazyka C (1969). Odtud se odehrává vývoj systému UNIX, viz např. ”Historie systémů typu UNIX” na mých stránkách.

IV. generace (1980-současnost) - mikroprocesory a osobní počítače
---
* příchod osobních počítačů (původně nazývaných mikropočítače), výrazný cenový rozdíl
* ”uživatelsky přátelský SW” (GUI vs. CLI)
* síťově a distribuované systémy

dominantní OS: MS DOS, UNIX, později NT

* MS DOS, na IBM PC kompatibilních strojích s i8088, ... <souč.> CPU
- původní verze značně primitivní, pak některé vlastnosti převzaty z UNIXu (adresáře, místo FCB služby UNIXového typu)
- grafická nadstavba Windows
- velké rozšíření - částečně díky obchodní politice MS (nezveřejňování API, per-procesor licence, začleňování kódu do aplikací pro nekompatibilitu s jinými systémy)

* UNIX - dominantní na ne-Intelovských strojích. S příchodem volně šířených klonů (Linux, *BSD) se výrazně rozšiřuje i na PC
- užíván jako síťový systém - viz níže

* oba systémy ovládány zadáváním příkazů na příkazovém řádku

- výzkum v Xerox PARC vedl ke vzniku GUI a uživatelsky přátelského SW
- první úspěšný systém Apple Macintosh
- jím inspirované grafické prostředí MS Windows nad MS DOSem (1985-1995), postupným vývojem samostatný systém (od Windows 95 po Windows Me)
- nový systém Windows NT (2000, XP), z velké části kompatibilní ale moderní architektura

V průběhu 80 let růst sítí osobních počítačů, vede ke vzniku síťových a distribuovaných OS.

Dělení OS

podle úrovně sdílení CPU

* jednoprocesový - např. MS DOS; v daném čase je v paměti aktivní jeden program

* multiprocesový (multiprogramový) - např. UNIX, Win NT; aby se využilo systémových zdrojů (sdílení CPU, paměti) je v hlavní paměti více než jeden program (= původní motivace). V současnosti nejčastěji v systémech pro více uživatelů (= současná motivace - uživatelské pohodlí).

podle typu interakce/požadavků na odezvu

* dávkový systém - uživatelské úlohy se zadávají jako sekvenční dávky, během zpracování není interakce mezi uživatelem a jeho úlohou
 - dnes jako jedna z možností pro superpočítače, clustery apod., aby byl systém vytížen; možnost vzdáleného zadávání úloh

* interaktivní - dovolí uživatelům interakci s jejich úlohami. Víceprocesové interaktivní systémy by měly poskytovat odpověď během několika sekund - CPU alokováno každému procesu malé časové kvantum = sdílení času

 - všechny běžně užívané systémy: Windows, UNIX & Linux...

* OS reálného času (real time, RT)
 - konkrétní aplikace mají přísné požadavky na čas odpovědi (řídící počítače, multimédia)
 - OS které se těmto požadavkům podřizují = OS reálného času

  V řídících aplikacích často dobře definované, časově ohraničené požadavky na čas odpovědi (jinak problém: co kdyby ”zastav motor výtahu” přišlo pozdě?), jiným aplikacím stačí ”nejlepší snaha systému”.

  2 typy - soft, hard RT systémy:

- hard real time (RT) system: zaručuje odezvu systému v ohraničeném čase => všechna zpoždění a režie systému ohraničeny => omezení funkčnosti OS; není systém souborů, virtuální paměť apod. Nelze zároveň sdílení času, žádný z univerzálních systémů neposkytuje, nebudeme se zde zabývat. Použití: řízení výroby, robotika, telekomunikace.

- soft RT system: RT úlohy mají prioritu před vším ostatním (často i před činností jádra OS), avšak nezaručuje odezvu v ohraničeném čase. Může být spolu se sdílením času, např. některé systémy typu UNIX (RT Linux). Použití: multimédia, virtuální realita.

existují i jiná dělení, např.:

* podle velikosti HW - pro superpočítače ... pro vestavěné počítače, mobilní telefony, inteligentní čipové karty; dnes už často není rozdíl, pro vestavěné systémy se užívá kromě specializovaných systémů také Linux apod.
* podle míry distribuovanosti

- klasické = centralizované jednoprocesorové a víceprocesorové
- paralelní = v aplikačních oblastech, kde požadavky překračují možnosti jednoprocesorových systémů (technologické limity - rychlost logických obvodů) - více procesorů paralelně vykonává jednu úlohu
- síťové OS -- nejsou podstatně odlišné od jednoprocesorových; mají síťové rozhraní + nízkoúrovňové ovládání + programy pro vzdálené přihlášení a přístup k souborovému systému. Uživatel se přihlašuje na vzdálený stroj, kopíruje soubory mezi stroji...
- distribuované systémy - z více počítačů na síti vytvářejí virtuální uniprocesor, uživatel neví, kde běží programy nebo kde jsou umístěné soubory

* podle počtu uživatelů na jednouživatelské (DOS, OS/2, NT(?)), víceuživatelské (UNIX, VMS...)
* podle funkcí - univerzální, specializované (databázové, ...)

Komponenty systému

* procesy; proces = běžící program; potřebuje minimálně:
  - čas CPU = být vykonáván,
  - paměť = mít kde běžet (kód a data)
  - vstupy a výstupy = soubory a I/O zařízení
* správa hlavní paměti
  - alokace a dealokace paměti podle potřeby
  - udržuje informaci která část paměti je používána a kým
* soubory
  - vytváření a rušení souborů
  - vytváření a rušení adresářů
  - primitiva pro manipulaci se soubory a adresáři
  - správa volného prostoru vnější paměti, alokace
  - mapování souborů na vnější paměť
  - rozvrhování diskových operací
* I/O podsystém
  - správa paměti pro buffering, caching, spooling
  - společné rozhraní ovladačů zařízení
  - ovladače pro specifická zařízení
* síť (networking) [nebudeme se zde zabývat]
* ochrana a bezpečnost = ke zdrojům smí přistupovat pouze autorizované procesy
  - specifikace přístupu
  - mechanismus ochrany (souborů, paměti)
* uživatelské rozhraní - původně součást monolitického OS, nyní většinou samostatný program
  - CLI (command line interface) - zadávání příkazů
  - GUI (graphical user interface) - ikony reprezentující programy, soubory, fce systému; výběr z nabízených možností

Vyvolání služby systému

Většina moderních CPU 2 režimy:

* privilegovaný (režim jádra - všechny instrukce dovoleny),
* uživatelský (problémový - I/O a některé další instrukce zakázány, CPU je neprovede).

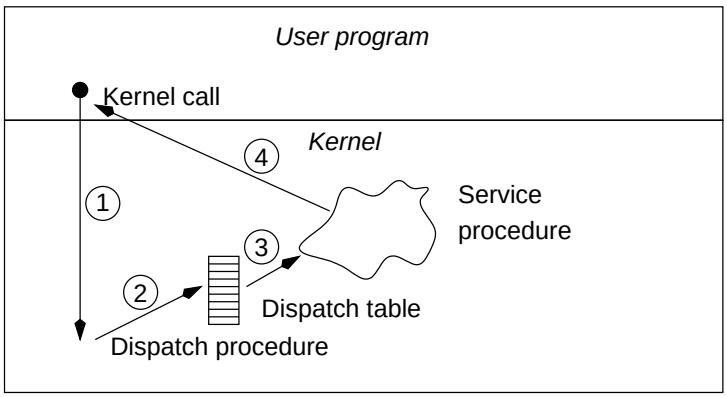
V uživatelském režimu běží aplikace a systémové programy (dále jen ”aplikace”), v privilegovaném režimu běží pouze vlastní OS (dále ”jádro OS”).

Aplikace nemohou samy provádět I/O apod., o to musí požádat jádro OS vyvoláním služby systému.

Jak aplikace vyvolá službu:

- parametry se uloží na určené místo (registry, zásobník)
- vyvolá se speciální instrukce, která vyvolá obslužnou proceduru v jádře a zároveň se přepne do privilegovaného režimu (1)

- OS zjistí která služba je vyvolána, převezme parametry, provede službu (2, 3)
- vrátí se zpátky do aplikace, zároveň přepne CPU zpět do uživatelského režimu (4)



Např. zápis řetězce na obrazovku (viz POT, systém CP/M & procesor Z80):

```
@vzít z Linuxu
                ld      c, 9       ; do C číslo služby 9 = vypsat řetězec
                ld      de, string ; do DE adresa řetězce
                call    5          ; volání systému
                ....
          ;
          string: db      'Retezec$'
```

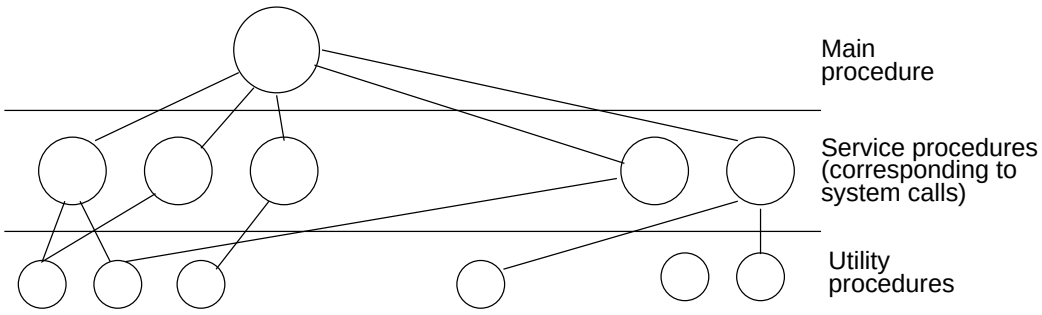
Různé OS poskytují různé služby, i mechanismy volání jádra se liší. Programovací jazyky zakrývají služby systému tak, aby se jevily jako běžné knihovní fce.

Architektury (= struktura) OS

Zde si popíšeme nejčastější možnosti, může být OS strukturován.

* monolitický OS
- jeden spustitelný soubor, který běží na holém počítači
- uvnitř OS existují moduly pro jednotlivé funkce (fs, proc...)
- napsáno jako jeden program, řízení se předává voláním podprogramů.
- většina starších systémů (UNIX a Linux, VMS, OS/360, MS DOS)

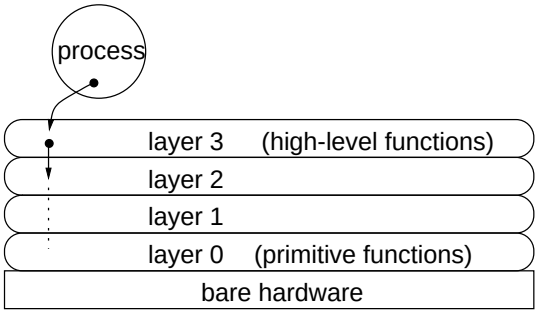
Problém: monolitický OS je "jedna velká hromada", proto pokusy různě strukturovat.



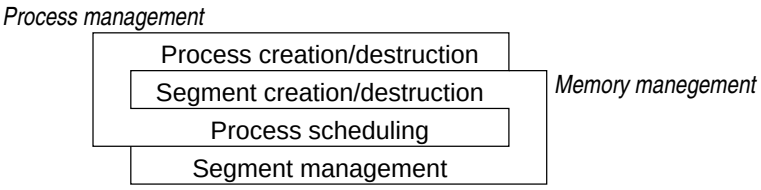
* vrstvený
- hierarchie procesů - nejníže primitivní vrstvy komunikující s HW, každá vyšší úroveň poskytuje abstraktnější virtuální stroj
- vrstvení může být logické nebo s HW podporou; pokud je s HW podporou - nižší vrstva volána z vyšší podobným způsobem jako systémové volání, vrstvy nelze obcházet
- výhoda - možnost výstavby systému od nejnižších vrstev, vyšší vrstvy

využívají primitiv poskytovaných nižšími vrstvami
- např. THE, MULTICS

Např. operační systém THE: na úrovni 0 virtualizace CPU (vrstva provádí přepínání mezi procesy, tj. vyšší vrstvy už mohou předpokládat existenci procesů), na úrovni 1 virtualizace paměti (vyšší vrstvy se nemusejí zabývat umístěním částí procesů v paměti) atd.

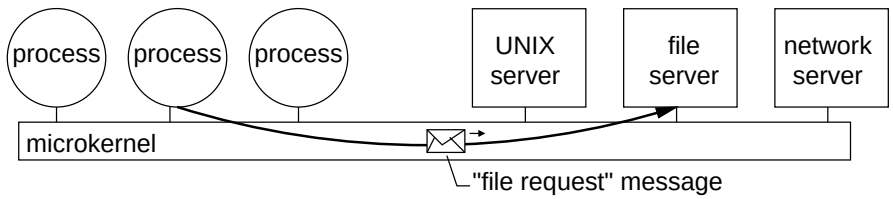


* funkční hierarchie
- problém vrstvených systémů - obtížné rozčlenit OS do striktní hierarchie vrstev, vznikají vzájemné závislosti
- např. správa procesů vs. správa paměti: pro procesy je zapotřebí paměť; pokud je paměť plná, nesmí se některé procesy vykonávat
- některé moduly vykonávají více funkcí, mohou být na více úrovních v hierarchii
- např. Pilot



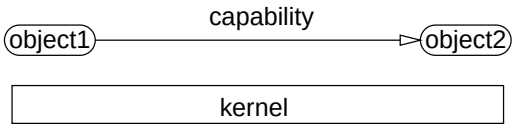
* model klient-server = systémy založené na mikrojádře
- většinu činností OS vykonávají samostatné procesy mimo jádro (servery, např. systém souborů)
- mikrojádro = vrstva nad holým strojem, poskytuje pouze nejdůležitější nízkoúrovňové funkce potřebné pro implementaci OS (nízkoúrovňovou správu procesů, adresový prostor, komunikaci mezi adresními prostory, někdy obsluhu přerušení a vstupy/výstupy)
- pouze mikrojádro v privilegovaném režimu

- výhody - vynucuje si modulárnější strukturu, snadnější tvorba distribuovaných OS (klienti a servery mohou komunikovat po síti)
- nevýhody - obecně složitější návrh systému, režie
- mnoho moderních OS: QNX, Hurd, L4, OSF/1, MkLinux, Amoeba...



* objektově orientovaná struktura
  - systém je množina objektů (soubory, HW zařízení...)
  - capability [čti kejpa-] = odkaz na objekt + množina práv definujících operace, spravuje jádro
  - jádro si vynucuje tento abstraktní pohled, kontroluje přístupová práva

- iAPX 432, částečně Windows NT (resp. Windows 2000 resp. Windows XP)



Existuje ještě celá řada možností jak strukturovat OS, ale většina současných OS monolitická (Linux), založená na mikrojádře (GNU Hurd) nebo objektově orientovaná (Windows NT v rámci možností).

❇