ARCHITEKTÚRA OPERAČNÉHO SYSTÉMU

Štruktúra OS

- Rôzne pohľady na OS
  - *Funkčný* (čo robí)
  - *Komponentný* (pohľad designera)
  - Cez *služby* (pohľad programátora)
  - *Štrukturálny* (pohľad implementátorov)
  - a iné

Funkční pohľad - Funkcie OS

- **Vykonanie programov a ich obsluha:**
  - Odštartovanie programu, obsluha jeho vykonania a prezentacia výsledkov
- **I/O operácie:**
  - Mechanizmus pre spustenie a obsluhu I/O
- **Správa súborového systému:**
  - Tvorba, manipulácia a údržba súborov

Funkcie pokr.

- **Komunikácie:**
  - medzi procesmi jedného užívateľa
  - medzi procesmi rôznych užívateľov
- **Získavanie a obsluha prerušení :**
  - pri porušení ochrany
  - bezpečnosť v prípade výpadku napájania
  - odhalenie nežiaducich stavov – došiel papier v tlačiarni

Funkcie pokr

- **Prideľovania prostriedkov :**
  - vrátane prideľovania CPU, pamäte a I/O zariadení
- **Účtovanie:**
  - za účelom platby a štatistiky
- **Ochrana:**
  - integrity dát užívateľov
  - pred neoprávneným prístupom
  - udržiava záznamy o nesprávnych pokusov o prihlásení

Komponentný pohľad na OS

- **Komponenty**
  - procesy
  - pamäť
  - I/O
    - zariadenia
    - súbory
  - ochrana
    - užívatelia
- Tieto komponenty v praxi nie sú striktne odlíšené

Procesy

- *Proces* je
  - *dynamická entita,* definovaná vykonaním programu
  - typicky, *program* + *run-time* informácie
    - t.j. mapovanie pamäte, hodnota PC, atď.
- OS sa snaží spracovávať všetky procesy rovnakým spôsobom
- Procesy sú
  - užívateľské
  - systémové
- Jeden proces môže vytvoriť ďalšie procesy
    - shell procesy
    - užívateľské procesy
    - systémové procesy

Správa procesov

- Operácie
  - Tvorba a ukončenie
  - Zastavenie a spustenie
    - kvôli prerušeniam, prepinaním kontextu ...
  - Synchronizácia procesov
    - pri prístupe k zdieľaním prostriedkom, pri I/O
  - Komunikácie procesov
  - Odhalenie a obsluha uviaznutia

Správa pamäte

- *Veľmi dôležitá!!*
  - snaha spracovávať čo najviac informácií na vyššej úrovni hierarchie
- Operácie
  - *prideľovanie* hlavnej pamäte aktívnym procesom
  - *spravovanie* voľnej a obsadenej pamäte
  - *zrušenie pridelenia pamäte*
  - politiky pre pridelenie a zrušenia pridelenia

Správa sekundárnych pamäti a správa I/O

- Operácie pre sekundárne pamäte
  - Správa voľných sektorov na disku
  - Prideľovanie diskového priestoru súborom
  - Plánovanie prístupu na disk za prítomnosti viacerých požiadaviek na I/O
- I/O operácie
  - Mechanizmus pre správu bufrov v radičoch
  - OS akceptuje požiadavku na I/O a odovzdá ju príslušnému driveru
  - Device drivers – ovládače zariadení

Súbory

- Užívateľské a systémové programy a dáta
- Podpora súborového systému
  - Logická organizácia dát
  - Pomoc pri organizovaní informácií pri obsluhe virtuálnej pamäte

Správa súborov

- Podporuje
  - Tvorbu a zrušenie súborov a adresárov
  - Primitíva pre manipuláciu so súbormi a adresármi
  - Prenos súborov medzi OP a disku pri správe štruktúr
  - Spolupráca so správcom pamäte
  - Zálohovanie a ochrana

- Ochrana – už prediskutovaná

OS z hľadiska používateľa /programátora

- Príkazový interpreter
- Dve úrovne špecifikácie
  - Systémové volania
    - interfejsy, cez ktoré procesy volajú určité funkcie OS
  - Systémové programy
- Štandardizačné práce na tejto úrovni
  - e.g. Posix, network protocols, ...

Príkazový interpreter 1

- Môže byť súčasťou jadra alebo môže byť proces, cez ktorý sa pristupuje k jadru, nazvaný *shell*
- Prvý prípad – MS DOS, druhy – Unix
- Veľké rozdiely medzi rôznymi OS v tomto smere – vrátane GUI

Príklad z MS-DOSu:

- Jednoduchší ako *multiprogramové* OS
  - Vyvolá sa príkazový interpreter
  - Ak príkaz má vykonať program
    - interpreter použije systémové volanie, ktoré zavedie program, ktorý sa bude vykonávať
    - niekedy, keď pamäť nepostačuje, veľké častí interpretera sa prepíšu.
  - Pri ukončení programu sa použije systémové volanie pre ukončenie procesu.
  - Po ukončení procesu sa použije ďalšie systémové volanie pre opätovné zavedenie príkazového interpretera a pokračuje sa obdobným spôsobom s ďalším príkazom

Príkazový interpreter pokr.

- Vidíme ho vždy keď sa prihlásime
- Príkazy obyčajne zahrňujú:
  - tvorbu a zrušenie procesu (implicitne)
  - I/O obsluha
  - manipulácia so súbormi
  - komunikácie
    - napr e-mail.
  - ochrana
    - zmena prístupu read/write k súborom, adresárom, ...

Príkazový interpreter

- Interpreter *obsahuje* kód pre vykonanie príkazu

  V tom prípade

  - interpreter akceptuje príkaz,
  - zoberie parametre príkazu (meno súboru , ..) a ide do príslušného segmentu kódu.

  - Existuje limit v počte príkazov, ktoré môžu byť spracované, pretože to vplýva na veľkosť interpretera

Príkazový interpreter pokr.

- Interpreter používa príkaz ako volanie systémovej rutiny
  - odovzdáva parametre
- Tento prístup dovoľuje rozšírenie
  - pridanie nového príkazu znamená len
    - pridanie novej procedúry
    - mapovanie syntaxe príkazu na štartovaciu adresu procedúry

Shell: vzor 3

- Interpreter vytvorí nový proces
  - Príklad z Unixu : **delete** G
    - interpreter
      - nájde systémový súbor, nazvaný **delete**,
      - odovzdá parameter G,
      - vykoná program.
  - Obmedzenia
    - záťaž pamäte pri ukladaní nových procedúr,
    - obmedzený počet parametrov, ktorý môže byť odovzdávaný.

Riadenie procesov

- *load, execute, end* (normálne), *abort*, *create/terminate* processes,
- *get* a *set* attribúov procesu
- *wait* (čakanie na určený čas),
- *wait* event, *signal* event
  - (čakanie na ukončenie V/V operácie a signal pre čakajúci proces),
- *allocate* memory

Práca so súbormi

- *create*/*delete* file
- *open*/*close* file
- *read*/*write* a *reposition*
- *get*/*set* file attributes (ochrana, umiestnenie v súborovom systéme...)

Manipulácia so zariadeniami

- *request* and *release* device, *read*/*write*, and *reposition* (napr. magnetická páska)
- *get* and *set* attributes a logické *attach*/*detach* device (potom, čo sa požiada resp. uvoľní zariadenie)
  - Tieto systémové volania sú veľmi podobné spôsobom, akými sa spracovávajú súbory
  - Veľa systémov sa pozará na súbory a zariadenia rovnakým spôsobom
  - Zariadenia sú označené ako „špeciálne súbory“ (napr. Linux, Unix)

Manipulácia s informáciami

- *get*/*set* time (informácie o čase a dátume),
- *get*/*set* system data (počet procesov, atribúty ako využitie pamäte a CPU určítimi procesmi, zmena priority procesu pre plánovanie apod.)
- *get*/*set* process and device attributes (vrátane spustenia a ukončenia tlačových úloh, resetovania tlačiarni po doplnení papiera apod.

Komunikácie

- *create/delete* links (medzi procesmi ako napr. keď ku danému programu a tlačiarni je vytvorené spojenie)
- *send/receive* messages (medzi procesmi, pre výmenu buď systémových alebo užívateľských informácií)
- Prenos stavových informácií (napr. stav určitého komunikačného zariadenia)
- Spôsob komunikácie môže byť:
  - Pomocou *správ,* keď procesy explicitne posielajú a dostavajú správy
  - Pomocou *zdieľanej pamäte*, kde sa prístup k informáciam vykonáva čítaním a zápisom – *read, write*

Systémové programy

- Každá z vypočítaných funkcií je obyčajne podporovaná systémovým programom
- Ešte sú
  - aplikačné programy - programy, ktoré využívajú OS a sú skoro jeho súčasťou

Aplikácie

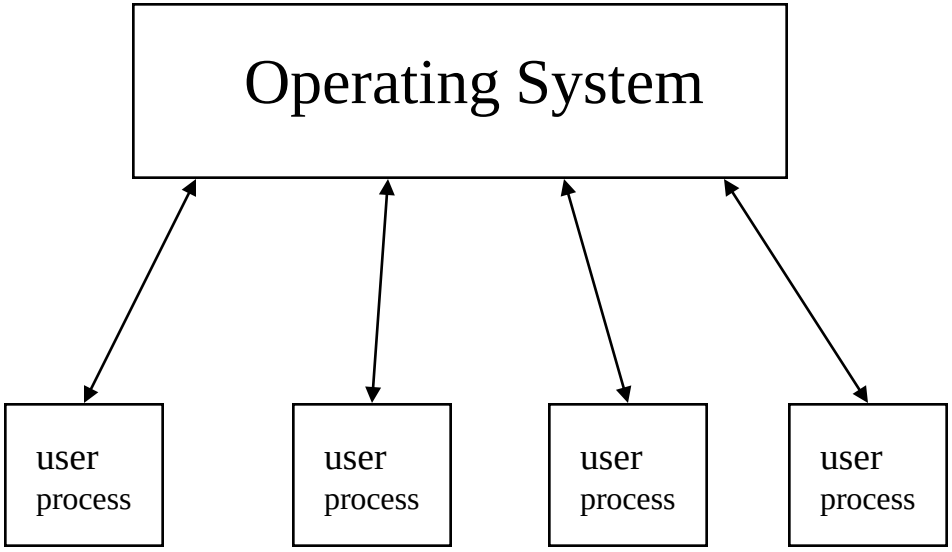
- **Podpora programovacích jazykov** - kompilátory, asemblery, ladiace programy
- **Zavadzacie a spojovacie programy (loadery a linkery)**
- **Aplikčné programy:**
  - Editory – textové a grafické, tabuľkové procesory, databázové systémy, hry a mnoho iných aplikácií
  - **Príkazový interpreter**
  - **...**

OS z hľadiska implementatora

- OS ako klient-server systém
  - jeden server
  - viacej serverov
  - obsluhuje sám seba
- Vrstvová štruktúra
  - mikrojadro
- Virtuálne stroje
- Návrh a implementácia OS

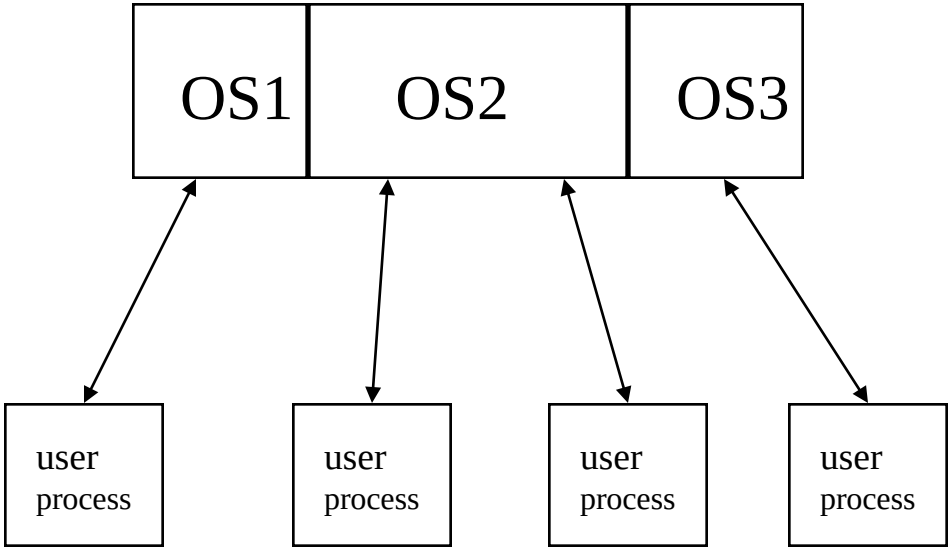
Klient-server

- Jeden server



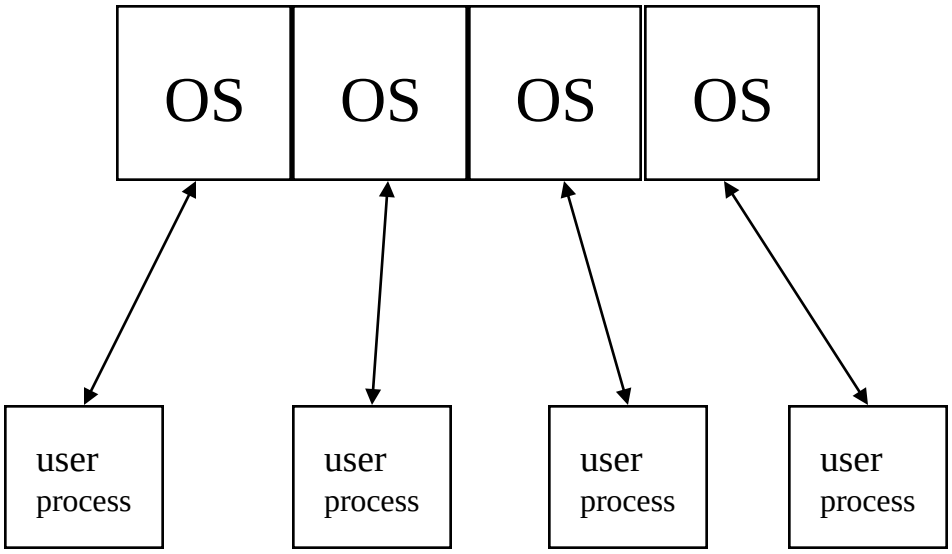
Klient-server

- Viacej serverov

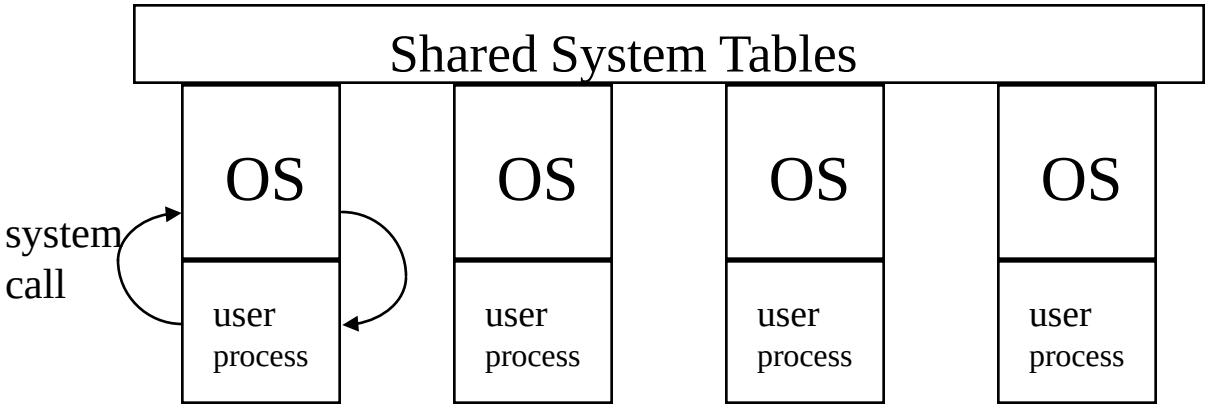


Klient-server

- Obsluhuje sám seba

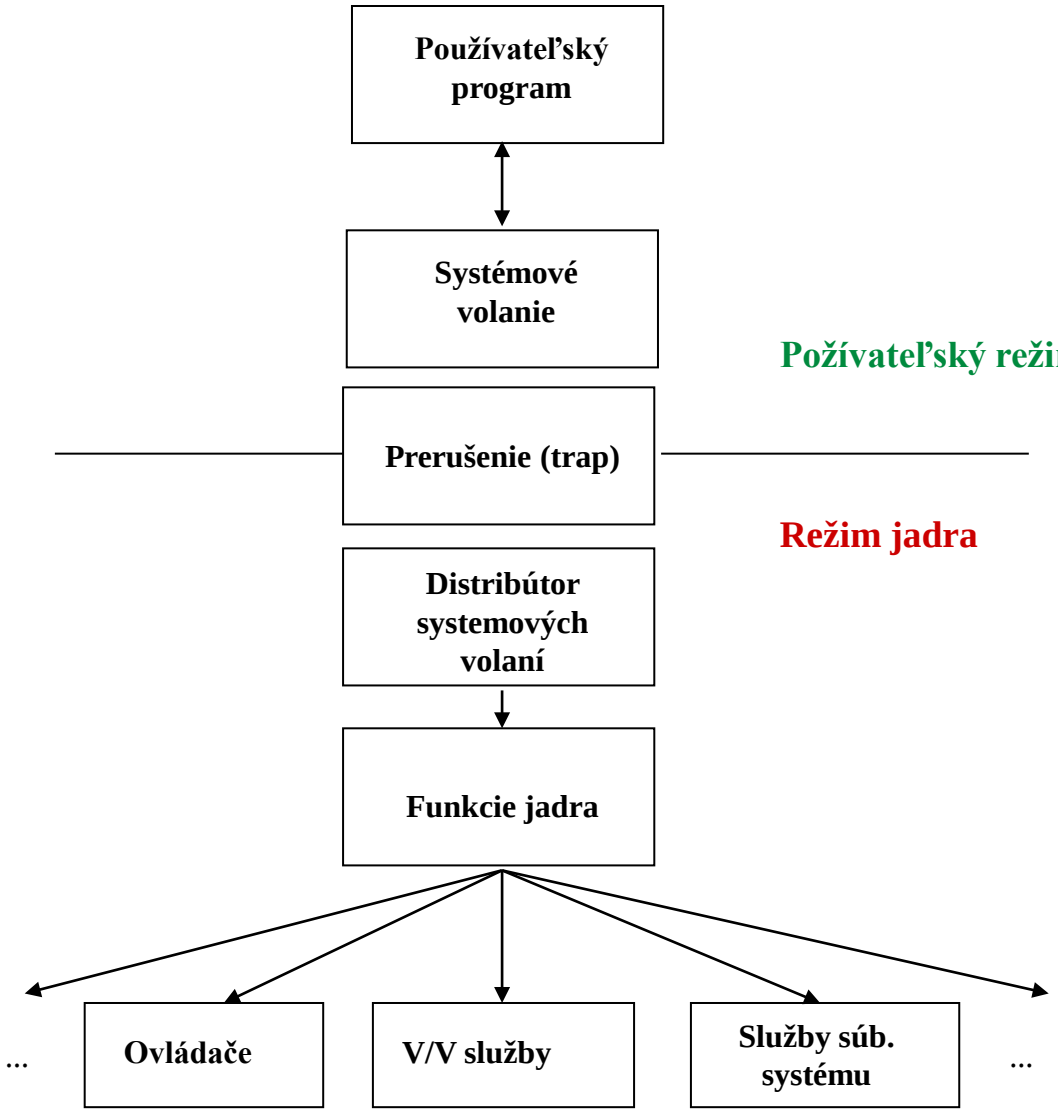


UNIXu


Shared System Tables
OS
OS
OS
OS
system
call
user
process
user
process
user
process
user
process

Vykonanie procesu v Unixe

- *fork* vytvorí nový proces
  - Urobí kópiu pamäte, PCB a pridá ho do frontu pripravených procesov
- Nový proces sa vykonáva
  - na začiatku v režime supervisora
  - *fork* sa potom vracia do užívateľského režimu
  - užívateľský proces pokračuje
  - užívateľský kód vykoná systémové volanie
  - proces sa prepne do režimu supervisora
  - OS obslúži systémové volanie
  - proces sa prepne do užívateľského režimu
  - pokračuje sa vykonávanie kódu procesu

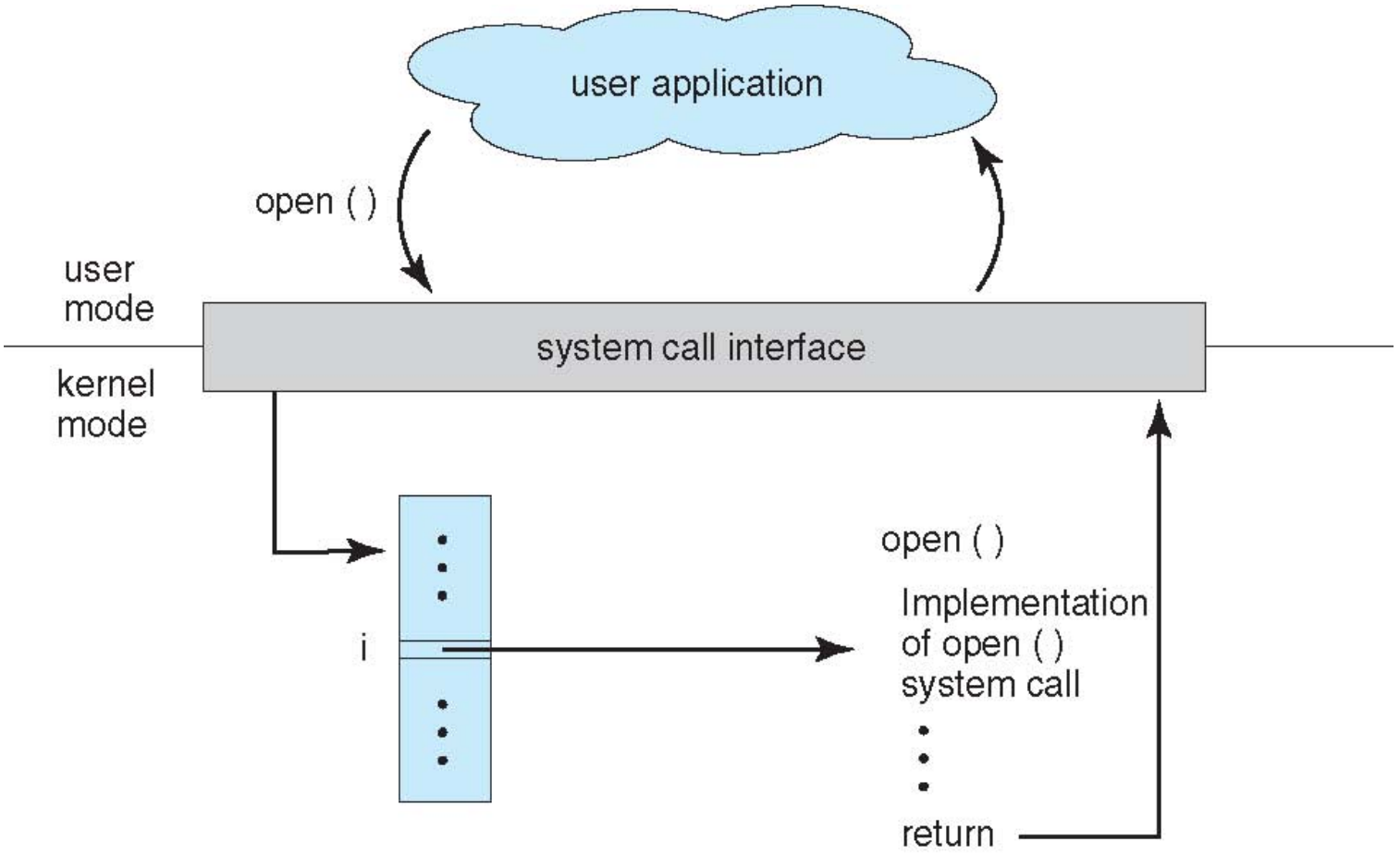


Prepínanie medzi režimom používateľa a režimom jadra

Systémové volania

- „Stub“ funkcie systémových volaní poskytujú **rozhranie** pre programovacie jazyky vyššej úrovne.
  - hlavná činnosť – generovať softvérové prerušenie (trap) a vykonať požadovanú činnosť v režime jadra
  - často sa nazývajú *wrappers*
  - *ich základná činnosť je vo väčšine OS rovnaká:*
    - Nastavenie parametrov,
    - Skok do jadra,
    - Kontrola návratovej hodnoty po návrate z jadra a
      - ak nenastala chyba: okamžitý návrat, ináč
      - ak nastala chyba: nastaví hodnotu chyby do globálnej premennej "errno" a vráti hodnotu -1

Systémové volania pokr.



Príklad systémového volania - x86 Linux read (glibc 2.1.3)


```
read: push %ebx
      mov 0x10(%esp,1),%edx          ; put the 3 parms in registers
      mov 0xc(%esp,1),%ecx
      mov 0x8(%esp,1),%ebx
      mov $0x3,%eax                  ; 3 is the syscall # for read
      int $0x80                       ; trap to kernel
      pop %ebx cmp $0xfffff001,%eax  ; check return value
      jae read_err
read_ret:   ret                       ; return if OK.
read_err:   push %ebx
      call read_next                  ; push PC on stack read_next:
      pop %ebx                        ; pop PC off stack to %ebx
      xor %edx,%edx                   ; clear %edx
      add $0x49a9,%ebx                ; the following is a bunch of
      sub %eax,%edx                  ; ...messy stuff that sets the push %edx ;
                                      ...value fo the errno variable
      call 0x4000dfc0 <__errno_location>
      pop %ecx pop %ebx
      mov %ecx,(%eax)
      or $0xffffffff,%eax             ; set return valu to -1
      jmp read_ret                    ; return
```

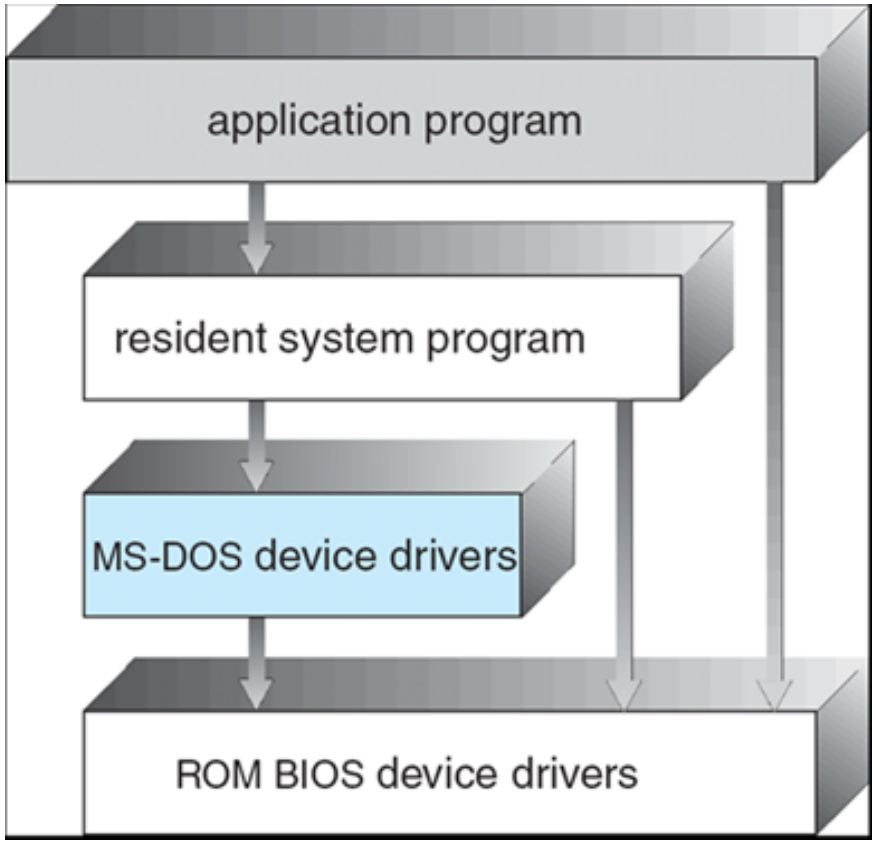
Typická štruktúra návrhu OS

- Neštruktúrovaný prístup
- Štruktúrovaný návrh
  - Jednotlivé moduly
  - Existuje hierarchia vrstiev, kde sú striktne definované volania procedúr, ležiacich v nižších vrstvách

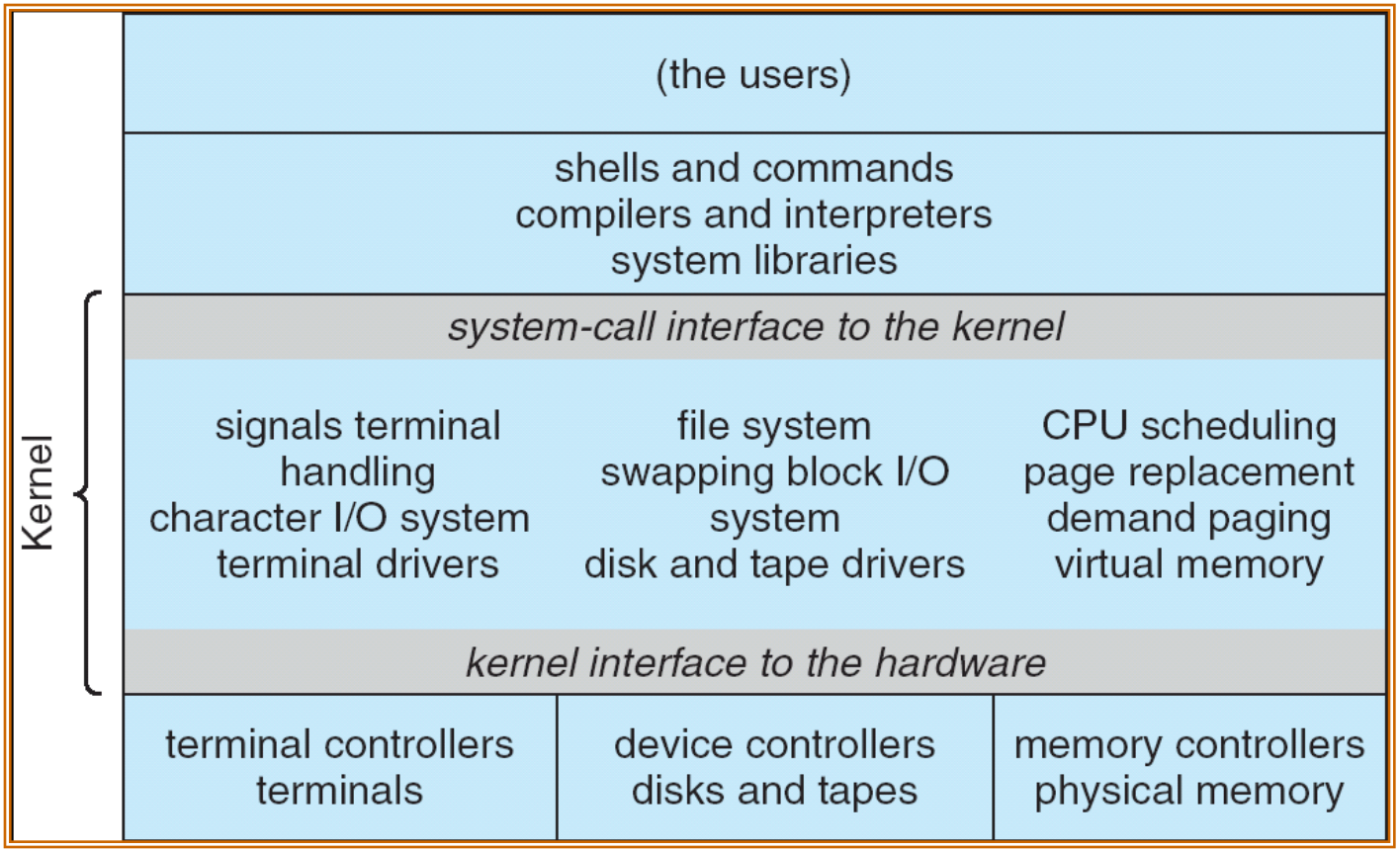
THE OS (Dijkstra)

- Layer 0 is hardware
- Layer 1 is the CPU scheduler
- Layer 2 is the memory manager
- Layer 3 is the operator-console device driver
- Layer 4 is the I/O buffer
  - Uses virtual memory to run, and the operator-console layers to flag errors
- Layer 5 is the user program layer

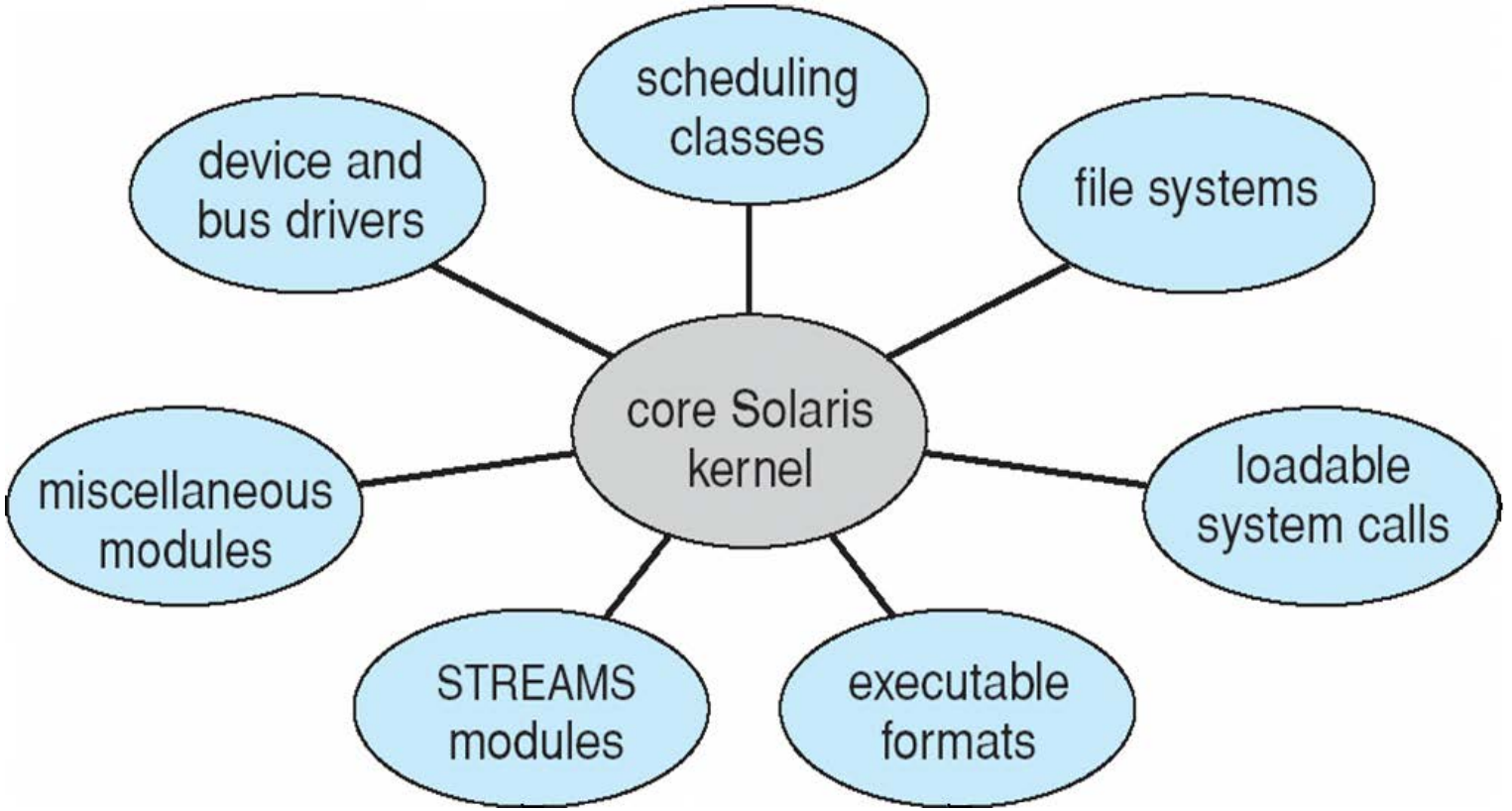
Architektúra OS MS-DOS - jednoduchá



Architektúra OS Unix - vrstvová

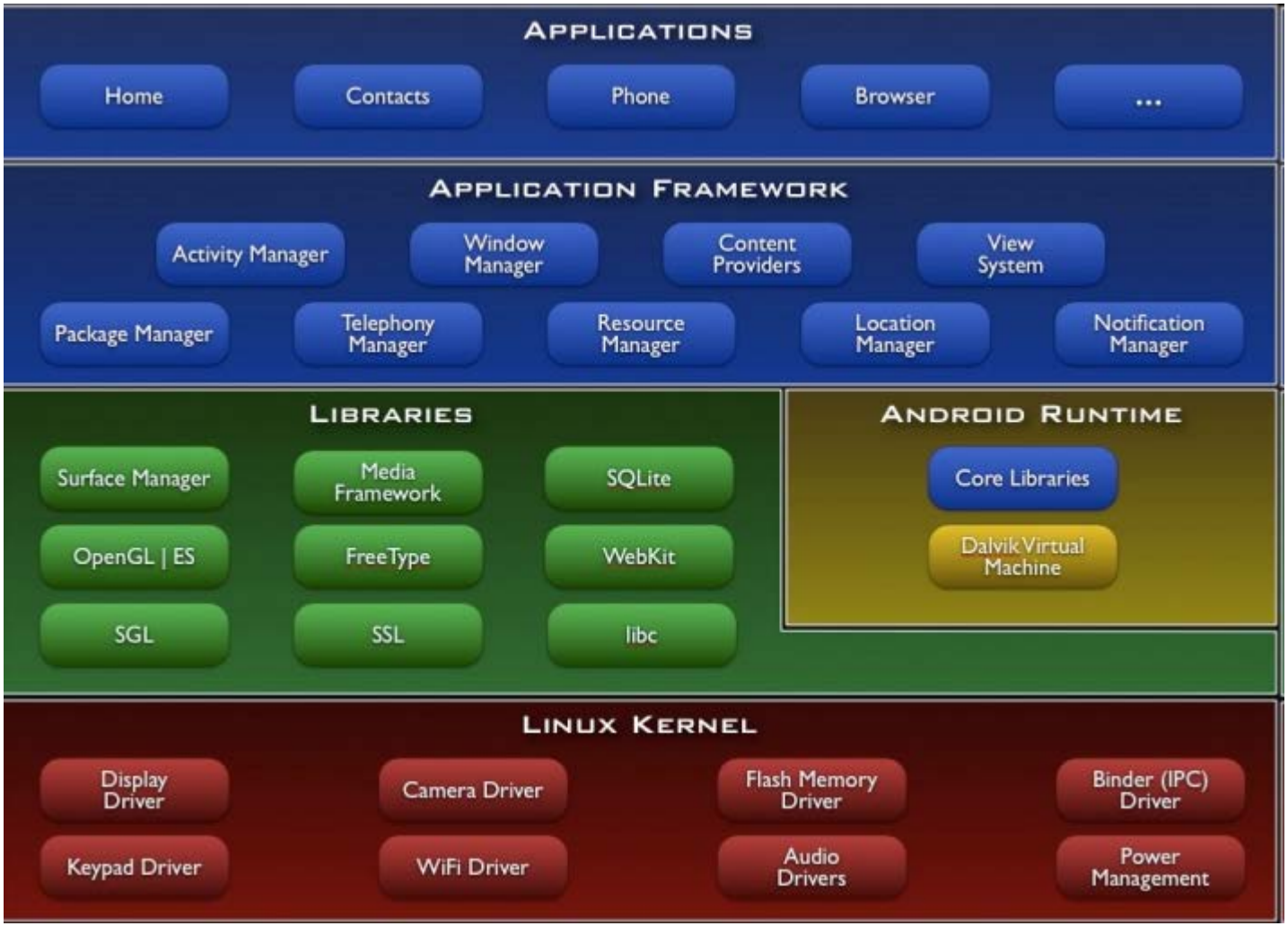


Architektúra OS Solaris - modulárna



Dynamické zavadzanie modulov počas behu systému

Architektúra OS Android – pre mobilné zariadenia



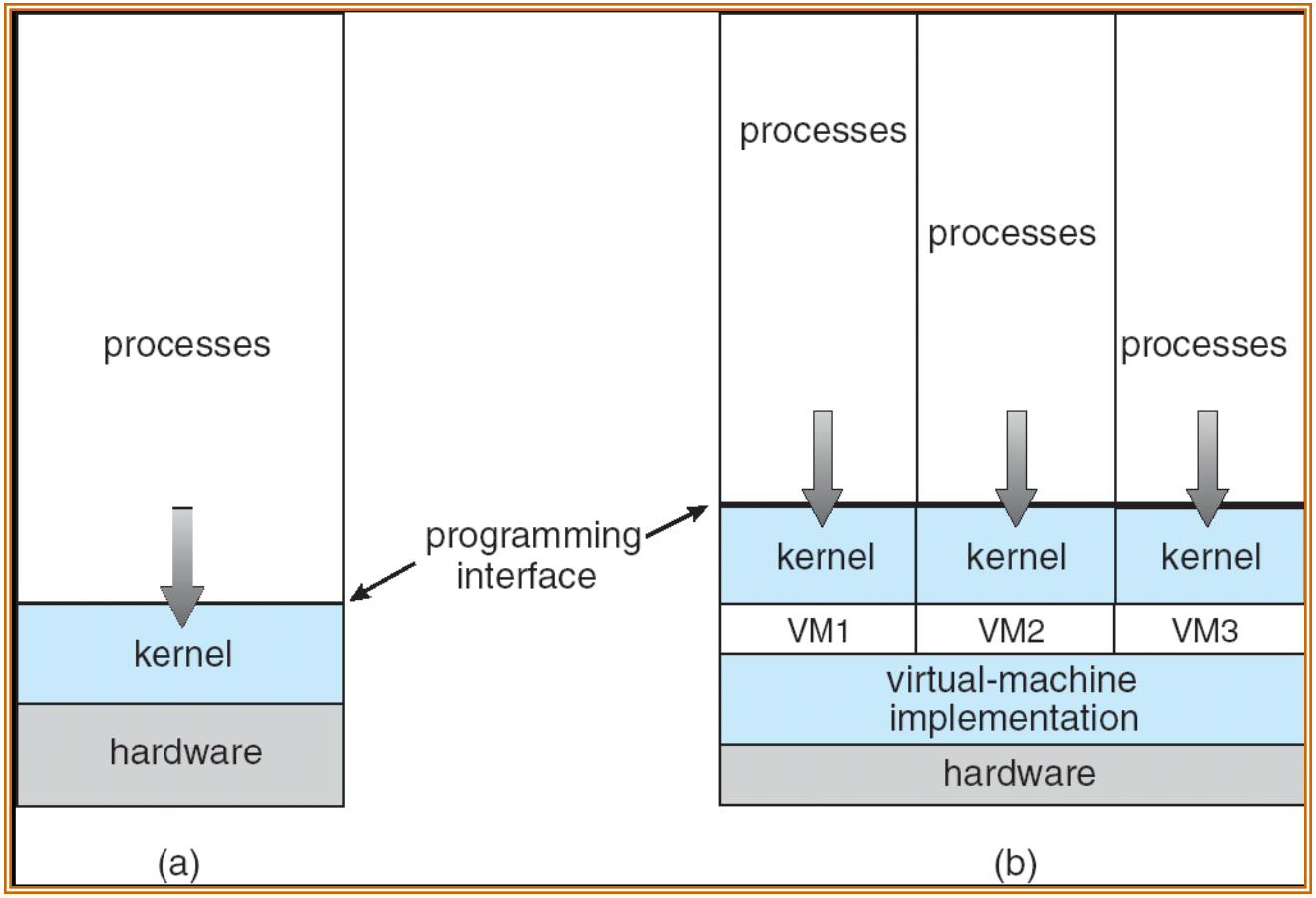
Štruktúrovaný návrh OS

- Všeobecná filozofia, postavená na prístupe, ktorý bol vysvetlený
  - Starostlivy návrh základných vrstiev a ich funkčnosti
    - Hardware je typicky level 0
    - Vrstva *i* má prístup len k vrstve *(i - 1)* alebo nižším
  - OS sa navrhuje systémom top-down
  - Prístup k vrstvám je cez príslušné systémové volania

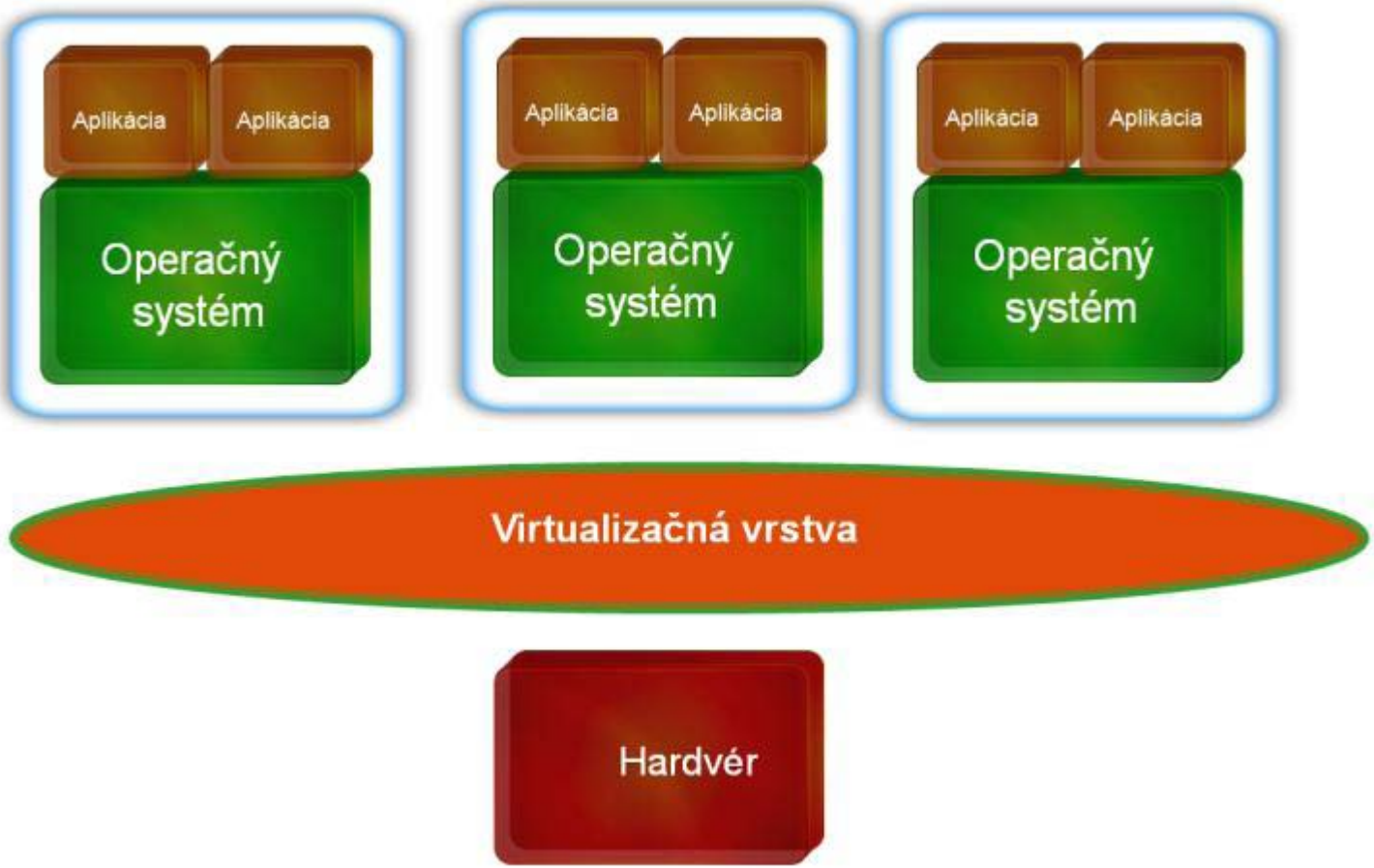
Virtuálny počítač

- OS ponuka abstraktný stroj
- Skrýva špecifika a detaily HW
- Idea virtuálnej mašiny zachádza do extrémov
  - Poskytuje úplnú kópiu (underlying) mašiny užívateľovi
    - Prvý virtuálny počítač - *Virtual Machine* (VM)- bol od IBM
    - Každý užívateľ má kompletný VM

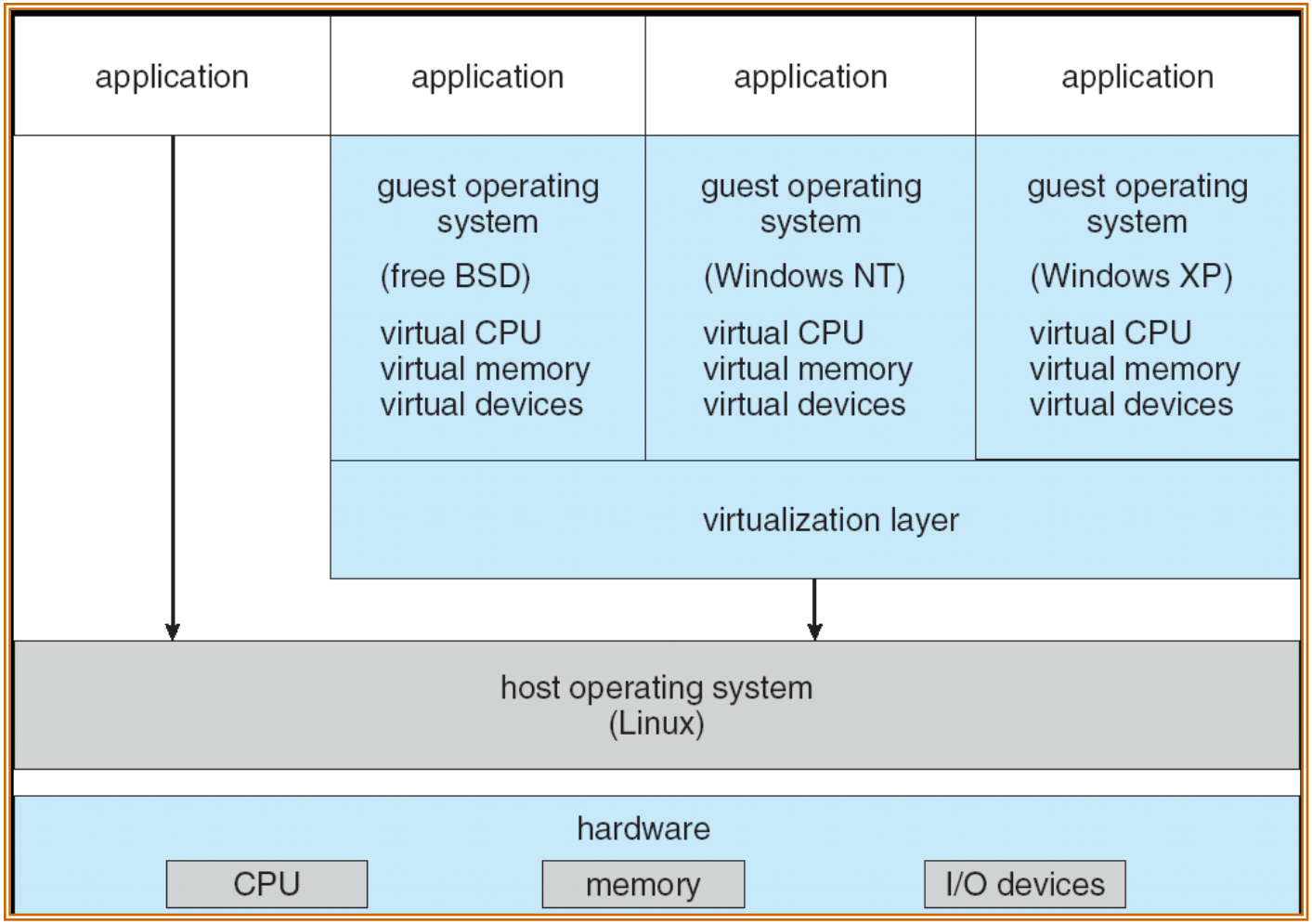
Nevirtuálny (a) a virtuálny stroj (b)



Viac virtuálnych strojov na jednom fyzickom počítači



Príklad - VMware architektúra



Príklad - VirtualBox

Mechanizmy pre vytvorenie tejto ilúzie

- Primárne
  - CPU plánovanie
    - zdieľanie CPU transparetným spôsobom
  - Správa pamäte
    - veľký virtuálny adresný priestor
    - ďalšie vlastnosti ako súborové systémy sú ponúknuté cez pomocné sub-systémy

Detailnejšie...

- User-level kód
  - Vykonáva sa ako je
- Supervisor-level kód
  - Vykonáva sa na úrovni užívateľa
  - Privilegované inštrukcie sú simulované
    - Generuje sa prerušenie do VM emulatora
  - Skryté registre sú simulované
  - I/O inštrukcie sú simulované
    - virtuálne disky

Politika vs. Mechanizmy

- **Politika** popisuje **čo** sa bude robiť
  - Rozhodne či prioritu budú mať programy náročné na I/O alebo náročné na výpočet
  - Politika môže byť vstupným parametrom a môže sa líšiť pre rôzne plánovacie algoritmy
- **Mechanizmy** popisujú ***ako*** sa majú veci robiť
  - Napr. plánovanie času CPU nepopisuje rozhodnutie, na základe ktorého procesy získavajú prioritu
- OS s mikrojadrom poskytujú len mechanizmy
- Pri návrhu OS sa dbá na striktne oddelenie mechanizmov od politiky

Generovanie systému

- Obnovenie OS s novými parametrami stroja
- Parametre stroja sú:
  - veľkosť pamäte, parametre I/O zariadení, mapy adries, ...
- Prístupy
  - prekompilovať
  - mať prekompilované parametrizované knižnice a iba zlinkovať
  - nastavenie parametrov z tabuľky počas nábehu