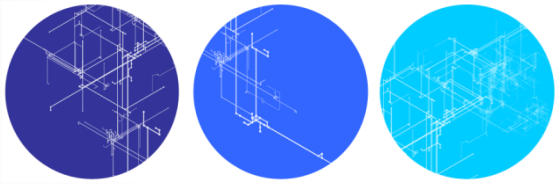
PB153
OPERAČNÍ SYSTÉMY A JEJICH ROZHRANÍ



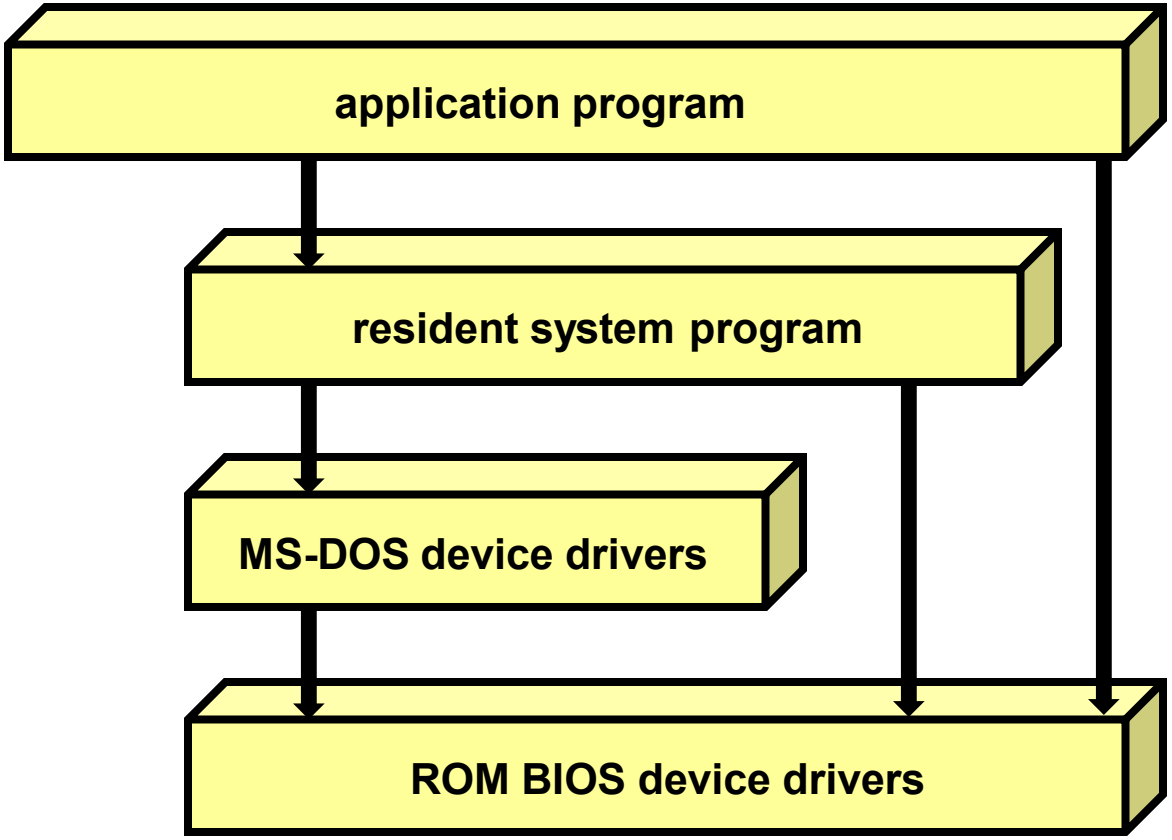
Principy výstavby OS

04

VNITŘNÍ STRUKTURA OS

- Existuje řada přístupů a implementací
  - jedno velké monolitické jádro
  - modulární, hierarchický přístup
  - malé jádro a samostatné procesy
- Struktura mnoha OS je poznamenána historií OS a původními záměry, které se mohou od současného stavu radikálně lišit

PŘÍKLAD: MS-DOS



PŘÍKLAD MS-DOS (2)

- Při programování pro OS MS-DOS využíváme služeb
  - spuštěných rezidentních programů
    - např. ovladač myši (poskytuje služby na INT 33h)
  - operačního systému
    - např. přístup k souborům (INT 21h)
  - BIOSu
    - např. nastavení grafického režimu (INT 10h)
  - přímo HW
    - např. přímo zápis do videopaměti pro zobrazení dat

PŘÍKLAD MS-DOS (3)

- Změna fontů v textovém režimu (bez využití služeb BIOSu, OS, přímo HW)

```
asm cli;
outport(0x3c4,0x0402);
outport(0x3c4,0x0704);
outport(0x3ce,0x0204);
outport(0x3ce,0x0005);
outport(0x3ce,0x0406);
for(i=0;i<=254;i++)
{ for(j=0;j<=15;j++)
{p=MK_FP(0xa000,32*i+j);
*p=font[y]; y++; } }
outport(0x3c4,0x0302);
outport(0x3c4,0x0304);
outport(0x3ce,0x0004);
outport(0x3ce,0x1005);
outport(0x3ce,0x0e06);
asm sti;
```

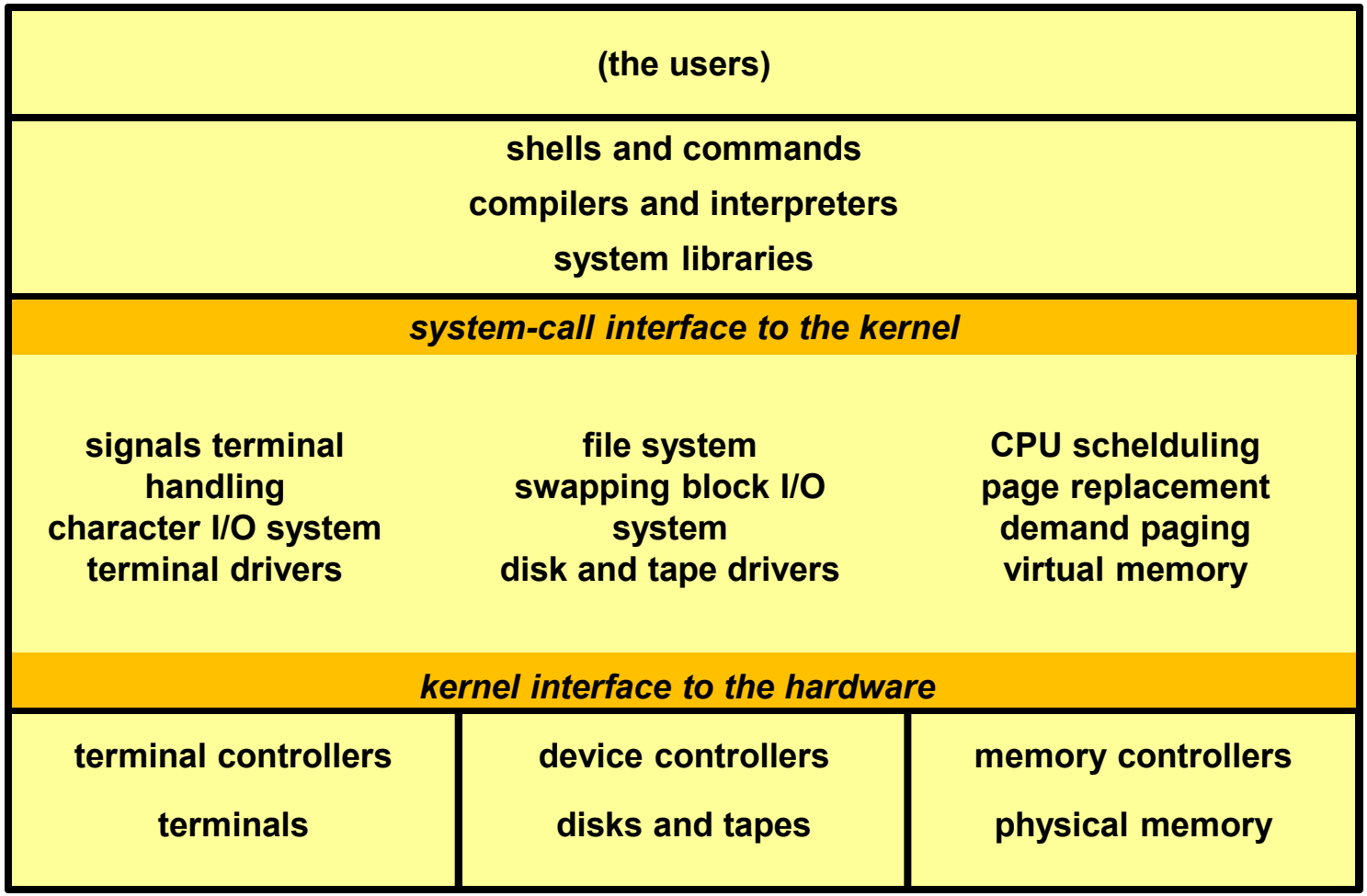
PŘÍKLAD MS-DOS (4)

- Hlavní cíl návrhu
  - maximální možná funkcionalita v co nejmenším prostoru
- Výsledek
  - modulová architektura není aplikovaná
  - i když MS-DOS má jistou strukturu, jeho rozhraní a jednotlivé komponenty nejsou důsledně separovány a uspořádány

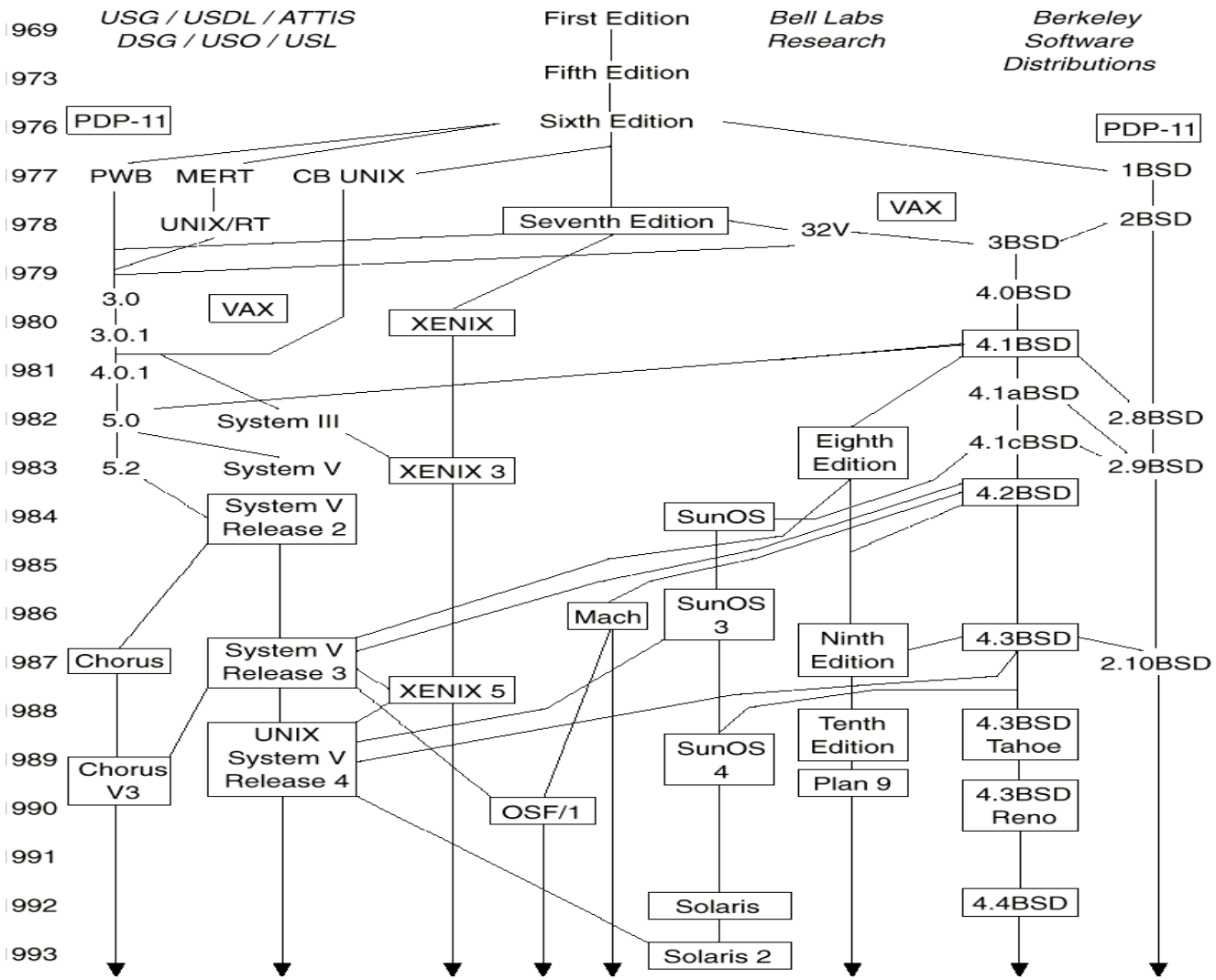
PŘÍKLAD UNIX

- Také omezen hardwarem
  - vznik v polovině 70. let
- OS Unix sestává ze 2 částí
  - systémové programy
  - jádro
    - vše, co se nachází pod rozhraním volání systému a nad fyzickým hardware
    - obstarává plnění funkcí z oblastí systému souborů, plánování CPU, správy paměti, …
    - vrstvová architektura sice existuje, ale hodně funkcí je na jedné úrovni

PŘÍKLAD UNIX (2)



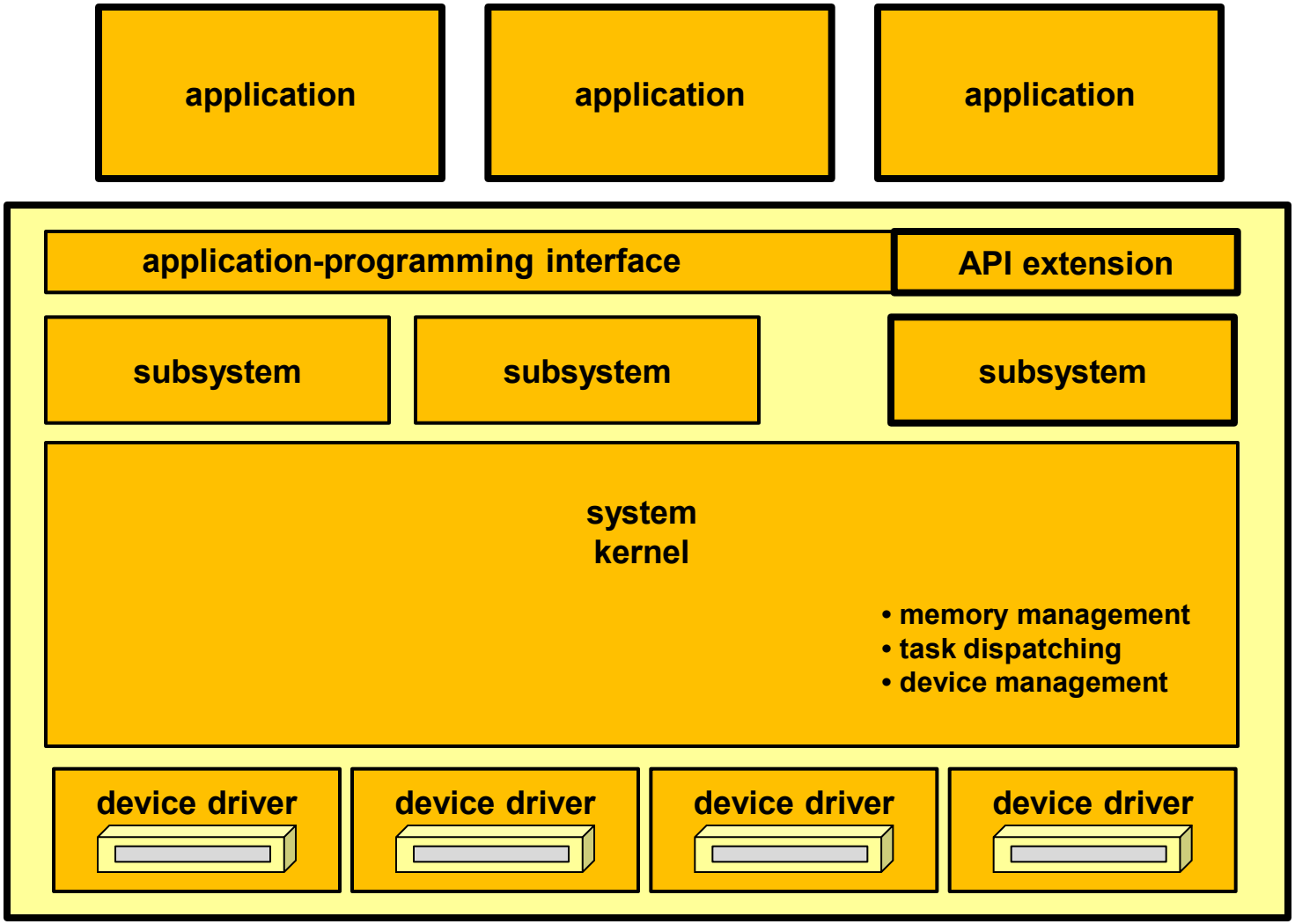
PŘÍKLAD: UNIX (3)



XENIX

- V roce 1979 koupil Microsoft licenci na Unix verze 7 od AT&T.
- V roce 1987 předal Microsft Xenix firmě SCO.
- Koncem 80. let byl Xenix pravděpodobně nejrozšířenějším OS unixového typu podle počtu strojů, na kterých běžel

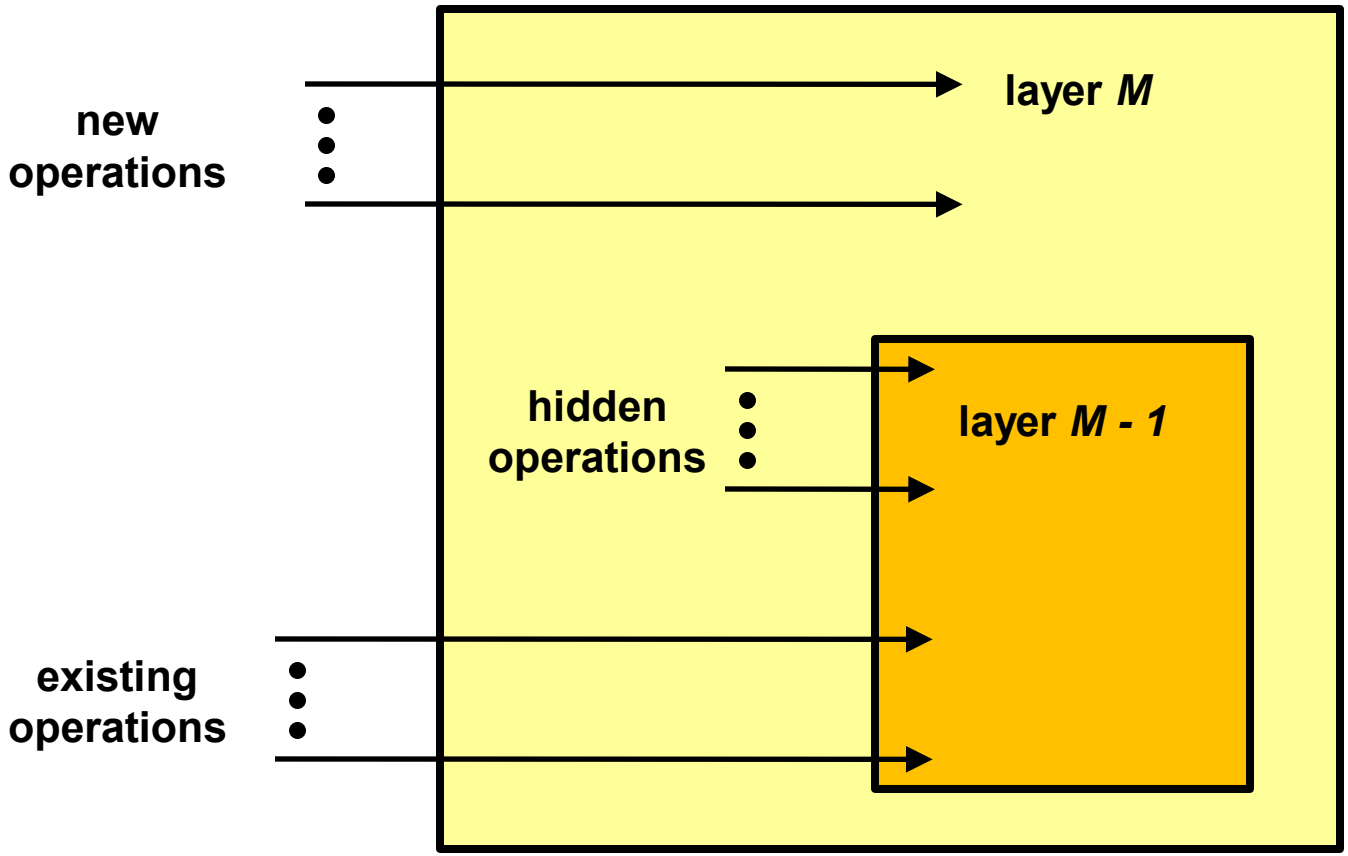
PŘÍKLAD OS/2


application
application
application
application-programming interface
API extension
subsystem
subsystem
subsystem
system
kernel
• memory management
• task dispatching
• device management
device driver
device driver
device driver
device driver

HIERARCHICKÁ VRSTVOVÁ ARCHITEKTURA

- OS se dělí do jistého počtu vrstev (úrovní)
- Každá vrstva je budována na funkcionalitě nižších vrstev
- Nejnižší vrstva (0) je hardware
- Nejvyšší vrstva je uživatelské rozhraní
- Pomocí principu modulů jsou vrstvy vybírány tak, aby každá používala funkcí (operací) a služeb pouze vrstvy $n - 1$

HIERARCHICKÁ ARCHITEKTURA



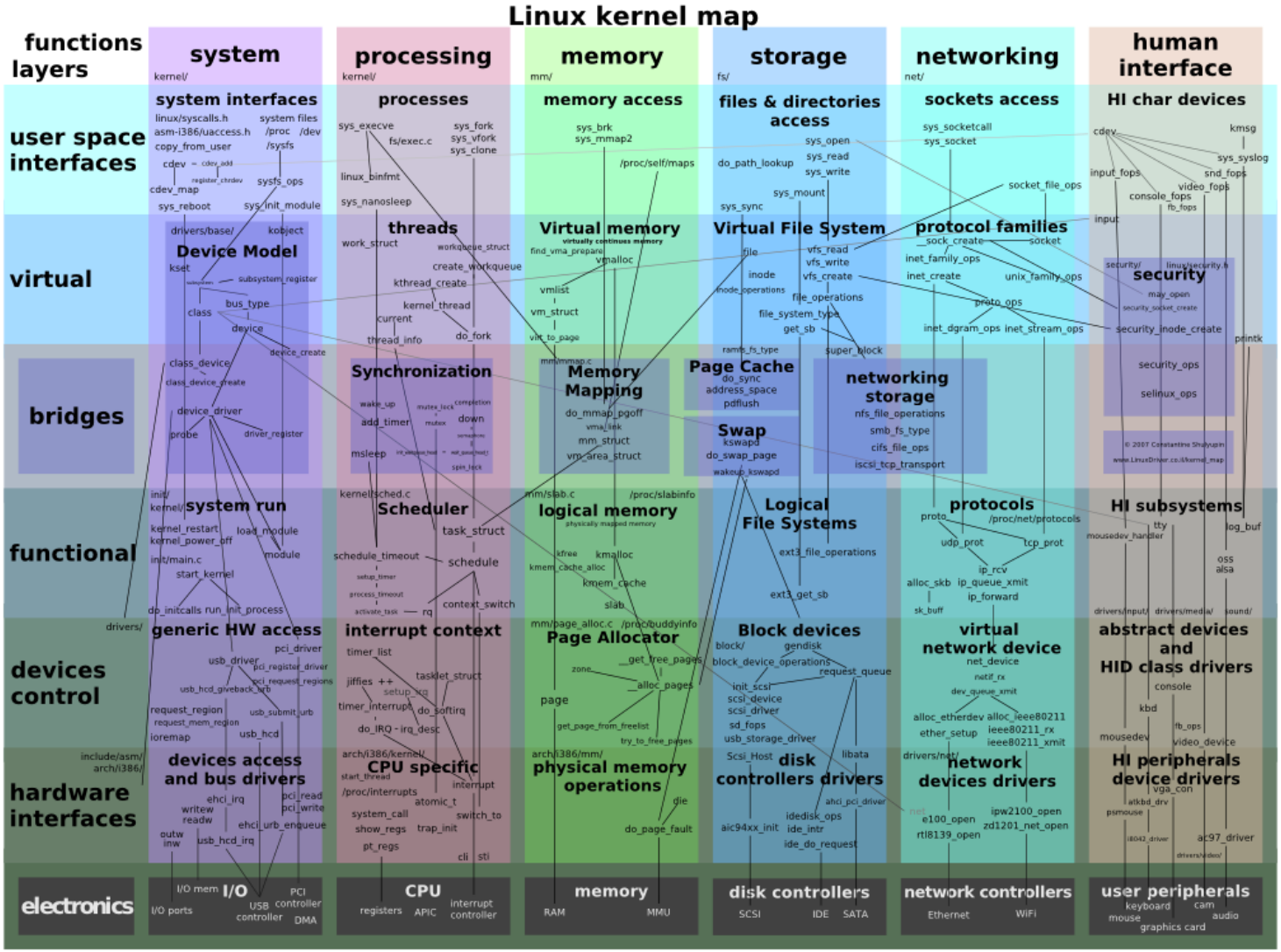
HIERARCHICKÁ STRUKTURA

- Řeší problém přílišné složitosti velkého systému
  - Provádí se dekompozice velkého problému na několik menších zvládnutelných problémů
- Každá úroveň řeší konzistentní podmnožinu funkcí
- Nižší vrstva nabízí vyšší vrstvě „primitivní“ funkce (služby)
- Nižší vrstva nemůže požadovat provedení služeb vyšší vrstvy
- Používají se přesně definovaná rozhraní
  - Jednu vrstvu lze uvnitř modifikovat, aniž to ovlivní ostatní vrstvy – princip modularity

HIERARCHICKÁ STRUKTURA

- Výhodou je modularita OS
- Nevýhodou je především vyšší režie a tím pomalejší vykonávání systémových volání
- Protože efektivita hraje v jádře OS významnou roli je třeba volit kompromis
  - pouze omezený počet úrovní pokrývající vyšší funkcionalitu
  - příklad: první verze Windows NT měly hierarchickou strukturu s řadou vrstev, avšak pro zvýšení výkonu OS bylo ve verzi NT 4.0 rozhodnuto přesunout více funkcionality do jádra a sloučit některé vrstvy

PŘÍKLAD: LINUX



PROVÁDĚNÍ SLUŽEB V KLASICKÉM OS

- Klasický OS (non-process kernel OS)
  - OS je prováděn jako samostatná entita v privilegovaném režimu
  - procesy – jen uživatelské programy
- Služba se provádí jako součást jádra
- Služba se provádí v rámci procesů
  - obecně lze celý OS provádět v kontextu uživatelského procesu
    - Leží v jeho adresovém prostoru
  - přerušení (volání služby OS) vyvolává implicitně pouze přepnutí režimu procesoru (z uživatelského do privilegovaného), ne změnu kontextu
  - k přepínání kontextu procesů dochází jen tehdy, je-li to nutné z hlediska plánování
  - pro volání procedur v rámci jádra se používá samostatný zásobník
  - program a data OS jsou ve sdíleném adresovém prostoru a sdílí je všechny uživatelské procesy

SLUŽBY V PROCESOVĚ KONSTRUOVANÉM OS

- OS je souhrnem systémových procesů
- Jádro tyto systémové procesy separuje, ale umožňuje jim synchronizaci a komunikaci
- Snaha o provádění co nejmenší části kódu v privilegovaném režimu procesoru
- V krajním případě je jádro pouze ústředna pro přepojování zpráv
- Takové řešení OS je snadno implementovatelné na multiprocesorových systémech
- Malé jádro - mikrojádro

STRUKTURA S MIKROJÁDREM

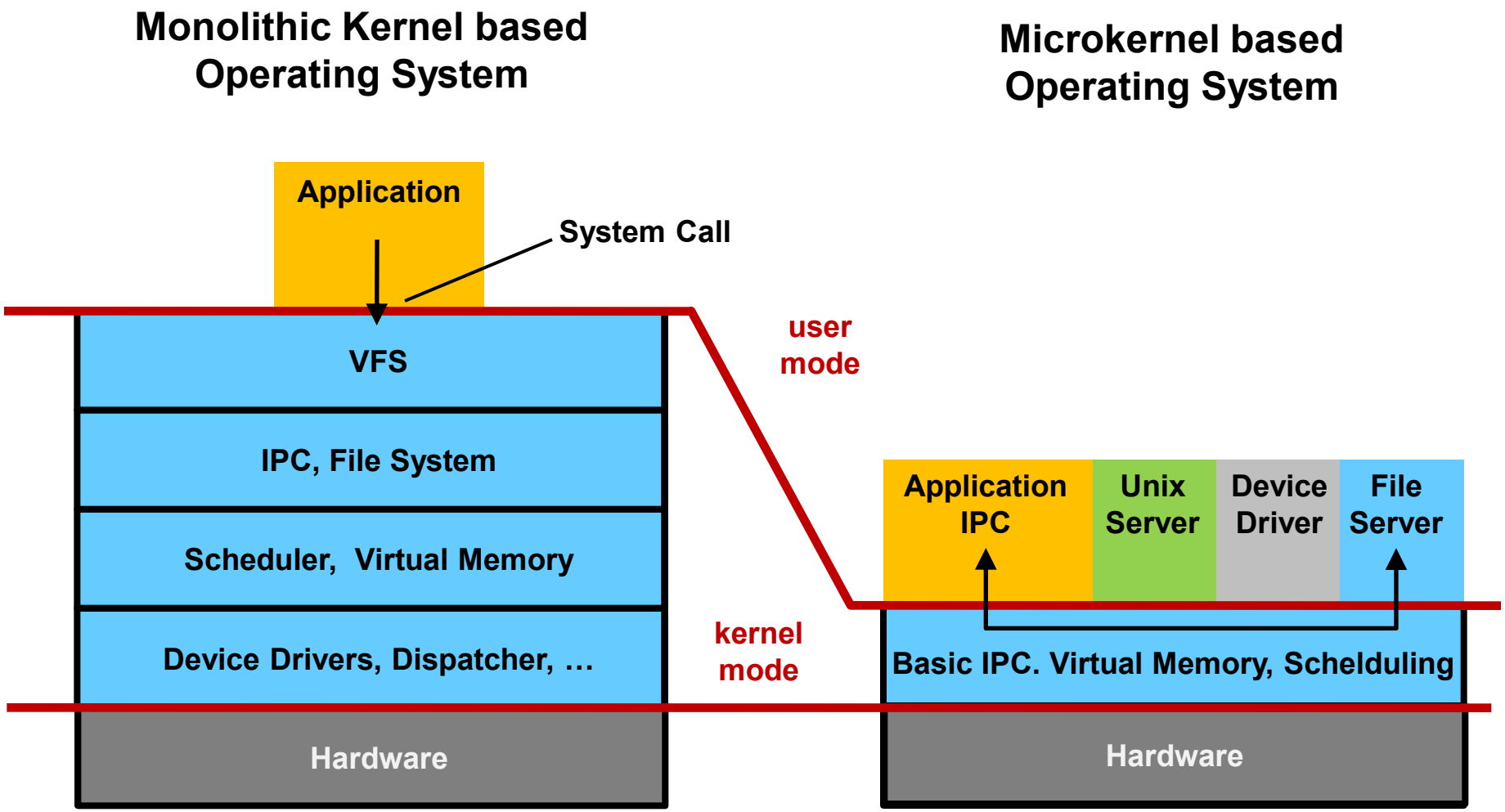
- Microkernel System Structure
- Malé jádro OS plnící pouze několik málo nezbytných funkcí
  - primitivní správa paměti (adresový prostor)
  - komunikace mezi procesy – Interprocess communication (IPC)
- Většina funkcí z jádra se přesouvá do „uživatelské“ oblasti
  - ovladače HW zařízení, služby systému souborů, virtualizace paměti …
  - mezi uživatelskými procesy se komunikuje předáváním zpráv

STRUKTURA S MIKROJÁDREM (2)

- Výhody mikrojádra
  - snadná přenositelnost OS, jádro je malé
  - vyšší spolehlivost (moduly mají jasné API a jsou snadněji testovatelné)
  - vyšší bezpečnost (méně kódu OS běží v režimu jádra)
  - flexibilita (jednodušší modifikace, přidání, odebrání modulů)
  - všechny služby jsou poskytovány jednotně (výměnou zpráv)

- Nevýhoda mikrojádra
  - zvýšená režie
    - volání služeb je nahrazeno výměnou zpráv mezi procesy

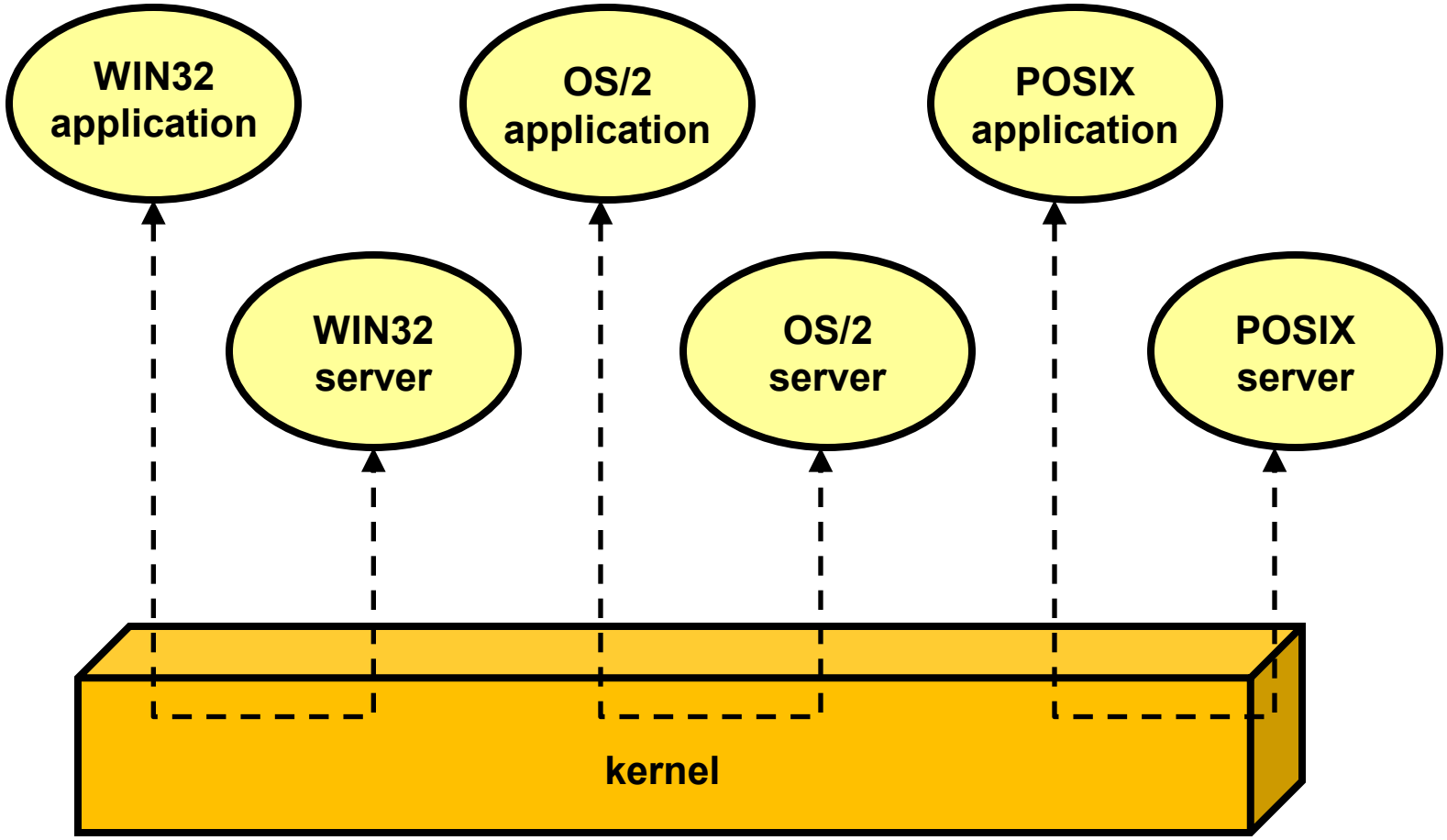
MIKROJÁDRO A MONOLITICKÉ JÁDRO



MACH

- Klasickým příkladem OS s mikrojádrem je Mach vytvořený v 80. letech
- Na přístupu Mach je založen např. Tru64 UNIX nebo realtimový OS QNX
- Windows NT používají hybridní strukturu
  - jádro má vrstevnou strukturu a zajišťuje komunikaci aplikace se „servery“
  - pro jednotlivé typy aplikací (Win32, OS/2, POSIX) existují „servery“ běžící v uživatelském režimu

PŘÍKLAD: WINDOWS NT



PŘÍKLAD: WINDOWS NT (pokr.)

- Další vývoj těchto subsystémů
  - OS/2 subsystém naposled ve Windows 2000
  - POSIX subsystém je v novějších Windows (ne variantách Home apod.) k dispozici ve formě „Subsystem for Unix-based Applications“ (SUA)
    - Do Windows 8. Od Windows 8.1 odstraněno.
  - Win32 se nyní jmenuje Windows API
    - A zahrnuje také API na 64bitových systémech

LINUX: MODULARITA

- Do linuxového jádra můžeme při běhu přidávat kód – moduly
  - LKM – Loadable Kernel Module
- Přesto je Linuxové jádro monolitické
  - Moduly běží stejně jako zbytek jádra v privilegovaném režimu
  - Jde o modularitu kódu jádra ne o modulární architekturu jádra (mikrojádro)

LINUX: MODULARITA

- LKM umožňují:
  - Přidávat funkčnost za běhu
    - Např. připojení nového USB zařízení
  - Snižují paměťové nároky jádra
    - Nahráváme jen moduly, které potřebujeme
    - Oproti speciálně zkompilovanému jádru však mají vyšší režii

LINUX: MODULARITA

- insmod, rmmod, lsmod, modinfo, depmod, modprobe

```
[root@localhost ~]# lsmod
Module                  Size  Used by
snd_intel8x0           28144  0
snd_ac97_codec         92136  1 snd_intel8x0
ac97_bus                1424  1 snd_ac97_codec
snd_seq                46960  0
snd_seq_device          6232  1 snd_seq
ppdev                   8200  0
snd_pcm                64932  2 snd_intel8x0,snd_ac97_codec
snd_timer              17992  2 snd_seq,snd_pcm
snd                    50908  6 snd_intel8x0,snd_ac97_codec,snd_seq,snd_seq_devi
ce,snd_pcm,snd_timer
soundcore               5672  1 snd
snd_page_alloc          7892  2 snd_intel8x0,snd_pcm
parport_pc             22748  0
i2c_piix4              11968  0
parport                29300  2 ppdev,parport_pc
i2c_core               23160  1 i2c_piix4
pcnet32                28000  0
mii                     4120  1 pcnet32
floppy                 47700  0
```

LINUX: MODULARITA

- Linux moduly obvykle nepodepisuje

```
[root@localhost ipv4]# modinfo ipip.ko
filename:       ipip.ko
license:        GPL
srcversion:     C977B3330409AAC23EC3ECC
depends:        tunnel4
vermagic:       2.6.31.5-127.fc12.i686.PAE SMP mod_unload 686
```

- Ale všímá si licence …

```
[root@localhost ~]# insmod procview.ko
procview: module license 'unspecified' taints kernel.
Disabling lock debugging due to kernel taint
```

Podepisování modulů v Linuxu

```
[root@localhost nfs]# modinfo nfs.ko
filename:       /lib/modules/3.13.5-202.fc20.x86_64/kernel/fs/nfs/nfs.ko
license:        GPL
author:         Olaf Kirch <okir@monad.swb.de>
alias:          nfs4
alias:          fs-nfs4
alias:          fs-nfs
depends:        fscache,sunrpc,lockd
intree:         Y
vermagic:       3.13.5-202.fc20.x86_64 SMP mod_unload
signer:         Fedora kernel signing key
sig_key:        87:85:EB:98:7C:F3:67:AA:CA:FF:90:88:B2:E4:25:94:3B:75:A4:DB
sig_hashalgo:   sha256
parm:           callback_tcpport:portnr
parm:           nfs_idmap_cache_timeout:int
parm:           nfs4_disable_idmapping:Turn off NFSv4 idmapping when using 'sec=sys' (bool)
parm:           max_session_slots:Maximum number of outstanding NFSv4.1 requests the client will negotiate (ushort)
parm:           send_implementation_id:Send implementation ID with NFSv4.1 exchange_id (ushort)
parm:           nfs4_unique_id:nfs_client_id4 uniquifier string (string)
parm:           recover_lost_locks:If the server reports that a lock might be lost, try to recover it risking data corruption.
bool)
parm:           enable_ino64:bool
```

```
[   35.106002] emptymodul: module verification failed: signature and/or  required key missing - tainting kernel
[   35.111168] Hello world in kernel space.
```

Podepisování modulů v Linuxu

- Podepisování k dispozici od roku 2004
  - Ale nevyužíváno často a nebylo součástí standardního jádra do verze 3.7
- Prosazuje RedHat (Fedora)
  - Aby umožnil tzv. Bezpečný Boot (Secure Boot)
- 2 základní režimy

Table 1. Module States

Module State	Permissive Mode	Enforcing Mode
Unsigned	Ok	EKEYREJECTED
Signed, no public key	ENOKEY	ENOKEY
Validly signed, public key	Ok	Ok
Invalidly signed, public key	EKEYREJECTED	EKEYREJECTED
Validly signed, expired key	EKEYEXPIRED	EKEYEXPIRED
Signed, hash algorithm unavailable	ENOPKG	ENOPKG
Signed, pubkey algorithm unavailable	ENOPKG	ENOPKG
Signature without sig packet	ENOMSG	ENOMSG
Corrupt signature	EBADMSG	EBADMSG
Corrupt file	ELIBBAD	ELIBBAD

LINUX: MODIFIKACE JÁDRA ZA BĚHU

- /dev/kmem
  - Možnost přímo číst/měnit paměť jádra za běhu
  - Přístupné pouze pro administrátora, přesto nebezpečné
  - V řadě distribucí už /dev/kmem nenajdeme
- LKM
  - Běží v privilegovaném režimu procesoru jako zbytek jádra
  - Možnost změny chování jádra
  - Rootkity

LINUX: TABULKA SYSTÉMOVÝCH VOLÁNÍ

- Nejjednodušší způsob jak implementovat rootkit je modifikovat tabulku rutin obsluhujících systémová volání (sys_call_table) a navázat se na volání jako open, readdir, …
- Snaha omezit možnost LKM modifikovat tabulku systémových volání
- Dnes není tento symbol exportován a není tak možné ho v LKM přímo použít a získat tak ukazatel na tabulku

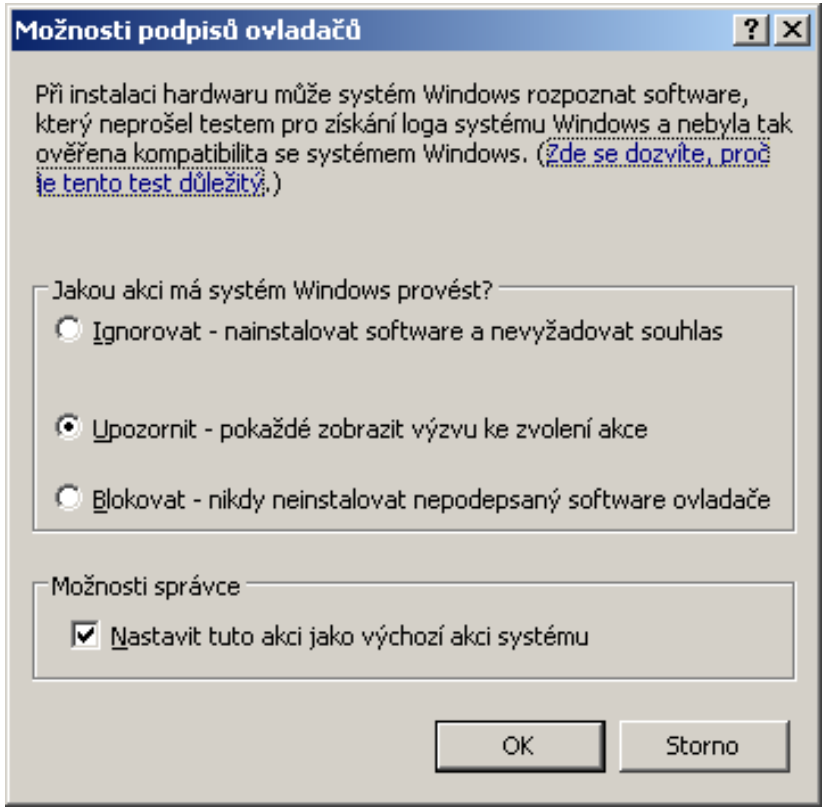
WINDOWS: MODULARITA

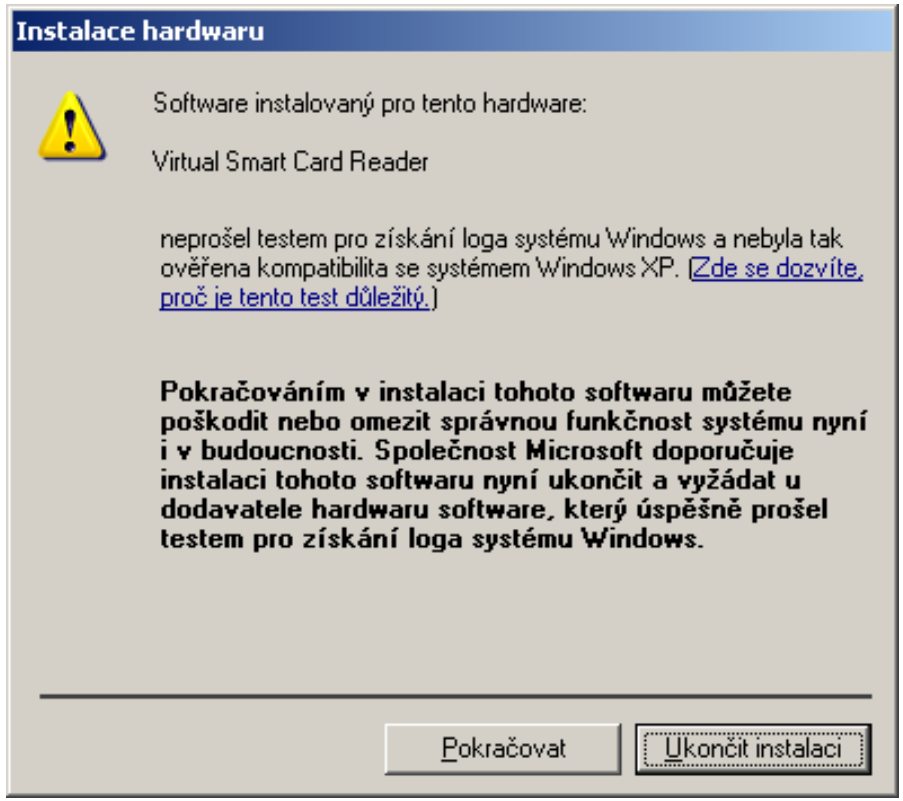
- Do jádra Windows můžeme za běhu vkládat ovladače
- Ty běží v privilegovaném režimu jádra

WINDOWS: MODULARITA (PŘ.)

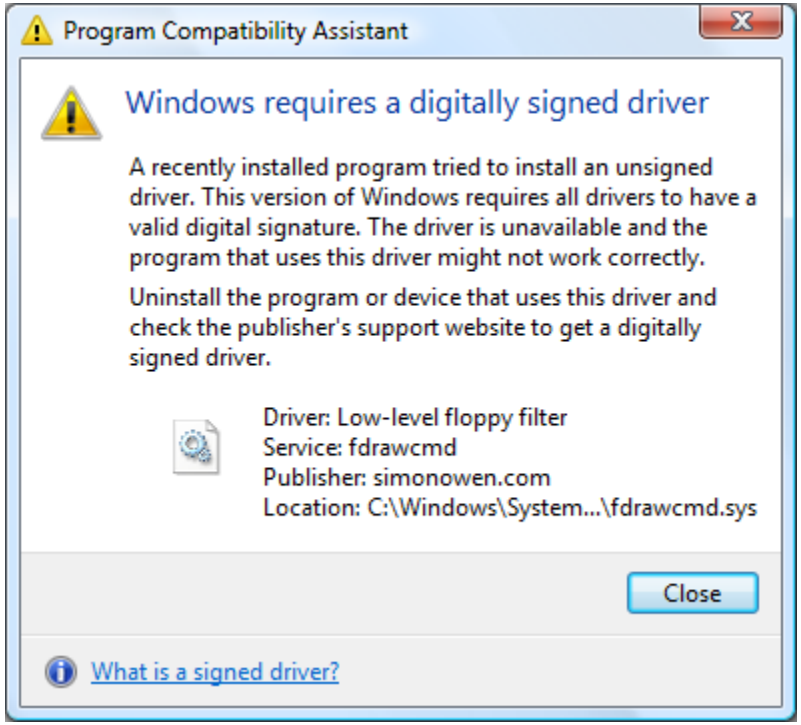
Driver Name	Loaded from (could be only name)	Address	Size	Hidden	References
ACPI.sys	ACPI.sys	0xB9F79000	188416	-	[ACPI.sys]
ACPIEC.sys	ACPIEC.sys	0xBA4C4000	12288	-	[ACPIEC.sys] [ntkrnlpa.exe]
ACPI_HAL	ACPI_HAL	0x806E4000	134400	-	[hal.dll] [ntkrnlpa.exe]
Afc.sys	C:\WINDOWS\system32\drivers\Afc.sys	0xBA448000	32768	-	[Afc.sys]
afd.sys	C:\WINDOWS\System32\drivers\afd.sys	0xA6FF7000	139264	-	[afd.sys]
atapi.sys	atapi.sys	0xB9CDC000	98304	-	[atapi.sys] [ntkrnlpa.exe]
ATMFD.DLL	C:\WINDOWS\System32\ATMFD.DLL	0xBFFA0000	286720	-	
audstub.sys	C:\WINDOWS\system32\DRIVERS\audstub.sys	0xBA797000	4096	-	[audstub.sys] [ntkrnlpa.exe]
BATTC.SYS	C:\WINDOWS\system32\DRIVERS\BATTC.SYS	0xBA4C0000	16384	-	
Beep.SYS	C:\WINDOWS\System32\Drivers\Beep.SYS	0xBA63A000	8192	-	[Beep.SYS] [ntkrnlpa.exe]
BOOTVID.dll	C:\WINDOWS\system32\BOOTVID.dll	0xBA4B8000	12288	-	
Cdfs.SYS	C:\WINDOWS\System32\Drivers\Cdfs.SYS	0xBA198000	65536	-	[Cdfs.SYS] [ntkrnlpa.exe]
cdrom.sys	C:\WINDOWS\system32\DRIVERS\cdrom.sys	0xBA1C8000	65536	-	[cdrom.sys] [CLASSPNP.SY...
CLASSPNP.SYS	C:\WINDOWS\system32\DRIVERS\CLASSPNP.SYS	0xBA0E8000	53248	-	
CmBatt.sys	C:\WINDOWS\system32\DRIVERS\CmBatt.sys	0xB9A4B000	16384	-	[CmBatt.sys] [ntkrnlpa.exe]
cmdguard.sys	C:\WINDOWS\System32\DRIVERS\cmdguard.sys	0xA7622000	126976	-	[cmdguard.sys] [ntkrnlpa.e...
cmdhlp.sys	C:\WINDOWS\System32\DRIVERS\cmdhlp.sys	0xBA3A8000	20480	-	[cmdhlp.sys]
compbatt.sys	compbatt.sys	0xBA4BC000	12288	-	[compbatt.sys] [ntkrnlpa.e...
cwgsd.sys	cwgsd.sys	0xB9D31000	1228800	-	[cwgsd.sys] [ntkrnlpa.exe]
disk.sys	disk.sys	0xBA0D8000	36864	-	[disk.sys] [CLASSPNP.SYS] ...
drmk.sys	C:\WINDOWS\system32\drivers\drmk.sys	0xBA318000	61440	-	
dump_atapi.sys	C:\WINDOWS\System32\Drivers\dump_atapi.sys	0xA6D9F000	98304	-	
dump_WMILIB.SYS	C:\WINDOWS\System32\Drivers\dump_WMILIB....	0xBA65C000	8192	-	
Dxapi.sys	C:\WINDOWS\System32\drivers\Dxapi.sys	0xA7059000	12288	-	
dxg.sys	C:\WINDOWS\System32\drivers\dxg.sys	0xBF000000	73728	-	
dxgthk.sys	C:\WINDOWS\System32\drivers\dxgthk.sys	0xBA7F4000	4096	-	

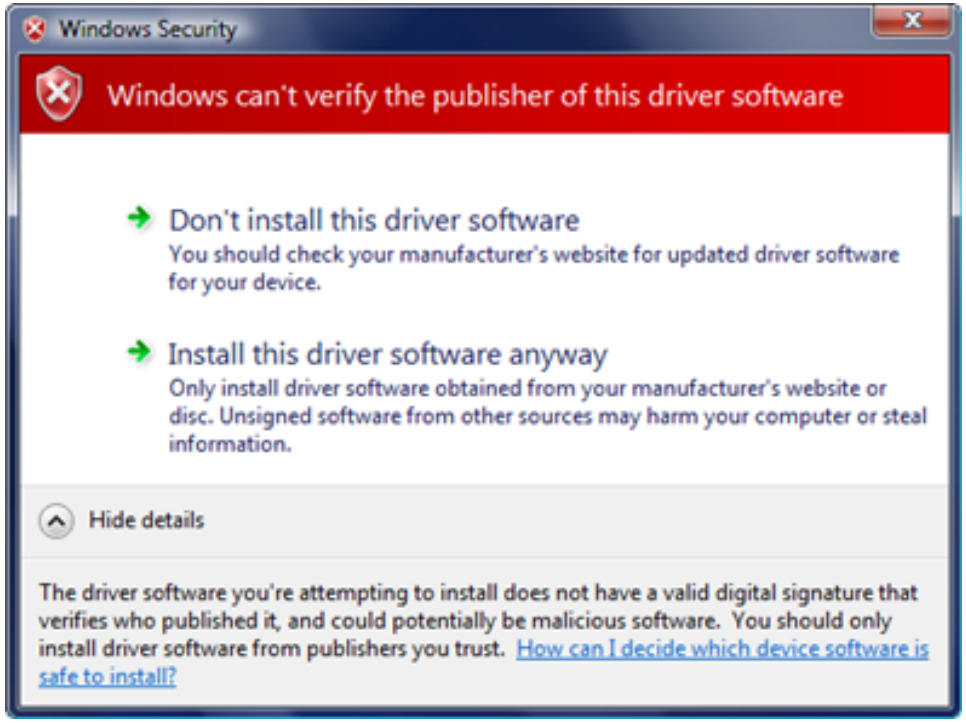
OVLADAČE VE WINDOWS XP



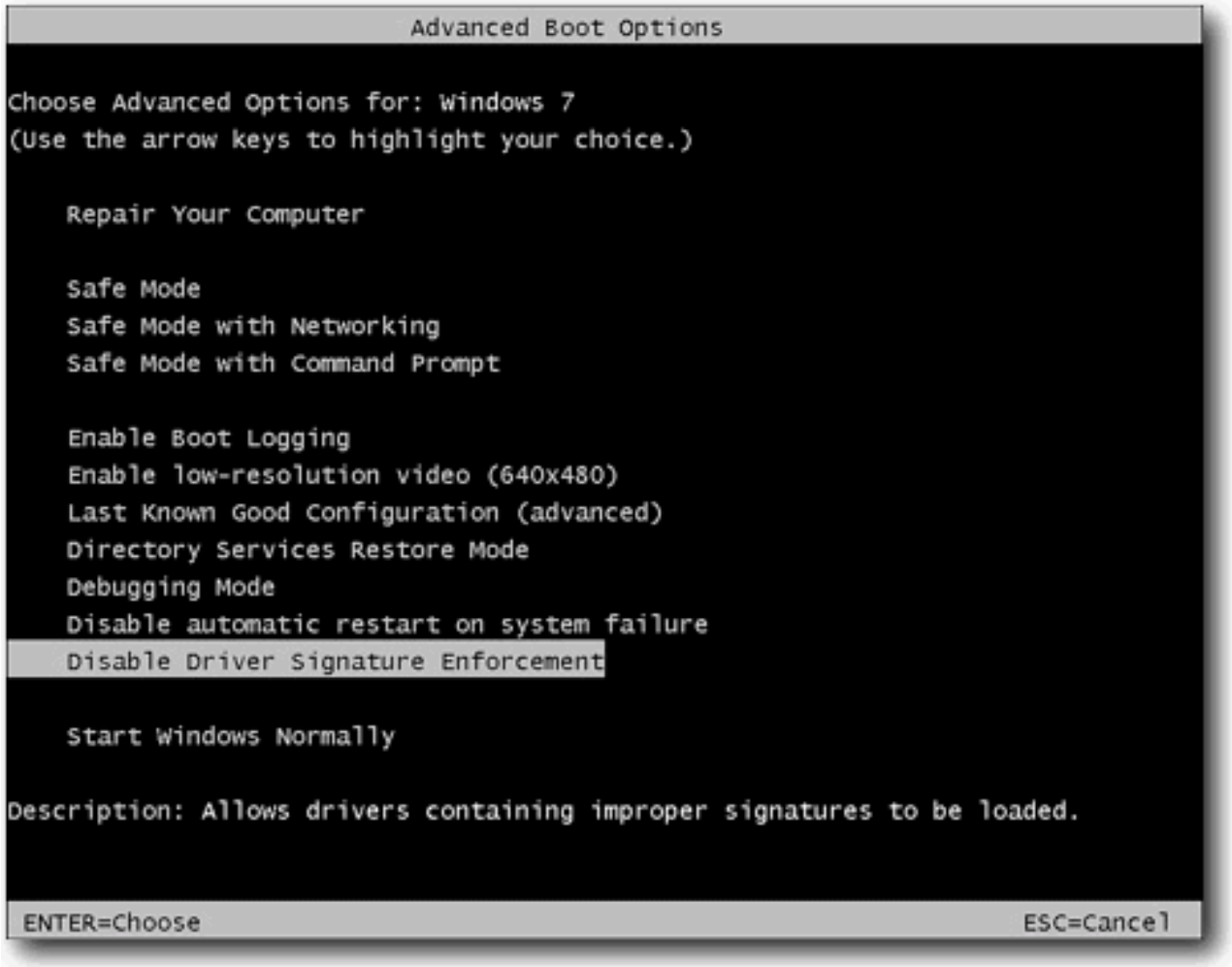


OVLADAČE A NOVĚJŠÍ WINDOWS





PODPISY VYŽADOVÁNY U 64BITOVÝCH SYSTÉMŮ



PODPISY OVLADAČŮ

- U Windows nejde při podepisování o čistotu jádra či licence
- Jde především o
  - Spolehlivost systému – tj. kvalitu kódu běžícího v privilegovaném režimu
  - A s tím související bezpečnost jádra/systému
  - DRM (!)
- User mode drivers – snaha snížit množství kódu běžící přímo v jádře

IMPLEMENTACE SYSTÉMU

- Tradičně býval OS napsaný v symbolickém strojovém jazyku (assembleru)
- OS se stále častěji píší v běžných programovacích jazycích vysoké úrovně (obvykle C/C++)
  - lze naprogramovat rychleji
  - výsledek je kompaktnější
  - OS je srozumitelnější a lze ho snadněji ladit
  - je snadněji přenositelný na jinou architekturu

SYSTEM GENERATION (SYSGEN)

- Operační systém je navržen tak, aby mohl běžet na jisté třídě architektur / sestav počítače
- OS musí být konfigurovatelný na konkrétní sestavu
- Program SYSGEN
  - Získává informace týkající se konkrétní konfigurace konkrétního hardwarového systému
- Bootování
  - Spuštění činnosti počítače zavedením jádra a předáním řízení na vstupní bod jádra pro spuštění činnosti
- Bootstrap program
  - Program uchovávaný v ROM, který je schopný nalézt jádro, zavést ho do paměti a spustit jeho provedení

BOOTOVÁNÍ IBM PC

- Řídí BIOS
  - provede se inicializace HW komponent
  - na základě uložené konfigurace zjistíme z kterého zařízení se má OS zavést
  - v případě pevného disku se spustí kód uložený v Master Boot Recordu (MBR)
    - tento kód například zjistí, která partition je aktivní a spustí boot sektor této partition. Kód uložený v boot sektoru načte soubory s jádrem OS do paměti
    - nebo např. LILO/Grub umožní interaktivně vybrat který OS bude zaveden (bootsektor které partition se má spustit?; kde je soubor s jádrem OS?)
    - tento kód může být delší než je délka MBR, musí pak být uložen v jiné oblasti disku

PŘÍKLAD: BOOTOVÁNÍ LINUXU (IBM PC)

- BIOS kontroluje HW
- Z vybraného zařízení se získá a spustí zavaděč („boot loader“)
  - Např. se z pevného disku přečte prvních 512 bajtů (tzv. MBR) a spustí se tento „kód“ (fáze 1).

HLAVNÍ ZAVÁDĚCÍ ZÁZNAM

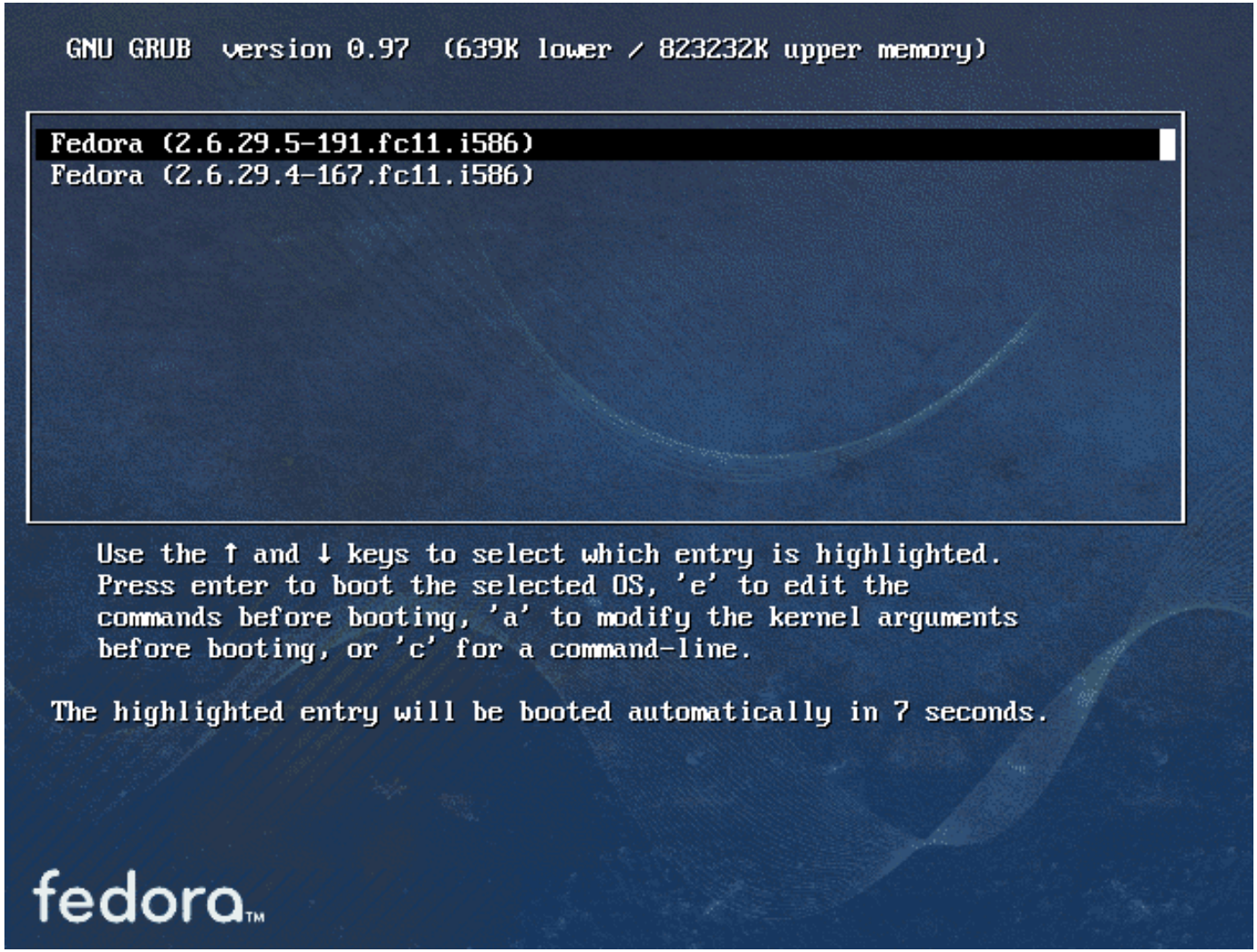
Struktura MBR

Adresa			Popis		Délka v bajtech
Hex	Oct	Dec			
0000	0000	0	Kód zavaděče		**440** (max 446)
01B8	0670	440	Volitelná délka disku		4
01BC	0674	444	Obvykle nuly; 0x0000		2
01BE	0676	446	**Tabulka primárnách oddílů** (4 položky po 16 bajtech, IBM schéma oddílů)		**64**
01FE	0776	510	55h	Signatura MBR; 0xAA55	2
01FF	0777	511	Aah		
Celková délka MBR: 446 + 64 + 2 =					**512**

PŘÍKLAD: BOOTOVÁNÍ LINUXU (IBM PC) - POKRAČOVÁNÍ

- Spuštění zavadeče
  - Ve volitelné fázi 1.5 se zavede kód pro přístup k disku (BIOS nemusí plně umět).
  - Ve fázi 2 se zavede zbytek kódu zavaděče.
- Výběr OS a parametrů
  - GRUB je schopen pracovat s ext2, ext3, ext4 souborovými systémy
  - LILO souborové systémy nezná a pracuje s přímými adresami souborů na disku

VÝBĚR OS - GRUB



BOOTOVANÍ LINUXU (IBM PC)

- Spuštění jádra
  - Z disku je přečten obraz jádra
  - Tento obraz se dekomprimuje
  - Provede se základní inicializace (např. tabulka stránek paměti)
  - Spustí se jádro (volá se funkce start_kernel())
- Jádro
  - Nastaví systém (ovladače přerušení, inicializace zařízení a ovladačů zařízení) a spustí plánovač.
  - Vytvoří proces Init (číslo procesu 1) – např. /sbin/init

GPT

- MBR je limitován velikostí disku 2TB
  - 32 bitů pro počet sektorů x 512 bitů na sektor
- Intel vyvinul nový standard GUID partition table
  - GPT
  - Redundance: tabulka na začátku a konci disku
  - Chránící MBR položka (oddíl EEh přes celý disk)
  - Minimálně 128 položek pro oddíly
- Podporováno na většině dnešních OS
  - GPT nepodporuje Win XP (32 + 64 bit) a Win Server 2003 (32 bit)

GUID jednotlivých oddílů

Operating system	Partition type	Globally unique identifier (GUID)[A]
(None)	Unused entry	00000000-0000-0000-0000-000000000000
	MBR partition scheme	024DEE41-33E7-11D3-9D69-0008C781F39F
	EFI System partition	C12A7328-F81F-11D2-BA4B-00A0C93EC93B
	BIOS Boot partition[B]	21686148-6449-6E6F-744E-656564454649
	Intel Fast Flash (iFFS) partition (for Intel Rapid Start technology)[20][21]	D3BFE2DE-3DAF-11DF-BA40-E3A556D89593
	Sony boot partition[F]	F4019732-066E-4E12-8273-346C5641494F
	Lenovo boot partition[F]	BFBFAFE7-A34F-448A-9A5B-6213EB736C22
Windows	Microsoft Reserved Partition (MSR)	E3C9E316-0B5C-4DB8-817D-F92DF00215AE
	Basic data partition[C]	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
	Logical Disk Manager (LDM) metadata partition	5808C8AA-7E8F-42E0-85D2-E1E90434CFB3
	Logical Disk Manager data partition	AF9B60A0-1431-4F62-BC68-3311714A69AD
	Windows Recovery Environment	DE94BBA4-06D1-4D40-A16A-BFD50179D6AC
	IBM General Parallel File System (GPFS) partition	37AFFC90-EF7D-4E96-91C3-2D7AE055B174
HP-UX	Data partition	75894C1E-3AEB-11D3-B7C1-7B03A0000000
	Service Partition	E2A1E728-32E3-11D6-A682-7B03A0000000
Linux	Linux filesystem data[C]	0FC63DAF-8483-4772-8E79-3D69D8477DE4
	RAID partition	A19D880F-05FC-4D3B-A006-743F0F84911E
	Swap partition	0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
	Logical Volume Manager (LVM) partition	E6D6D379-F507-44C2-A23C-238F2A3DF928
	`/home` partition[22]	933AC7E1-2EB4-4F13-B844-0E14E2AEF915
	`/srv` partition[22]	3B8F8425-20E0-4F3B-907F-1A25A76F98E8
	Plain dm-crypt partition[23][24]	7FFEC5C9-2D00-49B7-8941-3EA10A5586B7
	LUKS partition[23][24]	CA7D7CCB-63ED-4C53-861C-1742536059CC

Zdroj: Wikipedia (GPT)

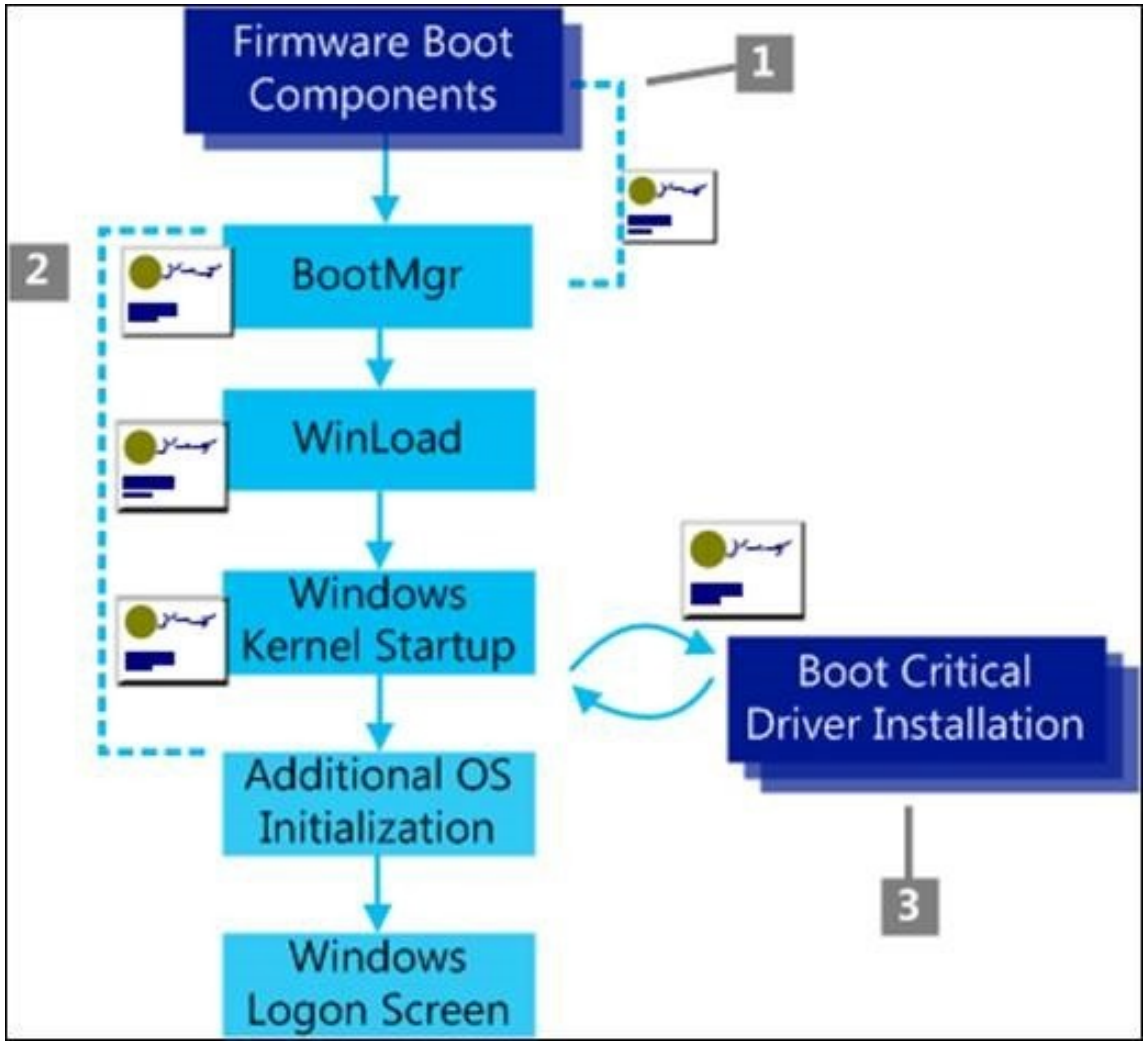
Bezpečný boot (secure boot)

- Snaha o bezpečné zavedení OS
  - Ochrana samotného zavádění komponent OS
  - Prevence rootkitů (bootkitů)
- Součást tzv. bezpečného bootu
  - Bezpečnost i dalších komponent (už. programů)

Secure Boot is a technology where the system firmware checks that the system boot loader is signed with a cryptographic key authorized by a database contained in the firmware. With adequate signature verification in the next-stage boot loader(s), kernel, and, potentially, user space, it is possible to prevent the execution of unsigned code.

Zdroj: Fedora 18. UEFI Secure Boot Guide

Bezpečný boot ve Windows 8



Zdroj: MSDN

Bezpečný boot ve Fedora Linux (18+)

- Postupné zavádění komponent
  - Shim, Grub, jádro, moduly jádra

- Shim: This is signed by the UEFI signing service. We do not have control over this key. The shim contains the Fedora Boot CA public key.
- GRUB: This is signed by the "Fedora Boot Signer" key, which chains off the Fedora Boot CA key. GRUB doesn't contain any keys, it calls into shim for its verification.
- Kernel: This is also signed by the Fedora Boot Signer. The kernel contains the public key used to sign kernel modules.
- Kernel Modules: These are signed with a private key generated during build. This key is not saved, a new key is used with each kernel build.

Zdroj: Fedora 18. UEFI Secure Boot Guide

Bezpečný boot – boj proti malware?

LINUX VENDOR Red Hat will pay Verisign to sign Fedora 18 so it can work with Microsoft's UEFI Secure Boot.

Microsoft has ordered all PC makers shipping Windows 8 PCs to have its UEFI Secure Boot feature enabled, which poses a significant hurdle for users should they want to install a different operating system. Now the Red Hat sponsored Fedora project has said it intends to pay Verisign $99 to sign its software.
Microsoft's intention with UEFI Secure Boot is to only allow trusted software to interact with PC hardware, including bootloaders and drivers. However many observers have pointed out that the company can also use this requirement to make it extremely difficult for users to install alternative operating systems.
…

While Verisign will be receiving $99 from Fedora and presumably the Ubuntu project and other Linux distributions, it is Microsoft that is the real winner, so far, since it is clear that UEFI Secure Boot will hurt not just Linux distributions but any operating systems that don't have the resources to sign everything to meet Microsoft's unilaterally imposed security regime.

Zdroj: http://www.theinquirer.net/inquirer/news/2181774/red-hat-pay-verisign-sign-fedora-microsofts-uefi-secure-boot

Výukovou pomůcku zpracovalo

Servisní středisko pro e-learning na MU

CZ.1.07/2.2.00/28.0041
Centrum interaktivních a multimediálních studijních opor pro inovaci výuky a efektivní učení

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ