Základy operačních systémů

I.

KIV/ZOS 2012

Kontaktní informace

- Ing. Ladislav Pešička
- UL401
- *pesicka@kiv.zcu.cz*
  - *Předmět zprávy začít: ZOS*
- Úřední hodiny
  - Út 09:30 až 10:30
  - Čt 14:30 až 15:30
- Veškeré informace v coursewaru

Požadavky na zápočet

- 2 zápočtové testy
  - pass / failed
  - 1 náhradní termín
- Semestrální práce
  - Info na cvičeních
  - Program + dokumentace

1. zápočtový test

- časově koncem října
- v době cvičení po ½
- napsat složitější script v /bin/bash
- a teoretická otázka
- Např:

```
skript -p1 s1.txt
skript -p2
skript -p3 > vys.txt
```

2. zápočtový test

- teoretický
- časově začátek prosince
- otázky z přednášek
- řešení příkladů podobných těm na cvičení

Zkouška

- Písemný test
  - Test na 60 min. bez pomůcek
  - zvolit správnou odpověď, odpovědět na otázku, doplnit diagram atd..
  - Ústní konzultace v případě potřeby

ZOS

- Obecné principy
  - Není zaměřen na 1 systém, vychází z Unixu
  - Není hodnocením, který systém je lepší
- KIV/OS, KIV/PPR
  - Pokračováním, Unix / Linux, paralelizace
- Praxe
  - Základy práce s Linuxem
  - Práce se sdílenými zdroji, ošetření kritické sekce

ZOS přednášky

- Úvod. Historie OS, rozdělení OS, architektura a komponenty OS
- **Proces**. Implementace procesu. Konstrukce pro vytváření procesů. Vlákna.
- Problém **kritické sekce**. Spin-lock. Semafory. Kooperace procesů.
- Implementace **semaforů**. Monitory a jejich implementace.
- Zprávy, RPC. Klasické problémy meziprocesové komunikace: Problém večeřících filosofů.
- **Plánování procesů**
- Plánování procesů v interaktivních systémech. Uvíznutí (deadlock).
- Uvíznutí: detekce a zotavení, dynamické zabránění, prevence. Správa hlavní paměti, základní mechanismy.
- Správa hlavní paměti -- **virtuální paměť**.
- **Vstupy a výstupy**.
- **Systémy souborů**.
- **Bezpečnost** v OS.
- Případová studie: UNIX (Linux). + další témata: **virtualizace**,..

ZOS cvičení

- Základy Linuxu
  - distribuce, jádro, struktura
  - uživatelské ovládání
  - příkazy, spojování příkazů
  - příkazové skripty
- Paralelní procesy, souběhy a ošetření
  - ošetření kritické sekce, uvíznutí, ...
  - reálná implementace Java, C, ..

- Kde všude můžete nalézt OS?

- Ukázky zařízení

(s využítím materiálu
Introduction to embedded systems)

Sonicare Elite toothbrush.

Microprocessor: 8-bit

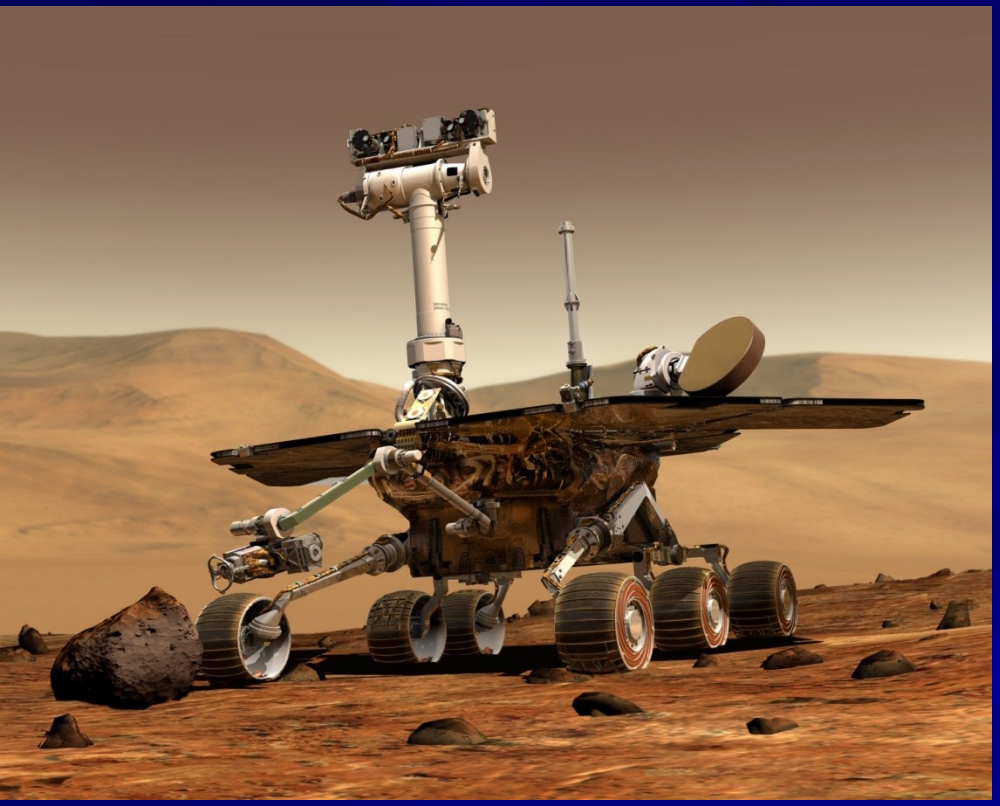
Has a programmable
speed control,
timer,
charge gauge

OS? NE

Product: Radiant Systems Point-of-Sale (POS) Terminal

Microprocessor:
Intel X86 Celeron

OS: Windows XP Embedded



Photograph courtesy of NASA/JPL CALTECH

NASA's Twin Mars Rovers.

Microprocessor:
Radiation Hardened
20Mhz PowerPC

Commercial Real-time OS

Software and OS was developed during multi-year flight to Mars
and downloaded using a radio link

Kuka robot arms welding a Mercedes

Kuka Industrial Robot Arm

Microprocessor: X86

OS: Windows CE OS & Others

LG VoIP Phone

Microprocessor: Freescale i.MX21 ARM

OS: Windows CE

OS – příklady použití

- Servery, pracovní stanice, notebooky
  - MS Windows, GNU/Linux, Solaris
- Mobilní zařízení, tablety
  - Windows CE, Symbian, Linux (Android aj.)
- Routery, AP, SOHO síťová zařízení
  - Cisco IOS, Linux, VxWorks
- Embedded zařízení
  - Bankomaty, stravovací systémy, lékařské přístroje
  - Windows CE, Windows XP embedded, Linux

Co všechno tvoří OS?

- Není všeobecná definice

- Vše co dodavatel poskytuje jako OS ?
  - Windows – kalkulačka, hra miny, malování, …
- Program, běžící po celou dobu běhu systému ?
  - Ale Linux, moduly, zavádění na žádost v případě potřeby

- SLOC (Source lines of code)
  - Windows XP: 40 milionů řádků
  - Linux kernel 2.6.19 11 mil. ř.
  - Distribuce Debian 4.0 283 mil. ř.

Operační Systém - definice

OS je softwarová vrstva (základní programové vybavení), jejíž úlohou je spravovat hardware a poskytovat k němu programům jednotné rozhraní

- OS zprostředkovává aplikacím přístup k hardwaru
- OS koordinuje a poskytuje služby aplikacím
  - Analogie – dopravní systém, vláda, ..
- OS je program, který slouží jako prostředník mezi aplikacemi a hardwarem počítače.

Privilegovaný a uživatelský režim

- Jádro OS běží v tzv. privilegovaném režimu
  - Všechny instrukce povoleny
  - *Privileg. režim není v MS DOS, různé embedded systémy*
  - *Někdy část OS v uživatelském režimu*
  - *Interpretované systémy (JVM)*

- Aplikace – v uživatelském režimu
  - Některé instrukce zakázány
    např. přímý přístup k disku, jeho zformátování zákeřnou aplikací
  - Aplikace musí požádat OS o přístup k souboru, ten rozhodne

- OS může zasahovat do běhu aplikací
- Aplikace může požádat OS o službu

OS

- Dva základní pohledy na OS
- **Rozšířený stroj** (shora dolů)
- **Správce zdrojů** (zdola nahoru)

OS jako rozšířený stroj

- Holý počítač
  - Primitivní a obtížně programovatelný (I/O)
  - Např. disky …
    - Práce s hlavičkou disku
    - Alokace dealokace bloků dat
    - Víc programů chce sdílet stejné médium
- Jako programátor chceme
  - Jednoduchý pohled – pojmenované soubory
  - OS skrývá před aplikacemi podrobnosti o HW (přerušení, správu paměti..)

OS jako rozšířený stroj

- Strojové instrukce (holý stroj)
- Vysokoúrovňové služby (rozšířené instrukce)
  - Systémová volání
- Z pohledu programátora
  - Pojmenované soubory
  - Neomezená paměť
  - Transparentní I/O operace
- ZOS zkoumá, jako jsou služby v OS implementovány

OS jako správce zdrojů

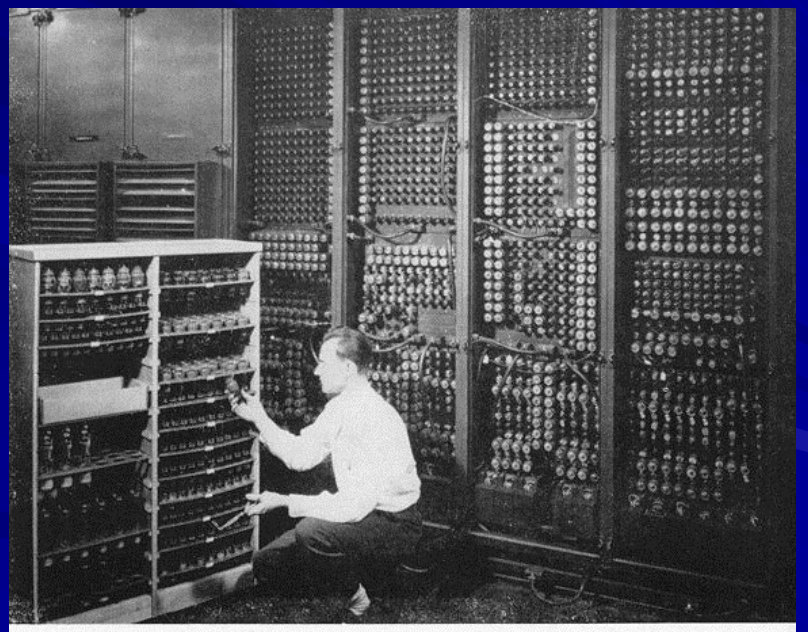
- OS jako poskytovatel / správce zdrojů (resource manager)

- Různé zdroje (čas CPU, paměť, I/O zařízení)
- OS – správná a řízená alokace zdrojů programům, které je požadují (přístupová práva)
- Konfliktní požadavky na zdroje
  - V jakém pořadí vyřízeny
  - Efektivnost, spravedlivost

Historický vývoj

- Vývoj hw -> vývoj OS

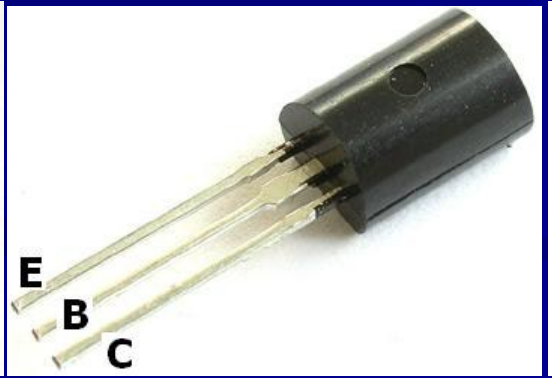
- 1. počítač – ENIAC, 15.2.1946
  - Tělocvična
  - 18 000 elektronek
  - Regály, chlazení
    - letecké motory
  - 5000 operací/s

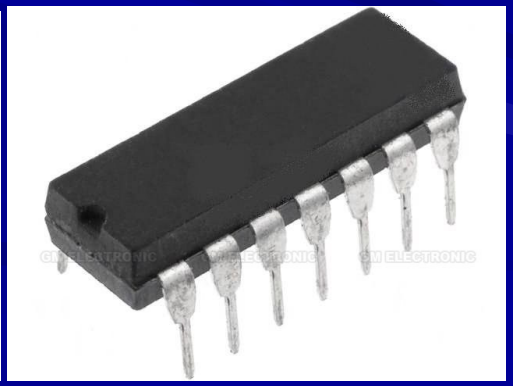


Replacing a bad tube meant checking among ENIAC's 19,000 possibilities.

Generace počítačů

1. Elektronky
2. Tranzistory
3. Integrované obvody
4. LSI, VLSI (mikroprocesory,..)





1.Generace (1945-55)

- Elektronky, propojovací desky
- Programování
  - V absolutním jazyce
  - Propojování zdířek na desce
  - Později děrné štítky, assemblery, knihovny, FORTRAN
  - Numerické kalkulace
- Způsob práce
  - **Stejní lidé** – stroj navrhli, postavili, programovali !
  - Zatrhnout blok času na rozvrhu, doufat, že to vyjde
- OS ještě neexistují

2. Generace (1955-65)

- Tranzistory, dávkové OS
- Vyšší spolehlivost; klimatizované sály
- Oddělení návrhářů, výroby, operátorů, programátorů, údržby
- Mil $ - velké firmy, vlády, univerzity
- Způsob práce
  - Vyděrovat štítky s programem
  - Krabici dát operátorovi
  - Výsledek vytisknut na tiskárně
- Optimalizace
  - Na levném stroji štítky přenést na magnetickou pásku

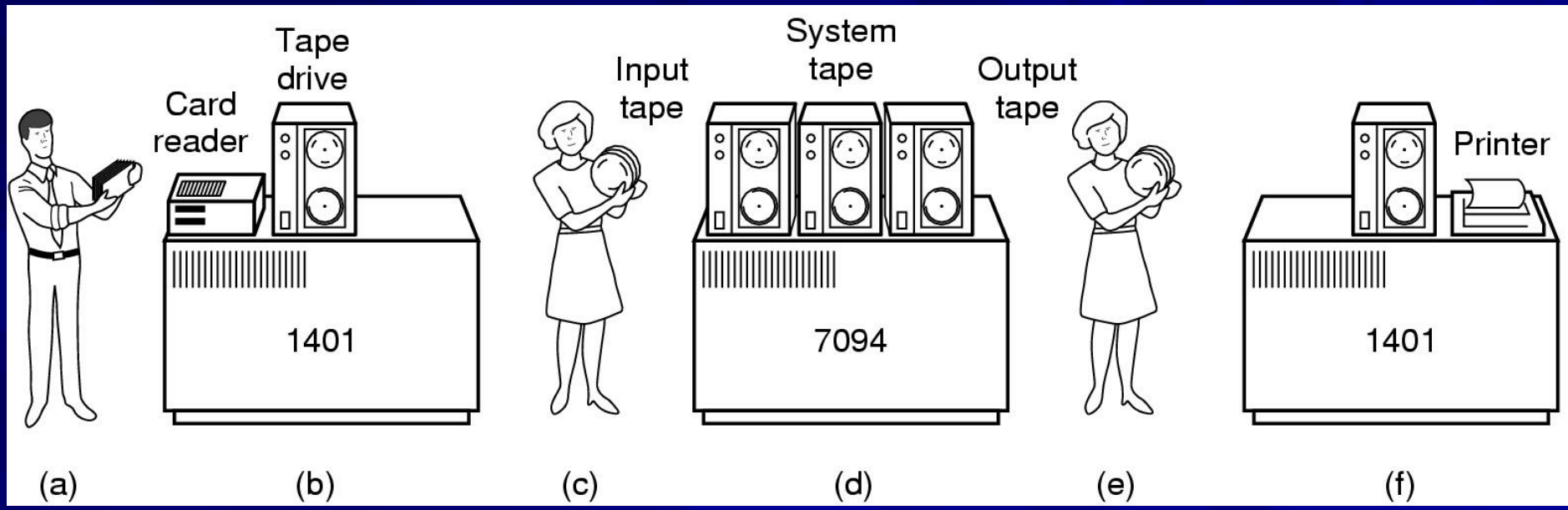
2. generace

- Sekvenční vykonávání dávek
- Ochrana systému – kdokoliv dokázal shodit
- OS IBSYS = IBM SYSTÉM FOR 7094

- Pokud úloha prováděla I/O, CPU čekal..
  - Čas CPU je drahý

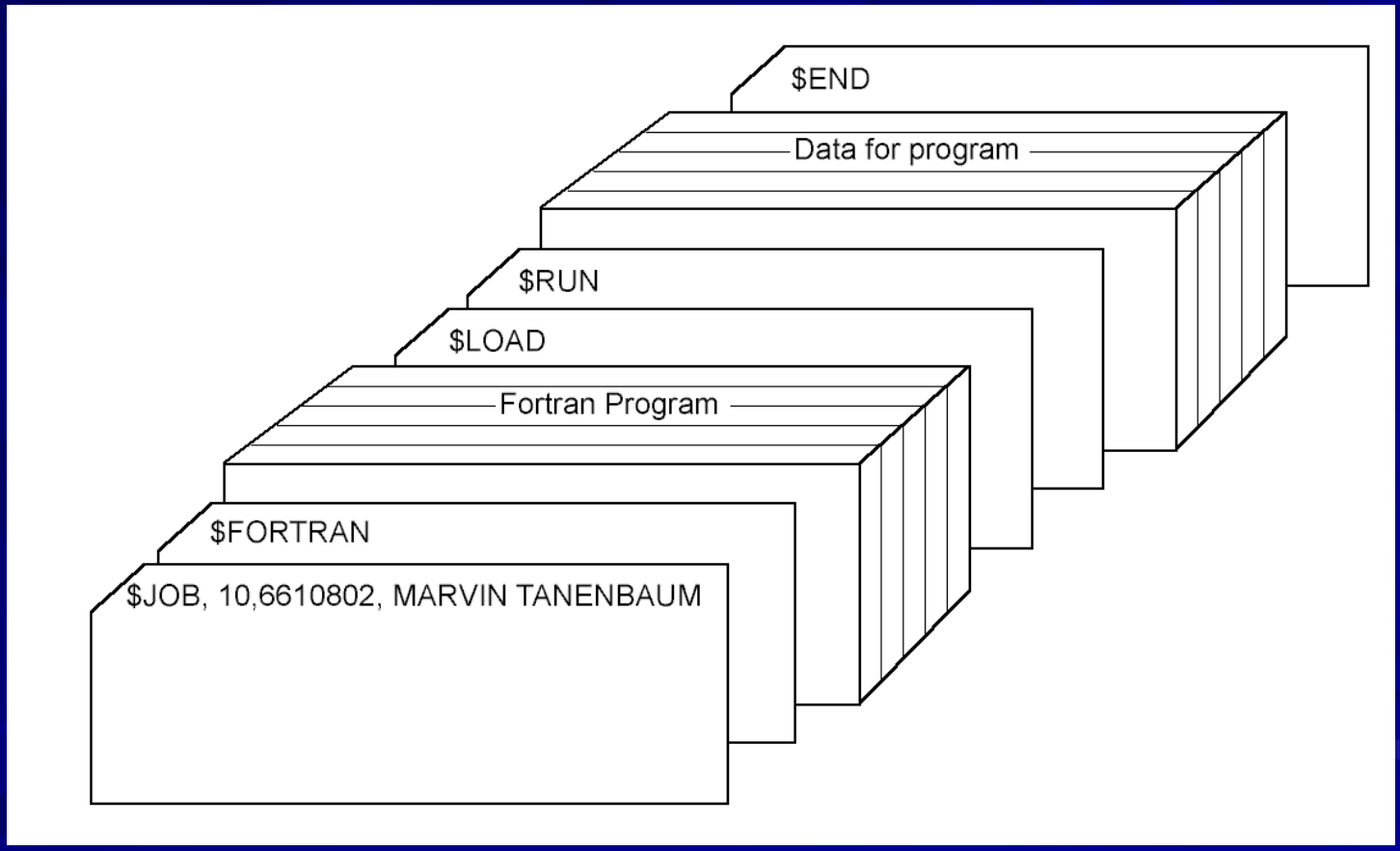
- Viz slidy Tanenbaum

Dávkové systémy



– vezmi štítky k 1401
– štítky se zkopírují na pásek (tape)
– pásek na 7094, provádí výpočet
– output tape na 1401 vytiskne výstup

History of Operating Systems



- Struktura typické dávky – 2nd generation

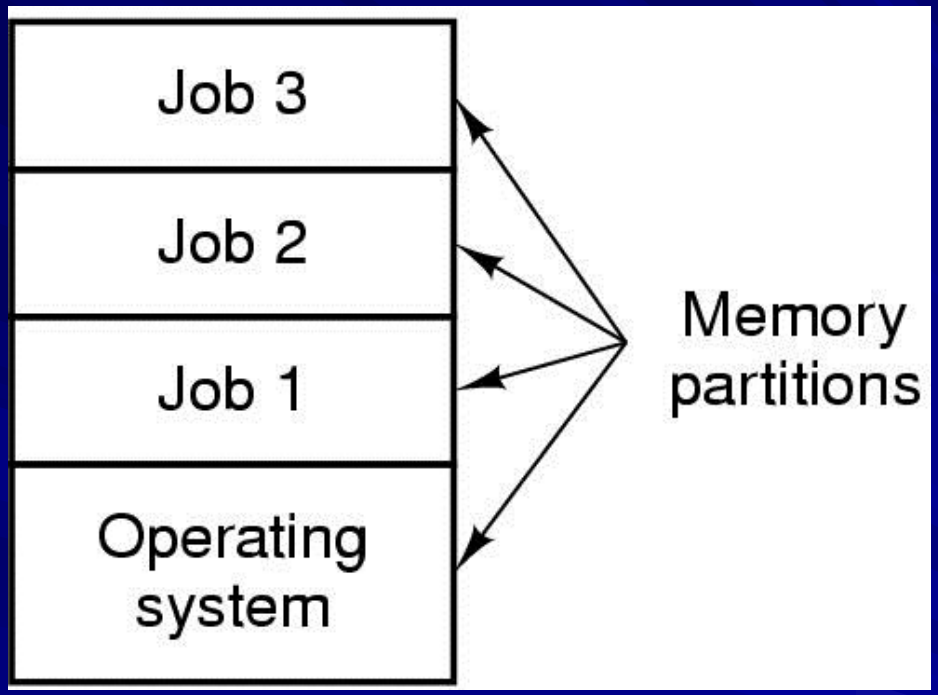
3. Generace (1965-80)

- Integrované obvody, multiprogramování
- Do té doby 2 řady počítačů
  - Vědecké výpočty
  - Komerční stroje – banky, pojišťovny
- IBM 360 – sjednocení
  - Malé i velké stroje
  - Komplexnost – spousta chyb

3. generace

- Multiprogramování
  - Doba čekání na I/O neefektivní (věda OK, banky 80-90% čekání)
  - Více úloh v paměti
    - Napřed konstantní počet
    - HW pro ochranu paměti
- Každá úloha ve vlastní oblasti paměti; zatímco jedna I/O, druhá počítá …

History of Operating Systems



- Multiprogramming system
  - three jobs in memory – 3rd generation

3. generace

- Spooling
  - Na vstupu – ze štítků na disk, úloha se zavede z disku
  - Na výstupu – výsledky na disk před výtiskem na tiskárně
- Stále dávkové systémy
  - Dodání úlohy, výsledek – několik hodin

3. generace

- Systémy se sdílením času
  (time shared system)
  - Varianta multiprogramování
  - CPU střídavě vykonává úlohy
  - Každý uživatel má on-line terminál

- CTSS (MIT 1962) *Compatible Time Sharing Sys.*
- MULTICS

Minipočítače

- DEC PDP (1961)
  - Cca 3.5 mil Kč , „jako housky“
  - Až PDP11 – nekompatibilní navzájem

- Výzkumník Bell Labs pracující na MULTICSu Ken Thompson – našel nepoužívanou PDP-7, napsal omezenou jednouživat. verzi MULTICSu vznik UNIXu a jazyka C (1969)

4. Generace (1980)

- Mikroprocesory, PC
- GUI x CLI
- Síťové a distribuované systémy
- MS DOS, Unix, Windows NT
- UNIX – dominantní na nonIntel;
- Linux, BSD – rozšíření i na PC
  - Výzkum Xerox PARC – vznik GUI
  - Apple Macintosh

- film „Piráti ze Silicon Valley“

Dělení OS

- Dle úrovně sdílení CPU

- Jednoprocesový
  - MS DOS, v daném čase v paměti aktivní 1 program
- Multiprocesový
  - Efektivnost využití zdrojů
  - Práce více uživatelů

Dělení OS

- Dle typu interakce

- Dávkový systém
  - Sekvenční dávky, není interakce
  - i dnes má smysl, viz. meta.cesnet.cz
- Interaktivní
  - Interakce uživatel – úloha
  - Víceprocesové – interakce max. do několika sekund (Win, Linux, ..)

OS reálného času (!)

- Výsledek má smysl, pouze pokud je získán v nějakém čase
- Přísné požadavky aplikací na čas odpovědi
  - Řídící počítače, multimedia
- Časově ohraničené požadavky na odpověď
  - Řízení válcovny plechu, výtahu mrakodrapu ☺
- Nejlepší snaha systému
  - Multimedia, virtuální realita
- Př: RTLinux, RTX Windows, VxWorks

Hard realtime OS

- Zaručena odezva v **ohraničeném** čase
- Všechna zpoždění a režie systému ohraničeny

Omezení na OS:

- Často není systém souborů,
- Není virtuální paměť
- Nelze zároveň sdílení času

- Řízení výroby, robotika, telekomunikace

Soft realtime OS

- Priorita RT úloh před ostatními
- Nezaručuje odezvu v daném čase
- Lze v systémech sdílení času
- RT Linux
- Multimédia, virtuální realita

Další dělení OS

- Dle velikosti HW
  - Superpočítač, telefon, čipová karta
- Míra distribuovanosti
  - Klasické - centralizované 1 a více CPU
  - Paralelní
  - Síťové
  - Distribuované
    - virtuální uniprocesor
    - Uživatel neví kde běží programy, kde jsou soubory

Další dělení OS

- Podle počtu uživatelů
  - Jedno a víceuživatelské
- Podle funkcí
  - Univerzální
  - Specializované (např. Cisco IOS)

Základní funkce operačního systému (!)

- správa procesů
- správa paměti
- správa souborů
- správa zařízení - I/O subsystém
- síťování (networking)
- ochrana a bezpečnost
- uživatelské rozhraní

Správa procesů

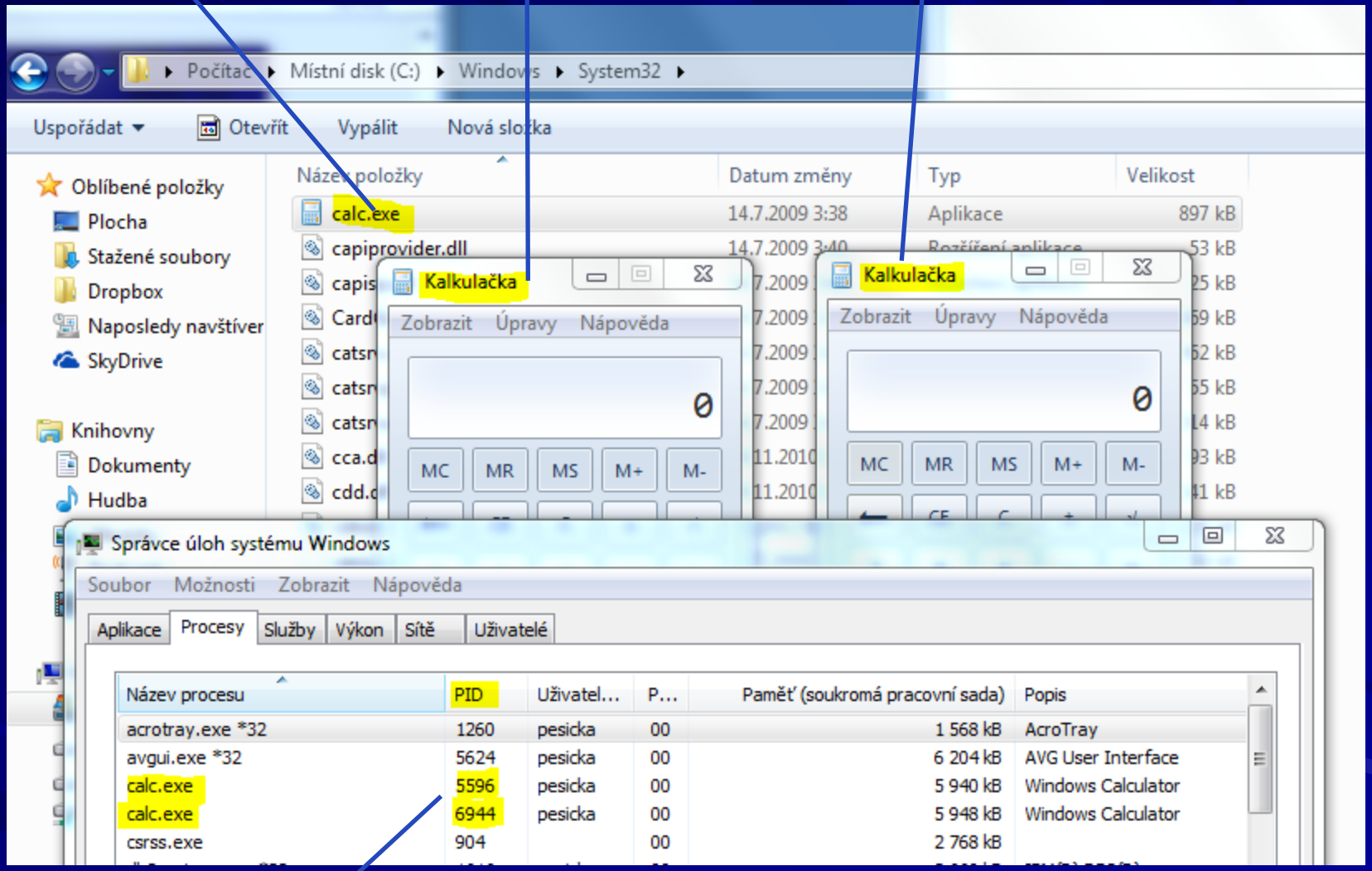
- Program
  - Spustitelný kód, v binární podobě
  - Nejčastěji uložený na disku
  - Např. C:\windows\system32\calc.exe

- proces – instance běžícího programu
  - Přidělen čas CPU
  - Potřebuje paměťový prostor
  - vstupy a výstupy
  - *Dle jednoho programu můžeme spustit více procesů*

PROGRAM

PROCES

PROCES



PID (ID procesu) – základní identifikátor procesu !!

Správa paměti

- správa hlavní paměti
  - Přidělování paměti procesům
    - alokace / dealokace paměti dle potřeby
    - Virtuální adresování (stránkování, segmentace)
  - Správa informace o volné a obsazené paměti
    - Která část paměti je volná, která obsazená a kým
  - Odebírání paměti skončenému programu
  - Ochrana paměti
    - Přístup pouze pro oprávněné procesy

Soubory

- soubory
  - vytváření a rušení souborů
  - vytváření a rušení adresářů
  - primitiva pro manipulaci
    - se soubory
    - s adresáři
  - správa volného prostoru vnější paměti
  - mapování souborů na vnější paměť
  - rozvrhování diskových operací

I/O subsystém

- I/O subsystém
  - správa paměti pro buffering, caching, spooling
  - společné rozhraní ovladačů zařízení
  - ovladače pro specifická zařízení

ovladače – kámen úrazu každého OS

Ochrana a bezpečnost

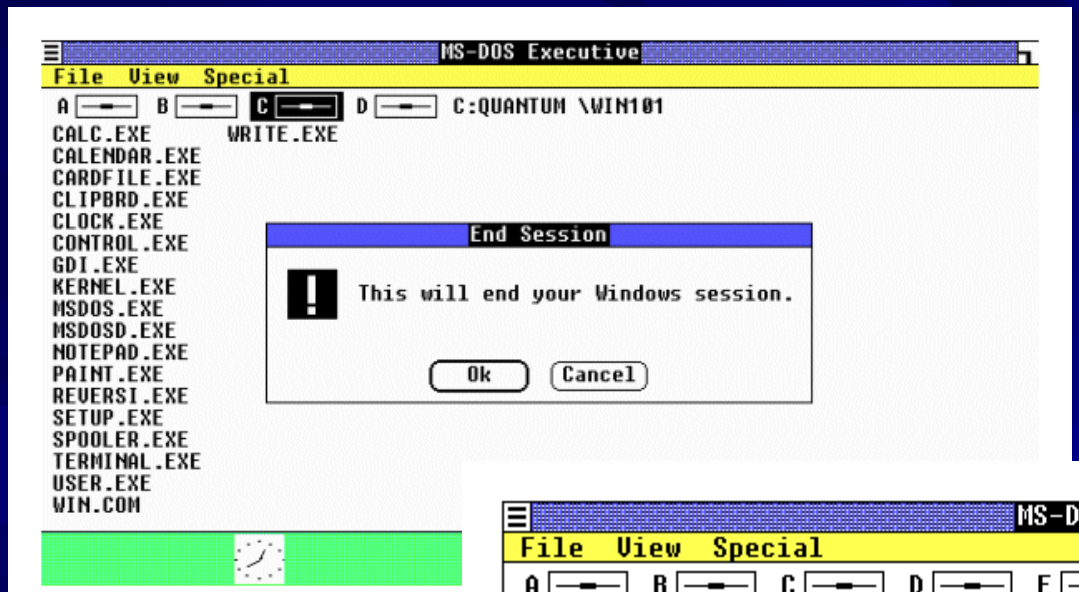
- ochrana a bezpečnost
  - ke zdrojům smí přistupovat pouze autorizované procesy
  - specifikace přístupu
  - mechanismus ochrany (souborů, paměti)

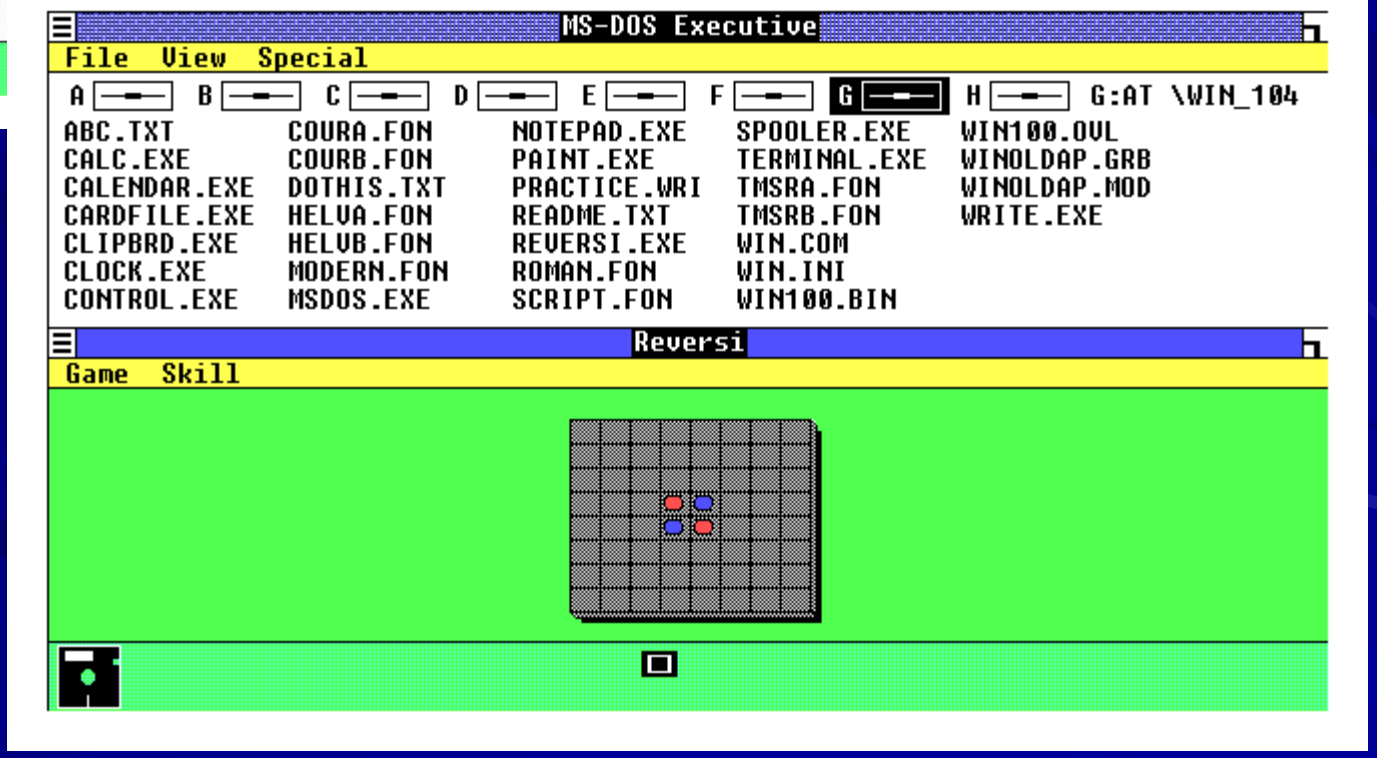
ACL, capabilities, ...

Uživatelské rozhraní

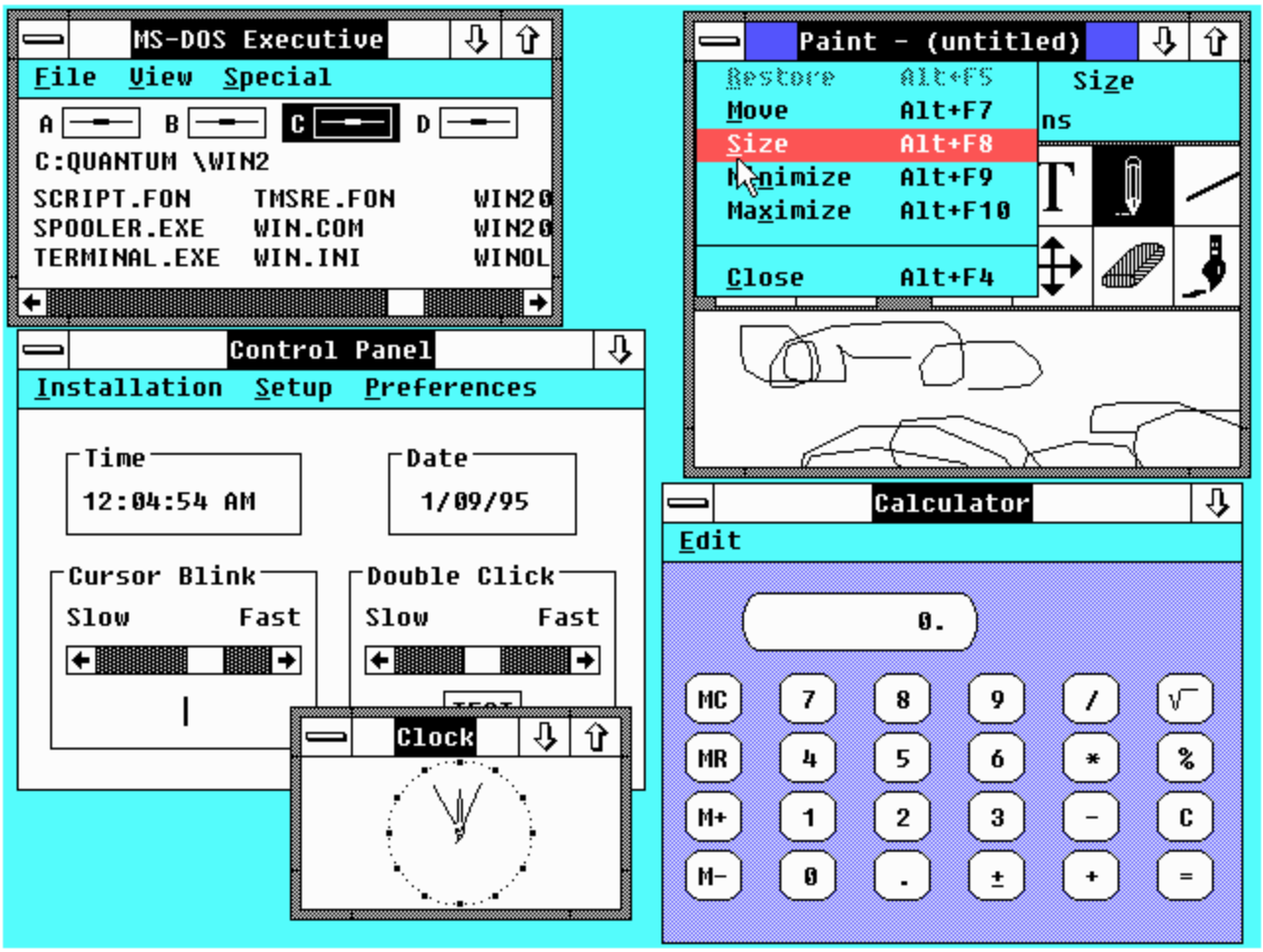
- uživatelské rozhraní

- CLI (command line interface)
- GUI (graphical user interface)
  - *ukázky z www.zive.cz*

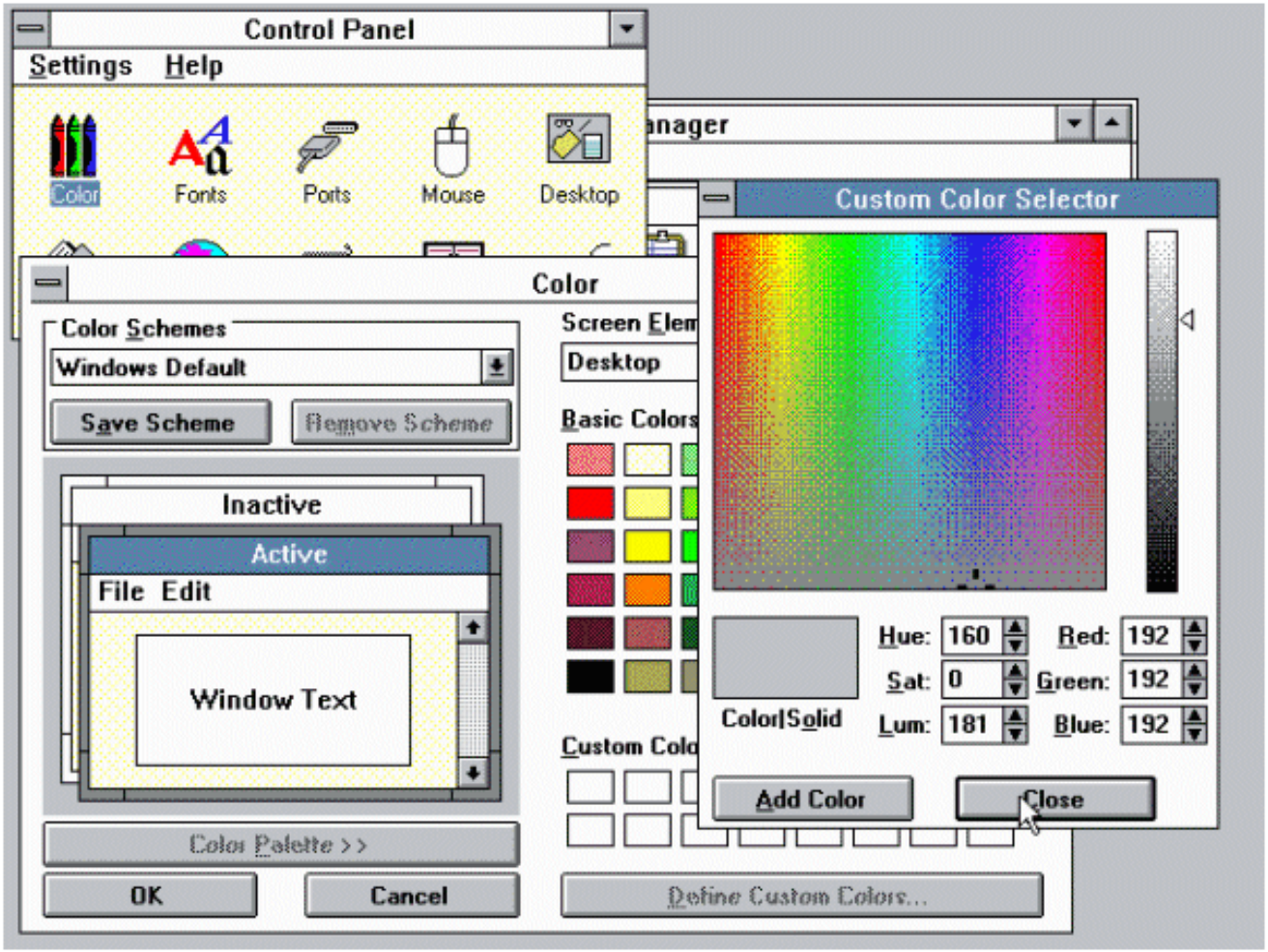

MS-DOS Executive
File View Special
A B C D C:QUANTUM \WIN101
CALC.EXE
CALENDAR.EXE
CARDFILE.EXE
CLIPBRD.EXE
CLOCK.EXE
CONTROL.EXE
GDI.EXE
KERNEL.EXE
MSDOS.EXE
MSDOSD.EXE
NOTEPAD.EXE
PAINT.EXE
REVERSI.EXE
SETUP.EXE
SPOOLER.EXE
TERMINAL.EXE
USER.EXE
WIN.COM
WRITE.EXE
End Session
This will end your Windows session.
Ok
Cancel


MS-DOS Executive
File View Special
A B C D E F G H G:AT \WIN_104
ABC.TXT
CALC.EXE
CALENDAR.EXE
CARDFILE.EXE
CLIPBRD.EXE
CLOCK.EXE
CONTROL.EXE
COURA.FON
COURB.FON
DOTHIS.TXT
HELVA.FON
HELVB.FON
MODERN.FON
MSDOS.EXE
NOTEPAD.EXE
PAINT.EXE
PRACTICE.WRI
README.TXT
REVERSI.EXE
ROMAN.FON
SCRIPT.FON
SPOOLER.EXE
TERMINAL.EXE
TMSRA.FON
TMSRB.FON
WIN.COM
WIN.INI
WIN100.BIN
WIN100.OVL
WINOLDAP.GRB
WINOLDAP.MOD
WRITE.EXE
Reversi
Game Skill

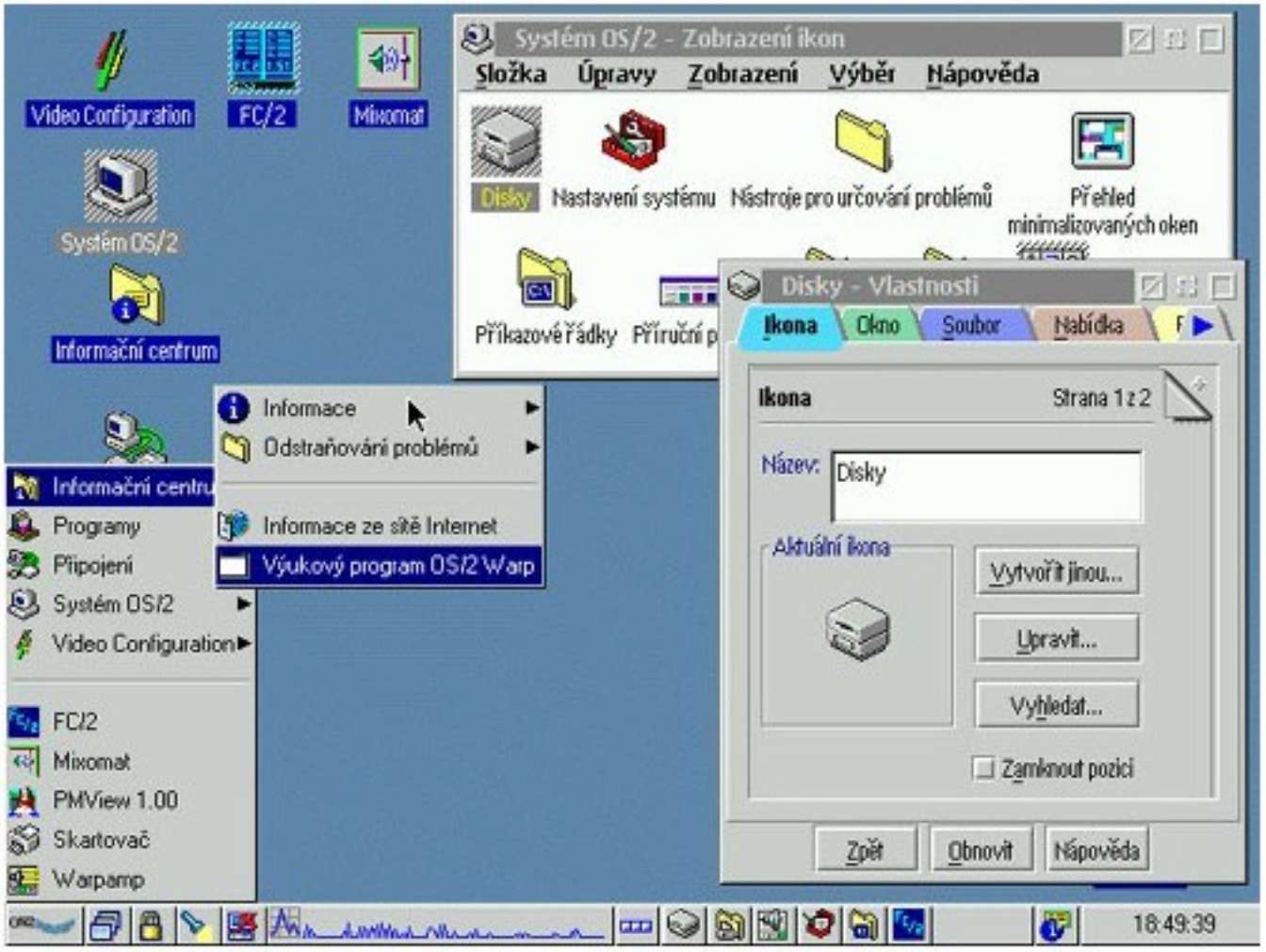
Win 2.0


MS-DOS Executive
File View Special
A B C D
C:QUANTUM \WIN2
SCRIPT.FON TMSRE.FON WIN20
SPOOLER.EXE WIN.COM WIN20
TERMINAL.EXE WIN.INI WINOL
Paint - (untitled)
Restore Alt+F5
Move Alt+F7
Size Alt+F8
Minimize Alt+F9
Maximize Alt+F10
Close Alt+F4
Size
ns
Control Panel
Installation Setup Preferences
Time
12:04:54 AM
Date
1/09/95
Cursor Blink
Slow Fast
Double Click
Slow Fast
Clock
Calculator
Edit
0.
MC 7 8 9 / √
MR 4 5 6 * %
M+ 1 2 3 - C
M- 0 . ± + =

Win 3.1


Control Panel
Settings Help
Color
Fonts
Ports
Mouse
Desktop
Color
Color Schemes
Windows Default
Save Scheme
Remove Scheme
Inactive
Active
File Edit
Window Text
Color Palette >>
OK
Cancel
Basic Colors
Custom Colo
Define Custom Colors...
Custom Color Selector
Hue: 160
Sat: 0
Lum: 181
Red: 192
Green: 192
Blue: 192
Color|Solid
Add Color
Close

OS/2 Warp4


Video Configuration
FC/2
Mixomat
Systém OS/2
Informační centrum
Systém OS/2 - Zobrazení ikon
Složka
Úpravy
Zobrazení
Výběr
Nápověda
Disky
Nastavení systému
Nástroje pro určování problémů
Přehled minimalizovaných oken
Příkazové řádky
Disky - Vlastnosti
Ikona
Okno
Soubor
Nabídka
Ikona
Strana 1 z 2
Název:
Disky
Aktuální ikona
Vytvořit jinou...
Upravit...
Vyhledat...
Zamknout pozici
Zpět
Obnovit
Nápověda
Informace
Odstraňování problémů
Informace ze sítě Internet
Výukový program OS/2 Warp
Informační centru
Programy
Připojení
Systém OS/2
Video Configuration
FC/2
Mixomat
PMView 1.00
Skartovač
Warpamp
18:49:39

2 základní režimy OS

- **Uživatelský režim**
  - V tomto režimu běží aplikace (word, kalkulačka,..)
  - Nemůžou vykonávat všechny instrukce, např. přímý přístup k zařízení (tj. zapiš datový blok *x* na disk *y*)
    - Proč? Jinak by škodlivá aplikace mohla např. smazat disk
    - Jak se tomu zabrání? Aplikace musí požádat jádro o službu, jádro ověří, zda aplikace má na podobnou činnost oprávnění a jádro činnost provede
- **Režim jádra**
  - Zde jsou povoleny všechny instrukce
  - Běží v něm jádro, které mj. vykonává služby (systémová volání), o které je aplikace požádá

Jak se dostat z uživatelského režimu do režimu jádra?

- Jde o přepnutí „mezi dvěma světy“, v každém z nich platí jiná pravidla

- Softwarové přerušení – instrukce INT 0x80
  - Stejně jako při hardwarovém přerušení (např. stisk klávesy) se začne vykonávat kód přerušení a vykoná se příslušné systémové volání
- Speciální instrukce (sysenter)
  - Speciální instrukce mikroprocesoru

Systémové volání

- Pojem **systémové volání** znamená vyvolání služby operačního systému, kterou by naše uživatelská aplikace nemohla sama vykonat, např. již zmíněný přístup k souboru na disku.

- Aplikace může volat systémové volání přímo (*open, creat*), nebo prostřednictvím knihovní funkce (v C např. fopen), která následně požádá o systémové volání sama.

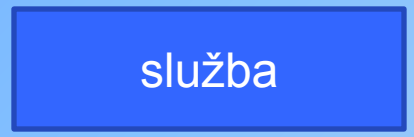
- Výhodou knihovní funkce je, že je na různých platformách stejná, ať už se vyvolání systémové služby děje různým způsobem na různých platformách.

Systémové volání - příklad

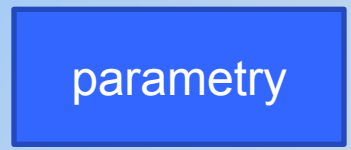
1. Do nějakého registru uložím číslo služby, kterou chci vyvolat
   - Je to podobné klasickému číselníku
   - Např. služba 1- vytvoření procesu, 2- otevření souboru, 3- zápis do souboru, 4- čtení ze souboru, 5- výpis řetězce na obrazovku atd.
2. Do dalších registrů uložím další potřebné parametry
   - Např. kde je jméno souboru který chci otevřít
   - Nebo kde začíná řetězec, který chci vypsat
3. Provedu instrukci, která mě přepne do světa jádra
   - tedy INT 0x80 nebo sysenter
4. V režimu jádra se zpracovává požadovaná služba
   - Může se stát, že se aplikace zablokuje, např. čekání na klávesu
5. Návrat, uživatelský proces pokračuje dále

Příklad

Jen ideový, v reálném systému se příslušné registry mohou jmenovat jinak




```
LD CX, 5      .. Budeme volat službu 5 (tisk řetězce)
LD BX, 100    .. Od adresy 100 bude uložený řetězec
LD AX, 10     .. Délka řetězce, co se bude tisknout
INT 0x80      .. Vyvolání sw přerušení, přechod do světa jádra
   …          .. Vykonání systémového volání
…             .. Kód naší aplikace pokračuje dále
```



Příklad reálný – Linux - getpid.c

```
int pid;

int main() {

    __asm__(
            "movl $20, %eax  \n"     /* getpid system call */
            "int  $0x80      \n"     /* syscall */
            "movl %eax, pid  \n"     /* get result */
            );
   printf("Test volani systemove sluzby...\nPID: %d\n", pid);


  return 0;
}
```

- do registru EAX dáme číslo služby 20
- systémové volání přes int 0x80
- v registru EAX máme návratovou hodnotu pro naši službu (getpid)

Jak zjistím číslo služby?

např.
http://www.cli.di.unipi.it/~gadducci/SOL-11/Local/referenceCards/LINUX_System_Call_Quick_Reference.pdf
(kolik různých volání jste napočítali?)

Možnosti systémového volání

- předchozí slide – inline assembler a INT 0x80
- použití instrukce syscall()

```
id1 = syscall (SYS_getpid);
```

- přímo volání getpid() .. wrapper v libc knihovně

```
id2 = getpid();
```

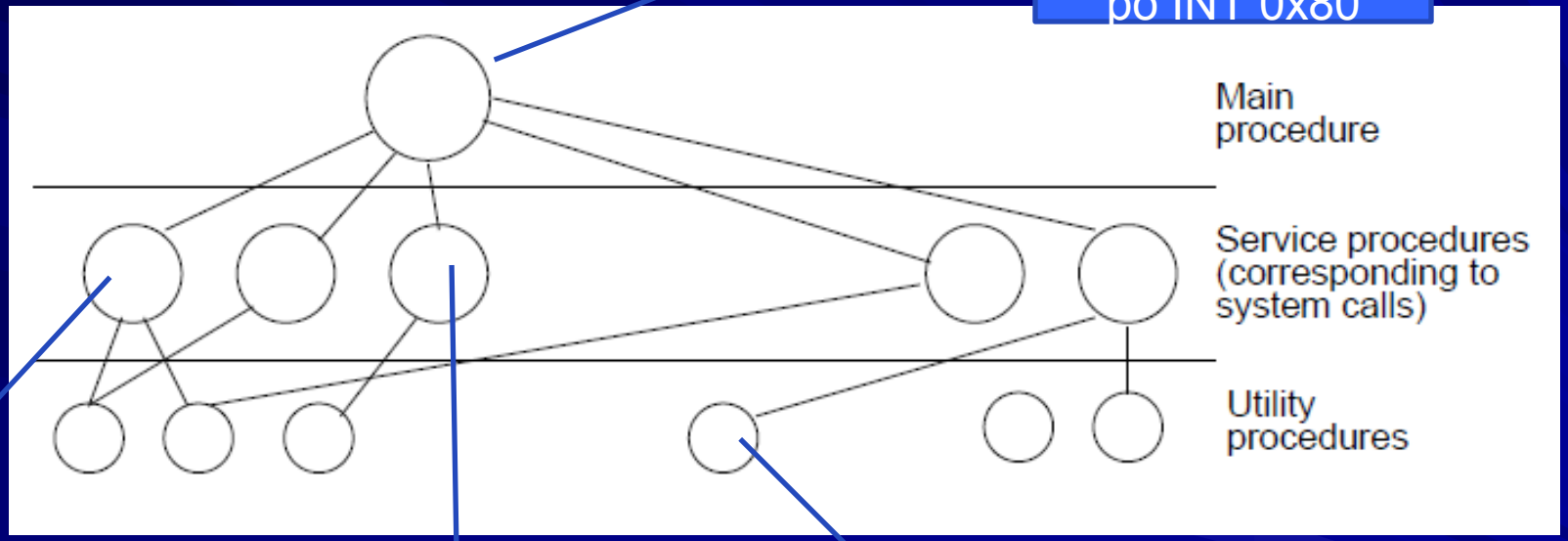
Další možnosti uložení parametrů

- Musím nějak jádru říci, kterou službu chci a další parametry
- Informaci můžeme uložit
  - Do registrů
  - Na zásobník
  - Na předem danou adresu v paměti
  - Kombinací uvedených principů
- Příklad informace:
  chci po OS službu 2 (natočení piva) číslo služby uložím do CX, do registru AX uložím velikost piva (malé), do registru BX stupeň (desítka)…
- *Jádro ví, že když je zavolána služba 2, tak jak má interpretovat obsah registrů AX a BX*

Jak jádro implementuje jednotlivé služby?

Ukážeme si na monolitickém jádru:

Vstupní bod jádra, sem se dostaneme např. po INT 0x80

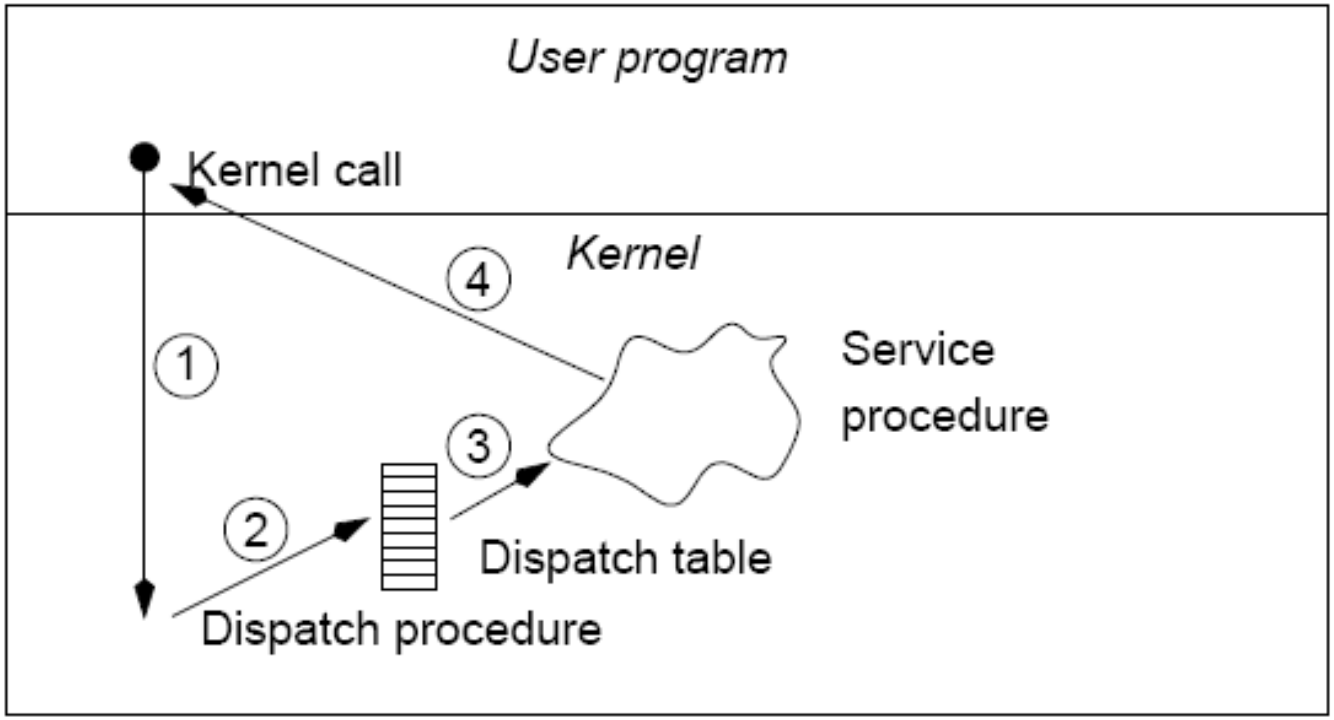


Služba 1
Např. otevři soubor

Služba 3
Např. natoč pivo

Pomocné procedury,
Může je využívat i více služeb

Vyvolání služby systému (opakování)



Vyvolání služby systému (opakování)

- Parametry uložíme na určené místo
  - registry, zásobník
- Provedeme speciální instrukci (1)
  - vyvolá obsluhu v jádře
  - přepne do privilegovaného režimu
- OS převezme parametry, zjistí, která služba je vyvolána a provede službu
- návrat zpět
  - Přepnutí do uživatelského režimu

Poznámka

- Systémové volání nevyžaduje přepnutí kontextu na jiný proces
- Je zpracováno v kontextu procesu, který jej vyvolal

Co znamená INT x?

- instrukce v assembleru pro x86 procesory, která generuje sw přerušení
- x je v rozsahu 0 až 255
- paměť od 0 do je 256 4bytových ukazatelů (celkem 1KB), obsahují adresu pro obsluhu přerušení – vektor přerušení
- HW interrupty jsou mapovány na dané vektory prostřednictvím programovatelného řadiče přerušení

INT 0x80

- v 16kové soustavě 80, dekadicky 128
- pro vykonání systémového volání
- do registru EAX se dá číslo systémového volání, které chceme vyvolat

Jak se aplikace dostane do režimu jádra? (opakování)

- Softwarové přerušení
  - Volající proces způsobí softwarové přerušení
  - Na platformě x86: instrukce int 0x80
  - Přerušení se začne obsluhovat, procesor se přepne do režimu jádra a začne se provádět kód jádra
- Speciální instrukce
  - Novější, rychlejší
  - Platforma x86: instrukce sysenter, sysexit

Může se lišit na různých platformách

Přerušení (interrupt)

- Přerušení patří k základním mechanismům používaným v OS
- **Asynchronní** obsluha události, procesor **přeruší** vykonávání sledu instrukcí (části kódu, které se právě věnuje), **vykoná** obsluhu přerušení (tj. instrukce v obslužné rutině přerušení) a **pokračuje** předchozí činností
- Analogie:
  - vařím oběd (vykonávám instrukce běžného procesu),
  - zazvoní telefon (přijde přerušení, je to asynchronní událost – kdykoliv)
  - Vyřídím telefon (obsluha přerušení)
  - Pokračuji ve vaření oběda (návrat k předchozí činnosti)
- Některé systémy mají víceúrovňová přerušení (vnoření)
  - (telefon přebije volání, že na někoho v sousedním pokoji spadla skříň)

Druhy přerušení (!!)

- **Hardwarové přerušení (vnější)**
  - Přichází z I/O zařízení, např. stisknutí klávesy na klávesnici
  - **Asynchronní** událost – uživatel stiskne klávesu, kdy se mu zachce
  - Vyžádá si pozornost procesoru bez ohledu na právě zpracovávanou úlohu
  - Doručovány prostřednictvím **řadiče přerušení** (umí stanovit **prioritu** přerušením,aj.)
- **Vnitřní přerušení**
  - Vyvolá je sám procesor
  - Např. pokus o dělení nulou, výpadek stránky paměti (!!)
- **Softwarové přerušení**
  - Speciální strojová instrukce (např. zmiňovaný příklad INT 0x80)
  - Je **synchronní**, vyvolané záměrně programem (chce službu OS)
    volání služeb operačního systému z běžícího procesu (!!)
    uživatelská úloha nemůže sama skočit do prostoru jádra OS, ale má právě k tomu softwarové přerušení

- Doporučuji přečíst:
  http://cs.wikipedia.org/wiki/P%C5%99eru%C5%A1en%C3%AD

Kdy v OS použiji přerušení? (to samé z jiného úhlu pohledu)

- **Systémové volání** (volání služby OS)
  - Využiji softwarového přerušení a instrukce INT
- **Výpadek stránky paměti**
  - V logickém adresním prostoru procesu se odkazuji na stránku, která není namapovaná do paměti RAM (rámec), ale je odložená na disku
  - Dojde k přerušení – výpadek stránky
    - Běžící proces se pozastaví
    - Ošetří se přerušení – z disku se stránka natáhne do paměti (když je operační pamět plná, tak nějaký rámec vyhodíme dle nám známých algoritmů ☺)
    - Pokračuje původní proces přístupem nyní už do paměti RAM
- **Obsluha HW zařízení**
  - Zařízení si **žádá pozornost**
  - Klávesnice: stisknuta klávesa
  - Zvukovka : potřebuji poslat další data k přehrání
  - Síťová karta: došel paket

Vektor přerušení

- I/O zařízení signalizuje přerušení *(něco potřebuji)*
- Přerušení přijde na nějaké lince přerušení (IRQ, můžeme si představit jeden drát ke klávesnici, jiný drát k sériovému portu, další k časovači atd.)
- Víme číslo drátu (např. IRQ 1), ale potřebujeme vědět, na jaké adrese začíná obslužný program přerušení
- Kdo to ví? … vektor přerušení
- **Vektor přerušení** je vlastně index do pole, obsahující adresu obslužné rutiny, vykonané při daném typu přerušení

Poznámka k přerušením

„signál“ operačnímu systému, že nastala nějaká událost, která vyžaduje ošetření (vykonání určitého kódu) - asynchronní

- Hardwarové přerušení
  - původ v HW
  - Stisk klávesy, pohnutí myši
  - Časovač (timer)
  - Disk, síťová karta, ztráta napájení,..
- Softwarová přerušení
  - Pochází ze SW
  - Obvykle z procesu v uživatelském režimu

Poznámka k přerušením

Příchod přerušení, z tabulky přerušení pozná, kde leží obslužný kód pro dané přerušení

Pozn. pro sw přerušení 0x80 ukazuje v tabulce přerušení (vektor přerušení) na vstupní bod OS

Maskování přerušení – v době obsluhy přerušení (musí být rychlá) lze zamaskovat méně důležitá přerušení

Sw přerušení jsou nemaskovatelná

Přijde-li přerušení… (!!)

- Přijde signalizace přerušení
- Dokončena rozpracovaná strojová instrukce
- Na zásobník uložena adresa následující instrukce
- Podle tabulky přerušení vyvolána obsluha přerušení
- Obsluha
  - Rychlá
  - Na konci stejný stav procesoru jako na začátku
- Instrukce návratu RET, IRET
  - Vyzvedne ze zásobníku návratovou adresu a na ní pokračuje
- Přerušená úloha (mimo zpoždění) nepozná, že proběhla obsluha přerušení

Knihovní funkce

- mechanismy volání jádra se v různých OS liší
- knihovní funkce
  - programovací jazyky zakrývají služby systému, aby se jevily jako běžné knihovní funkce
  - Jazyk C: printf(“Retezec”);
    - Knihovní funkce se sama postará, aby vyvolala vhodnou službu OS na dané platformě pro výpis řetězce
- Ne vždy musí volání knihovní funkce znamenat systémové volání, např. matematické funkce

Architektury OS

OS = jádro + systémové nástroje

Jádro se zavádí do operační paměti při startu a zůstává v činnosti po celou dobu běhu systému

Základní rozdělení:

- Monolitické jádro – jádro je jeden funkční celek
- Mikrojádro – malé jádro, oddělitelné části pracují jako samostatné procesy v user space
- Hybridní jádro - kombinace

Architektura OS

Linux

Rodina OS:	Unix-like
Aktuální verze:	3.5 / 21. července 2012
Podporované platformy:	IA-32, x86-64, PowerPC, ARM, m68k, DEC Alpha, SPARC, hppa, IA-64, MIPS, s390 a další
Typ kernelu:	Monolitické jádro
Implicitní uživatelské rozhraní:	GNOME, KDE, Xfce a jiné
Licence:	GNU GPL a jiné
Stav:	Aktuální

Windows 7

Web:	Windows 7
Vyvíjí:	Microsoft
Rodina OS:	Windows NT
Druh:	Uzavřený vývoj
Aktuální verze:	Service pack 1 SP1 / 15.3.2011
Způsob aktualizace:	Windows Update
Správce balíčků:	Windows Installer
Podporované platformy:	x86, x86_64
Typ kernelu:	Hybridní jádro
Implicitní uživatelské rozhraní:	Grafické uživatelské rozhraní
Licence:	Microsoft EULA
Stav:	finální verze

Mac OS X

Web:	www.apple.com/macosx/
Vyvíjí:	Apple Inc.
Rodina OS:	BSD
Druh:	Uzavřený vývoj (s využitím open source komponent)
Aktuální verze:	10.8 / 19. července 2012
Podporované platformy:	x86, x86-64, PowerPC (32bitový i 64bitový)
Typ kernelu:	Hybridní jádro
Implicitní uživatelské rozhraní:	Aqua
Licence:	Apple SLA (část pod APSL)
Stav:	Aktivní

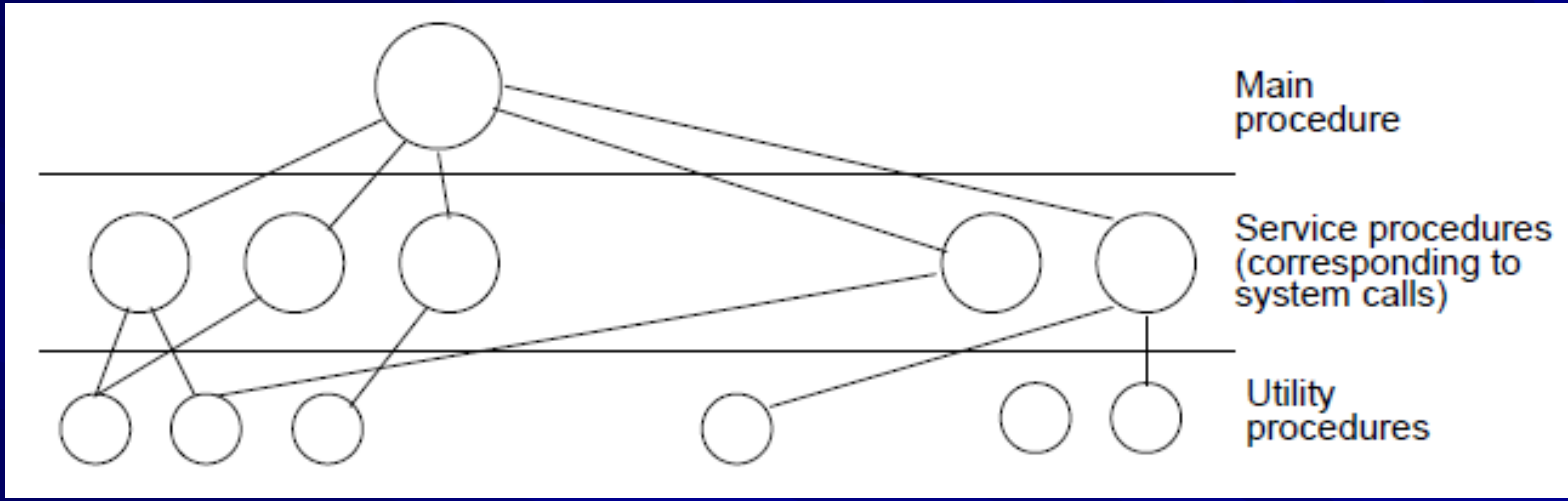
Monolitické jádro

- Jeden spustitelný soubor
- Uvnitř moduly pro jednotlivé funkce
- Jeden program, řízení se předává voláním podprogramů
- Příklady: UNIX, Linux, MS DOS

Typickou součástí jádra je např. souborový systém

Linux je monolitické jádro OS, s podporou zavádění modulů za běhu systému

Monolitické jádro (!)

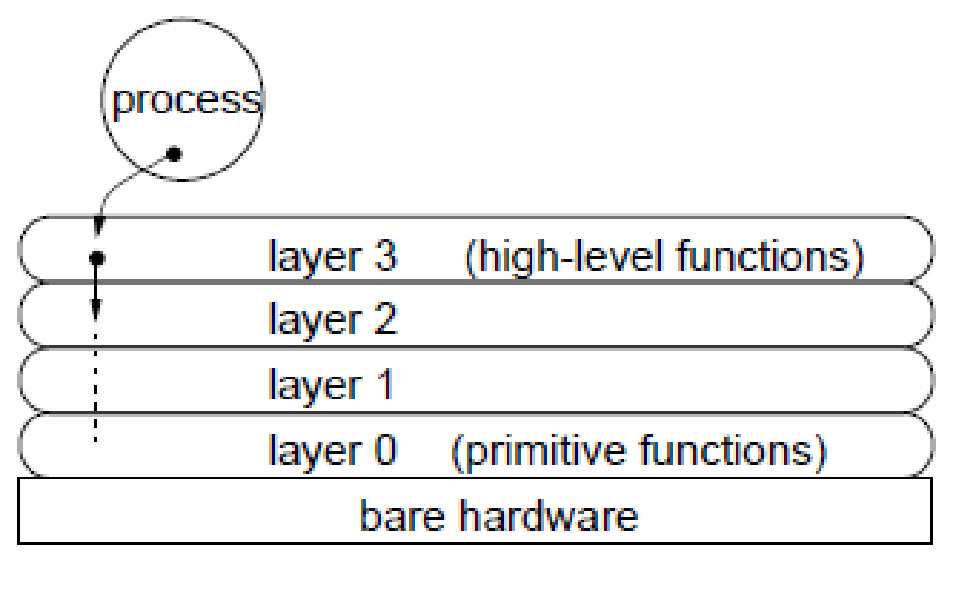


Main procedure – vstupní bod jádra, na základě čísla služby zavolá servisní proceduru
Service procedure – odpovídá jednotlivým systémovým voláním (zobrazení řetězce, čtení ze souboru, aj.)
Service procedure volá pro splnění svých cílů různé pomocné utility procedures (lze je opakovaně využít v různých voláních)

Vrstvené jádro

- Výstavba systému od nejnižších vrstev
- Vyšší vrstvy využívají primitiv poskytovaných nižšími vrstvami
- Hierarchie procesů
  - Nejníže vrstvy komunikující s HW
  - Každá vyšší úroveň poskytuje abstraktnější virtuální stroj
  - Může být s HW podporou – pak nelze vrstvy obcházet (obdoba systémového volání)
- Příklady: THE, MULTICS

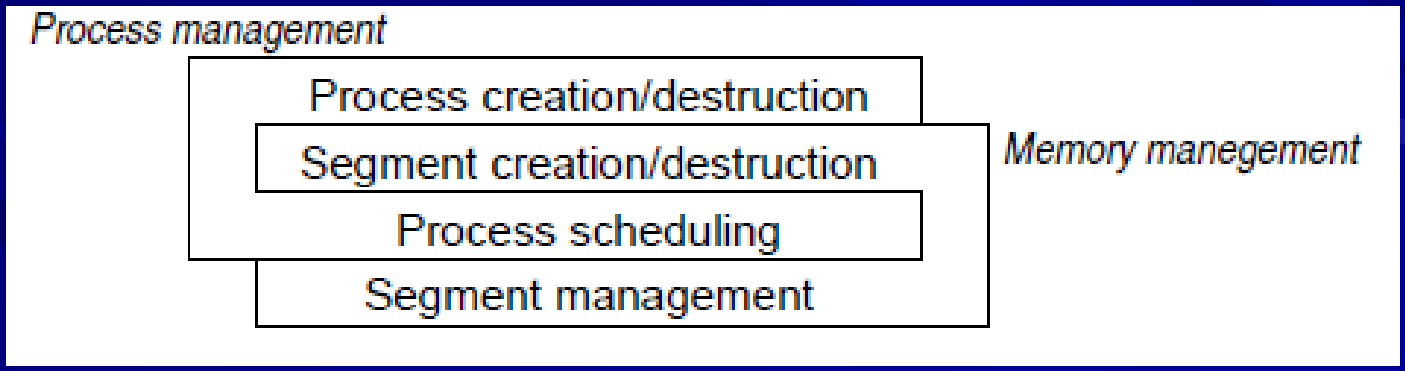
Vrstvené jádro



OS THE:
Úroveň 0 .. Virtualizace CPU (přepínání mezi procesy)
vyšší vrstva už předpokládá existenci procesů
Úroveň 1 .. Virtualizace paměti
vyšší vrstvy už nemusí řešit umístění části procesů v paměti

Funkční hierarchie

- Problém jak rozčlenit do vrstev
  - „dřív slepice, nebo vejce?“
  - správa procesů vs. správa paměti
- Některé moduly vykonávají více funkcí, mohou být na více úrovních v hierarchii
- Př: Pilot



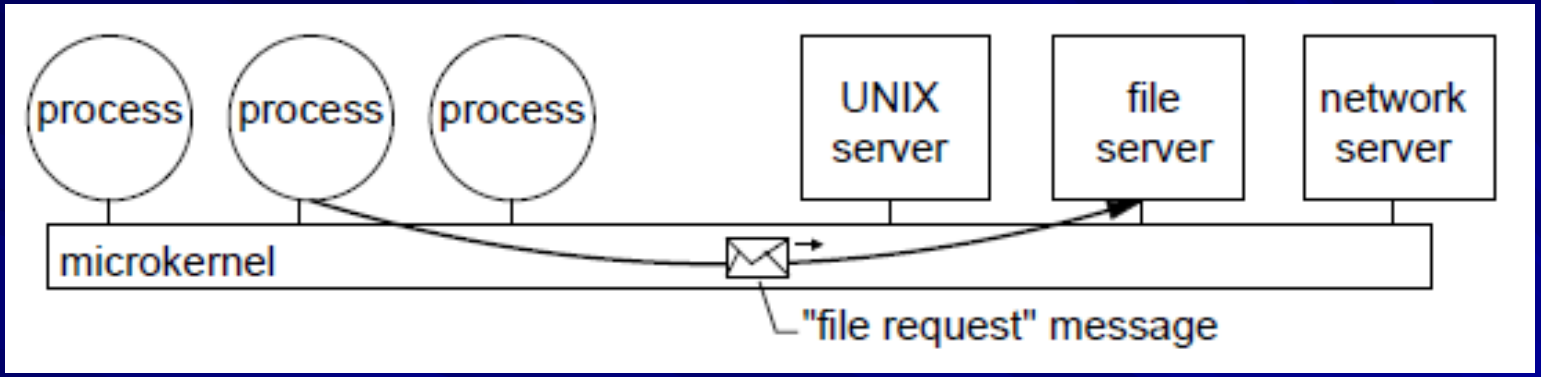
Mikrojádro (!)

- Model klient – server
- Většinu činností OS vykonávají samostatné procesy mimo jádro (servery, např. systém souborů)
- Mikrojádro
  - Poskytuje pouze nejdůležitější nízkoúrovňové funkce
    - Nízkoúrovňová správa procesů
    - Adresový prostor, komunikace mezi adresovými prostory
    - Někdy obsluha přerušení, vstupy/výstupy
  - Pouze mikrojádro běží v privilegovaném režimu
    - Méně pádů systému

Mikrojádro

- Výhody
  - vynucuje modulární strukturu
  - Snadnější tvorba distribuovaných OS (komunikace přes síť)
- Nevýhody
  - Složitejší návrh systému
  - Režie
- Příklady: QNX, Hurd, OSF/1, MINIX, Amoeba

Mikrojádro



Mikrojádro – základní služby, běží v privilegovaném režimu

1. proces vyžaduje službu
2. mikrojádro předá požadavek příslušnému serveru
3. server vykoná požadavek

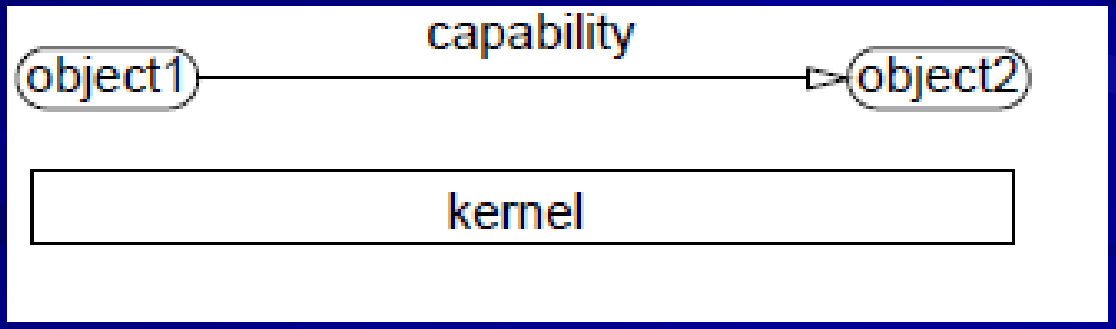
Snadná vyměnitelnost serveru za jiný

Chyba serveru nemusí být fatální pro celý operační systém (není v jádře)

Server může event. běžet na jiném uzlu sítě (distribuov. syst.)

Objektově orientovaná struktura

- Systém je množina objektů (soubory, HW zařízení)
- Capability = odkaz na objekt + množina práv definujících operace, spravuje jádro
- Jádro si vynucuje tento abstraktní pohled
- Jádro kontroluje přístupová práva
- Př: částečně Windows NT (Win 2000, XP,..) - hybridní



Hybridní jádro

- Kombinuje vlastnosti monolitického a mikrojádra
- Část kódu součástí jádra (monolitické)
- Jiná část jako samostatné procesy (mikrojádro)

- Příklady
  - Windows NT (Win 2000, Win XP, Windows Server 2003, Windows Vista,..)
  - Windows CE (Windows Mobile)
  - BeOS

GNU/Linux, GNU/Hurd

GNU/Linux

- GNU programy, např. gcc (R. Stallman)
- Monolitické jádro OS – Linux (Linus Torvald)

GNU/Hurd

- GNU programy, např. gcc
- Mikrojádro Hurd

Linux

- Pojem Linux jako takový označuje jádro operačního systému
- Pokud hovoříme o operačním systému, správně bychom měli říkat GNU/Linux, ale toto přesné označení používá jen málo distribucí, např. *Debian GNU/Linux*

- Flame war – debata mezi Linusem Torvaldsem a Andrewem Tanenbaumem, že Linux měl být raději mikrokernel

Linux - odkazy

Interaktivní mapa Linuxového jádra:

http://www.makelinux.net/kernel_map

Jaké jádro je nyní aktuální? (např. 2.6.31)

http://kernel.org/

Spuštění operačního systému bez instalace:

Live CD, Live DVD, Live USB

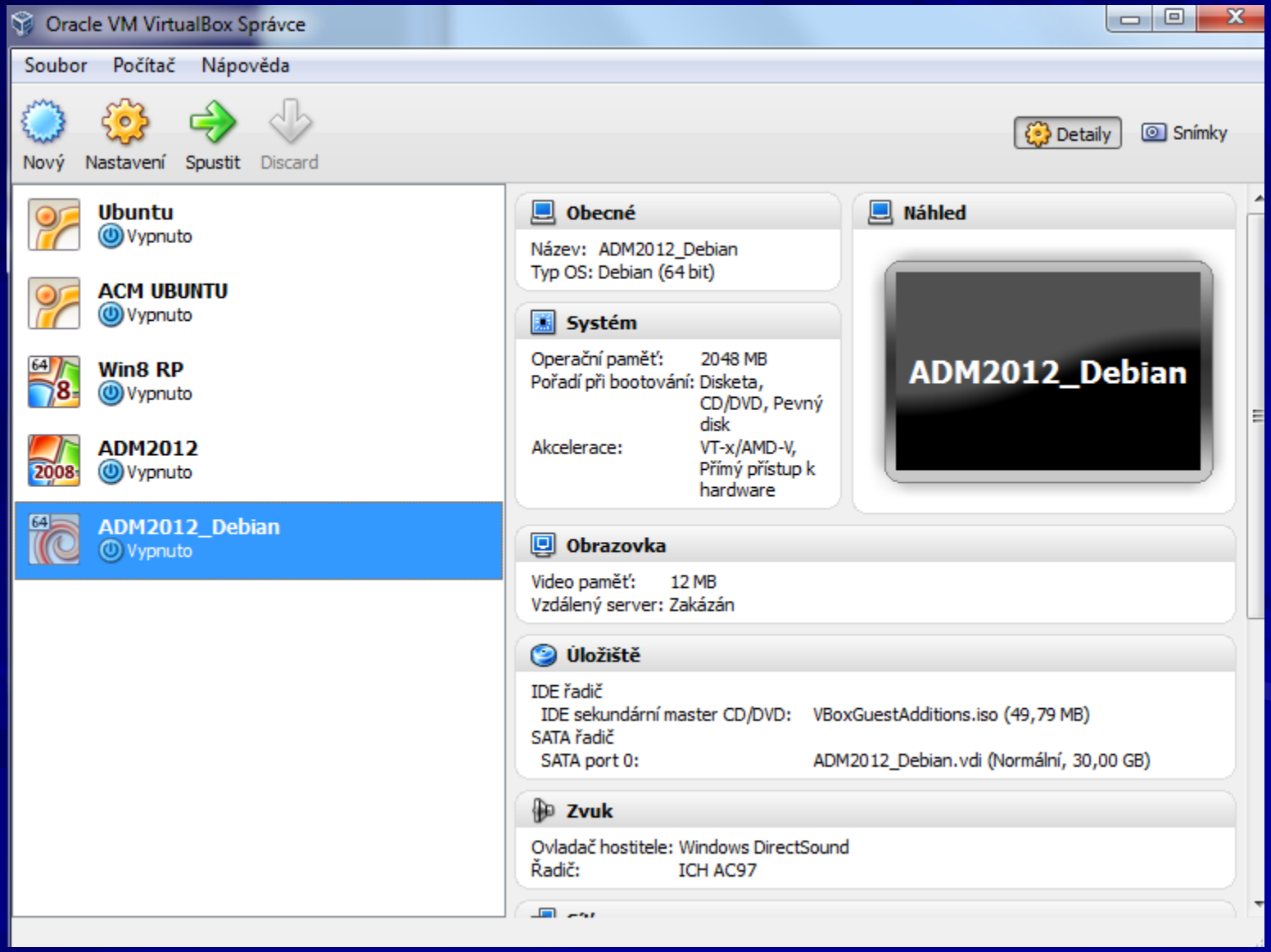
Např. Knoppix (http://knoppix.org/)

Virtualizace

- Možnost nainstalovat si virtuální počítač a provozovat v něm jiný operační systém, včetně přístupu k síti aj.

- VirtualBox
- VmWare
- Xen (např. virtualizace serverů), KVM aj.

VirtualBox


Oracle VM VirtualBox Správce
Soubor
Počítač
Nápověda
Nový
Nastavení
Spustit
Discard
Detaily
Snímky
Ubuntu
Vypnuto
ACM UBUNTU
Vypnuto
Win8 RP
Vypnuto
ADM2012
Vypnuto
ADM2012_Debian
Vypnuto
Obecné
Název: ADM2012_Debian
Typ OS: Debian (64 bit)
Systém
Operační paměť: 2048 MB
Pořadí při bootování: Disketa, CD/DVD, Pevný disk
Akcelerace: VT-x/AMD-V, Přímý přístup k hardware
Náhled
ADM2012_Debian
Obrazovka
Video paměť: 12 MB
Vzdálený server: Zakázán
Úložiště
IDE řadič
IDE sekundární master CD/DVD: VBoxGuestAdditions.iso (49,79 MB)
SATA řadič
SATA port 0: ADM2012_Debian.vdi (Normální, 30,00 GB)
Zvuk
Ovladač hostitele: Windows DirectSound
Řadič: ICH AC97

Literatura, použité zdroje

Obrázky z některých slidů (20, 21, 24) pocházejí z knížky

Andrew S. Tanenbaum: Modern Operating Systems

vřele doporučuji tuto knihu, nebo se alespoň podívat na slidy ke knize dostupné mj. na webu předmětu v *Přednášky -> Odkazy*