KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Administrowanie sieciami lokalnymi i serwerami

Jacek Kobus

Wydział Fizyki, Astronomii i Informatyki Stosowanej UMK

http://jkob.fizyka.umk.pl/_downloads/ssk-admin[4].pdf

ostatnia aktualizacja: 10-02-2022

Program

1. Uruchamianie systemu operacyjnego (w tym GNU/Linux)
   - sprawdzanie sprzętu i ładowanie systemu operacyjnego
   - uruchamianie usług i zarządzanie usługami (poziomy pracy, systemd)
2. Metody uruchamiania systemu Linux (dysk, USB, PXE, tryb rescue)
   - konfigurowanie i instalacja GRUB/GRUB2
   - protokoły BOOTP (ARP/RARP), DHCP, TFTP, PXE
3. Wykrywanie i dynamiczna konfiguracja urządzeń (udev)
4. Konfiguracja interfejsów sieciowych i mostu (przełącznika ethernetowego)
5. Usługa nazw domenowych (DNS)
6. Bezpieczne rejestrowanie się i budowa tuneli (SSH)
7. Sieciowy system plików (NFS)

Literatura

[1] D. J. Barrett, R. E. Silverman, and R. G. Byrnes. *SSH, the Secure Shell: The Definitive Guide*. O'Reilly, Sebastopol, 2005.

[2] *Free OnLine Books*. http://www.linux.org/docs/online_books.html/.

[3] R. Love. *Linux kernel. Przewodnik programisty*. Wydawnictwo Helion, Gliwice, 2004.

[4] M. Mitchell, J. Oldham, and A. Samuel. *Advanced Linux Programming*. http://www.advancedlinuxprogramming.com/.

[5] D. Mosberger i S. Eranian. *IA-64 Linux Kernel. Design and Implementation*. Prentice-Hall PTR, Upper Saddle River, 2002.

[6] A. Silberschatz i P. B. Galvin. *Podstawy systemów operacyjnych*. Wydawnictwo Naukowo-Techniczne, wyd.5, Warszawa, 2002.

[7] A. S. Tanenbaum and H. Bos. *Systemy operacyjne*. Wydawnictwo Helion, Gliwice, 2010.

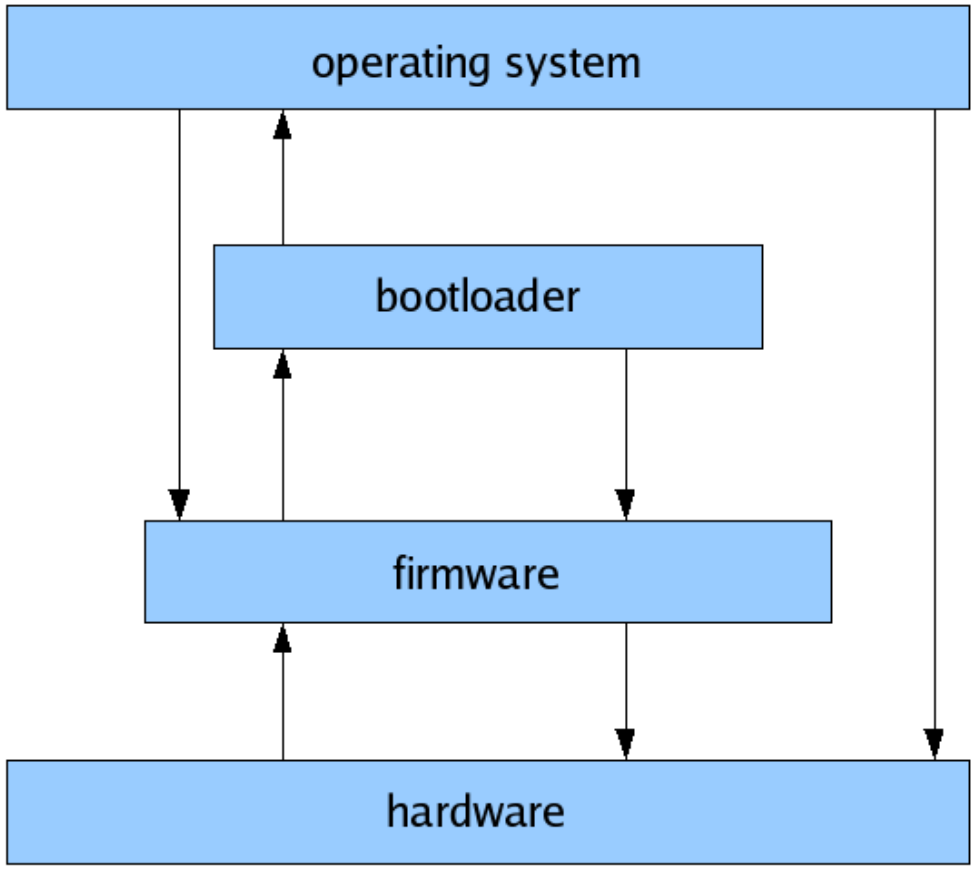
Uruchamiania systemu operacyjnego[1]

W jaki sposób ładowane jest jądro systemu operacyjnego do pamięci komputera?

Proces uruchamiania/ładowania systemu operacyjnego określa się (żargonowo) terminem bootowanie:

- *boot* ← *bootstrap* ← *bootstrap load*
- *to bootstrap – to pull oneself up by one's bootstraps, i.e improve one's situation by one's own efforts*
- *bootstrap* – ucho z tyłu cholewki buta

[1] D. Mosberger and S. Eranian, IA-64 Linux Kernel, Prentice-Hall PTR, Upper Saddle River, 2002, strony http://en.wikipedia.org/wiki/Booting i pokrewne



Boot components

Uruchamianie systemu komputerowego

1. Włączenie komputera (autotest zasilacza: *power good*)
2. Procesor pobiera zawartość komórki pamięci CS:IP F000:FFF0 (0xFFFF0) i wykonuje skok do wskazanego adresu, gdzie jest BIOS
3. POST sprawdza czy uruchomienie komputera miało charakter
   - *twardy* – pełen test sprzętu (POST *Power-On Self Test*)
   - *miękki* – pomijany jest test pamięci[2]
4. POST sprawdza BIOS, pamięć CMOS i jej baterię
5. POST inicjuje kartę grafiki i ją sprawdza
6. POST identyfikuje BIOS (wyświetlana jest wersja, producent, data)
7. POST testuje pamięć operacyjną (wyświetlana jest jej wielkość)
8. POST sprawdza i inicjuje urządzenia sprzętowe (przydział IRQ, adresów I/O, lista zainstalownych urządzeń PCI)

[2]Komórka pamięci 0000:0472 zawiera liczbę 1234h.

9. Pobranie ustawień z CMOS
10. Poszukiwanie systemu do uruchomienia (kolejność bootowania); poszukiwany jest sektor, który zawiera program ładujący (dwa ostatnie bajty to 0x55 oraz 0xAA, czyli 0xAA55)
11. Ładowanie programu ładującego do pamięci głównej i jego uruchomienie
12. Ładowanie do pamięci programu ładującego fazy drugiej, który odpowiedzialny jest za załadowanie i uruchomienie systemu operacyjnego
13. Inicjacja pracy systemu: `/etc/rc.d/rc.S` (BSD), poziomy pracy systemu (sysVinit), Upstart (Ubuntu 6.10, CentOS 6.x), systemd (Debian/Fedora/CentOS 7)

Zob. także:

- Inside the Linux boot process
- Booting Linux (ps)/Booting Linux (pdf)

Uruchamianie systemu komputerowego: faza II

Ładowanie do pamięci i przekazywanie sterowania do głównego kodu startowego zapisanego w pierwszym bloku (sektorze, C=0, H=0, S=1) dysku, tzw. MBR-e (*Master Boot Record*) przy pomocy funkcji przerwania INT 13h (19):

```
INT 13h: Bootstrap Loader
Reads track 0, sector 1 into 0000:7C00, jumps to this address
```

Struktura MBR-u: 446 B (lub 440 dla Windows NT) – program rozruchowy (główny kod startowy), 64 B – tabela partycji (aktywna partycja), 2 B – *magic number* (AA55h).

Kod startowy czyta tablicę partycji, odszukuje pierwszy sektor aktywnej partycji i ładuje kopię tego sektora do pamięci.

Dawne problemy:[3]

- *1024 limit* (504 MiB); rozw.: *enhanced* BIOS wspierający translację
- *8 GB limit*; rozw.: funkcja przerwania INT 13h zastąpiona przez funkcję *extended* INT 13h

[3] https://www.win.tue.nl/ aeb/linux/Large-Disk-4.html

1024 limit

Standard	Bits For Cylinder Number	Bits for Head Number	Bits for Sector Number	Total Bits for Geometry
IDE/ATA	16	**4**	8	28
BIOS Int 13h	**10**	8	**6**	24
Combination (Smaller of Each)	**10**	**4**	**6**	**20**

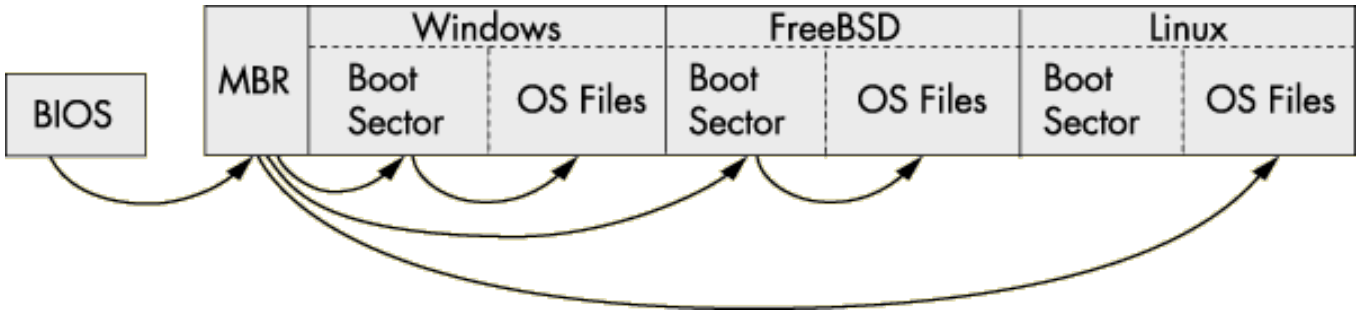
Standard	Maximum Cylinders	Maximum Heads	Maximum Sectors	Maximum Capacity
IDE/ATA	65,536	**16**	256	128 GiB
BIOS Int 13h	**1,024**	256	**63**	7.88 GiB
Combination (Smaller of Each)	**1,024**	**16**	**63**	**504 MiB**

Wybór systemu operacyjnego

- specjalizowany MBR
- specjalizowany sektor rozruchowy (*volume boot sector*)
- rozruch łańcuchowy

  Każdy sektor rozruchowy (*bootujący*) jest odpowiedzialny za ładowanie właściwego SO, który po uruchomieniu pozwala na kontynuację bootowania z przekazaniem sterowania do sektora rozruchowego innej partycji.

Fazy uruchamiania systemu operacyjnego[4]



[4]Roderick Smith *Multibooting with GRUB*, Linux Magazine

Programy ładujące fazy pierwszej

- BIOS (*Basic Input/Output System*) – dla urządzeń x86, x86_64
- Coreboot (project wspierany przez FSF, wcześniej LinuxBIOS)[5]
- Das U-Boot (*Universal Bootloader*) – dla urządzeń z osadzonym systemem operacyjnym
- EFI (*Extensible Firmware Interface*), UEFI (*Unified EFI*) – specyfikacja opracowana przez firmę Intel
- Etherboot → gPXE → iPXE – sieciowe programy ładujące (*Preboot Execution Environment*)
- OpenBIOS, OpenBoot – realizacja standardu Open Firmware (IEEE 1275-1994)
- SLOF (*Slimline*) – Open Firmware wytwarzany przez IBM, wspiera architekturę PowerPC

[5] https://puri.sm/: telefony, notebooki, klucze zabezpieczające marki Librem firmy Purism (od 2017)
https://trmm.net/Heads: *Heads is an open source custom firmware and OS configuration for laptops*

Programy ładujące fazy drugiej

- loadlin – program ładujący system Linux działający w systemie DOS/Windows
- LILO (*LInux LOder* – (były) domyślny program ładujący dla większości dystrybucji (następca programu loadlin)
- SILO (*SPARC Improved bootLOader*) – program ładujący system Linux (także Solaris) dla architektury SPARC
- GNU GRUB (GRUB Legacy), GRUB/GRUB2 – domyślny program ładujący system Linux
- NTLDR – program ładujący dla systemów Windows NT (do Windows Server 2003)
- BOOTMGR – program ładujący dla systemów: Windows Vista, Windows Server 2008, Windows 7/8/10

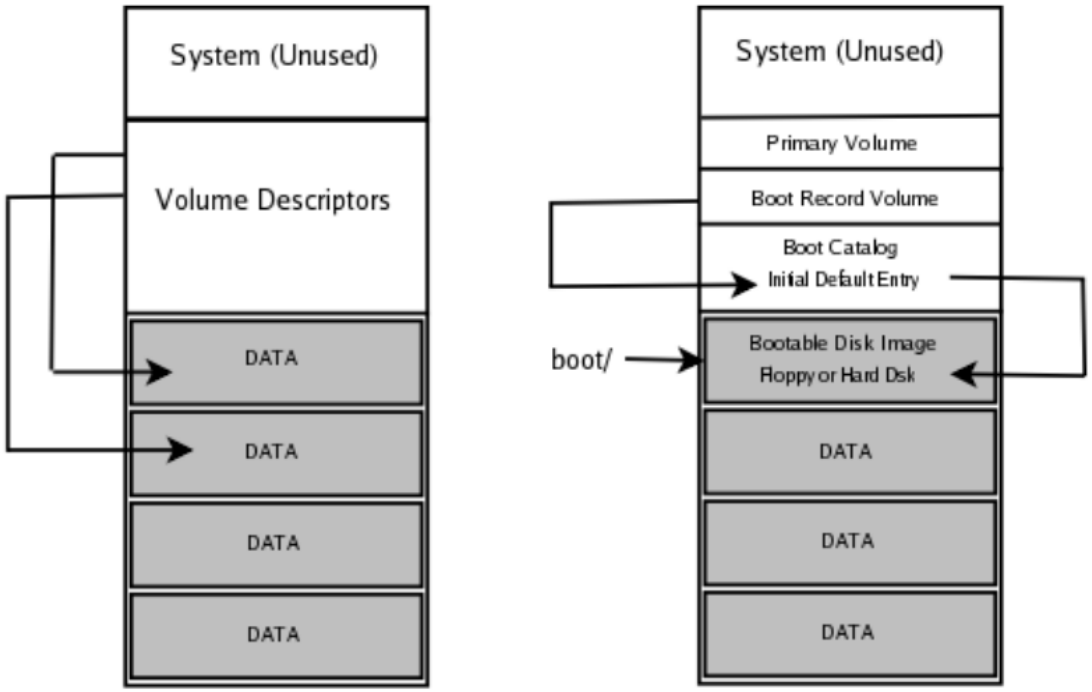
Programy ładujące *LINUX fazy drugiej[6]

- SYSLINUX – program ładujący SO Linux z partycji FAT systemów MS-DOS/Windows
- ISOLINUX – program ładujący SO Linux z partycji ISO 9660/El Torito (z CD-ROM-u)

  Standard El Torito pozwala na bootowanie w dwóch trybach: z i bez emulacji dyskietki. W trybie z emulacją dyskietki jej obraz jest pobierany z CD-ROM-u, a następnie udostępniany systemowi jako wirtualna stacja dyskietek (obraz jest faktycznie systemem plików FAT, więc do ładowania SO jest używany SYSLINUX). W trybie bez emulacji informacja potrzebna do bootowania jest zgromadzona bezpośrednio na CD, a do jej wykorzystania potrzebny jest program ISOLINUX.

[6] http://www.syslinux.org/wiki/index.php/SYSLINUX

Uruchamianie systemu komputerowego z CD/DVD[7]



[7] *Advanced Linux System Administration I, Lab work for LPI 201*, LinuxIT, http://www.nongnu.org/lpi-manuals/manual/pdf/GNU-FDL-OO-LPI-201-0.1.pdf

Uruchamianie systemu komputerowego

- EXTLINUX – program ładujący SO Linux z partycji ext2/3/4
- PXELINUX – program ładujący SO Linux z serwera w sieci

  Do załadowania programu niezbędna jest karta sieciowa wspierającą PXE (*Pre-eXecution Environment*) lub odpowiedni program czytany z nośnika (dyskietka/klucz USB) (http://rom-o-matic.net/)
- MEMDISK – program ładujący (*legacy*) SO via PXE; także kiedy BIOS systemu komputerowego nie wspiera obrazów ISOLINUX-a.

  MEMDISK emuluje dysk używając wysokiej pamięci, umieszczając sterownik dysku w niskiej pamięci i zmieniając obsługę przerwań: INT 13h (sterownik dysku) i INT 15h (sprawdzanie pamięci); zob. http://etherboot.org/wiki/bootingmemdisk

BIOS

- BIOS to zbiór funkcji tworzących interfejs pomiędzy systemem operacyjnym (SO) i sprzętem; BIOS był pierwotnie wykorzystywany do uruchamiania komputerów klasy PC i zapewnienia komunikacji między SO i urządzeniami wejścia/wyjścia (I/O)

  Wcześniej BIOS był używany przez system operacyjny CP/M dla mikrokomputerów 8-bitowych.

- BIOS był modernizowany wraz z rozwojem rynku komputerów zgodnych z PC
- BIOS jest specyficzny dla architektury procesorów x86, gdyż bazuje na 16-bitowym rzeczywistym (*real mode*) trybie dostępu do pamięci (adresowanie ograniczone do 1 MB)
- szeregowe inicjowanie urządzeń, uruchamianie systemu operacyjnego tylko z dysków mniejszych, niż 2.2 TB.

Coreboot[8]

- projekt FSF; zastąpienie BIOS-u przez otwarte oprogramowanie firmowe pozwalające na ładowanie systemów 32- i 64-bitowych
- brak wsparcia dla wywołań BIOS (duże ograniczenie na liczbę możliwych zadań); SeaBIOS (używany przez qemu) dostarcza tych wywołań (nowoczesne systemy operacyjne z nich nie korzystają)
- wersja x86 wykonuje się w trybie 32-bitowym po wykonaniu 10 instrukcji

[8] http://www.coreboot.org/users.html

UEFI (Unified Extensible Firmware Interface)

- specyfikacja definiująca niezależny od architektury procesora interfejs pomiędzy SO i sprzętem
- stanowi (od około 2005) rozszerzenie specyfikacji EFI wprowadzonej przez firmę Intel (połowa lat 1990) i rozwijanej przez Unified EFI Forum
- wspiera szyfrowanie, uwierzytelnianie sieciowe, *User Interface Architecture (Human Interface Infrastructure)*
- UEFI wspiera bezpieczne uruchamianie systemu operacyjnego (*Secure Boot*)
- najnowsza wersja: 2.8 (3/2019)

UEFI – zalety

- uruchamianie systemu z dużych dysków:
  od 2.2 TB ($2^{32} \times 512 = 2$ TiB) do 9.4 ZB ($2^{73} = 8$ ZiB)
- obsługa pamięci dla systemów x86-32 oraz x86-64
- szybsze uruchamianie systemu (?)
- architektura niezależna od CPU
- moduły obsługi urządzeń (*drivers*) niezależne od CPU
- elastyczne środowisko pre-OS (wsparcie sieciowe)
- architektura modułowa
- *secure boot*

UEFI: *boot sector* $\longrightarrow$ *boot manager*

UEFI

- tablice – UEFI dostarcza tablic z danymi dotyczącymi platformy sprzętowej oraz usług, które są dostępne dla programu ładującego i SO
- *boot services* – wsparcie dla tekstowych i graficznych konsol na różnych urządzeniach, obsługa magistrali, urządzeń blokowych i plików
- *runtime services* – dostęp do zmiennych, czasu, pamięci wirtualnej
- protokoły – wszystkie moduły obsługi urządzeń (*drivers*) muszą świadczyć usługi via protokoły, czyli specjalne zbiory programowych interfejsów wykorzystywanych do komunikacji między binarnymi modułami

UEFI

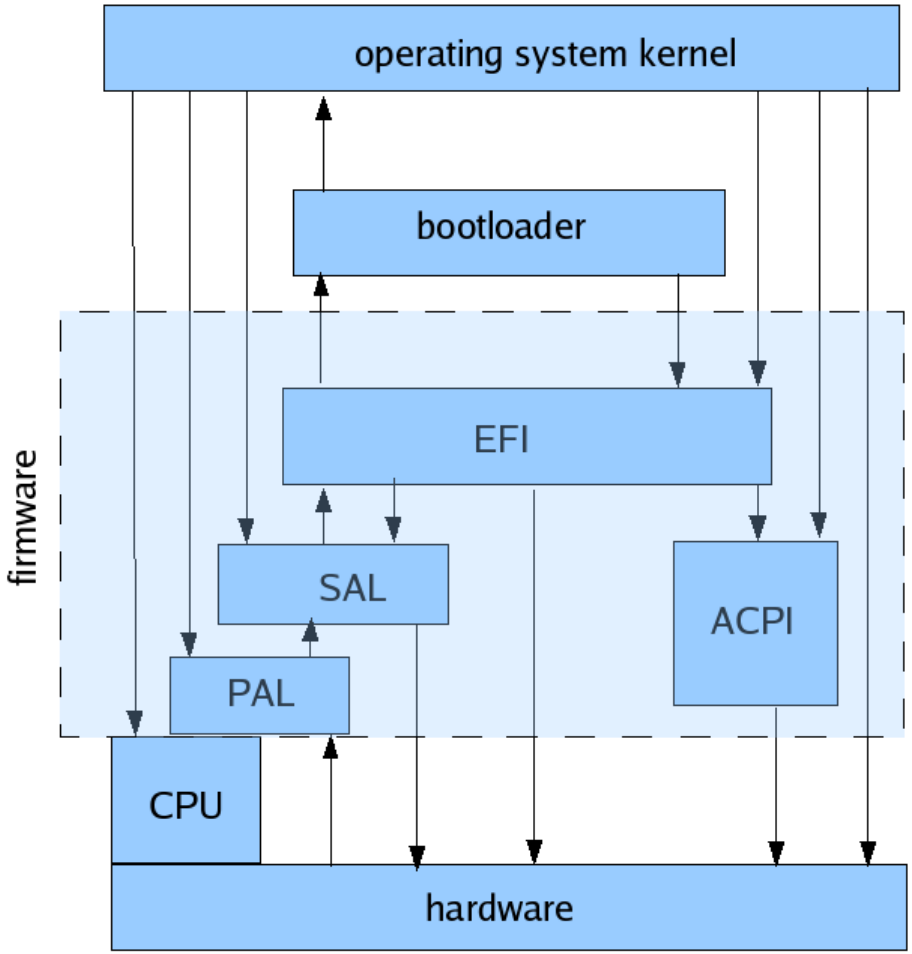
- moduły obsługi urządzeń – oprócz modułów zależnych od architektury specyfikacja EFI określa środowisko programów obsługi urządzeń niezależnych od architektury
- *boot manager* – menadżer odpowiedzialny za wybór i uruchamienie SO
- dyski – wsparcie dla tablic partycji MSDOS oraz GPT (*GUID Partition Table*)
- nshell – powłoka EFI
- rozszerzenia – rozszerzenia do EFI mogą być załadowane z dowolnej pamięci nieulotnej dostępnej w systemie
- *BIOS compatibility mode* – imitowanie BIOS-u i uruchamianie via MBR (CMS *Compatibility Support Module*)

EFI – programy ładujące

Umieszczane są we właściwych sobie podkatalogach katalogu EFI na systemowej partycji EFI (ESP, *EFI System Partition*), która jest pierwszą partycją na dysku bootującym (VFAT). Odmiany:

- dla systemów Windows
- dla OS X
- ELILO (od roku 2000), EFILINUX dla systemów Linux
- GRUB Legacy (zmodyfikowany)
- GRUB2 (w zależności od kompilacji) wspiera systemy BIOS i EFI
- refind-efi, gummieboot – nie programy ładujące, ale zarządzające programami ładującymi
- EFISTUB Kernel – jądro systemu jako swój własny progam ładujący (od wersji $\geqslant$ 3.3); `CONFIG_EFI_STUB=y`[9]

[9] https://wiki.archlinux.org/index.php/UEFI_Bootloaders; zob. /boot/config-.x86_64



IA-64 firmware components

UEFI – wady

- UEFI to niezależny system operacyjny (podobnej wielkości co Linux)
- działa także po uruchomieniu SO (*Ring -2 Hypervisor*)
- idealne miejsce do ukrycia wrogiego oprogramowania

Linus Torvalds on EFI (July 26, 2006):[10]

> [...] this other Intel brain-damage (the first one being ACPI)
>
> [...] the original EFI code in the kernel basically duplicates all the BIOS interfaces
>
> [...] don't get me wrong - the problem with EFI is that it actually superficially looks much better than the BIOS, but in practice it ends up being one of those things where it has few real advantages, and often just a lot of extra complexity because of the 'new and improved' interfaces that were largely defined by a committee.
>
> [...] so EFI has this cool shell, a loadable driver framework, and other nice features. Where 'nice' obviously means 'much more complex than the simple things they designed in the late seventies back when people were stupid and just wanted things to work'. Of course, it's somewhat questionable whether people have actually gotten smarter or stupider in the last 30 years. It's not enough time for evolution to have increased our brain capacity, but it certainly is enough time for most people to no longer understand how hardware works any more.
>
> [...] not that I'd ever claim that the BIOS is wonderful either, but at least everybody knows that the BIOS is just a bootloader, and doesn't try to make it anything else.

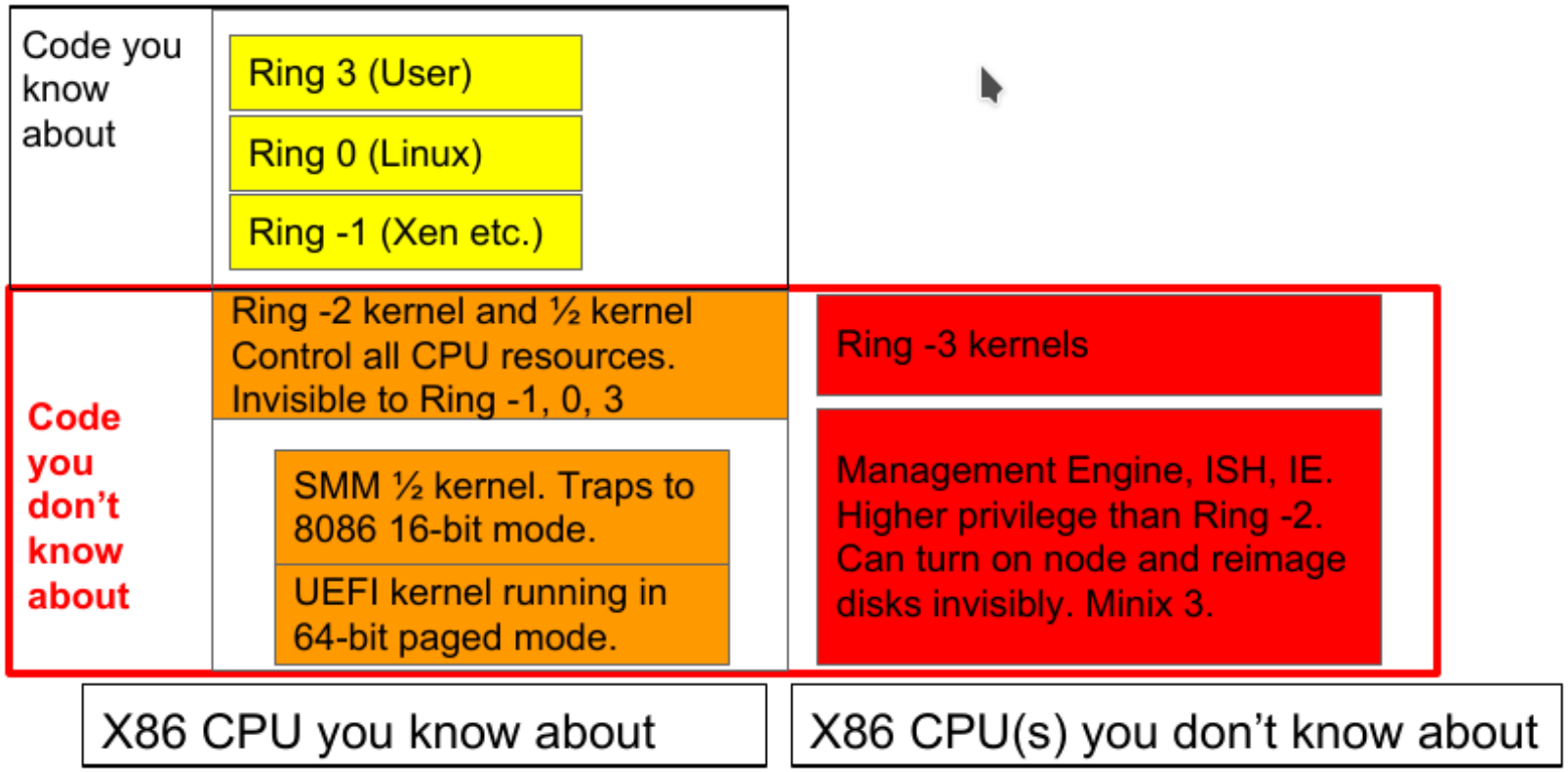
[10] http://kerneltrap.org/node/6884

Why replace UEFI with coreboot?[11]

- While Intel's edk2 tree that is the base of UEFI firmware is open source, the firmware that vendors install on their machines is proprietary and closed source. Updates for bugs fixes or security vulnerabilities are at the vendor's convienence; user specific enhancements are likely not possible; and the code is not auditable.
- UEFI is much more complex than the BIOS that it replaced. It consists of millions of lines of code and is an entire operating system, with network device drivers, graphics, USB, TCP, https, etc, etc, etc. All of these features represents increased śurface area"for attacks, as well as unnecessary complexity in the boot process.
- coreboot is open source and focuses on just the code necessary to bring the system up from reset. This minimal code base has a much smaller surface area and is possible to audit.

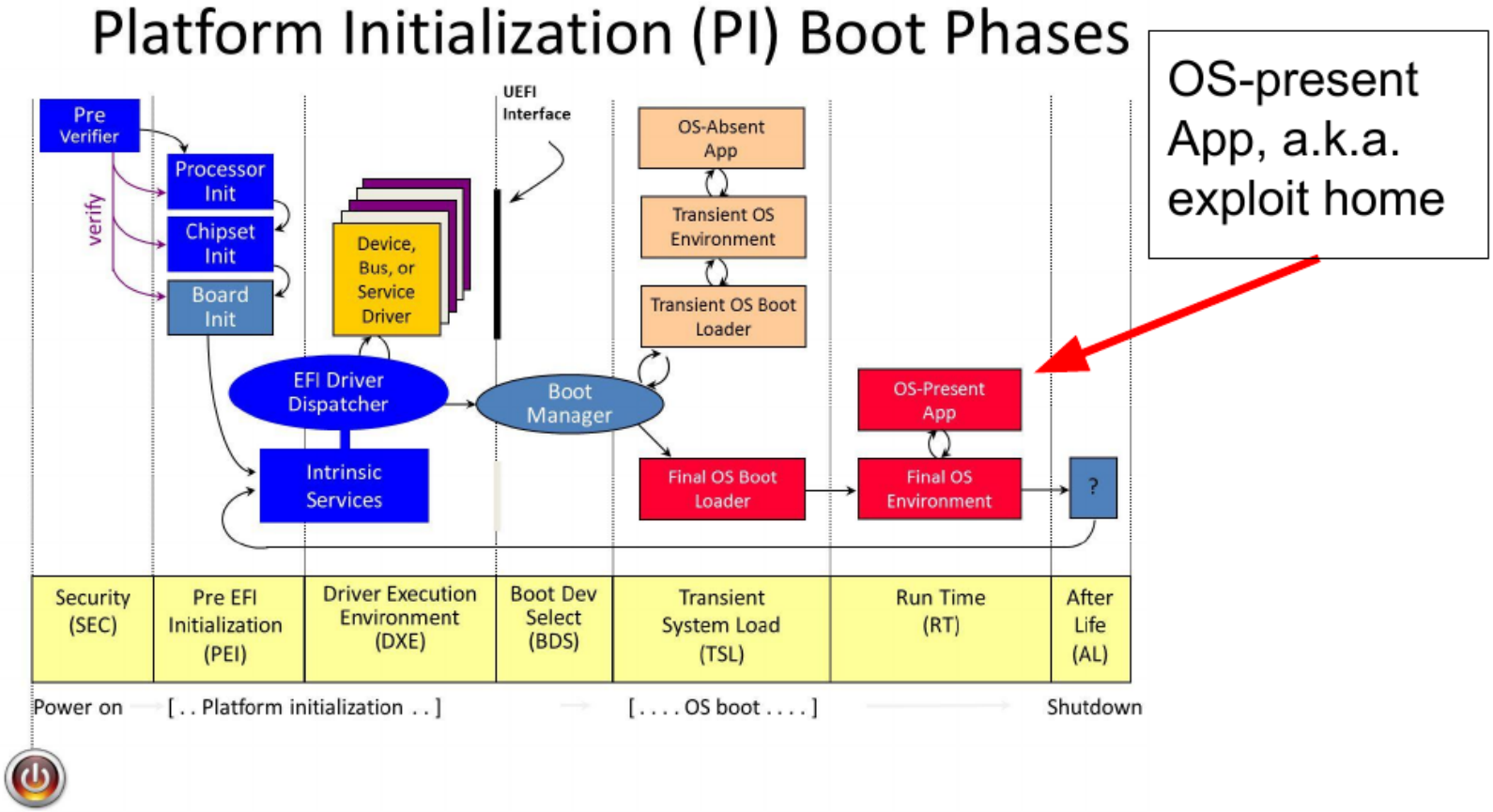
[11] https://trmm.net/Heads_FAQ

Architektura współczesnego procesora[12]



[12] https://schd.ws/hosted_files/osseu17/84/Replace%20UEFI%20with%20Linux.pdf

Etapy uruchamiania systemu wg UEFI



LinuxBoot i u-root[13]

Google zaproponował zastąpienie UEFI przez NERF, *Non-Extensible Reduced Firmware*, tj. uproszczone oprogramowanie firmowe EFI.

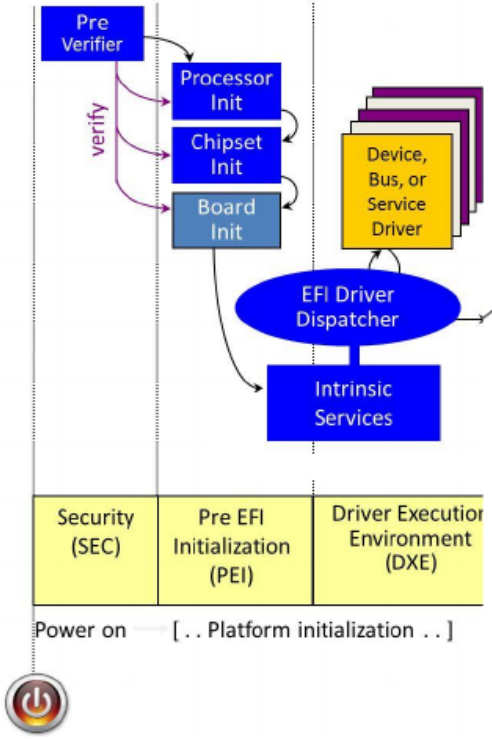
Na NERF składają się

- LinuxBoot (*LinuxBoot Project*) – oprogramowanie oparte o jądro Linuksa, które zastępuje fazy DXE oraz BDS procesu uruchomieniowego UEFI
- u-root – initramfs z narzędziami napisanymi w języku Go

UEFI może zostać zupełnie wyeliminowane, jeśli dwie pierwsze fazy procesu uruchomieniowego można zastąpić przez Coreboot.

[13] https://lwn.net/Articles/738649/
https://lwn.net/Articles/748586/
https://github.com/u-root/u-root
https://osseu17.sched.com/event/ByYt/replace-your-exploit-ridden-firmware-with-linux-ronald-minnich-google

Etapy uruchamiania systemu wg NERF

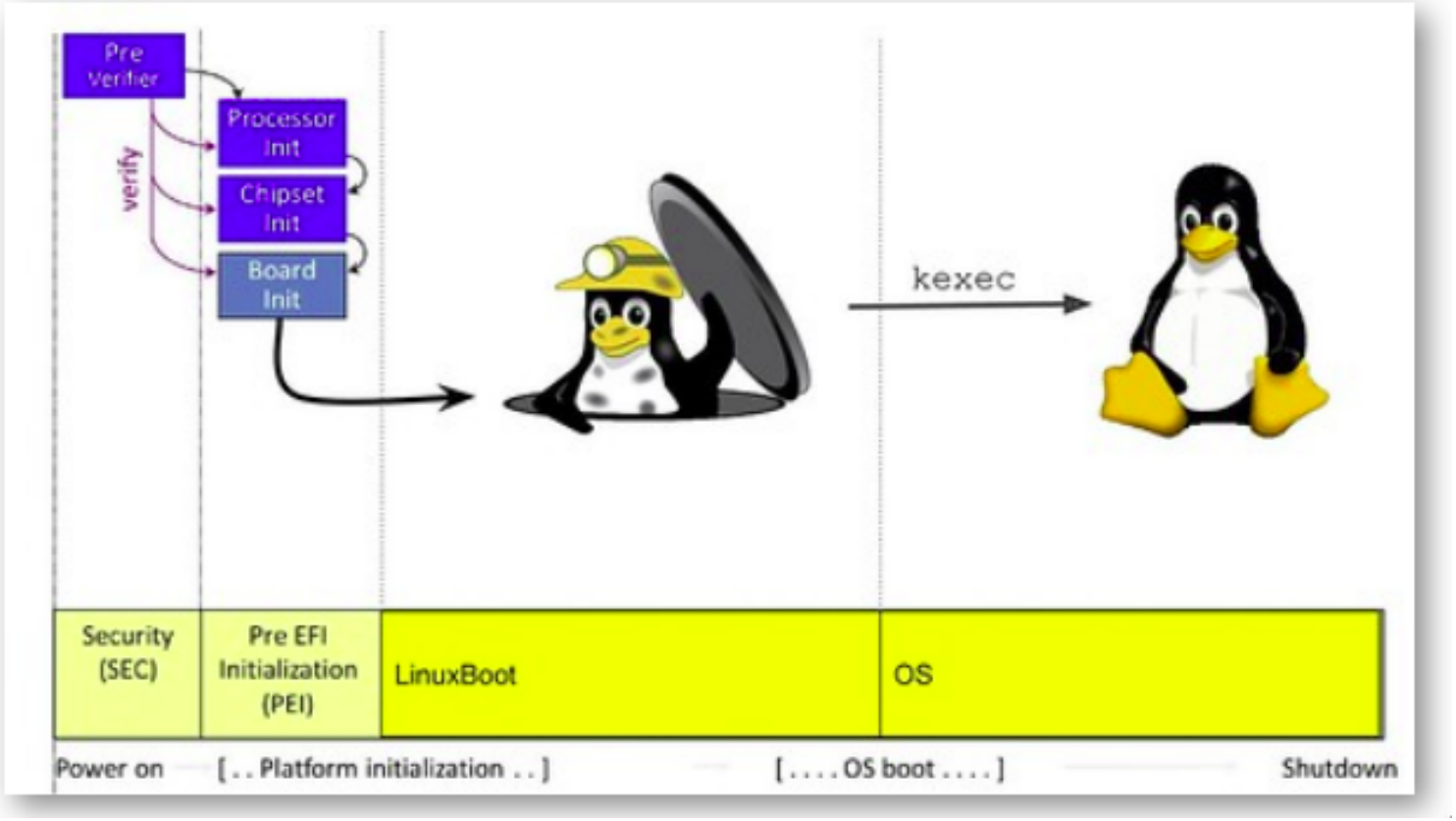


“Boot Manager” -> Linux kernel

Go-based userland (u-root.tk)

We allow DXE dispatcher if ACPI and some device initialization require it. We remove most DXEs. We kexec next kernel.

Etapy uruchamiania systemu wg NERF[14]



[14] https://trmm.net/LinuxBoot_34c3

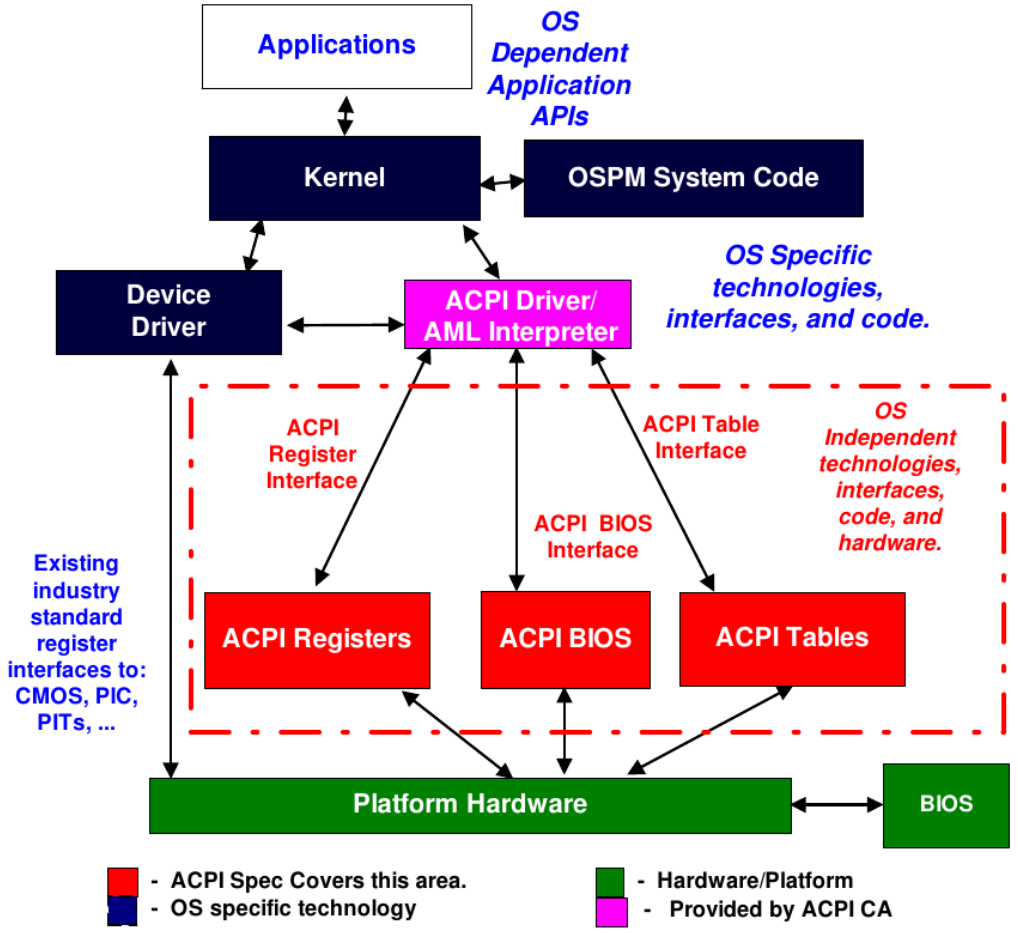
ACPI (*Advanced Control and Power Interface*)

Standard opisuje (zależne od producenta) cechy sprzętu w ogólny sposób i pozwala nim sterować z poziomu systemu operacyjnego.
Standard definiuje

- zestaw rejestrów (realizowanych sprzętowo)
- interfejs do BIOS-u, który określa tablice konfiguracyjne, metody sterowania, sposób konfigurowania płyty głównej i numerowania obsługiwanych przez nią urządzeń
- stany zasilania systemu i urządzeń
- model cieplny
- język programowania AML (A(CPI) Machine Language), przestrzeń nazw (*filesystem-like namespace*)
- zob.: `/proc/acpi` oraz `/sys/firmware/acpi`

Działanie ACPI wymaga, aby BIOS był wyposażony w oprogramowanie ACPI, a system operacyjny był zgodny z ACPI (*ACPI-compatible*).

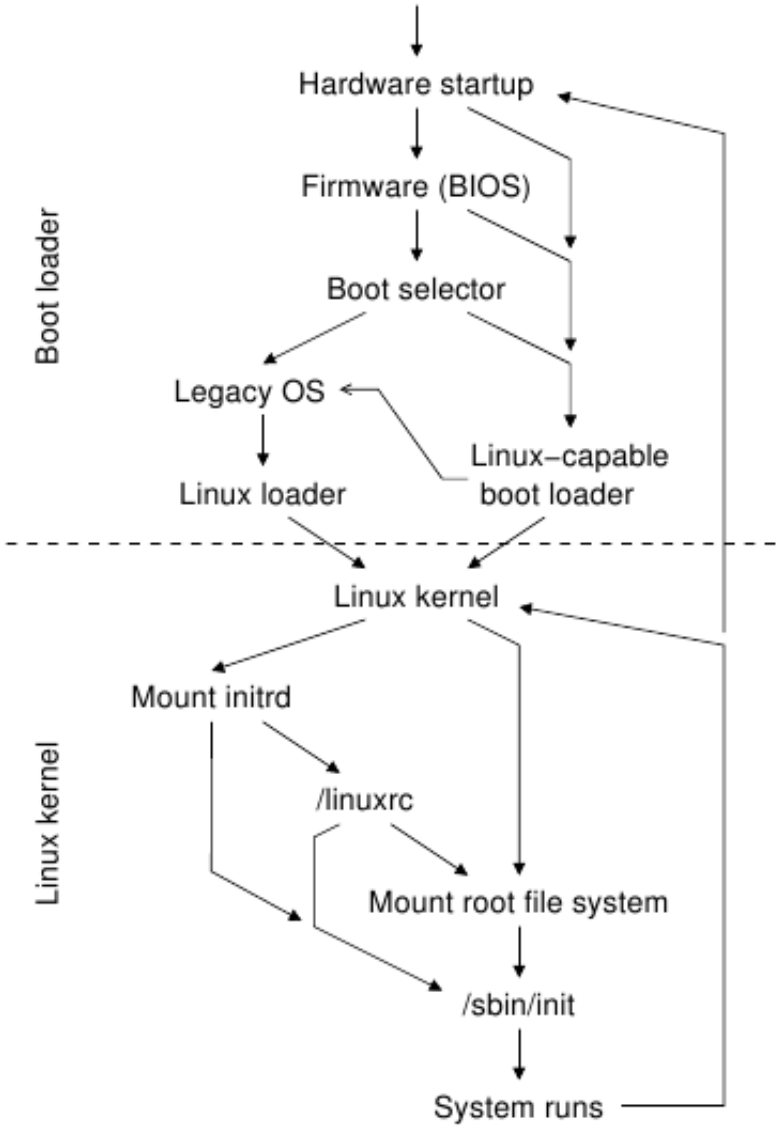
ACPI – schemat[15]



[15] http://www.acpi.info/presentations/ACPI_Overview.pdf

ACPI – zalety

- włączanie/wyłączanie urządzeń z poziomu użytkownika (`powertop`)
- obniżenie poziomu zużycia energii przez system, kiedy bateria osiąga określony stan rozładowania
- system operacyjny może regulować częstotliwość zegara procesora
- system operacyjny może regulować pobór mocy przez płytę główną i urządzenia uaktywniając je w razie potrzeby
- możliwość realizowania stanu *stand-by* z zachowaniem podtrzymywania zasilania wybranych urządzeń (np. modemu)
- możliwość realizowania stanu *hibernation* i szybkiego powrotu z (głębokiego) uśpienia
- obsługa funkcji PnP (urządzenia podlegają zarządzaniu po podłączeniu do systemu)


Hardware startup
Firmware (BIOS)
Boot selector
Boot loader
Legacy OS
Linux–capable
boot loader
Linux loader
Linux kernel
Mount initrd
/linuxrc
Linux kernel
Mount root file system
/sbin/init
System runs

Uruchamianie systemu operacyjnego

`start()` (`arch/x86/boot/head.S`)

- określa dostępną pamięć RAM
- inicjuje klawiaturę (opóźnienia i częstotliwość powtarzania)
- inicjuje kartę graficzną
- inicjuje kontroler dysku i określa parametry dysku twardego
- inicjuje mysz i sprawdza obsługę ACPI przez BIOS
- programuje układ kontrolera przerwań APIC (*Advanced Programmable Interrupt Controller*)
- wykonuje skok do funkcji `startup32()` (`arch/x86/boot/compressed/head_32.S`), która dokonuje dekompresji obrazu jądra

Uruchamianie systemu operacyjnego

`startup32()` (`arch/i386/kernel/head.S`)

- inicjuje rejestry segmentacji
- wywołuje funkcję `setup_idt()`
- umieszcza parametry systemu uzyskane z BIOS-u
- identyfikuje model procesora
- wykonuje skok do funkcji `start_kernel` (np. *Linux version 2.6.10...*)

`start_kernel()` (`init/main.c`)

- inicjuje alokator pamięci `bootmem allocator` (przydział ciągłych obszarów pamięci)
- parsuje parametry wywołania jądra
- inicjuje obsługę wyjątków `trap_init()`
- inicjuje obsługę przerwań `init_IRQ()`
- inicjuje tablicę stron `paging_init()`
- inicjuje deskryptory stron `mem_init()`
- inicjuje pamięć buforów `kmem_cache_init()` i `kmem_cache_size_init()`
- inicjuje czas i datę systemową `time_init()`
- otwiera konsolę
- budzi pozostałe procesory, inicjuje `cpu_idle()`
- tworzy wątek jądra dla procesu 1 (`kernel_thread()` w `arch/x86/kernel/process.c`), który tworzy inne wątki i wykonuje program `/sbin/init` ew. `/usr/lib/systemd/systemd`
- montuje initrd/initramfs, ładuje moduły obsługi urządzeń i wykonuje `pivot_root()`
- wywołuje `/sbin/init` ew. `/usr/lib/systemd/systemd`, inicjuje pozostałe podsystemy jądra, zwolnia pamięć używaną przy uruchamianiu jądra
- pojawia się znak zachęty (tekstowy lub graficzny)

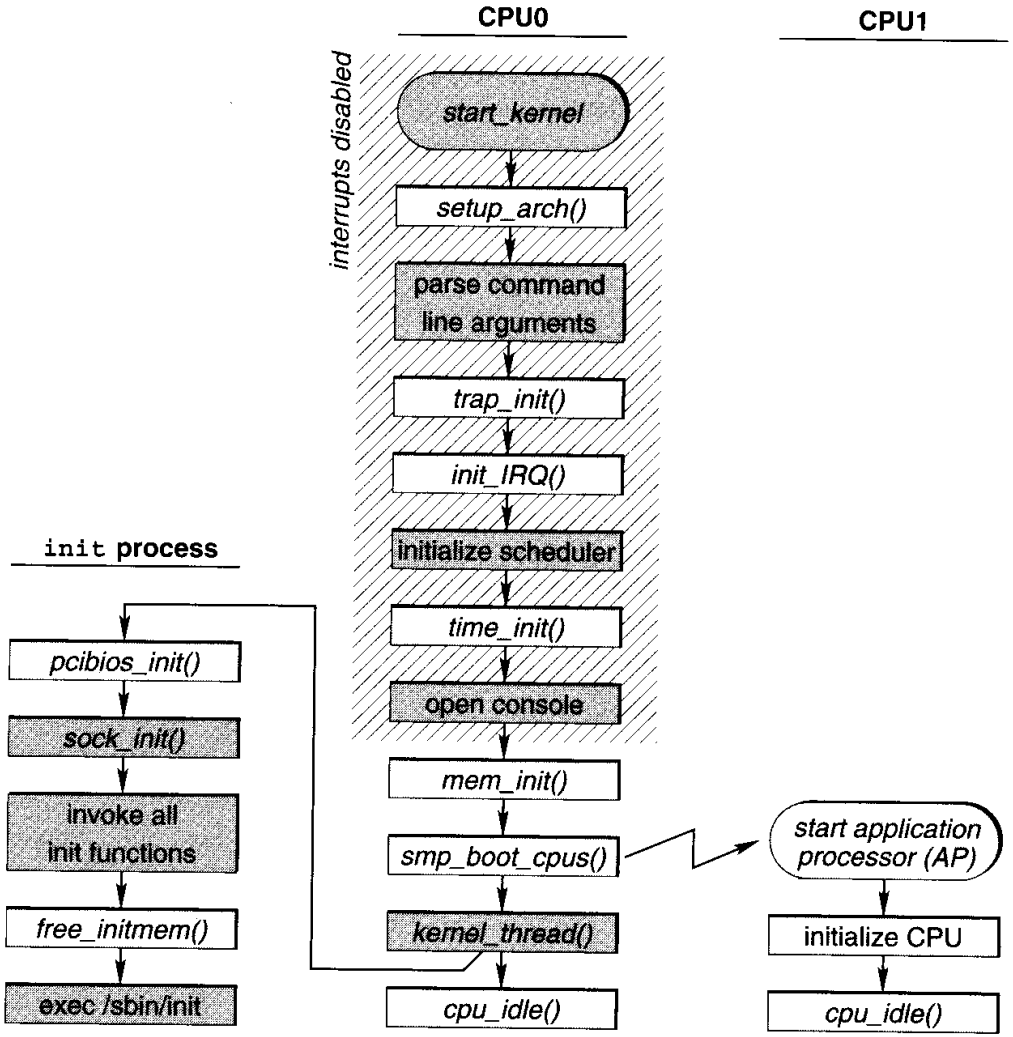


Figure 10.16. Overview of the Linux bootstrap procedure.

Inicjowanie pracy systemu

- `init` jest odpowiedzialny za prawidłową inicjację wszystkich procesów systemowych (jest pierwszym procesem w przestrzeni użytkownika i ojcem wszystkich procesów).
- `init` jest uruchamiany bezpośrednio przez jądro; pozostałe procesy są uruchamiane przez `init` lub przez jakiś z jego potomków
- `init` uruchamia (zwykle) /init (dawniej linuxrc) znajdujący się w initramfs (initrd)
- /init uruchamia /sbin/init po zamontowaniu partycji systemowej (tj. systemu plików, który zawiera pliki systemu operacyjnego); jego działanie jest kontrolowane przez /etc/inittab.

W nowszych wersjach jądra program `/sbin/init` został zastąpiony przez `/usr/lib/systemd/systemd`.

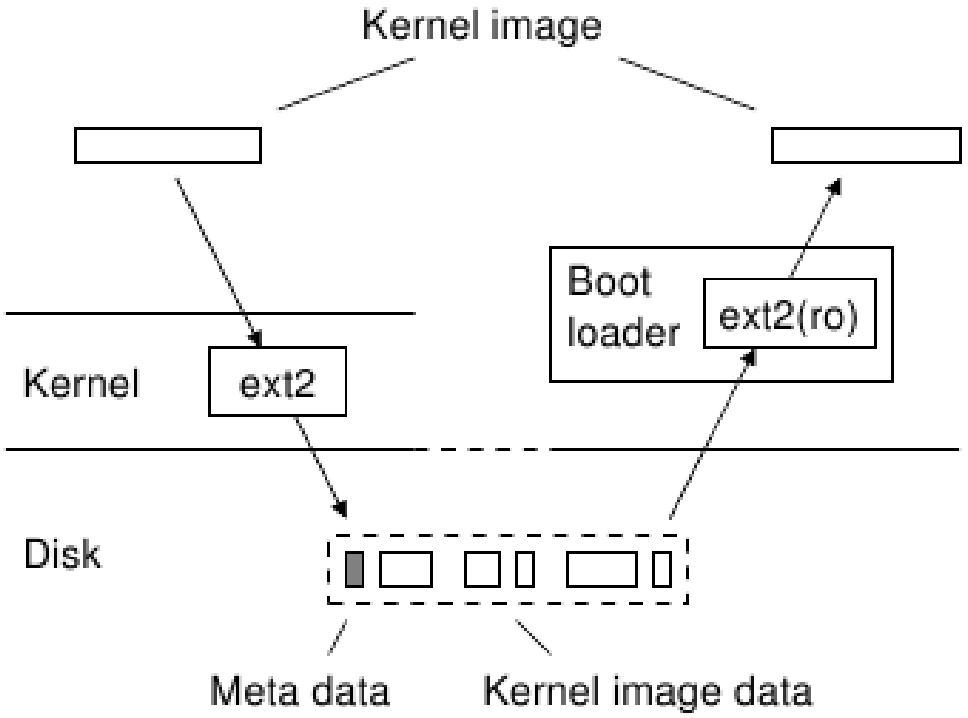
Zob.: http://www-128.ibm.com/developerworks/library/l-linuxboot/index.html

Rodzaje programów uruchomieniowych

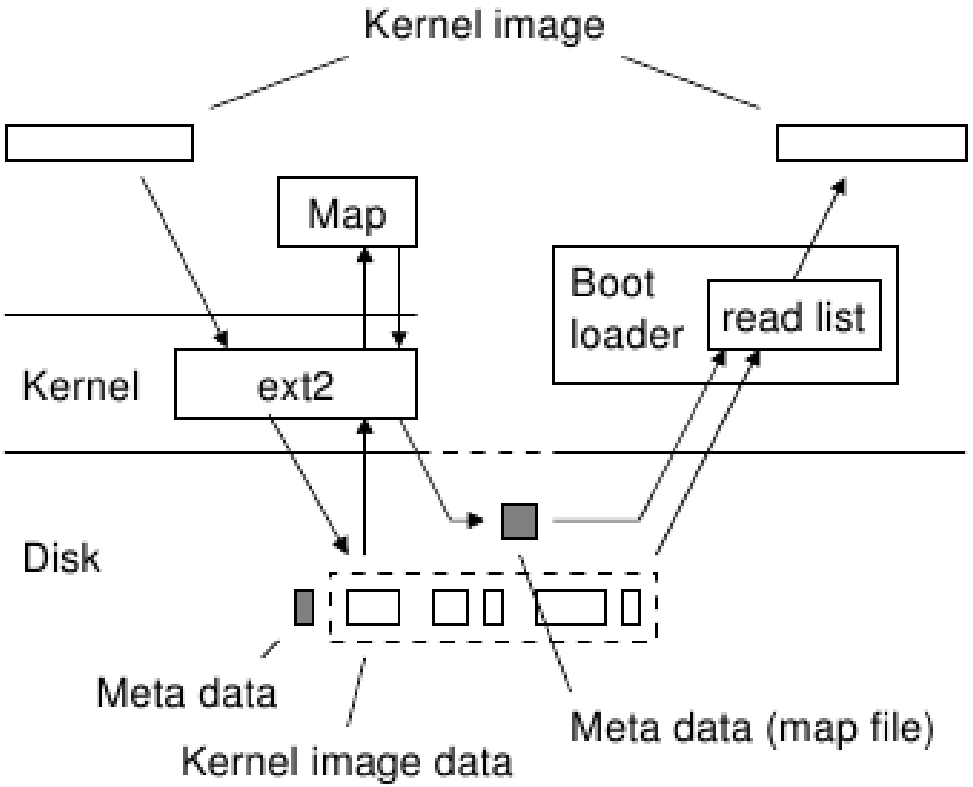
- specjalizowane, przystosowane do uruchamiania systemu z konkretnego nośnika, np. syslinux, isolinux
- ogólnego przeznaczenia pracujące pod kontrolą innego systemu operacyjnego (systemu operacyjnego gospodarza), np. loadlin
- ogólnego przeznaczenia zależne od systemu plików, np. grub, silo
- ogólnego przeznaczenia niezależne od systemu plików, np. lilo

Porównanie programów ładujących:
http://en.wikipedia.org/wiki/Comparison_of_boot_loaders

System aware bootloader



System unaware bootloader



LILO (*LInux LOader*)

Linuksowy program do ładowania systemu operacyjnego LILO:

- część I: instalowana w MBR lub w pierwszym sektorze aktywnej partycji; kopiuje do RAM pozostałą część LILO
- część II: `boot.b` instalowana na partycji zawierającej obraz systemu operacyjnego (partycja / z katalogiem `boot`) lub osobna partycja (`/boot`)
  - z dysku pobierana jest lista możliwych systemów do uruchomienia (`/etc/lilo.conf`)
  - uruchamianie wybranego systemu (np. *Loading Linux*)

LILO nie wspiera bootowania przez sieć!

/etc/lilo.conf

```
boot=/dev/hda2
map=/boot/map
install=/boot/boot.b
# timeout in 1/10s
timeout=100
prompt
message=/boot/message
default=linux
#restricted #password=******
image=/boot/vmlinuz-2.4.20
        label=linux
        initrd=/boot/initrd-2.4.20.img
        read-only
        root=/dev/hda2

other=/dev/hda1
        label=win98
```

GRUB[16] – zalety

- może być instalowany w MBR-e, sektorze rozruchowym partycji; korzysta ze zwykłych plików instalowanych na partycji SO, które mogą być modyfikowane bez konieczności reinstalacji programu ładującego
- obsługuje wiele systemów plików (ext2/3/4, FAT12/FAT16/FAT32, HFS, ISO9660, JFS, ReiserFS, UDF, FOOTNOTESIZE, NTFS, HFS+, ZFS), wspiera cpio, tar
- może być umieszczany na partycjach RAID (*software* lub LVM)
- wspiera automatyczną dekompresję plików (gzip, xz)
- niezależny od sposobu translacji geometrii dysku, obsługa trybu LBA (*Logical Block Addressing*)
- prawidłowo rozpoznaje wielkość dostępnej pamięci RAM
- może modyfikować tablicę partycji i zmieniać kolejność partycji, które widzi system operacyjny
- umożliwia przekazywanie parametrów do jądra uruchamianego systemu

[16] GRUB *GNU GRand Unified Bootloader*, https://www.ibm.com/developerworks/library/l-grub2/index.html

GRUB – pliki

```
# ls -la /boot/grub
-rw-r--r-- 1 root root     64 cze  5  2010 device.map
-rw-r--r-- 1 root root   7584 cze  5  2010 e2fs_stage1_5
-rw-r--r-- 1 root root   7456 cze  5  2010 fat_stage1_5
-rw-r--r-- 1 root root   6720 cze  5  2010 ffs_stage1_5
-rw------- 1 root root   1153 lut  2 22:03 grub.conf
-rw-r--r-- 1 root root   6720 cze  5  2010 iso9660_stage1_5
-rw-r--r-- 1 root root   8224 cze  5  2010 jfs_stage1_5
lrwxrwxrwx 1 root root     11 cze  5  2010 menu.lst -> ./grub.conf
-rw-r--r-- 1 root root   6880 cze  5  2010 minix_stage1_5
-rw-r--r-- 1 root root   9248 cze  5  2010 reiserfs_stage1_5
-rw-r--r-- 1 root root  55808 mar 16  2009 splash.xpm.gz
-rw-r--r-- 1 root root    512 cze  5  2010 stage1
-rw-r--r-- 1 root root 104988 cze  5  2010 stage2
-rw-r--r-- 1 root root   7072 cze  5  2010 ufs2_stage1_5
-rw-r--r-- 1 root root   6272 cze  5  2010 vstafs_stage1_5
-rw-r--r-- 1 root root   8872 cze  5  2010 xfs_stage1_5
```

GRUB – fazy działania[17]

1. z MBR-a jest pobierany i uruchamiany program `stage1`
2. program ładujący zna adres do pliku stopnia (fazy) 1.5, który jest umieszczany zaraz za MBR-em w obszarze zwanym *DOS compatibility space* (62 sektory)
3. program ładujący uzyskuje dostęp do systemu plików, z którego pobierany jest plik fazy 2. (`stage2`)
4. dostępna jest powłoka GRUB-a, pobierany jest plik konfiguracyjny (`/boot/grub.conf`), wyświetlana lista możliwych systemów operacyjnych do uruchomienia

[17] Barry Nauta, *Bootloaders – an introduction*, 2008

GRUB – /boot/grub/grub.conf

```
default=0
# timeout in 1s (-1 disables timeout)
timeout=10
splashimage=(hd0,1)/boot/grub/splash.xpm.gz
title Fedora Core (2.6.10-1.9_FC2)
        root (hd0,1)
        kernel /boot/vmlinuz-2.6.10-1.9_FC2 ro root=/dev/hda2
        initrd /boot/initrd-2.6.10-1.9_FC2.img
title Windows 95/98/NT/2000
map (hd0,0) (hd0,2)
map (hd0,2) (hd0,0)
makeactive
rootnoverify (hd0,2)
chainloader +1


# cat /boot/grub/device.map
(hd0)     /dev/sda

# cat /boot/grub/device.map
(hd0)     /dev/xvda
```

GRUB2 – /boot/grub2/grub.cfg

```
### BEGIN /etc/grub.d/10_linux ###
menuentry 'Fedora (4.7.9-200.fc24.x86_64) 24 (Twenty Four)' --class fedora --class gnu-linux \
     --class gnu --class os --unrestricted $menuentry_id_option 'gnulinux-4.6.7-300.fc24....' {
load_video
set gfxpayload=keep
insmod gzio
insmod part_msdos
insmod ext2
set root='hd1,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
  search --no-floppy --fs-uuid --set=root --hint-bios=hd1,msdos1 --hint-efi=hd1,msdos1 \
                              --hint-baremetal=ahci1,msdos1  9fc4d52e-9f74-4a59-a553-c08d36831708
else
  search --no-floppy --fs-uuid --set=root 9fc4d52e-9f74-4a59-a553-c08d36831708
fi
linux16 /boot/vmlinuz-4.7.9-200.fc24.x86_64 root=/dev/sdb1 ro rd.lvm.lv=fedora_scobie/root \
                                            rd.lvm.lv=fedora_scobie/swap rhgb quiet LANG=en_US.UTF-8
initrd16 /boot/initramfs-4.7.9-200.fc24.x86_64.img
}
```

```
# cat /boot/grub2/device.map
(hd0)      /dev/sda
(system-slash)      /dev/mapper/system-slash
```

GRUB2 – własności[18]

- modularny, nie potrzebuje *stage 1.5*, moduły są ładowane w razie potrzeby (obsługa LVM, RAID, bootowanie przez sieć)
- partycje są numerowane od 1 (poprzedzone przez msdos/gpt)
- urządzenia blokowe są numerowane od 0
- instalacja GRUB2 na dysku z tablicą GPT wymaga utworzenia oddzielnej partycji (500 MB) typu bios_grub (*BIOS boot partition*, EF02), gdzie instalowany jest program rozruchowy (ściślej, plik /boot/grub2/i386-pc/core.img)
- instalacja GRUB2 wymaga wolnego miejsca za MBR-m; pierwsza partycja powinna zaczynać się od sektora 2048 (ew. 4096), tzw. problem *MBR gap*[19]
- zarządzanie via `grub2-mkconfig`, `grub2-mkimage`, `grub2-install`, ...

[18] https://docs.fedoraproject.org/en-US/Fedora/22/pdf/Multiboot_Guide/Fedora-22-Multiboot_Guide-en-US.pdf
[19] https://wiki.archlinux.org/index.php/GRUB

SYS/ISO/EXT/PXELINUX

SYSLINUX poszukuje pliku konfiguracyjnego w następującej kolejności:

```
/boot/syslinux/syslinux.cfg
/syslinux/syslinux.cfg
/syslinux.cfg
```

Przykładowy plik konfiguracyjny:

```
default Diskless-CentOS56-i386
label Diskless-CentOS56-i386
    kernel Diskless-CentOS56-i386/vmlinuz
    append initrd=Diskless-CentOS56-i386/initrd.img root=/dev/ram0 \
           init=disklessrc ramdisk_size=128 ETHERNET=eth0  \
           NFSROOT=158.75.5.240:/arc-data/images/diskless/i386/CentOS56
```

Lista parametrów jądra:
http://www.kernel.org/doc/Documentation/kernel-parameters.txt

SYS/ISO/EXT/PXELINUX

Przykładowy plik konfiguracyjny:

```
# install CentOS 5.5
default linux
prompt 1
# timeout in  1/10s
timeout 500
display pxelinux.cfg/boot.msg
F1 pxelinux.cfg/boot.msg
...
F5 pxelinux.cfg/rescue.msg
label linux
  kernel i386/vmlinuz
  append initrd=i386/initrd.img
label ks
  kernel i386/vmlinuz
  append ks load_ramdisk=1 initrd=i386/initrd.img network \
           ks=http://www.fizyka.umk.pl/~jkob/Linux/kickstart-centos/
label memtest86
  kernel memtest
  append -
```

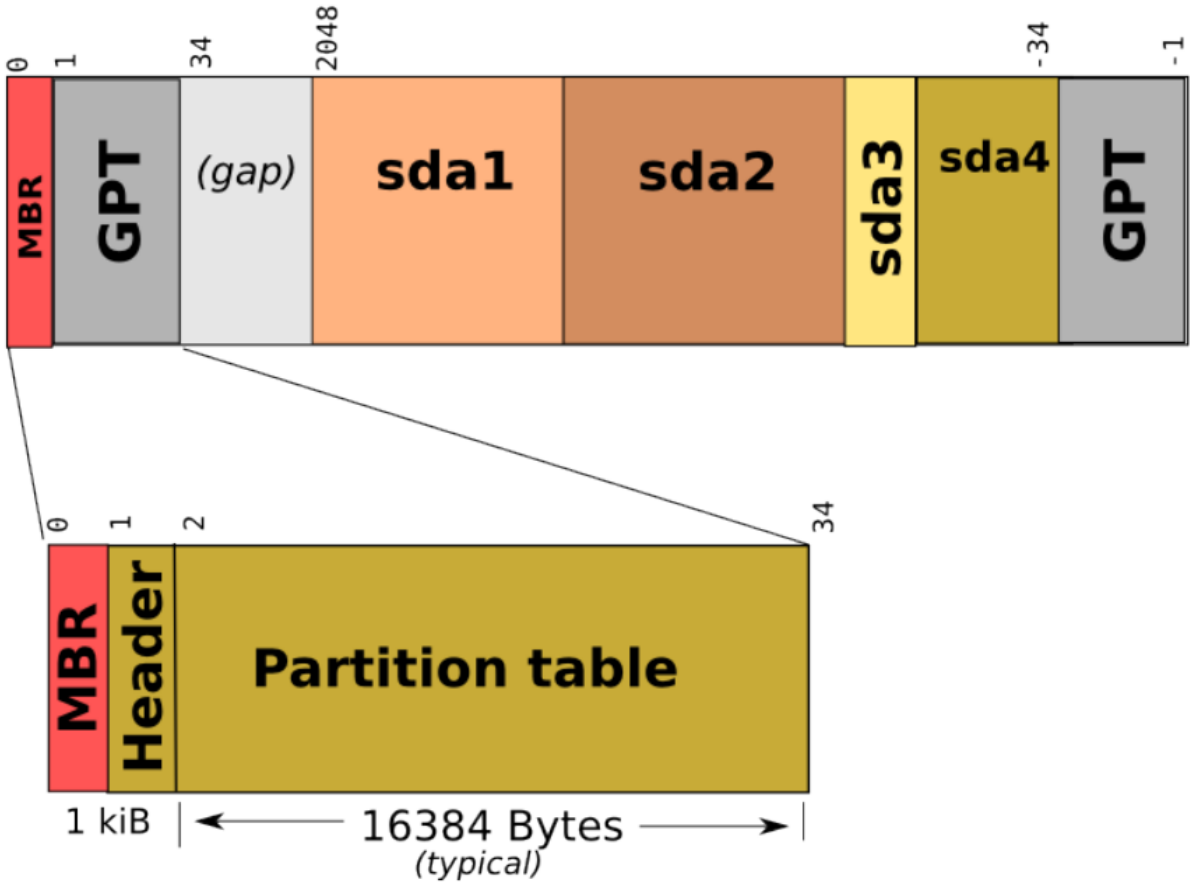
Tablica partycji GPT

Tablica MSDOS: partycja ograniczona do $2^{32} \times 512$ B=2 TiB.
Standard UEFI: GPT (*GUID Partition Table*).

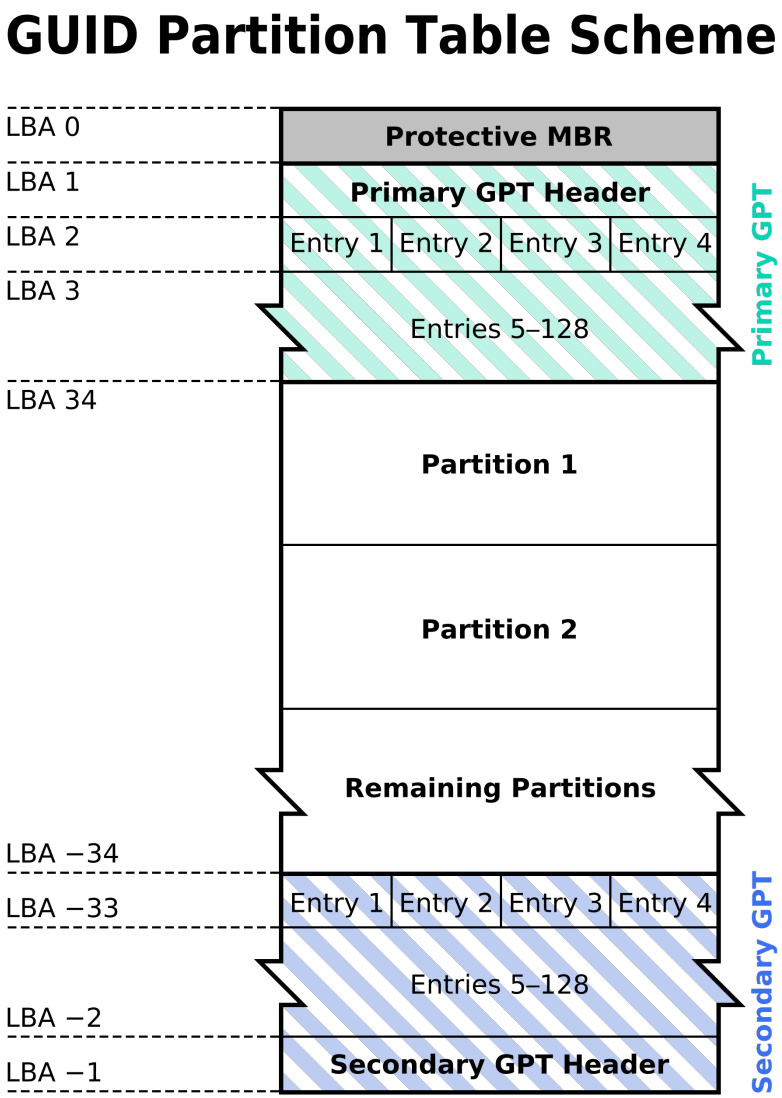
Zalety:

- LBA 0: określa MBR dla zachowania wstecznej kompatybilności, *protective MBR* z partycją typu 0xEE wypełniającą cały dysk lub partycję długości 2 TiB
- LBA 1: nagłówek określa maksymalną liczbę partycji (domyślnie 128), wielkość opisu partycji (domyślnie 128 B), sumy kontrolne CRC32 i UUID dysku
- opis partycji: 64-bitowy początek i koniec (LBA), typ (UUID), nazwa (do 36 znaków kodowych UTF-16LE)
- partycje mogą być większe niż 2 TiB (do 8 ZiB)
- partycje są identyfikowane przez GUID (przenaszalność dysków)
- typy partycji są identyfikowane prze GUID (brak konfliktów)
- przechowuje kopię tablicy partycji na końcu dysku

Struktura dysku z tablicą GPT[20]



[20] http://www.redhat.com: Bonneville, Getting Beyond 2 Terabytes Using gpt with storage devices


GUID Partition Table Scheme
LBA 0
LBA 1
LBA 2
LBA 3
LBA 34
LBA −34
LBA −33
LBA −2
LBA −1
Protective MBR
Primary GPT Header
Entry 1
Entry 2
Entry 3
Entry 4
Entries 5–128
Partition 1
Partition 2
Remaining Partitions
Entry 1
Entry 2
Entry 3
Entry 4
Entries 5–128
Secondary GPT Header
Primary GPT
Secondary GPT

Tablica partycji GPT

Typy partycji:

Linux/Windows data	EBD0A0A2-B9E5-4433-87C0-68B6B72699C7
Linux swap	0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
Linux LVM	E6D6D379-F507-44C2-A23C-238F2A3DF928
Linux RAID	A19D880F-05FC-4D3B-A006-743F0F84911E

Tablica partycji GPT: przykład

```
# gdisk -l /dev/sda
GPT fdisk (gdisk) version 0.8.2

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.
Disk /dev/sda: 625142448 sectors, 298.1 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): B11C509D-52A5-4E2B-9864-B8C56DB9E314
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 625142414
Partitions will be aligned on 2048-sector boundaries
Total free space is 146029 sectors (71.3 MiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            4096            8191   2.0 MiB     EF02  pri
   2            8192          976895   473.0 MiB   EF00  pri
   3          976896       625000447   297.6 GiB   8E00  pri
```

Tablica partycji GPT

- narzędzia: parted, gdisk, sgdisk, blkid
- generowanie UUID: `uuidgen [-rt]`
- czy GUID jest identyczny z UUID?
  http://stackoverflow.com/questions/246930/is-there-any-difference-between-a-guid-and-a-uuid
- kiedy bootowanie z dysku GPT wymaga systemu wspierającego UEFI?
  http://www.rodsbooks.com/gdisk/bios.html
- zob. także:
  http://wiki.archlinux.org/index.php/GRUB
  http://www.ata-atapi.com/hiwtab.htm
  http://en.wikipedia.org/wiki/GUID_Partition_Table
  http://www.rodsbooks.com/gdisk/index.html

Pliki obrazu initrd/initramfs[21]

Czasy, kiedy parametr root przyjmował wartości /dev/fd0 lub /dev/hda1 minęły. Obecnie system może znajdować się na CDROM-e, kluczu USB, dysku SCSI, SATA, macierzy dyskowej RAID, na dysku sieciowym (NFS).

Zamontowanie systemowej partycji wymaga wcześniejszego załadowania modułu lub modułów do jej prawidłowej obsługi (rozmaite kontrolery dysków SCSI, obsługa DHCP, DNS, połączenie sieciowe, szyfrowanie).

Plik obrazu initrd/initramfs jest wykorzystywany do przechowywania programu linuxrc/init/systemd i innych programów i modułów potrzebnych do uruchomienia systemu, dzięki czemu nie trzeba modyfikować jądra przy zmianie sposobu bootowania.

[21] http://linuxdevices.com/articles/AT4017834659.html

Pliki initrd dla jąder serii do 2.4

Plik initrd.img to skompresowamy obraz partycji ext2, który służy do zainicjowania fragmentu pamięci RAM związanego z urządzeniem /dev/initrd (zobacz `man initrd`).

initrd.img jest ładowany do pamięci RAM (inicjowany) przez program ładujący przed uruchomieniem jądra.

Dostęp:

```
mkdir initrd
mv initrd.img initrd.img.gz
gunzip initrd.img.gz
mount -o loop initrd.img initrd
```

Pliki initrd dla jąder serii $\geqslant$ 2.6

Plik initramfs.img to skompresowane archiwum cpio partycji initramfs (system plików ramfs/tmpfs).

Dostęp:[22]

```
mkdir initramfs
cd initramfs
gzip -dc < ../initramfs.img | cpio -i

lsinitrd | less
lsinitrd -f /etc/ld.so.conf
```

Uwaga! Dla Redhat Enterprise Linux 7:

```
mkdir initramfs
cd initramfs
/usr/lib/dracut/skipcpio ../initramfs.img | zcat | cpio -i
```

Tworzenie:

```
mkinitrd|dracut  <initrd-image> <kernel-version>
```

[22] Skrypt ułatwiający modyfikowanie plików initrd.img/initramfs.img: http://jkob.fizyka.umk.pl/labul/scripts/initramfs.sh

initrd kontra initramfs

Wady dysku tworzone w pamięci opercyjnej:

- określony rozmiar
- określony blokowy system plików (potrzebny sterownik)
- operacje I/O via pamięć podręczna
- linuxrc wykonywany z PID=0

Zalety initramfs:

- system plików zbudowany w oparciu o pamięć podręczną stron (zastosowanie systemu plików ramfs)
- wielkość zależna od potrzeb
- pliki są przechowywane bezpośrednio w pamięci bez konieczności stosowania urządzenia blokowego i systemu plików
- init/systemd wykonywany jako pierwszy proces w przestrzeni użytkownika (PID=1)

Zalety initramfs[23]

tmpfs puts everything into the kernel internal caches and grows and shrinks to accommodate the files it contains and is able to swap unneeded pages out to swap space. It has maximum size limits which can be adjusted on the fly via 'mount -o remount ...'

If you compare it to ramfs (which was the template to create tmpfs) you gain swapping and limit checking. Another similar thing is the RAM disk (/dev/ram*), which simulates a fixed size hard disk in physical RAM, where you have to create an ordinary filesystem on top. Ramdisks cannot swap and you do not have the possibility to resize them.

Since tmpfs lives completely in the page cache and on swap, all tmpfs pages currently in memory will show up as cached. It will not show up as shared or something like that. Further on you can check the actual RAM+swap use of a tmpfs instance with df(1) and du(1).

[23] Zob. /usr/src/linux/Documentation/filesystems/ramfs-rootfs-initramfs.txt, /usr/src/linux/Documentation/filesystems/tmpfs.txt

Analiza plików initrd/initramfs

Powiększanie się skryptu uruchomieniowego (i plików obrazu):

- linuxrc (RH8.0, 2002): 432 B (3 MB)
- init (F9, 2008): 2592 B
- init (F13, 2010): 2592 B (12 MB)
- init (F16, 2011): 10271 B (17 MB)
- init=/usr/lib/systemd/systemd (F20, 2014): 1210216 B (18 MB); initramfs-0-rescue-...img (42 MB)

Analiza plików initrd/initramfs

initrd-2.4.18-14.img (RedHat 8.0): /linuxrc

```
#!/bin/nash

echo "Loading jbd module"
insmod /lib/jbd.o
echo "Loading ext3 module"
insmod /lib/ext3.o
echo Mounting /proc filesystem
mount -t proc /proc /proc
echo Creating block devices
mkdevices /dev
echo Creating root device
mkrootdev /dev/root
echo 0x0100 > /proc/sys/kernel/real-root-dev
echo Mounting root filesystem
mount -o defaults --ro -t ext3 /dev/root /sysroot
pivot_root /sysroot /sysroot/initrd
umount /initrd/proc
```

initrd-2.6.30.10-105.2.23.fc11.i686.PAE:/init

```
!/bin/nash
mount -t proc /proc /proc
setquiet
echo Mounting proc filesystem
echo Mounting sysfs filesystem
mount -t sysfs /sys /sys
echo Creating /dev
mount -o mode=0755 -t tmpfs /dev /dev
mkdir /dev/pts
mount -t devpts -o gid=5,mode=620 /dev/pts /dev/pts
mkdir /dev/shm
mkdir /dev/mapper
echo Creating initial device nodes
mknod /dev/null c 1 3
mknod /dev/zero c 1 5
...
mknod /dev/ttyS0 c 4 64
daemonize --ignore-missing /bin/plymouthd
echo Setting up hotplug.
hotplug
echo "Loading i2c-core module"
modprobe -q i2c-core
echo "Loading output module"
modprobe -q output
echo "Loading video module"
modprobe -q video
echo "Loading i2c-algo-bit module"
modprobe -q i2c-algo-bit
echo "Loading drm module"
modprobe -q drm
echo "Loading i915 module"
modprobe -q i915
/lib/udev/console_init tty0
plymouth --show-splash
echo Creating block device nodes.
mkblkdevs
echo Creating character device nodes.
mkchardevs
echo "Loading pata_acpi module"
modprobe -q pata_acpi
echo "Loading ata_generic module"
modprobe -q ata_generic
echo Making device-mapper control node
mkdmnod
modprobe scsi_wait_scan
rmmod scsi_wait_scan
mkblkdevs
echo Scanning logical volumes
lvm vgscan --ignorelockingfailure
echo Activating logical volumes
lvm vgchange -ay --ignorelockingfailure  vg00
resume /dev/vg00/swap
echo Creating root device.
mkrootdev -t ext3 -o defaults,ro /dev/vg00/root
echo Mounting root filesystem.
mount /sysroot
cond -ne 0 plymouth --hide-splash
echo Setting up other filesystems.
setuproot
loadpolicy
plymouth --newroot=/sysroot
echo Switching to new root and running init.
switchroot
echo Booting has failed.
```

Zawartość /boot/initramfs-3.13.6-200.fc20.x86_64

```
total used in directory 18 available 343086
drwxr-xr-x  12 root root 1024 Mar 10 19:20 .
dr-xr-xr-x.  6 root root 3072 Mar 10 19:20 ..
lrwxrwxrwx   1 root root    7 Mar 10 19:20 bin -> usr/bin
drwxr-xr-x   2 root root 1024 Mar 10 19:20 dev
drwxr-xr-x  12 root root 1024 Mar 10 19:20 etc
lrwxrwxrwx   1 root root   24 Mar 10 19:20 init -> /usr/lib/systemd/systemd
lrwxrwxrwx   1 root root    7 Mar 10 19:20 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Mar 10 19:20 lib64 -> usr/lib64
drwxr-xr-x   2 root root 1024 Mar 10 19:20 proc
drwxr-xr-x   2 root root 1024 Mar 10 19:20 root
drwxr-xr-x   2 root root 1024 Mar 10 19:20 run
lrwxrwxrwx   1 root root    8 Mar 10 19:20 sbin -> usr/sbin
-rwxr-xr-x   1 root root 3017 Mar 10 19:20 shutdown
drwxr-xr-x   2 root root 1024 Mar 10 19:20 sys
drwxr-xr-x   2 root root 1024 Mar 10 19:20 sysroot
drwxr-xr-x   2 root root 1024 Mar 10 19:20 tmp
drwxr-xr-x   7 root root 1024 Mar 10 19:20 usr
drwxr-xr-x   2 root root 1024 Mar 10 19:20 var
```

Zawartość /boot/initramfs-3.13.6-200.fc20.x86_64

```
total used in directory 18 available 343086
drwxr-xr-x  12 root root 1024 Mar 10 19:20 .
dr-xr-xr-x.  6 root root 3072 Mar 10 19:20 ..
lrwxrwxrwx   1 root root    7 Mar 10 19:20 bin -> usr/bin
drwxr-xr-x   2 root root 1024 Mar 10 19:20 dev
drwxr-xr-x  12 root root 1024 Mar 10 19:20 etc
lrwxrwxrwx   1 root root   24 Mar 10 19:20 init -> /usr/lib/systemd/systemd
lrwxrwxrwx   1 root root    7 Mar 10 19:20 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Mar 10 19:20 lib64 -> usr/lib64
drwxr-xr-x   2 root root 1024 Mar 10 19:20 proc
drwxr-xr-x   2 root root 1024 Mar 10 19:20 root
drwxr-xr-x   2 root root 1024 Mar 10 19:20 run
lrwxrwxrwx   1 root root    8 Mar 10 19:20 sbin -> usr/sbin
-rwxr-xr-x   1 root root 3017 Mar 10 19:20 shutdown
drwxr-xr-x   2 root root 1024 Mar 10 19:20 sys
drwxr-xr-x   2 root root 1024 Mar 10 19:20 sysroot
drwxr-xr-x   2 root root 1024 Mar 10 19:20 tmp
drwxr-xr-x   7 root root 1024 Mar 10 19:20 usr
drwxr-xr-x   2 root root 1024 Mar 10 19:20 var
```

Zawartość /boot/initramfs-4.13.9-200.fc26.x86_64.img

```
lrwxrwxrwx   1 root     root            7 Aug 11 14:19 bin -> usr/bin
drwxr-xr-x   2 root     root            0 Aug 11 14:19 dev
drwxr-xr-x  13 root     root            0 Aug 11 14:19 etc
drwxr-xr-x   3 root     root            0 Aug 11 14:19 home
lrwxrwxrwx   1 root     root           23 Aug 11 14:19 init -> usr/lib/systemd/systemd
lrwxrwxrwx   1 root     root            7 Aug 11 14:19 lib -> usr/lib
lrwxrwxrwx   1 root     root            9 Aug 11 14:19 lib64 -> usr/lib64
drwxr-xr-x   2 root     root            0 Aug 11 14:19 proc
drwxr-xr-x   2 root     root            0 Aug 11 14:19 root
drwxr-xr-x   2 root     root            0 Aug 11 14:19 run
lrwxrwxrwx   1 root     root            8 Aug 11 14:19 sbin -> usr/sbin
-rwxr-xr-x   1 root     root         3113 Aug 11 13:44 shutdown
drwxr-xr-x   2 root     root            0 Aug 11 14:19 sys
drwxr-xr-x   2 root     root            0 Aug 11 14:19 sysroot
drwxr-xr-x   2 root     root            0 Aug 11 14:19 tmp
drwxr-xr-x   6 root     root            0 Aug 11 14:19 usr
drwxr-xr-x   3 root     root            0 Aug 11 14:19 var
```

Zawartość /boot/initramfs-3.10.0-123.el7.x86_64

```
drwxr-xr-x 12 root root 4,0K 10-21 20:55 .
drwxrwxr-x  4 jkob jkob 4,0K 10-21 20:55 ..
lrwxrwxrwx  1 root root    7 10-21 20:55 bin -> usr/bin
drwxr-xr-x  2 root root 4,0K 10-21 20:55 dev
drwxr-xr-x 12 root root 4,0K 10-21 20:55 etc
lrwxrwxrwx  1 root root   23 10-21 20:55 init -> usr/lib/systemd/systemd
lrwxrwxrwx  1 root root    7 10-21 20:55 lib -> usr/lib
lrwxrwxrwx  1 root root    9 10-21 20:55 lib64 -> usr/lib64
drwxr-xr-x  2 root root 4,0K 10-21 20:55 proc
drwxr-xr-x  2 root root 4,0K 10-21 20:55 root
drwxr-xr-x  4 root root 4,0K 10-21 20:55 run
lrwxrwxrwx  1 root root    8 10-21 20:55 sbin -> usr/sbin
-rwxr-xr-x  1 root root 3,0K 10-21 20:55 shutdown
drwxr-xr-x  2 root root 4,0K 10-21 20:55 sys
drwxr-xr-x  2 root root 4,0K 10-21 20:55 sysroot
drwxr-xr-x  2 root root 4,0K 10-21 20:55 tmp
drwxr-xr-x  7 root root 4,0K 10-21 20:55 usr
drwxr-xr-x  3 root root 4,0K 10-21 20:55 var
```

Bootowanie w sieci IP

Problem:
Jak załadować system operacyjny na bezdyskowy węzeł w sieci IP?

Rozwiązania:[24]

- RARP + TFTP
- BOOTP + TFTP
- PXE + DHCP + TFTP

[24] RARP (*Reverse Resolution Address Protocol*): RFC 903
BOOTP (*BOOTstrap Protocol*): RFC 951 i RFC 1084, uaktualnienia w RFC 1395 i RFC 1497
DHCP (*Dynamic Host Configuration Protocol*): RFC 1541 i RFC 2132 (także 1532-1534)
TFTP (*Trivial File Tranfer Protocol*): RFC 1350, RFC 2347, RFC 2348, RFC 2349

RARP + TFTP

W lokalnej sieci znajduje się serwer (rarpd), który dysponuje danymi wiążącymi adresy sprzętowe klientów (bezdyskowych hostów) z przyporządkowanymi im adresami IP (`/etc/ethers` lub baza NIS+).

```
# cat /etc/ethers
08:00:20:1f:0d:4d 192.168.2.33
08:00:20:21:9c:95 terminal1
...
```

Serwer odpowiada na zapytanie klienta, jeśli (przy domyślnej konfiguracji serwera TFTP) w katalogu `/tftpboot/` znajduje się plik (bootowalny obraz) o nazwie równej numerowi IP zapisanemu szesnastkowo i dużymi literami (zobacz `gethostip`).

```
lrwxrwxrwx  1 root  root      25 lip 15  2005 C0A80281.SUN4M -> kernel-sparc-2.2.25-xterm
-rw-r--r--  1 root  root 1471121 lip 15  2005 kernel-sparc-2.2.25-xterm
```

Klient po otrzymaniu od serwera RARP adresu IP pobiera via TFTP (z tego samego serwera) przeznaczony dla niego plik z programem rozruchowym.

RARP – ograniczenia[25]

- proces użytkownika komunikuje się z warstwą łącza danych bezpośrednio (zależność od systemu)
- zapytanie RARP zwraca jedynie IP (ale nie z powodu braku miejsca w pakiecie)
- nie można przekazywać zapytań RARP do centralnego serwera (jeden serwer na domenę rozgłoszeniową)
- klient wymaga obsługi ICMP i TFTP

[25] http://www.ecse.rpi.edu/Homepages/shivkuma/teaching/sp2001/ip2001-Lecture11-6pp.pdf

BOOTP

- komunikacja klient-serwer wykorzystuje protokoły UDP/IP
- oprogramowanie IP klienta wysyła rozgłoszenie
- serwer wykorzystuje port 67, a klient – 68
- w lokalnej sieci znajduje się serwer (bootpd), który dysponuje danymi wiążącymi adresy sprzętowe klientów z przyporządkowanymi im adresami IP znajdującymi się w pliku `/etc/bootptab`:

```
H12:  ht=ethernet: ha=00036EABABAB: ip=192.168.9.12:
   sm=255.255.255.0: gw=192.168.9.240:
H13:  ht=ethernet: ha=00036ECDCDCD: ip=192.168.9.13:
   sm=255.255.255.0: gw=192.168.9.240:
...
```

- pobieranie obrazu via TFTP

Ograniczenie: nie można dynamicznie przydzielać adresów IP

DHCP

W lokalnej sieci znajduje się serwer (dhcpd), który dostarcza klientom danych umożliwiających konfigurację interfejsu sieciowego, załadowanie programu uruchomieniowego, zamontowanie partycji systemowej. Domyślnie demon dhcpd korzysta z pliku konfiguracyjnego `/etc/dhcpd.conf` i korzysta z portu 67 (klient z portu 68).

```
NAME
       dhcpd - Dynamic Host Configuration Protocol Server

SYNOPSIS
       dhcpd  [  -p  port ] [ -f ] [ -d ] [ -q ] [ -t | -T ] [ -4 | -6 ]
              [ -s server ] [ -cf config-file ] [ -lf lease-file ]
              [ -pf pid-file ] [ --no-pid ] [ -tf trace-output-file ]
              [ -user user ] [ -group group ] [ -chroot dir ]
              [  -play  trace-playback-file ] [ if0 [ ...ifN ] ]
```

DHCP: działanie

DHCP Lease Stages[26]

1. Lease Request – The client sends a broadcast requesting an IP address (DHCPDISCOVER broadcast message)
2. Lease Offer – The server sends
   - IP address
   - Netmask
   - Default Gateway address
   - DNS server addresse(s)
   - NetBIOS Name server (NBNS) address(es).
   - Lease period in hours
   - IP address of DHCP server.

   and marks the offered address as unavailable. The message sent is a DHCPOFFER unicast/broadcast message.

[26] http://www.comptechdoc.org/independent/networking/guide/netdhcp.html

DHCP Lease Stages (cont)

3. Lease Acceptance – The first offer received by the client is accepted. The acceptance is sent from the client as a broadcast (DHCPREQUEST message) including the IP address of the DHCP server that sent the accepted offer. Other DHCP servers retract their offers and mark the offered address as available and the accepted address as unavailable.
4. Server lease acknowledgement – The server sends a DHCPACK or a DHCPNACK if an unavailable address was requested.

DHCP Lease Stages (cont)

DHCP discover message – the initial broadcast sent by the client to obtain a DHCP lease.

It contains the client MAC address and computer name. This is a broadcast using 255.255.255.255 as the destination address and 0.0.0.0 as the source address. The request is sent, then the client waits one second for an offer. The request is repeated at 9, 13, and 16 second intervals with additional 0 to 1000 milliseconds of randomness. The attempt is repeated every 5 minutes thereafter.

The client uses its own port 68 as the source port with port 67 as the destination port on the server to send the request to the server. The server uses its own port 67 as the source port with port 68 as the destination port on the client to reply to the client. Therefore the server is listening and sending on its own port 67 and the client is listening and sending on its own port 68.

DHCP Lease Renewal

- After 50% of the lease time has passed, the client will attempt to renew the lease with the original DHCP server that it obtained the lease from using a DHCPREQUEST message.
- Any time the client boots and the lease is 50% or more passed, the client will attempt to renew the lease.
- At 87.5% of the lease completion, the client will attempt to contact any DHCP server for a new lease.
- If the lease expires, the client will send a request as in the initial boot when the client had no IP address.

  If this fails, the client TCP/IP stack will cease functioning.

DHCP: RFC 1541

```
Message         Use
-------         ---
DHCPDISCOVER -  Client broadcast to locate available servers.
DHCPOFFER    -  Server to client in response to DHCPDISCOVER with
                offer of configuration parameters.
DHCPREQUEST  -  Client broadcast to servers requesting offered
                parameters from one server and implicitly declining
                offers from all others.
DHCPACK      -  Server to client with configuration parameters,
                including committed network address.
DHCPNAK      -  Server to client refusing request for configuration
                parameters (e.g., requested network address already
                allocated).
DHCPDECLINE  -  Client to server indicating configuration parameters
                (e.g., network address) invalid.
DHCPRELEASE  -  Client to server relinquishing network address and
                cancelling remaining lease.
```

Plik konfiguracyjny `/etc/dhcpd.conf`, `/etc/dhcp/dhcpd.conf`

```
authoritative;

ddns-update-style none;
#ddns-update-style interim;
#ddns-update-style standard;
#ddns-update-style ad-hoc;      # deprecated, does not work

# Bootp queries are allowed by default.
allow bootp;

# By default a lease expires after one day
# max-lease-time 84600


# If a client does not ask for a specific expiration time a lease will
# be granted for default-lease-time (in seconds)
# default-lease-time 42300
```

Plik konfiguracyjny `/etc/dhcpd.conf`

```
subnet 10.15.0.0 netmask 255.255.0.0 {
  option subnet-mask             255.255.0.0;
  option broadcast-address       10.15.255.255;
  option domain-name-servers     158.75.5.250, 158.75.1.4;
  option domain-name             "fizyka.umk.pl";
  option routers                 10.15.0.1;

  pool {
    range 10.15.0.10 10.15.255.200;
    max-lease-time 1200;
    default-lease-time 1200;
  }
}
```

Plik konfiguracyjny `/etc/dhcpd.conf`

```
subnet 158.75.4.0 netmask 255.255.254.0 {
  option subnet-mask 255.255.254.0;
  option broadcast-address 158.75.5.255;
  option routers 158.75.5.190;
  option domain-name-servers 158.75.5.250, 158.75.1.4;
  option domain-name "fizyka.umk.pl";
}

subnet 158.75.104.0 netmask 255.255.255.0 {
  option subnet-mask 255.255.255.0;
  option broadcast-address 158.75.104.255;
  option routers 158.75.104.254;
  option domain-name-servers 158.75.5.250, 1158.75.5.252, 58.75.1.4;
  option domain-name "fizyka.umk.pl";
}
```

Plik konfiguracyjny `/etc/dhcpd.conf`

```
subnet 158.75.105.224 netmask 255.255.255.224 {
    option subnet-mask 255.255.255.224;
    option broadcast-address 158.75.105.255;
    option routers 158.75.105.254;
    option domain-name "fizyka.umk.pl";
    option domain-name-servers 158.75.5.252, 158.75.1.4;
    filename "centos-6.3-i386-install/pxelinux.0";
    next-server 158.75.5.240;
    default-lease-time 15000;
    max-lease-time 20000;

    pool {
#            range 158.75.105.243 158.75.105.253;
            range 158.75.105.225 158.75.105.253;
             max-lease-time 5400;
        }

}
```

Plik konfiguracyjny `/etc/dhcpd.conf`

```
option option-128 code 128 = string;
option option-129 code 129 = text;
subnet 192.168.2.0 netmask 255.255.255.0 {
        option subnet-mask              255.255.255.0;
        option broadcast-address        192.168.2.255;
        option domain-name-servers      158.75.5.252, 158.75.1.4;
        option domain-name              "fizyka.umk.pl";
        next-server                     158.75.5.240;
        option root-path                "158.75.5.240:/mnt/gv-images/images/xterminals/
        use-host-decl-names             on;
        option routers                  192.168.2.254;
        option log-servers              158.75.5.240;
        filename                        "/xterminals/lts/kernel";
}
```

Plik konfiguracyjny `/etc/dhcpd.conf`

```
host 501
{
  hardware ethernet aa:bb:cc:dd:ee:ff;
  fixed-address 158.75.4.1;
  option host-name "kleks";
}

host 1504
{
  #deny booting;
  #ignore booting
  allow booting;
  filename "pxelinux.0";
  next-server 192.168.9.241;
  hardware ethernet 00:c0:4f:c6:db:d1;
  fixed-address 192.168.9.4;
}
```

Plik konfiguracyjny `/etc/dhcpd.conf`

```
host 2001
{
  hardware ethernet 00:50:da:43:2f:00;
  fixed-address 192.168.2.5;
  filename "/xterminals/lts/vmlinuz-2.4.19-ltsp-1";
}
host 2002
{
  hardware ethernet 00:c0:4f:c7:76:4b;
  fixed-address 192.168.2.6;
  option option-128 e4:45:74:68:00:00;
  option option-129 "nic=3c509, vga=773";
}
```

DHCP: przykład działania

Fragment pliku /etc/dhcpd.conf

```
   pool {
             range 158.75.4.197 158.75.4.254;
             max-lease-time 3600;
             deny unknown-clients;
         }
```

Fragment pliku /var/log/messages

```
Feb 28 08:25:36 hel dhcpd: DHCPDISCOVER from 00:e0:4c:ea:b5:81 via eth1
Feb 28 08:25:37 hel dhcpd: DHCPOFFER on 158.75.4.252 to 00:e0:4c:ea:b5:81 via eth1
Feb 28 08:25:37 hel dhcpd: DHCPREQUEST for 158.75.4.252 (158.75.5.90) from \
                                                        00:e0:4c:ea:b5:81 via eth1
Feb 28 08:25:37 hel dhcpd: DHCPACK on 158.75.4.252 to 00:e0:4c:ea:b5:81 via eth1
Feb 28 08:53:00 hel dhcpd: DHCPREQUEST for 158.75.4.252 from 00:e0:4c:ea:b5:81 via eth1
Feb 28 08:53:00 hel dhcpd: DHCPACK on 158.75.4.252 to 00:e0:4c:ea:b5:81 via eth1
Feb 28 09:19:34 hel dhcpd: DHCPREQUEST for 158.75.4.252 from 00:e0:4c:ea:b5:81 via eth1
Feb 28 09:19:34 hel dhcpd: DHCPACK on 158.75.4.252 to 00:e0:4c:ea:b5:81 via eth1
Feb 28 09:45:22 hel dhcpd: DHCPREQUEST for 158.75.4.252 from 00:e0:4c:ea:b5:81 via eth1
Feb 28 09:45:22 hel dhcpd: DHCPACK on 158.75.4.252 to 00:e0:4c:ea:b5:81 via eth1
Feb 28 10:13:32 hel dhcpd: DHCPREQUEST for 158.75.4.252 from 00:e0:4c:ea:b5:81 via eth1
Feb 28 10:13:32 hel dhcpd: DHCPACK on 158.75.4.252 to 00:e0:4c:ea:b5:81 via eth1
```

Fragment pliku `/var/lib/dhcpd/dhcpd.leases`

```
lease 158.75.4.223 {
  starts 1 2014/03/31 08:28:32;
  ends 1 2014/03/31 08:43:32;
  cltt 1 2014/03/31 08:28:32;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 6c:62:6d:34:7f:62;
  uid "\001lbm4\177b";
  client-hostname "KombajnII";
}
lease 158.75.4.223 {
  starts 1 2014/03/31 08:36:02;
  ends 1 2014/03/31 08:51:02;
  cltt 1 2014/03/31 08:36:02;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 6c:62:6d:34:7f:62;
  uid "\001lbm4\177b";
  client-hostname "KombajnII";
}
lease 158.75.4.223 {
  starts 1 2014/03/31 08:43:32;
  ends 1 2014/03/31 08:58:32;
  cltt 1 2014/03/31 08:43:32;
  binding state active;
  next binding state free;
  rewind binding state free;
  hardware ethernet 6c:62:6d:34:7f:62;
  uid "\001lbm4\177b";
  client-hostname "KombajnII";
}
```

TFTP

W lokalnej sieci znajduje się serwer (tftpd), który umożliwia klientom pobieranie pliku zawierającego program uruchomnieniowy (możliwy jest też zapis danych przez klientów na serwerze). Serwer domyślnie nasłuchuje na (dobrze znanym) porcie 69. Cechy protokołu TFTP:

- w warstwie transportowej korzysta z UDP
- wysłanie żądania odczytu lub zapisu pliku jest równoznaczne z żądaniem nawiązania połączenia
- jeśli serwer odpowiada pozytywnie na to żądanie, to rozpoczyna się przesyłanie pakietów w blokach o równej (negocjowanej) długości (nie większej niż 65464 B)
- każdy przesłany pakiet musi zostać potwierdzony przez odbiorcę; nadawca może ponownie przesłać zaginiony pakiet
- przesłanie pakietu krótszego niż ustalony oznacza zakończenie transmisji

TFTP

Demon tftpd jest uruchamiany przez superdemona sieciowego xinetd, w sposób określony przez plik konfiguracyjny: `/etc/xinetd.d/tftp`

```
# default: off
# description: The tftp server serves files using the trivial file transfer \
#       protocol.  The tftp protocol is often used to boot diskless \
#       workstations, download configuration files to network-aware printers, \
#       and to start the installation process for some operating systems.
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
#       server_args             = -c -s /tftpboot -v -B 1468
        server_args             = -s /tftpboot -v -B 1468
        disable                 = no
        per_source              = 11
        cps                     = 100 2
        flags                   = IPv4
}
```

Główny plik konfiguracyjny: `/etc/xinetd.conf`

```
defaults
{
# enabled =
# disabled =

# Define general logging characteristics.
log_type = SYSLOG daemon info
log_on_failure = HOST
log_on_success = PID HOST DURATION EXIT

# Define access restriction defaults
#
# no_access =
# only_from =
# max_load = 0
cps = 50 10
instances = 50
per_source = 10
...
# Generally, banners are not used. This sets up their global defaults
# banner =
# banner_fail =
# banner_success =
}
includedir /etc/xinetd.d
```

TFTP

W CentOS 7 demon tftpd jest uruchamiany jako oddzielna usługa przez systemd.

```
# systemctl cat tftp.service
# /usr/lib/systemd/system/tftp.service
[Unit]
Description=Tftp Server
Requires=tftp.socket
Documentation=man:in.tftpd

[Service]
ExecStart=/usr/sbin/in.tftpd -s /var/lib/tftpboot
StandardInput=socket

[Install]
Also=tftp.socket
```

PXELINUX

Działanie PXELINUX-a wymaga[27]

- uruchomienia i konfiguracji serwera DHCP
- uruchomienia serwera TFTP obsługującego opcję tsize (man tftpd: `Report the size of the file that is about to be transferred`).
- umieszczenia w katalogu (lub podkatalogach) /var/lib/tftpboot pliku pxelinux.0, plików uruchomieniowych oraz typu initrd/initramfs
- utworzenia katalogu pxelinux.cfg i umieszczenia w nim plików typu syslinux.cfg dla poszczególnych hostów lub grup hostów

[27] http://syslinux.zytor.com/faq.php

Bootowanie w sieci IP: PXELINUX

PXE poszukuje pliku konfiguracyjnego o nazwie:

1. UUID klienta (jeśli BIOS wspiera); tylko nowsze wersje syslinux
2. typu sieci i adresu sieciowego
3. adresu IP (zapisanego szestnastkowo)
4. adresu IP bez kolejnych najmniej znaczących cyfr
5. default

Jeśli żaden plik nie zostanie odnaleziony, to (od wersji 3.20) PXE będzie (po pewnej zwłoce) próbował ponownie załadować system operacyjny.

Nazwy plików są podawane względem katalogu, gdzie pxelinux.0 został zainstalowany. Nazwy mogą mieć co najwyżej 127 znaków.

`gethostip adresIP` zamienia adres IP zapisany w formie X.X.X.X na 8 cyfr szestnastkowych.

PXELINUX

Poszukiwanie pliku konfiguracyjnego

```
/var/lib/tftpboot/pxelinux.cfg/b8945908-d6a6-41a9-611d-74a6ab80b83d
/var/lib/tftpboot/pxelinux.cfg/01-88-99-aa-bb-cc-dd
/var/lib/tftpboot/pxelinux.cfg/C000025B
/var/lib/tftpboot/pxelinux.cfg/C000025
/var/lib/tftpboot/pxelinux.cfg/C00002
/var/lib/tftpboot/pxelinux.cfg/C0000
/var/lib/tftpboot/pxelinux.cfg/C000
/var/lib/tftpboot/pxelinux.cfg/C00
/var/lib/tftpboot/pxelinux.cfg/C0
/var/lib/tftpboot/pxelinux.cfg/C
/var/lib/tftpboot/pxelinux.cfg/default
```

PXELINUX[28]

Przykładowy plik konfiguracyjny:

```
default linux-lts
prompt 1
timeout 5
display boot.msg
F1 boot.msg
F2 options.msg
F3 general.msg
F4 param.msg
label linux-lts
  kernel /xterminals/lts/vmlinuz-2.4.26-ltsp-pxe
  append init=/linuxrc rw root=/dev/ram0 initrd=/xterminals/lts/initrd-2.4.26-ltsp-pxe
```

Fragment z /var/log/messages na serwerze TFTP

```
... in.tftpd[7039]: RRQ from 192.168.2.22 filename pxelinux.0
... in.tftpd[7040]: RRQ from 192.168.2.22 filename pxelinux.cfg/01-00-c0-4f-63-35-33
... in.tftpd[7041]: RRQ from 192.168.2.22 filename pxelinux.cfg/C0A80216
... in.tftpd[7042]: RRQ from 192.168.2.22 filename boot.msg
... in.tftpd[7043]: RRQ from 192.168.2.22 filename /xterminals/lts/vmlinuz-2.4.26-ltsp-pxe
```

[28] http://www.syslinux.org/wiki/index.php?title=SYSLINUX

Poziomy pracy systemu

man init/telinit, inittab, runlevel

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit
```

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# Schedule a shutdown for 2 minutes from now.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run xdm in runlevel 5
x:5:respawn:/etc/X11/prefdm -daemon
```

Poziomy pracy systemu

- zmiana poziomu pracy systemu:

```
/sbin/init [ -a ] [ -s ] [ -b ] [ 0123456Ss ]
/sbin/telinit [ -t sec ] [ 0123456sSQqabcUu ]
```

- administrowanie poziomami pracy systemu:

```
runlevel

chkconfig --list [name]
chkconfig --add name
chkconfig --del name
chkconfig [--level levels] name <on|off|reset>
chkconfig [--level levels] name

ntsysv [--level <levels>]
```

Uruchamianie/zatrzymywanie/sprawdzanie usług

```
/etc/init.d/cups --help
Użycie: /etc/init.d/cups {start|stop|restart|condrestart|reload|status}
```

Dostęp do usługi z dowolnego miejsca:

```
service cups {start|stop|restart|condrestart|reload|status}
```

Uruchamianie/zatrzymywanie/sprawdzanie usług

Każda usługa uruchomiona poprzez skrypt uruchomieniowy tworzy w /var/-lock/subsys plik blokady.

```
service <initscript> status
```

sprawdza PID pliku wykonywalnego i obecność pliku w katalogu /var/lock/-subsys/. Jeśli nie można znaleźć PID, ale podsystem jest zablokowany, to pojawia się komunikat:

```
<service> dead but pid file exists
```

Jeśli nie ma pliku blokady, to usługa może być uruchomiona/zatrzymana. Jednak przy zmianie poziomu pracy systemu skrypt rc sprawdza, czy w /var/lock/subsys/ znajdują się pliki blokad powiązanych z usługami, które mają być włączone/wyłączone. Brak jakiegoś pliku spowoduje, że powiązana z nim usługa nie zostanie poprawnie zatrzymana/uruchomiona.

Uruchamianie/zatrzymywanie/sprawdzanie usług

Podkatalog /var/lock/subsys jest sprawdzany przy wykonywaniu komend `shutdown` oraz `reboot`. Kolejność czynności wykonywanych podczas operacji `shutdown` jest następująca:[29]

1. dla każdej znanej usługi wykonywana jest komenda `service <initscript> stop`
2. komenda `kill -SIGTERM` wymusza zakończenie pozostałych przy życiu procesów
3. następuje pięciosekundowa zwłoka
4. komenda usuwająca `kill -SIGKILL` usuwa pozostałe przy życiu procesy

Operacje powyższe przeprowadza skrypt `/etc/init.d/killall` dla każdej usługi, dla której znaleziono plik blokady.

[29] http://www.redhat.com/magazine/008jun05/departments/tips_tricks/

Uruchamianie usług: upstart[30]

Because the traditional System V init daemon (SysVinit) does not deal well with modern hardware, including hotplug devices, USB hard and flash drives, and network-mounted filesystems, Ubuntu replaced it with the Upstart init daemon.

Upstart is a new init daemon that allows services to be started in response to events rather than in bulk runlevels. It also has support for monitoring services and restarting them when they go awry. It has the potential to expand to provide cron support as well, and to manage session-level as well as system-level services.

While much of this would require massive restructuring to accomplish, Upstart is also very capable of emulating a SysV style init system and can be placed into Fedora right now without any changes to the init scripts.

[30] http://www.linux.com/feature/125977, http://fedoraproject.org/wiki/Features/Upstart, http://upstart.ubuntu.com/wiki/

upstart: konfiguracja

```
/etc/init:

control-alt-delete.conf
init-system-dbus.conf
kexec-disable.conf
plymouth-shutdown.conf
prefdm.conf
quit-plymouth.conf

rc.conf

rcS-emergency.conf
rcS-sulogin.conf
rcS.conf
readahead-collector.conf
readahead-disable-services.conf
readahead.conf
serial.conf
splash-manager.conf
start-ttys.conf
tty.conf
vmstat.conf
```

```
# rc - System V runlevel compatibility
#
# This task runs the old sysv-rc runlevel scripts.  It
# is usually started by the telinit compatibility wrapper.

start on runlevel [0123456]

stop on runlevel [!$RUNLEVEL]

task

export RUNLEVEL
console output
exec /etc/rc.d/rc $RUNLEVEL
```

upstart: użycie

```
# initctl emit runlevel RUNLEVEL=3
# telinit 3


# cat vmstat.conf

      start on vmstat-on
      stop on vmstat-off

      exec vmstat 1  >> /root/vmstat.log


# initctl emit vmstat-on|vmstat-off
# [initctl] start|stop vmstat
```

Uruchamianie usług: systemd[31]

- `/sbin/init` jako dowiązanie symboliczne do `/usr/lib/systemd/systemd` (Debian, Fedora, Gentoo, Ubuntu (>3/2015))
- systemd is a system and service manager for Linux, compatible with SysV and LSB[32] init scripts. systemd provides
  - aggressive parallelization capabilities
  - uses socket and D-Bus activation for starting services
  - offers on-demand starting of daemons
  - keeps track of processes using Linux cgroups
  - maintains mount and automount points
  - implements an elaborate transactional dependency-based service control logic
  - it can work as a drop-in replacement for sysvinit

[31] http://fedoraproject.org/wiki/Systemd, http://linuxconfau.blip.tv/file/4696791
https://www.freedesktop.org/wiki/Software/systemd/
[32] https://en.wikipedia.org/wiki/Linux_Standard_Base

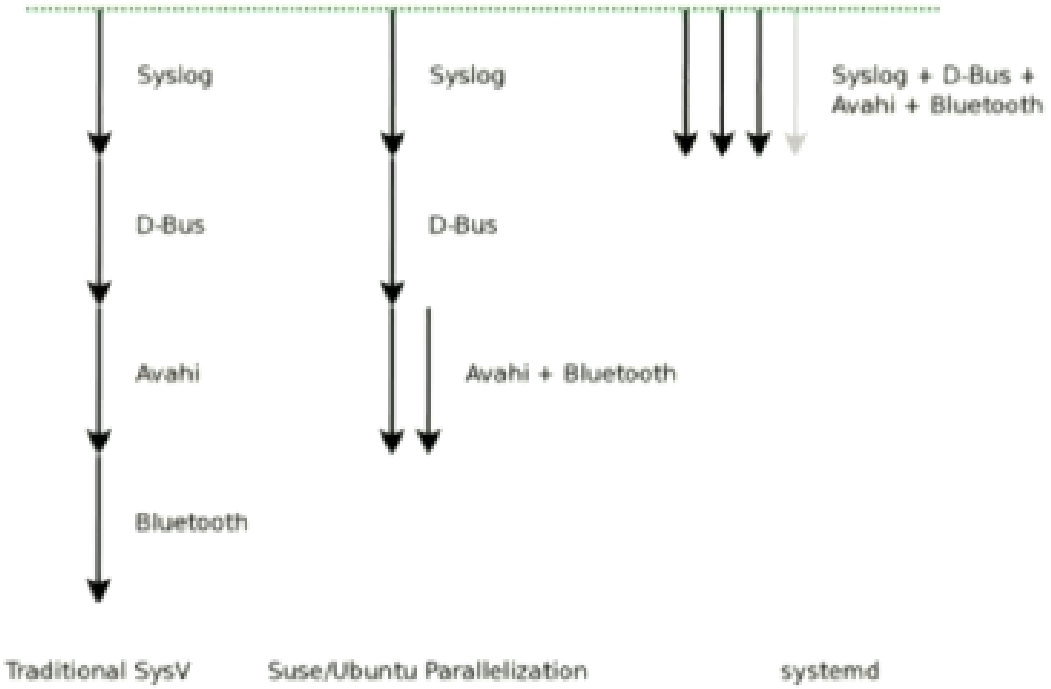
Uruchamianie usług: systemd – gniazda[33]

To make a TCP server the following steps are required:

- Create a socket with the socket() system call.
- Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
- Listen for connections with the listen() system call.
- Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
- Send and receive data using the read() and write() system calls.

[33] http://www.tutorialspoint.com/unix_sockets/socket_server_example.htm

Uruchamianie usług: systemd – zrównoleglenie



Uruchamianie usług: systemd – zalety[34]

- tworzenie potrzebnych gniazd we wczesnej fazie uruchamiania systemu likwiduje zależności między demonami
- aktywacja usługi via gniazdo, magistralę, sprzęt (*socket/bus/hardware activation*)
- możliwość restartowania usług bez utraty danych
- jądro odpowiedzialne za porządkowanie demonów (mechanizm obsługi kolejek)
- ładowanie modułów jądra na żądanie
- udostępnianie systemów plików via automontowanie
- brak skryptów podczas uruchamiania usług
- (hierarchiczna) kontrola procesów via podsystem cgroup (*control group*)

[34] Lennart Pettering: The Biggest Myths http://0pointer.de/blog/projects/the-biggest-myths

systemd – units[35]

- .service: A service unit describes how to manage a service or application on the server. This will include how to start or stop the service, under which circumstances it should be automatically started, and the dependency and ordering information for related software.
- .socket: A socket unit file describes a network or IPC socket, or a FIFO buffer that systemd uses for socket-based activation. These always have an associated .service file that will be started when activity is seen on the socket that this unit defines.
- .target: A target unit is used to provide synchronization points for other units when booting up or changing states. They also can be used to bring the system to a new state. Other units specify their relation to targets to become tied to the target's operations.

[35] https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files, https://www.redhat.com/en/about/videos/summit-2018-demystifying-systemd

systemd – units

- .device: A unit that describes a device that has been designated as needing systemd management by udev or the sysfs filesystem. Not all devices will have .device files. Some scenarios where .device units may be necessary are for ordering, mounting, and accessing the devices.
- .mount: This unit defines a mountpoint on the system to be managed by systemd. These are named after the mount path, with slashes changed to dashes. Entries within /etc/fstab can have units created automatically.
- .automount: An .automount unit configures a mountpoint that will be automatically mounted. These must be named after the mount point they refer to and must have a matching .mount unit to define the specifics of the mount.

systemd – units

- .path: This unit defines a path that can be used for path-based activation. By default, a .service unit of the same base name will be started when the path reaches the specified state. This uses inotify to monitor the path for changes.
- .swap: This unit describes swap space on the system. The name of these units must reflect the device or file path of the space.
- .timer: A .timer unit defines a timer that will be managed by systemd, similar to a cron job for delayed or scheduled activation. A matching unit will be started when the timer is reached.

systemd – units

- .slice: A .slice unit is associated with Linux Control Group nodes, allowing resources to be restricted or assigned to any processes associated with the slice. The name reflects its hierarchical position within the cgroup tree. Units are placed in certain slices by default depending on their type.
- .scope: Scope units are created automatically by systemd from information received from its bus interfaces. These are used to manage sets of system processes that are created externally.

  `man systemd.scope`:

  > Scope units are not configured via unit configuration files, but are only created programmatically using the bus interfaces of systemd. They are named similar to filenames. A unit whose name ends in ".scopeżefers to a scope unit. Scopes units manage a set of system processes. Unlike service units, scope units manage externally created processes, and do not fork off processes on its own.
  >
  > The main purpose of scope units is grouping worker processes of a system service for organization and for managing resources.

systemd – użycie

```
systemctl start|stop|reload|restart|condrestart name.service
service   start|stop|reload|restart|condrestart name

systemctl status|is-active name.service
service   status           name

systemctl enable|disable name.service
chkconfig name on|off

systemctl is-enabled name.service
chkconfig name

systemctl list-units [-t unit]; systemctl list-unit-files [-t unit]
ls /etc/rc.d/init.d

ls /etc/systemd/system/*.wants/name.service

systemctl daemon-reload
```

systemd – użycie

sysvinit Runlevel	systemd Target
0	runlevel9.target, poweroff.target, systemctl poweroff
1,s,single	runlevel1.target, rescue.target
2,4	runlevel2.target, runlevel4.target, multi-user.target
3	runlevel3.target, multi-user.target
5	runlevel5.target, graphical.target
6	runlevel6.target, reboot.target, systemctl reboot
emergency	emergency.target

```
systemctl isolate multi-user.target|runlevel3.target
init 3

rm /etc/systemd/system/default.target
ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target
systemctl set-default multi-user

systemctl [-f|--force] halt|reboot|poweroff

/usr/sbin/halt|poweroff|reboot --> /usr/bin/systemctl
```

Zob. *man halt, reboot, poweroff*: *These are legacy commands available for compatibility only.*

Czy systemd skraca czas uruchamiania systemu?

```
# systemd-analyze blame
         21.511s dnf-makecache.service
          2.781s systemd-udev-settle.service
          ...
           798ms rc-local.service
           622ms vboxdrv.service
           ...
           118ms upower.service
           106ms chronyd.service
           105ms named.service
           ...
             3ms dracut-shutdown.service
             3ms sys-fs-fuse-connections.mount
             2ms sys-kernel-config.mount

# systemd-analyze time
Startup finished in 1.288s (kernel) + 3.265s (initrd) + 7.821s (userspace) = 12.375s

# systemd-analyze plot > plot.svg
```

Czy systemd skraca czas uruchamiania systemu?

```
# systemd-analyze critical-chain
graphical.target @26.821s
|-multi-user.target @26.820s
  |-pmie.service @13.297s +434ms
    |-pmcd.service @12.987s +307ms
      |-network-online.target @12.983s
        |-network.target @6.096s
          |-wpa_supplicant.service @7.147s +42ms
            |-dbus-daemon.service @4.529s
              |-basic.target @4.525s
                |-sockets.target @4.525s
                  |-sssd-kcm.socket @4.524s
                    |-sysinit.target @4.473s
                      |-systemd-timesyncd.service @4.128s +344ms
                        |-systemd-tmpfiles-setup.service @4.051s +74ms
                          |-local-fs.target @4.047s
                            |-run-user-1000.mount @13.521s
                              |-local-fs-pre.target @3.645s
                                |-lvm2-monitor.service @847ms +1.443s
                                  |-lvm2-lvmetad.socket @846ms
                                    |-system.slice
                                      |--.slice
```

systemd: cgroups[36]

```
# lssubsys -am
cpuset /sys/fs/cgroup/cpuset
cpu,cpuacct /sys/fs/cgroup/cpu,cpuacct
blkio /sys/fs/cgroup/blkio
memory /sys/fs/cgroup/memory
devices /sys/fs/cgroup/devices
freezer /sys/fs/cgroup/freezer
net_cls,net_prio /sys/fs/cgroup/net_cls,net_prio
perf_event /sys/fs/cgroup/perf_event
hugetlb /sys/fs/cgroup/hugetlb
pids /sys/fs/cgroup/pids

# mount | grep cgroup
# lscgroup
# systemd-cgtop
# systemd-cgls

# systemctl [--runtime] set-property stress-1cpu.service CPUQuota=20%
# systemctl [--runtime] set-property user-1001.slice CPUAccounting=1 CPUQuota=50%
# systemctl show user-1001.slice
```

[36] http://lwn.net/Articles/679786/ www.freedesktop.org/wiki/Software/systemd/

systemd: configuracja usług

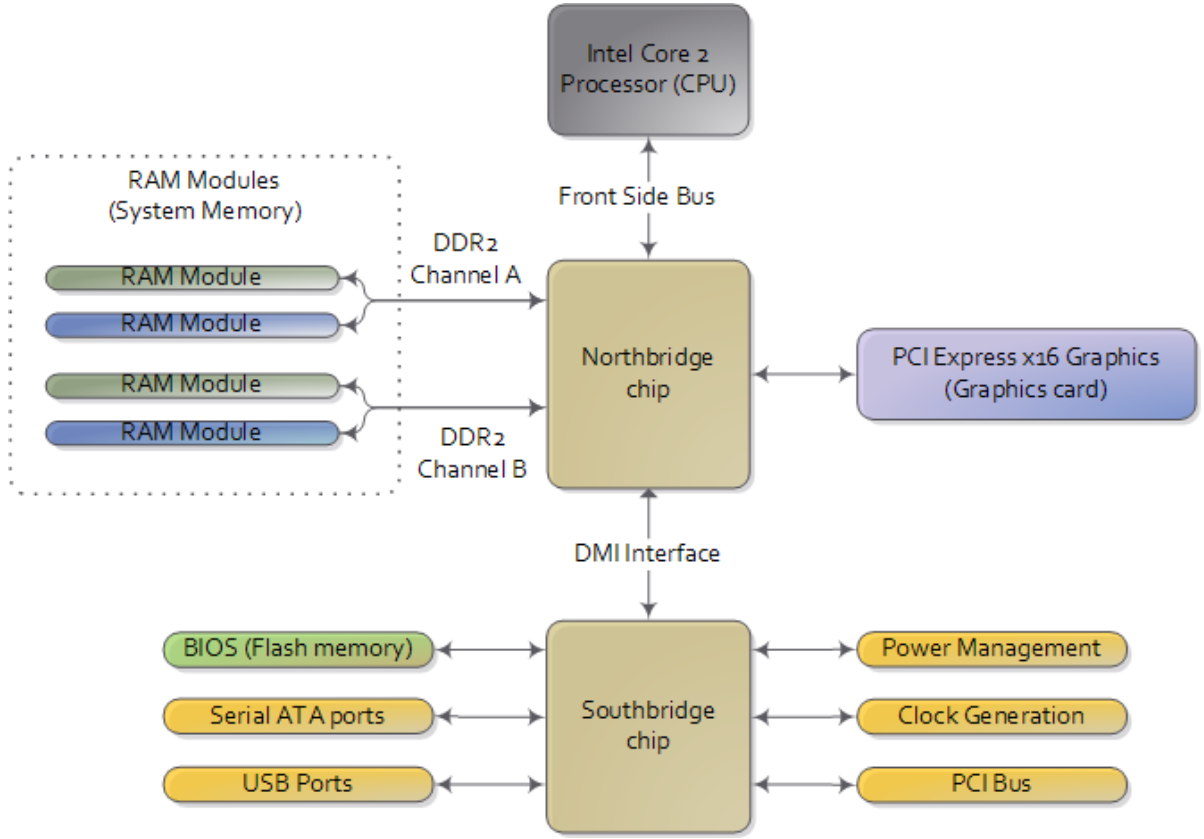
/etc/systemd/system/vmstat.service

```
[Unit]
Description=vmstat

[Service]
Type=notify
Environment=LANG=C
ExecStart=/usr/bin/vmstat 2
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGKILL
PrivateTmp=true
Restart=always
CPUAccounting=1
MemoryAccounting=1
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=vmstat-2

[Install]
WantedBy=multi-user.target
```

/etc/rsyslog.d/vmstat.conf

```
if $programname == 'vmstat-2' \
                then /tmp/vmstat.log
if $programname == 'vmstat-2' \
                then stop
```

Architektura płyty głównej[37]



[37] G. Duarte http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map

Magistrala PCI

PCI (*Peripheral Component Interconnect*) interfejs komponentów peryferyjnych:

- 32/64-bitowa szyna rozszerzeń dla komputerów zgodnych z IBM PC oraz Macintosh
- opracowana przez firmę Intel w 1992 r. (mikroprocesor Pentium)
- obsługuje standard podłącz i używaj (PnP, *Plug and Play*)
- szybkość transferu danych (MB/s=10^6 B/s, GB/s=10^9 B/s)

PCI32	33.33 MHz =	133.3 MB/s	66.66MHz = 266.6 MB/s
PCI64	33.33 MHz =	266.6 MB/s	66.66MHz = 533.3 MB/s
PCI-X	133.32 MHz =	1066.4 MB/s	266.64MHz = 2133.1 MB/s
PCIe 1.0 (×1)	2.50 GHz	250.0 MB/s	
PCIe 1.0 (×16)		4.0 GB/s	
PCIe 2.0 (×16)	5.00 GHz	8.0 GB/s	
PCIe 3.0 (×16)	8.00 GHz	15.754 GB/s	
PCIe 4.0 (×16)	16.00 GHz	31.508 GB/s	

Magistrala przyszłości – OpenCAPI (2016-10-15)[38]

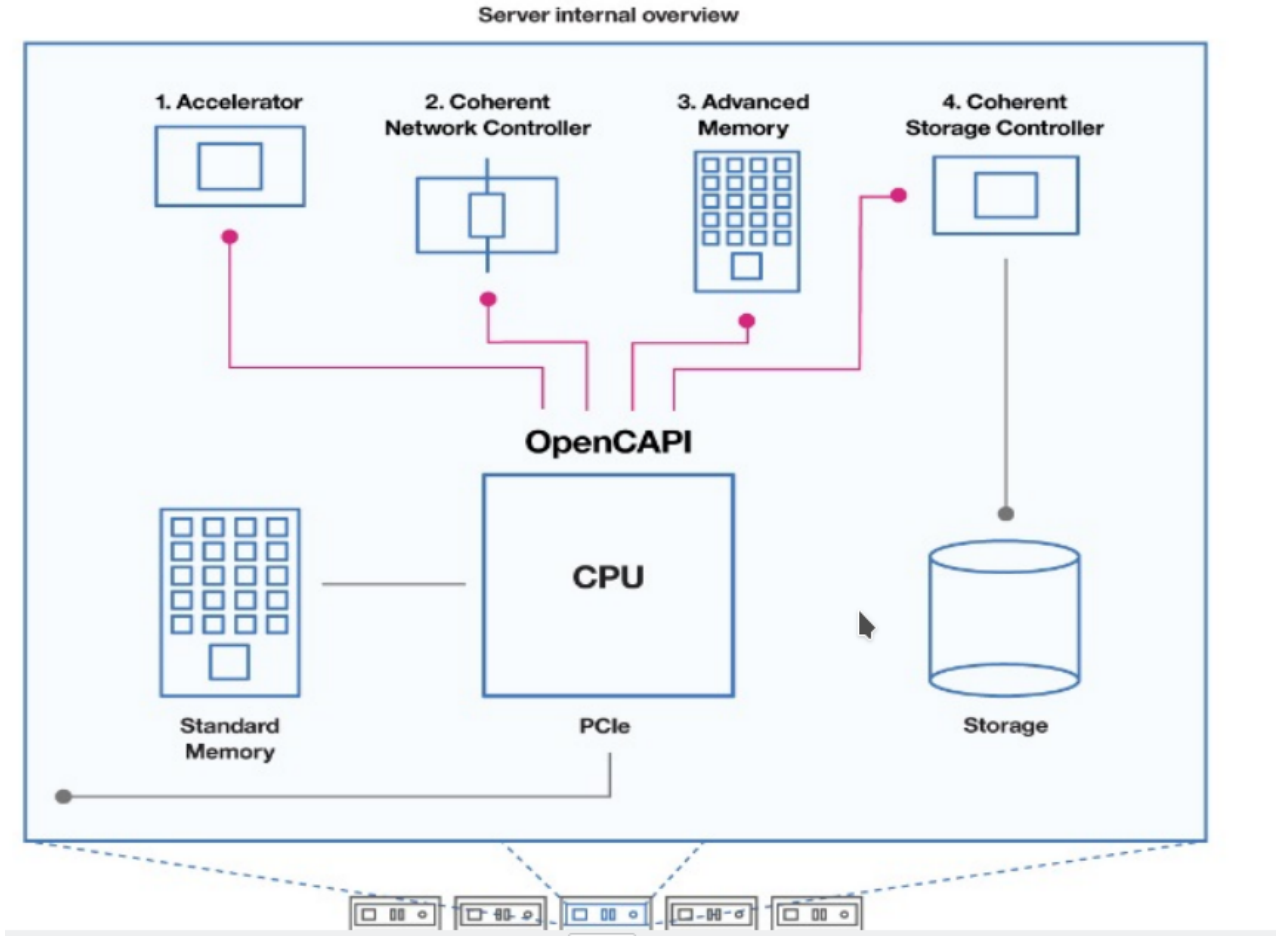
IBM has unveiled OpenCAPI, an open-standard, high-speed bus interface for connecting devices in servers. The announcement coincides with the formation of a consortium of the same name that will manage the new standard, and which initially includes tech heavyweights Hewlett Packard Enterprise (HPE), Dell EMC, NVIDIA, Mellanox, Micron, Xilinx, and Google. The first OpenCAPI-supported devices and servers are expected to show up in 2017.

Unlike CAPI (*Coherent Accelerator Processor Interface*), IBM's coherent memory communication protocol that rode on top of PCI Express (PCIe), OpenCAPI is both a new hardware interface and a new protocol, and would obviate the need for PCIe on the motherboard if all the chips were OpenCAPI compliant. The general idea is the same as IBM's original CAPI: to be able to link together processors, coprocessors, network adapters, flash modules, and other devices in the server as equal peers, and which can directly access system memory in the same manner as a CPU.

Zob. https://www.top500.org/news/tech-companies-sign-on-to-new-standard-for-high-performance-server-bus/

[38] https://opencapi.org

Heterogeniczna architektura OpenCAPI[39]



[39] https://www.slideshare.net/insideHPC/open-capi-a-new-standard-for-high-performance-attachment-of-memory-acceleration-and-networks

Uniwersalna szyna szeregowa (USB, *Universal Serial Bus*):

- standard szyny zewnętrznej do podłączania do komputera do 127 urządzeń peryferyjnych (jeden IRQ)
- standard opracowany w 1995 r. wspólnie przez wiodących producentów sprzętu komputerowego i telekomunikacyjnego (Compaq, DEC, IBM, Intel, Microsoft, NEC, Northern Telecom, Philips)
- szybkość: 12 Mb/s i 1.5 Mb/s (USB 1.1), 480 Mb/s (USB 2.0), 4.8 Gb/s (USB 3.0)
- długość kabla: USB 1.1 – 3 m, USB 2.0 – 5 m
- transmisja w trybie półdupleksu (USB 1.1 i 2.0) i pełnego dupleksu (USB 3.0)
- wyprowadzenie funkcji PnP poza komputer
- łatwość rozmnażania portów i wydłużania połączenia poprzez zastosowanie maksymalnie pięciu koncentratorów (*USB hubs*)
- USB łączy drukarki, skanery, kamery wideo, dyski, stacje dyskietek, klawiatury, myszy, joysticki, telefony, modemy, napędy CD-ROM, napędy taśmowe, urządzenia wideo MPEG-2, *data gloves*, digitalizatory (*digitizers*), itp.

Wyświetlanie informacji o urządzeniach: lspci i lsusb

We współczesnych systemach urządzenia są obsługiwane przez magistrale PCI i USB.

Komendy **lspci** i **lsusb** służą do wypisania zarejestrowanych przez system urządzeń podłączonych do tych magistral.

```
# lspci -D

0000:00:00.0 Host bridge: Intel Corporation Mobile 945GM/PM/GMS, 943/940GML and 945GT Express Memory Controller Hub (re
0000:00:1c.0 PCI bridge: Intel Corporation N10/ICH 7 Family PCI Express Port 1 (rev 01)
0000:00:1f.0 ISA bridge: Intel Corporation 82801GBM (ICH7-M) LPC Interface Bridge (rev 01)
0000:00:1f.2 IDE interface: Intel Corporation 82801GBM/GHM (ICH7 Family) SATA IDE Controller (rev 01)
0000:09:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5752 Gigabit Ethernet PCI Express (rev 02)
0000:0c:00.0 Network controller: Intel Corporation PRO/Wireless 3945ABG [Golan] Network Connection (rev 02)

# lspci -nn
00:1f.2 IDE interface [0101]: Intel Corporation 82801GBM/GHM (ICH7 Family) \
                                               SATA IDE Controller [8086:27c4] (rev 01)
09:00.0 Ethernet controller [0200]: Broadcom Corporation NetXtreme BCM5752 \
                                               Gigabit Ethernet PCI Express [14e4:1600] (rev 02)
0c:00.0 Network controller [0280]: Intel Corporation PRO/Wireless 3945ABG [Golan] \
                                               Network Connection [8086:4222] (rev 02)
```

Pojemność numeracji: `domain(16):bus(8):device(5).function(3)`

Wyświetlanie informacji o urządzeniach: lspci

Powiązanie pomiędzy identyfikatorami urządzeń PCI, a ich pełnymi nazwami znajdują się w pliku /usr/share/hwdata/pci.ids.

```
# vendor  vendor_name
#       device  device_name                             <-- single tab
#               subvendor subdevice  subsystem_name     <-- two tabs
14e4  Broadcom Corporation
...
        1600  NetXtreme BCM5752 Gigabit Ethernet PCI Express
                1028 01c1  Precision 490
                1028 01c2  Latitude D620
                103c 3015  PCIe LAN on Motherboard
                107b 5048  E4500 Onboard
...
8086  Intel Corporation
...
        4220  PRO/Wireless 2200BG [Calexico2] Network Connection
                103c 12f6  Compaq nw8240/nx8220
                8086 2712  IBM ThinkPad R50e
                8086 2721  Dell B130 laptop integrated WLAN
```

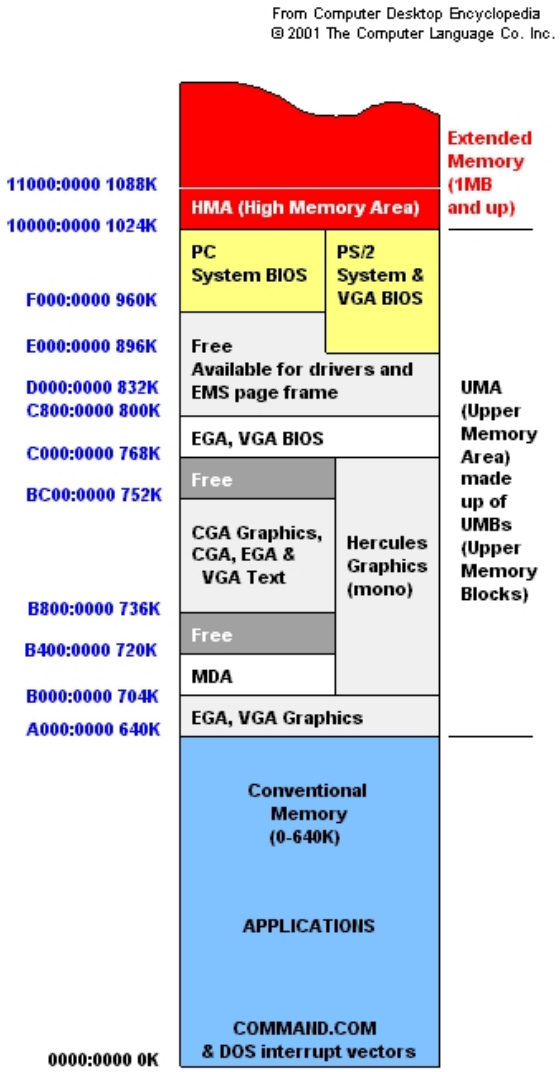
Wyświetlanie informacji o urządzeniach: lspci

```
# lspci  -v -s 09:00.0
09:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5752 Gigabit Ethernet PCI
Express (rev 02)
        Subsystem: Dell Device 01cc
        Flags: bus master, fast devsel, latency 0, IRQ 30
        Memory at efcf0000 (64-bit, non-prefetchable) [size=64K]
        Expansion ROM at <ignored> [disabled]
        Capabilities: <access denied>
        Kernel driver in use: tg3
        Kernel modules: tg3

# lspci -v -s 0c:00.0

0c:00.0 Network controller: Intel Corporation PRO/Wireless 3945ABG [Golan] Network
Connection (rev 02)
        Subsystem: Intel Corporation Device 1021
        Flags: bus master, fast devsel, latency 0, IRQ 29
        Memory at efdff000 (32-bit, non-prefetchable) [size=4K]
        Capabilities: <access denied>
        Kernel driver in use: iwl3945
        Kernel modules: iwl3945
```

Obsługa urządzeń

- Sterownik urządzenia (*device controller*) związany jest z konkretnym urządzeniem i rozporządza lokalnym buforem i zbiorem rejestrów o specjalnym przeznaczeniu. Odpowiada za przesyłanie danych między urządzeniem zewnętrznym, a własnym buforem.
- Moduł sterujący/obsługi urządzenia (*driver*) jest odpowiedzialny od strony systemu operacyjnego za komunikację ze sterownikiem urządzenia.

Zob.: `/proc/iomem`, `/proc/ioports`




```
       0.0000 M        0.0039 M  Reserved
       0.0039 M        0.6133 M  System RAM
       0.6133 M        0.6250 M  Reserved
       0.6250 M        0.7500 M  PCI Bus 0000:00
       0.7500 M        0.8076 M  Video ROM
       0.8125 M        0.8281 M  pnp 00:00
       0.8281 M        0.8437 M  pnp 00:00
       0.8438 M        0.8594 M  pnp 00:00
       0.8594 M        0.8750 M  pnp 00:00
       0.8750 M        1.0000 M  Reserved
       0.9375 M        1.0000 M  System ROM
       1.0000 M     2785.2656 M  System RAM
     352.0000 M      360.6260 M  Kernel code
     360.6260 M      368.3481 M  Kernel data
     370.6484 M      372.5781 M  Kernel bss
    2785.2656 M     3000.4531 M  Reserved
    3000.4531 M     3000.7656 M  ACPI Non-volatile Storage
    3000.7656 M     3020.4648 M  Reserved
    3020.4648 M     3022.4961 M  ACPI Non-volatile Storage
    3022.4961 M     3022.9961 M  ACPI Tables
...
    4077.2500 M     4077.2969 M  PCI Bus 0000:00
    4077.2500 M     4077.2695 M  TPM
    4077.2695 M     4077.2969 M  pnp 00:01
    4078.0000 M     4078.0039 M  Local APIC
    4078.0000 M     4078.0039 M  Reserved
    4092.0000 M     4096.0000 M  Reserved
```

Wyświetlanie informacji o urządzeniach: lspci

W jaki sposób lspci wykrywa urządzenia podłączone do magistrali PCI?

```
# strace -e trace=open lspci
open("/usr/share/hwdata/usb.ids", O_RDONLY) = 3
...
open("/sys/bus/pci/devices/0000:09:00.0/resource", O_RDONLY) = 4
open("/sys/bus/pci/devices/0000:09:00.0/irq", O_RDONLY) = 4
open("/sys/bus/pci/devices/0000:09:00.0/vendor", O_RDONLY) = 4
open("/sys/bus/pci/devices/0000:09:00.0/device", O_RDONLY) = 4
open("/sys/bus/pci/devices/0000:09:00.0/class", O_RDONLY) = 4
open("/sys/bus/pci/devices/0000:09:00.0/config", O_RDONLY) = 3
...
open("/usr/share/hwdata/pci.ids", O_RDONLY) = 4
```

Zawartość najważniejszych plików

```
irq: 30
vendor: 0x14e4
device: 0x1600
class:  0x020000
```

Wyświetlanie informacji o urządzeniach: lsusb

```
# lsusb
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 008: ID 046d:c052 Logitech, Inc.
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 003: ID 0b97:7762 O2 Micro, Inc. Oz776 SmartCard Reader
Bus 003 Device 002: ID 0b97:7761 O2 Micro, Inc. Oz776 1.1 Hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 413c:a005 Dell Computer Corp. Internal 2.0 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Powiązanie pomiędzy identyfikatorami urządzeń USB, a ich pełnymi nazwami znajdują się w pliku /usr/share/hwdata/usb.ids.

Wyświetlanie informacji o urządzeniach: lsusb

```
# lsusb -v -s 04:008

Bus 004 Device 008: ID 046d:c052 Logitech, Inc.
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x046d Logitech, Inc.
  idProduct          0xc052
  bcdDevice           27.20
  iManufacturer           1 Logitech
  iProduct                2 USB Optical Mouse
  iSerial                 0
  bNumConfigurations      1
  ...
  ...
  ...
```

udev – oczekiwania

- odejście od statycznej obsługi urządzeń, jądro może usuwać i dodawać (prawie) każde urządzenie w trakcie pracy systemu
- każdą zmianę stanu urządzenia trzeba móc przekazać do przestrzeni użytkownika; wyświetlanie informacji o urządzeniach w trybie użytkownika
- udostępnianie informacji o urządzeniach przez jądro (katalog /sys)
- po podłączeniu i wykryciu urządzenie musi być skonfigurowane
- śledzenie stanu urządzenia; powiadamianie użytkowników (pewnych) urządzeń o zmianie ich stanu

```
NAME
       udev - Dynamic device management

DESCRIPTION
       udev supplies the system software with device events, manages permissions of device
       nodes and may create additional symlinks in the /dev directory, or renames network
       interfaces. The kernel usually just assigns unpredictable device names based on the
       order of discovery. Meaningful symlinks or network device names provide a way to
       reliably identify devices based on their properties or current configuration.

       The udev daemon, systemd-udevd.service(8), receives device uevents directly from
       the kernel whenever a device is added or removed from the system, or it changes its
       state. When udev receives a device event, it matches its configured set of rules
       against various device attributes to identify the device. Rules that match may
       provide additional device information to be stored in the udev database or to be
       used to create meaningful symlink names.

       All device information udev processes is stored in the udev database and sent out
       to possible event subscribers. Access to all stored data and the event sources is
       provided by the library libudev.

CONFIGURATION
       udev configuration files are placed in /etc/udev/ and /lib/udev/.
```

udev – własności[40]

- od wersji jądra 2.5 wszystkie urządzenia fizyczne i wirtualne są widoczne w hierarchiczny sposób poprzez sysfs.
- w przestrzeni użytkownika można odnotowywać zdarzenie, że jakieś urządzenie zostało dodane lub ujęte, możliwa staje się dynamiczna obsługa urządzeń z przestrzeni (w trybie) użytkownika
- obsługa urządzeń jest realizowana przez podsystem udev (dawniej devfs); od 4/2012 drzewo udev włączone do systemd

[40] http://w3.linux-magazine.com/issue/71/Dynamic_Device_Management_in%20Udev.pdf, http://doc.opensuse.org/documentation/html/openSUSE/opensuse-reference/cha.udev.html

udev – zalety

- zachowanie stałej nazwy urządzenia podczas jego wędrówki po drzewie urządzeń
- powiadamianie zewnętrznych systemów o zmianie urządzenia
- elastyczny schemat nazywania urządzeń; wyniesienie polityki nazewniczej poza jądro
- możliwość stosowania przez jądro dynamicznych dużych/małych (głównych/drugorzędnych) numerów urządzeń

udev i netlink

Dlaczego komunikacja udev-jądro nie wykorzystuje mechanizmów syscall, ioctl, /proc, ale podsystem gniazd netlink (AF_NETLINK, RFC 3549)?

Zalety:

- pełen dupleks
- obsługa w trybie asynchronicznym
- jądro może inicjować sesje
- łatwe rozgłaszanie zdarzeń jądra poprzez rozgłoszenia grupowe
- gniazda typu BSD
- łatwość użycia i dodawania nowych cech

udev: katalog /dev

- zawiera węzły (pliki urządzeń), poprzez które można dotrzeć do urządzeń w jądrze
- udev dba, żeby zawartość katalogu odpowiadała aktualnej sytuacji
- każde urządzenie jest opisane jednym plikiem; odłączenie urządzenia powoduje usunięcie węzła
- /dev ma charakter tymczasowy, pliki pojawiają/znikają po włączeniu/-wyłączeniu systemu (w najnowszych systemach /dev jest realizowany jako system plików devtmpfs)

Katalog /dev nie zawiera plików (typowych) urządzeń sieciowych!

udev: urządzenia blokowe i znakowe

Urządzenia blokowe i znakowe są traktowane przez system operacyjny jak specjalnego rodzaju pliki definiowane przez typ urządzenia (blokowe, znakowe) oraz dwa numery: główny/podrzędny (duży/mały) (*major/minor number*).

Wszystkie duże i małe numery urządzeń znakowych, blokowych i in. mają przypisane odpowiednie nazwy (zob. Documentation/devices.txt).

Tworzenie urządzeń znakowych/blokowych:

```
mknod        /dev/null c 1 3
mknod -m 666 /dev/hda  b 3 0
mknod -m 666 /dev/hda1 b 3 1
```

Urządzenia blokowe: numeracja

```
  3 block       First MFM, RLL and IDE hard disk/CD-ROM interface
                  0 = /dev/hda          Master: whole disk (or CD-ROM)
                 64 = /dev/hdb          Slave: whole disk (or CD-ROM)
                For partitions, add to the whole disk device number:
                  0 = /dev/hd?          Whole disk
                  1 = /dev/hd?1         First partition
                  2 = /dev/hd?2         Second partition
                     ...
                 63 = /dev/hd?63        63rd partition

  8 block       SCSI disk devices (0-15)
                  0 = /dev/sda          First SCSI disk whole disk
                 16 = /dev/sdb          Second SCSI disk whole disk
                 32 = /dev/sdc          Third SCSI disk whole disk
                     ...
                240 = /dev/sdp          Sixteenth SCSI disk whole disk

8,65-71,128-135: 16 x 16  = 256 SCSI disks

# ls -la /dev/sdb*
brw-rw---- 1 root disk 8, 16 03-14 11:59 /dev/sdb
brw-rw---- 1 root disk 8, 17 03-14 11:59 /dev/sdb1
```

Linux 2.6: duży numer – 12 bitów, mały numer – 20 bitów

Nowsze wersje jądra: dynamiczny przydział numerów (udev)

Interfejs SCSI[41]

```
# strace -e trace=open lsscsi
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
open("/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
open("/sys/bus/scsi/devices/0:0:0:0/type", O_RDONLY) = 3
open("/sys/bus/scsi/devices/0:0:0:0/vendor", O_RDONLY) = 3
open("/sys/bus/scsi/devices/0:0:0:0/model", O_RDONLY) = 3
open("/sys/bus/scsi/devices/0:0:0:0/rev", O_RDONLY) = 3
open("/sys/devices/pci0000:00/0000:00:1f.2/ata1/host0/target0:0:0/0:0:0:0/block/sda/dev",O_RDONLY) =
[0:0:0:0]    disk    ATA      ST9320423AS      0002  /dev/sda
open("/sys/bus/scsi/devices/1:0:0:0/type", O_RDONLY) = 3
open("/sys/bus/scsi/devices/1:0:0:0/vendor", O_RDONLY) = 3
open("/sys/bus/scsi/devices/1:0:0:0/model", O_RDONLY) = 3
open("/sys/bus/scsi/devices/1:0:0:0/rev", O_RDONLY) = 3
open("/sys/devices/pci0000:00/0000:00:1f.2/ata2/host1/target1:0:0/1:0:0:0/block/sr0/dev",O_RDONLY) =
[1:0:0:0]    cd/dvd  HL-DT-ST DVD-ROM GDR8084N 1.02  /dev/sr0
```

[41] http://sg.danny.cz/scsi/lsscsi.html, https://www.cdsi.us.com/there-is-no-n-in-lun/, https://https://www.cdsi.us.com/why-lun-is-not-a-lun/, `man lsscsi`

Interfejs SCSI

```
# lsscsi
[0:0:0:0]    cd/dvd  MATSHITA CD-RW  CW-8124   DZ13  /dev/sr0
[2:0:0:0]    disk    Areca    ARC-1880-VOL#000 R001  /dev/sda
[2:0:16:0]   process Areca    RAID controller  R001  -
[5:0:0:0]    disk    SEAGATE  ST973401LSUN72G  0556  /dev/sdb
[5:0:1:0]    disk    SEAGATE  ST973401LSUN72G  0556  /dev/sdc
[5:0:2:0]    disk    SEAGATE  ST973401LSUN72G  0556  /dev/sdd
[5:0:3:0]    disk    FUJITSU  MAY2073RC        5204  /dev/sde
[6:0:0:0]    cd/dvd  AMI      Virtual CDROM    1.00  /dev/sr1
[7:0:0:0]    disk    AMI      Virtual Floppy   1.00  /dev/sdf

# lsscsi -t
[0:0:0:0]    cd/dvd  ata:                              /dev/sr0
[2:0:0:0]    disk                                       /dev/sda
[2:0:16:0]   process                                    -
[5:0:0:0]    disk    sas:0x5000c500031b3295             /dev/sdb
[5:0:1:0]    disk    sas:0x5000c500031b30d5             /dev/sdc
[5:0:2:0]    disk    sas:0x5000c50002e03421             /dev/sdd
[5:0:3:0]    disk    sas:0x500000e017b6f0d2             /dev/sde
[6:0:0:0]    cd/dvd  usb: 2-4:1.0                       /dev/sr1
[7:0:0:0]    disk    usb: 2-5:1.0                       /dev/sdf
```

Interfejs SCSI

```
[root@scobie ~]# lsscsi -t
[0:0:0:0]    disk    sata:5000cca77ecd1787           /dev/sda
[2:0:0:0]    disk    sata:5001b44a5ce85187           /dev/sdb

[root@scobie ~]# smartctl -a /dev/sda
...
Model Family:     Hitachi/HGST Travelstar Z7K500
Device Model:     HGST HTS725050A7E630
Serial Number:    TF755BWH0XTU4S
LU WWN Device Id: 5 000cca 77ecd1787
...

[root@scobie ~]# smartctl -a /dev/sdb
...
Model Family:     SanDisk based SSDs
Device Model:     SanDisk SSD U110 16GB
Serial Number:    134508402055
LU WWN Device Id: 5 001b44 a5ce85187
...
```

Interfejs SCSI

```
[root@scobie ~]# ls -la /dev/disk/by-id
...
lrwxrwxrwx 1 ... lvm-pv-uuid-Ht8kr1-m4Vt-Ux1W-fflL-ZEfQ-s46V-ZAnqp3 -> ../../sda5
lrwxrwxrwx 1 ... wwn-0x5000cca77ecd1787 -> ../../sda
lrwxrwxrwx 1 ... wwn-0x5000cca77ecd1787-part1 -> ../../sda1
lrwxrwxrwx 1 ... wwn-0x5000cca77ecd1787-part2 -> ../../sda2
lrwxrwxrwx 1 ... wwn-0x5000cca77ecd1787-part3 -> ../../sda3
lrwxrwxrwx 1 ... wwn-0x5000cca77ecd1787-part4 -> ../../sda4
lrwxrwxrwx 1 ... wwn-0x5000cca77ecd1787-part5 -> ../../sda5
lrwxrwxrwx 1 ... wwn-0x5001b44a5ce85187 -> ../../sdb
lrwxrwxrwx 1 ... wwn-0x5001b44a5ce85187-part1 -> ../../sdb1
...
```

Interfejs SCSI

Oznaczenia urządzeń SCSI:

```
/sys/devices/pci...00:1f.2/host0/target0:0:0/0:0:0:0/scsi_device/0:0:0:0
/sys/class/scsi_device/0:0:0:0/device/block/sda
/sys/class/scsi_device/h:c:i:l/device/block/sda
```

```
# lsscsi -txx
[0:0:0:0x0000000000000000]  disk    sata:5002538d422a5d34          /dev/sda
[2:0:0:0x0000000000000000]  disk    sata:5001b44a5ce85187          /dev/sdb
[3:0:0:0x0000000000000000]  disk    usb:2-2:1.0                    /dev/sdc
[3:0:0:0x0001000000000000]  disk    usb:2-2:1.0                    /dev/sdd
[3:0:0:0x0002000000000000]  disk    usb:2-2:1.0                    /dev/sde
[3:0:0:0x0003000000000000]  disk    usb:2-2:1.0                    /dev/sdf
```

SCSI/devfs

h=SCSI addapter number/host number

c=channel/bus number (zawsze zero!)

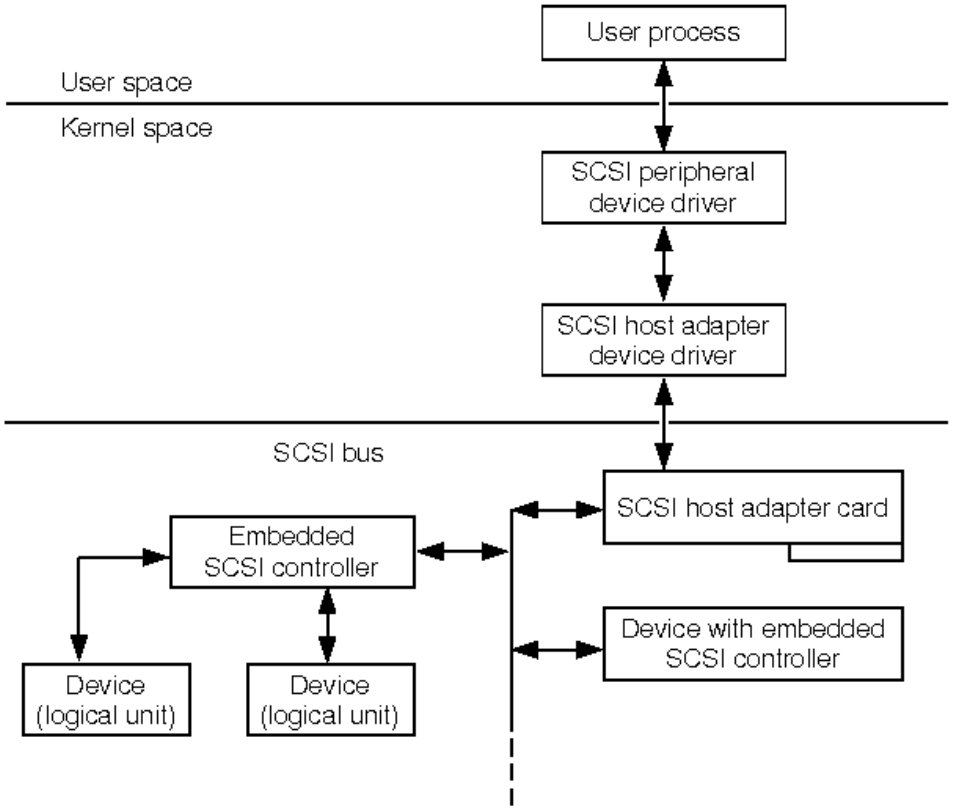
i=id number/target

l=lun/lun (logical unit number)

Interfejs SCSI: podstawowe pojęcia

- *host adapter, host bus adapter (HBA)* (adapter hosta) – karta, która zamienia niezależne od komputera komunikaty protokołu SCSI na informacje właściwe dla sprzętu komputera
- *SCSI initiators* – urządzenie, które może inicjować operacje I/O (jest nim HBA); typowa konfiguracja: jeden inicjator i wiele celów
- *SCSI target* – odnosi się do sterownika (osadzonego) zewnętrznego urządzenia SCSI, które jest adresowane; identyfikator celu (*target ID*) jest adresem osadzonego sterownika.
- *Logical Unit (LU)* – jednostka logiczna, która jest używana do adresowania operacji I/O do niektórych urządzeń wspierających różne media (logiczne dyski, przewijaki taśmowe)
- *SCSI bus* (magistrala SCSI) – kabel, który łączy adapter hosta z szeregiem sterowników SCSI powiązanych ze swoimi urządzeniami

Interfejs SCSI[42]



[42] http://uw714doc.sco.com/en/HDK_osdi/osdi_intro_top.html

Interfejsy SCSI i USB

Klucz USB w dwóch różnych gniazdach:

```
# lsscsi -t
[3:0:0:0]    disk    usb:2-2:1.0                       /dev/sdc
/dev/disk/by-path/pci-0000:00:14.0-usb-0:2:1.0-scsi-0:0:0:0 -> ../../sdc
/sys/devices/pci0000:00/0000:00:14.0/usb2/2-2/2-2:1.0/host3/target3:0:0/3:0:0:0/block/sdc
/sys/devices/pci0000:00/0000:00:14.0/usb2/b-p/b-p:c.i/host3/target3:0:0/3:0:0:0/block/sdc

[3:0:0:0]    disk    usb:2-3:1.0                       /dev/sdc
/dev/disk/by-path/pci-0000:00:14.0-usb-0:3:1.0-scsi-0:0:0:0 -> ../../sdc
/sys/devices/pci0000:00/0000:00:14.0/usb2/2-3/2-3:1.0/host3/target3:0:0/3:0:0:0/block/sdc
```

Klucz USB w gniaździe huba:

```
# lsscsi -t
[51:0:0:0]   disk    usb: 3-2.2:1.0                    /dev/sdb
/sys/class/scsi_device/51:0:0:0/device/block/sdb/sdb1
/sys/devices/pci...00:1d.1/usb3/3-2/3-2.2/3-2.2:1.0/host51/target51:0:0/51:0:0:0/block/sdb/sdb1
/sys/devices/pci...00:1d.1/usb3/b-p/b-p.P/b-p.P:c.i/host51/target51:0:0/51:0:0:0/block/sdb/sdb1
```

b=USB bus number, p=host port

P=hub port

c=configuration number, i=interface number

Interfejsy SCSI i USB

```
# lsscsi -v
[0:0:0:0]    disk    ATA      ST9320423AS      0002  /dev/sda
  dir: /sys/bus/scsi/devices/0:0:0:0  [/sys/devices/pci...:00:1f.2/ \
                                                host0/target0:0:0/0:0:0:0]
[1:0:0:0]    cd/dvd  HL-DT-ST DVD-ROM GDR8084N 1.02  /dev/sr0
  dir: /sys/bus/scsi/devices/1:0:0:0  [/sys/devices/pci...:00:1f.2/ \
                                                host1/target1:0:0/1:0:0:0]
[52:0:0:0]   disk    Generic  USB Flash Disk   2.00  /dev/sdb
  dir: /sys/bus/scsi/devices/52:0:0:0 [/sys/devices/pci...:00:1d.7/usb1/1-4/1-4:1.0/ \
                                                host52/target52:0:0/52:0:0:0]
```

W nowszych wersjach systemu udev nie ma już informacji o hubie USB:

```
root@scobie ~$ lsscsi -t
[0:0:0:0]    disk    sata:5002538d422a5d34           /dev/sda
[2:0:0:0]    disk    sata:5001b44a5ce85187           /dev/sdb
[3:0:0:0]    disk    usb:2-1:1.0                     /dev/sdc
[4:0:0:0]    disk    usb:2-2:1.0                     /dev/sdd
[4:0:0:1]    disk    usb:2-2:1.0                     /dev/sde
[4:0:0:2]    disk    usb:2-2:1.0                     /dev/sdf
[4:0:0:3]    disk    usb:2-2:1.0                     /dev/sdg
```

SCSI *rulez*[43]

- In SCSI, a lun is only unique under each target, and is not an independently unique identifier. If a computer has multiple SCSI adapters (buses), each adapter is represented by a unique ID. Therefore, specific communication with any SCSI device is accomplished using a precise address consisting of adapter ID, target ID, and lun – ASL for short.
- Before fibre channel, some SCSI vendors attempted to insert more than one bus in a single adapter by adding a parameter called "channel." The complete address then became ACSL. Although the "channel" feature was eventually dropped (and is therefore always zero), to this day almost all operating systems use the ACSL convention when addressing a device.
- This SCSI convention is deeply ingrained in all current operating systems, no matter if the connectivity is FC, SATA, SAS, or USB. All storage devices use the SCSI convention and protocol when accessing disks, even though there is hardly any real SCSI hardware present anymore. When LUNs are provisioned to hosts in an FC SAN, they must do so using the old SCSI addressing scheme. Fortunately, the mapping from LUN to SCSI address is reasonably straightforward.

[43] Wai Lam, https://www.cdsi.us.com/why-lun-is-not-a-lun/

udev + uevents

- wykrycie urządzenia powoduje udostępnienie przez jądro informacji o tym urządzeniu via sysfs
- tworzony jest oddzielny katalog z plikami zawierającymi różne atrybuty urządzenia
- dodanie/usunięcie urządzenia powoduje wysłanie przez jądro u-zdarzenia (uevent), które jest odbierane przez udev
- udev przetwarza zdarzenie wg reguł umieszczonych w /usr/lib/udev/rules.d/, /run/udev/rules.d/ i /etc/udev/rules.d/); pliki reguł muszą być postaci $*$.rules
- wszystkie pliki z regułami są sortowane i przetwarzane wg porządku leksykograficznego (katalog nie ma znaczenia)
- pliki o jednakowych nazwach zastępują siebie (pliki z /etc mają najwyższy priorytet, potem te z /run/)
- dowiązanie symboliczne /dev/null do jakiegoś pliku z regułami wyłącza je całkowicie

udev: nazwy urządzeń

Nazwy urządzeń mogą być określone w oparciu o

- etykietę lub numer seryjny
- numer urządzenia na szynie danych
- położenie w ramach topologii szyny
- nazwę stosowaną przez jądro
- wynik działania programu

udev – przydatne komendy

```
# udevadm monitor
# udevadm info --attribute-walk --path  $(udevadm info \
                                --query=path --name /dev/sd?)
# udevadm info --attribute-walk --name /dev/sd?
# udevadm control --reload-rules
# udevadm trigger --verbose [--dry-run]
# udevadm trigger --attr-match=serial=FCE7073C4301C68A \
                  --action=add|remove|change
```

```
# udev 095    --> udev 173
# udevcontrol --> udevadm control
# udevinfo    --> udevadm info
# udevmonitor --> udevadm monitor
```

Zob. http://jkob.fizyka.umk.pl/labul/scripts/udevctl

udev – reguły

```
# find /sys -name em1
/sys/class/net/em1
/sys/devices/pci0000:00/0000:00:19.0/net/em1

# udevadm  info --attribute-walk --path /sys/class/net/em1
  ...
  looking at device '/devices/pci0000:00/0000:00:19.0/net/em1':
    KERNEL=="em1"
    SUBSYSTEM=="net"
    ...
    ATTR{address}=="28:d2:44:56:0f:d5"
    ...
```

Zmiana nazwy interfejsu z em1 na eth0 wymaga dodania do pliku /etc/udev/rules.d/70-net.rules wiersza

```
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="28:d2:44:56:0f:d5", \
                                                        NAME:="eth0"
```

udev – ładowanie modułów[44]

- sterownik magistrali wyszukuje znajdujące się na niej urządzenia
- dla każdego znalezionego urządzenia tworzona jest odpowiednia struktura danych jądra i jest wysyłane u-zdarzenie do demona udev
- każde z urządzeń identyfikuje się poprzez odpowiednio sformatowany identyfikator, np.:

```
# cat /sys/class/net/em1/device/modalias
pci:v00008086d00001559sv000017AAsd00002214bc02sc00i00
```

- każdy z modułów jądra zawiera informacje o urządzeniach, które obsługuje (zob. `/lib/modules/$(uname -r)/modules.alias`.[45]

Ładowanie modułów ulega uproszczeniu:

```
# MODALIAS=pci:v00008086d00001559sv000017AAsd00002214bc02sc00i00
# modprobe $MODALIAS
```

```
DRIVER!="?*", ENV{MODALIAS}=="?*", RUN{builtin}="kmod load $env{MODALIAS}"
```

[44] https://doc.opensuse.org/documentation/leap/reference/html/book.opensuse.reference/cha.udev.html
[45] Do tworzenia tego pliku służy komenda depmod.

udev – reguły[46]

Przykłady:

- /etc/udev/rules.d/10-usbkey.rules

```
SUBSYSTEMS=="usb", ATTRS{serial}=="FCE7073C4301C68A", KERNEL=="sd?1", SYMLINK+="usb256m"
```

- /etc/udev/rules.d/10-usbkey.rules

```
SUBSYSTEMS=="usb", ATTRS{serial}=="FCE7073C4301C68A", KERNEL=="sd?1", \
                    SYMLINK+="usb256m", GROUP="usbstorage", \
                    RUN+="/usr/bin/at -M -f /home/jkob/bin/backup2usbctl now"
```

- /etc/udev/rules.d/70-net.rules

```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:18:de:39:06:9b", \
                                                KERNEL=="wlan*", NAME:="wlan0"

SUBSYSTEMS=="pci", PROGRAM="/sbin/biosdevname --smbios 2.6 --policy physical -i %k", \
                                        NAME="%c"  OPTIONS+="string_escape=replace"
```

[46] http://www.reactivated.net/writing_udev_rules.html

Interfejsy sieciowe: reguły wyznaczania nazw

systemd używa domyślnie następującego schematu przydzielania nazw interfejsom sieciowym (przy założeniu, że obsługa podsystemu biosdevname jest wyłączona, tj. biosdevname=0[47]):

1. nazwy urobione wg informacji pozyskanych z oprogramowania firmowego lub BIOS-u (np. eno1); w przeciwnym razie nazwy wg schematu 2
2. nazwy urobione wg informacji pozyskanych z oprogramowania firmowego lub BIOS-u o położeniu urządzenia na szynie PCI (np. ens1); w przeciwnym razie nazwy wg schematu 3
3. nazwy urobione wg informacji o fizycznym położeniu NIC na płycie głównej (np. enp2s0); w przeciwnym razie nazwy wg schematu 5
4. nazwy urobione wg adresu MAC (np. enx78e7d1ea46da) są dostępne na życzenie użytkownika
5. tradycyjny, nieprzewidywalny schemat nazewniczy, jeśli inne zawiodły (np. eth0)

[47] Biosdevname jest pomocniczym programem użytkowym dla udev, opracowanym przez firmę Dell (na licencji GPL). Dostarcza on spójnego mechanizmu nadawania nazw urządzeniom sieciowym na podstawie ich fizycznego położenia udostępnianego przez BIOS. Debian go nie używa!

Interfejsy sieciowe: reguły wyznaczania nazw

Nazwy interfejsów sieciowych są zależne od ich typu i poprzedzone prefiksami:

- `en` – Ethernet
- `wl` – wireless LAN (WLAN)
- `ww` – wireless wide area network (WWAN)

Pozostała część nazwy zależy od typu urządzenia:

Format	Description
o<index>	on-board device index number
s<slot>[f<function>][d<dev_id>]	hotplug slot index number
x<MAC>	MAC address
p<bus>s<slot>[f<function>][d<dev_id>]	PCI geographical location
p<bus>s<slot>[f<function>][u<port>]\ [..][c<config>][i<interface>]	USB port number chain

Interfejsy sieciowe: przykłady wyznaczania nazw

System	`device`	`biosdevname -i device`	lspci
Dell	em1	em1	
	p2p1	p2p1	
Supermicro	eno1	em1	
SunFire X4200	enp134s1f0	em1	86:01.0 ... Gigabit ...
	enp134s1f1	em2	86:01.1 ... Gigabit ...
	enp132s0	p1p1	84:00.0 ... 10-Gigabit ...
SunFire X4440	enp3s0	p1p1	03:00.0 ... 10-Gigabit ...
HP-ProLiant DL180 Gen9	ens2f0	p2p1	02:00.0 ... 10-Gigabit ...
	ens2f1	p2p2	02:00.1 ... 10-Gigabit ...

Interfejsy sieciowe: nazwy wg biosdevname

Urządzenie	Stara nazwa	Nowa nazwa	Przykład
Embedded NIC	`eth[0123...]`	`em[1234...]`	
PCI NIC	`eth[0123...]`	`p<slot>p<ethernet port>`	`p2p1, p2p2`

- Schemat stosowany domyślnie dla systemów firmy Dell oraz gdy biosdevname został *explicite* włączony.
- Jeśli schemat jest włączony, to można go wyłączyć przy bootowaniu używając `biosdevname=0 net.ifnames=0` jako argumentu jądra.
- Domyślna polityka przypisywania nazw: *physical*. Maszyny wirtualne: biosdevname zawsze wyłączony!
- nazwy zależne od wersji SMBIOS

```
# /sbin/biosdevname --policy physical -i eth0
p2p1
# /sbin/biosdevname --smbios 2.4 --policy physical -i eth0
em1
```

- CentOS 7 – wyłączenie nowego systemu przypisywania nazw

```
# cd /etc/udev/rules.d
# ln -s /dev/null 80-net-name-slot.rules
```

Interfejs sieciowy: zmiana nazwy via `ip`

Jeśli nie chcemy modyfikować konfiguracji udev w celu zmieny nazwy interfejsu sieciowego, to można tę nazwę zmienić korzystając z komendy `ip`.

Przykład:

```
ip link set wlp3s0 down
ip link set wlp3s0 name wlan0
ip link set wlan0 up
```

Interfejs sieciowy: zmiana nazwy via `ifrename`

```
NAME
       ifrename - rename network interfaces based on various static criteria

SYNOPSIS
       ifrename [-c configfile] [-p] [-d] [-u] [-v] [-V] [-D]
       ifrename [-c configfile] [-i interface] [-n newname]

DESCRIPTION
       Ifrename is a tool allowing you to assign a consistent name to each of
       your network interface.

       By default, interface names are dynamic, and each network interface is
       assigned the first available name (eth0, eth1, ...). The order network
       interfaces are created may vary. For built-in interfaces, the kernel
       boot time enumeration may vary. For removable interface, the user
       may plug them in any order.
```

Interfejs sieciowy: zmiana nazwy via `ifrename`

Przykładowa zawartość pliku `/etc/iftab`:

```
eth*  mac 00:15:c5:b1:9a:8a driver tg3
dummy* mac 2a:97:4c:16:d6:f9 driver dummy
```

Komenda `ifrename` musi zostać użyta po załadowaniu modułów tworzących interfejsy sieciowe, a uaktywnieniem (nie konfiguracją!) tych interfejsów (zob. także `ifrename -p`).

Przykłady:

```
# ifrename
# ifrename -i dummy0 -n testif0
# ifrename -i p4p1 -n eth0
```

Magistrala programowa D-BUS[48]

D-BUS:

- mechanizm komunikacji międzyprocesowej (IPC+RPC)
- część projektu freedesktop.org, rozwijany przez społeczność RedHat-a
- magistrala systemowa:
  - jedna magistrala dla wszystkich użytkowników
  - tylko dla usług systemowych
  - tylko dla zdarzeń niskiego poziomu (podłączanie do sieci, załączanie urządzeń USB, itp.)
- magistrala sesji
  - dla każdej sesji jedna
  - *provides desktop services to user applications*
  - powiązana z sesją X

[48] https://bootlin.com/pub/conferences/2016/meetup/dbus/josserand-dbus-meetup.pdf

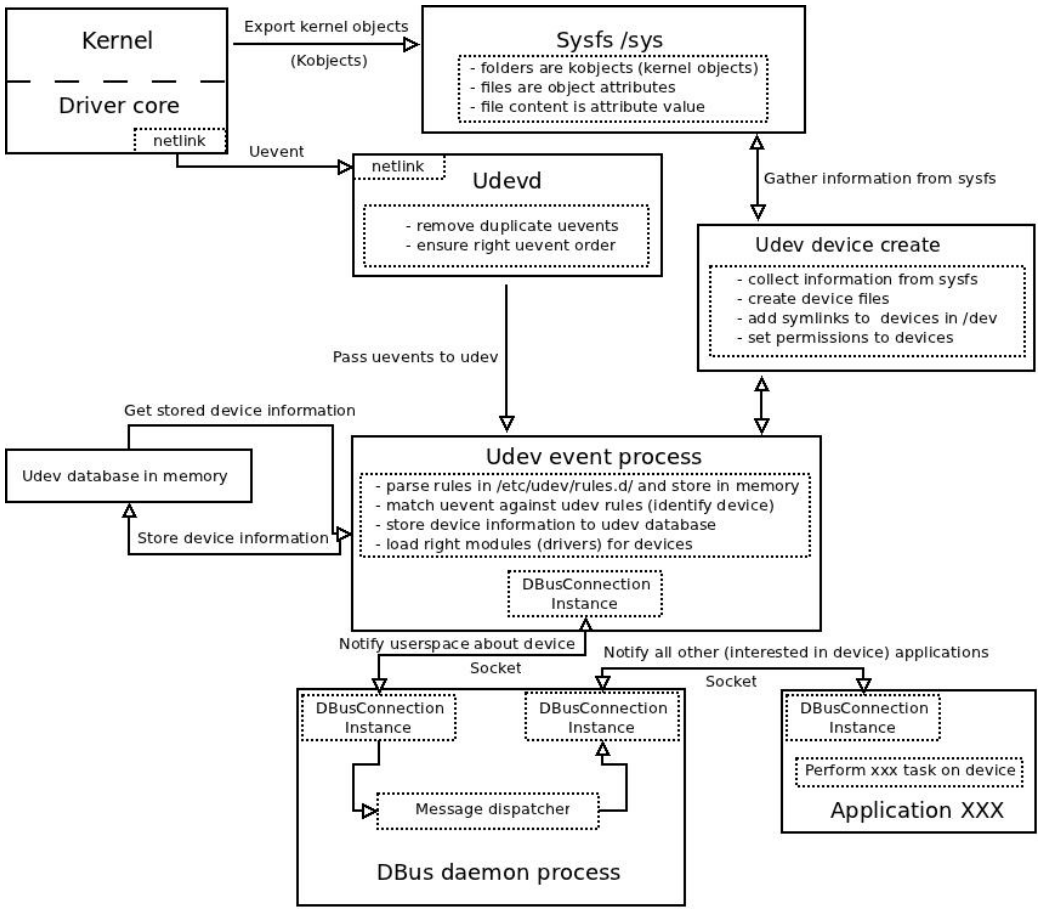
dbus-daemon

```
NAME   dbus-daemon - Message bus daemon
DESCRIPTION
       dbus-daemon is the D-Bus message bus daemon. (...) D-Bus is
       first a library that provides one-to-one communication between
       any two applications; dbus-daemon is an application that uses
       this library to implement a message bus daemon. Multiple
       programs connect to the message bus daemon and can exchange
       messages with one another.
       There are two standard message bus instances: the systemwide
       message bus (installed on many systems as the "messagebus" init
       service) and the per-user-login-session message bus (started
       each time a user logs in). dbus-daemon is used for both of
       these instances, but with a different configuration file.
```

Śledzenie zachowania demona: `dbus-monitor, bustle[-pcap]`

Przeglądanie obiektów D-bus (GUI): qdbusviewer, d-feet

udev + d-bus[49]



[49] http://blogas.sysadmin.lt/?p=141

Hardware Abstraction Layer (HAL)

```
NAME   hald - HAL daemon
DESCRIPTION
       hald is a daemon that maintains a database of the devices
       connected to the system system in real-time. The daemon
       connects to the D-Bus system message bus to provide an API that
       applications can use to discover, monitor and invoke operations
       on devices.
```

Ewolucja obsługi urządzeń w Linuksie:

$$\text{kudzu} \longrightarrow \text{hald} \longrightarrow \text{udevd}$$

$$\text{kernel} \longrightarrow \text{udevd} \longrightarrow \text{NetworkManager} \longleftrightarrow \text{D-bus} \longleftrightarrow \text{Applications}$$

HAL is a behemoth, do-it-all, daemon to access hardware. It is now obsoleted by udisks and upower, as well as libudev for device discovery.[50]

[50] http://fedoraproject.org/wiki/Features/HalRemoval

upower

```
NAME
       UPower - System-wide Power Management
DESCRIPTION
       UPower provides an interface to enumerate power sources on the system
       and control system-wide power management. Any application can access
       the org.freedesktop.UPower service on the system message bus.
```

```
# upower --enumerate
/org/freedesktop/UPower/devices/line_power_AC
/org/freedesktop/UPower/devices/battery_BAT0
/org/freedesktop/UPower/devices/battery_BAT1
/org/freedesktop/UPower/devices/DisplayDevice

# upower --show-info /org/freedesktop/UPower/devices/battery_BAT1

# upower --monitor|--monitor-detail
```

Notebooki: `man tlp`, `man tlp-stat`

udisks (CentOS 7/Fedora 31)

NAME

udisks - Disk Manager

DESCRIPTION

udisks provides interfaces to enumerate and perform operations on disks and storage devices. Any application (including unprivileged ones) can access the udisksd(8) daemon via the name org.freedesktop.UDisks2 on the system message bus[1]. In addition to the D-Bus API, a library, libudisks2 is also provided. This library can be used from C/C++ and any high-level language with GObjectIntrospection[2] support such as Javascript and Python. udisks is only indirectly involved in what devices and objects are shown in the user interface.

ACCESS CONTROL

By default, logged-in users in active log-in sessions are permitted to perform operations (for example, mounting, unlocking or modifying) on devices attached to the seat their session is on. Access-control is fine-grained and based on polkit(8), see the ‘‘Authorization Checks’’ chapter in the udisks documentation for more information. Note that the x-udisks-auth option can be used in the /etc/fstab and /etc/crypttab files to specify that additional authorization is required to mount resp. unlock the device (typically requiring the user to authenticate as an administrator).

udisksctl (CentOS 7/Fedora 31)

```
NAME
       udisksctl - The udisks command line tool
SYNOPSIS
       udisksctl status
       udisksctl info {--object-path OBJECT | --block-device DEVICE}
       udisksctl mount {--object-path OBJECT | --block-device DEVICE} \
                                [--filesystem-type TYPE] [--options OPTIONS...] ...
       udisksctl unmount {--object-path OBJECT | --block-device DEVICE} [--force] ...
       udisksctl loop-setup --file PATH [--read-only] [--offset OFFSET] [--size SIZE] ...
       udisksctl loop-delete {--object-path OBJECT | --block-device DEVICE} ...
       udisksctl power-off {--object-path OBJECT | --block-device DEVICE} ...
       udisksctl monitor
       udisksctl dump
       udisksctl help

DESCRIPTION
       udisksctl is a command-line program used to interact with the udisksd(8)
       daemon process.
```

polkit

NAME

polkit - Authorization Manager

OVERVIEW

polkit provides an authorization API intended to be used by privileged programs (“MECHANISMS”) offering service to unprivileged programs (“SUBJECTS”) often through some form of inter-process communication mechanism. In this scenario, the mechanism typically treats the subject as untrusted. For every request from a subject, the mechanism needs to determine if the request is authorized or if it should refuse to service the subject. Using the polkit APIs, a mechanism can offload this decision to a trusted party: The polkit authority.

The polkit authority is implemented as an system daemon, polkitd(8), which itself has little privilege as it is running as the polkitd system user. Mechanisms, subjects and authentication agents communicate with the authority using the system message bus.

In addition to acting as an authority, polkit allows users to obtain temporary authorization through authenticating either an administrative user or the owner of the session the client belongs to. This is useful for scenarios where a mechanism needs to verify that the operator of the system really is the user or really is an administrative user.

NetworkManager lub /etc/init.d/network

RHEL 6 i 7 (CentOS 6 i 7) umożliwia zarządzanie interfejsami sieciowymi poprzez `/etc/init.d/network` (`ifup`, `ifdown`, etc) oraz usługę NetworkManager.[51]

Zasady współdziałania:

- jeśli urządzenie jest zarządzane przez NetworkManagera i jest podłączone, to nic się nie dzieje
- jeśli urządzenie nie jest zarządzane przez NetworkManagera i nie jest podłączone, to stosowany jest tradycyjny sposób obsługi urządzenia
- jeśli urządzenie jest zarządzane przez NetworkManagera, to użycie skryptu `ifdown` jest równoznaczne z niejawnym wykorzystaniem NetworkManagera do wykonania zadania.

W RHEL 7 (CentOS 7) domyślnym zarządcą jest NetworkManager.

[51] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-NetworkManager_-and_the_Network_Scripts.html

/etc/init.d/network

Skrypt `network` korzysta z definicji zmiennych i funkcji zawartych w plikach:

- /etc/init.d/functions
- /etc/sysconfig/network
- /etc/sysconfig/network-scripts/network-functions
- /etc/sysconfig/network-scripts/ifcfg-DEV
- /etc/sysconfig/network-scripts/ifup|ifdown-DEV
- /etc/sysconfig/network-scripts/ifup|down-route
- /etc/sysconfig/static-routes
- /etc/sysconfig/network-scripts/init.ipv6-global

```
service network [status|[re]start|stop]
ifup|ifdown DEV
/etc/sysconfig/network-scripts/ifup|ifdown-ipv6 DEV
```

/etc/sysconfig/network

```
NETWORKING=yes
NETWORKDELAY=10
#GATEWAY=<gateway IP>
#GATEWAYDEV=eno1
#NETWORKING_IPV6=no  + net.ipv6.conf.all.disable_ipv6=1 (/etc/sysctl.conf)
#NOZEROCONF=yes|no
#IPV6_AUTOCONF=yes|no
#IPV6_ROUTER=yes|no
#IPV6FORWARDING=yes|no
#IPV6_AUTOTUNNEL=yes|no
#IPV6_DEFAULTGW=<IPv6 address[%interface]> (optional)
#IPV6_DEFAULTDEV=eth0
```

Opis zmiennych znajduje się w pliku: /usr/share/doc/initscripts-∗/-sysconfig.txt.

route-DEV

Jeśli przy uruchamianiu interfejsu trzeba dodać wpis(y) do tablicy routingu, to należy utworzyć plik route-DEV zawierający informacje o adresie sieci, masce i bramie, np.:

```
ADDRESS0=192.168.10.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.200.254

ADDRESS1=192.168.20.0
NETMASK1=255.255.255.0
GATEWAY1=192.168.200.254
```

ifcfg-DEV[52]

ifcfg-eth0 (statyczna konfiguracja interfejsu eth0)

```
DEVICE=eth0
BOOTPROTO=none                          # NETWORK=10.0.1.0
ONBOOT=yes                              # NETMASK=255.255.255.0
IPADDR=10.0.1.27                        # BROADCAST=10.0.1.255
PREFIX=24
```

ifcfg-eth0 (dynamiczna konfiguracja interfejsu eth0 via DHCP)

```
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
SEARCH="fizyka.umk.pl" # DOMAINNAME="fizyka.umk.pl"
                # oraz dopisanie wartości pobranych z serwera DHCP.
DNS1="8.8.8.8"
DNS2="8.8.4.4"
```

[52] http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-networkscripts-interfaces.html

sysconfig.txt: dodatkowe zmienne[53]

```
USERCTL=yes|no

BOOTPROTO=none|bootp|dhcp

NOZEROCONF=no|yes
  Set this to yes to set a dynamic link-local addresses over this device.

PERSISTENT_DHCLIENT=yes|no|1|0
  Without this option, or if it is 'no'/'0', and BOOTPROTO=dhcp, dhclient
  is run for the interface in "one-shot" mode; if the dhcp server does not
  respond for a configurable timeout, then dhclient exits and the interface
  is not brought up - the '-1' option is given to dhclient.

  If PERSISTENT_DHCLIENT=yes, then dhclient will keep on trying to contact
  the dhcp server when it does not respond - no '-1' option is given to
  dhclient. Note: this disables the automatic checking for the presence
  of a link before starting dhclient.
```

[53] Zob. /usr/share/doc/initscripts-∗/sysconfig.txt

sysconfig.txt: dodatkowe zmienne

```
DHCPRELEASE=yes|no|1|0
  With this option set to 'yes' (1), when a dhcp configured interface is
  brought down with 'ifdown', the lease will be released. Otherwise, leases
  are not released.
DHCP_HOSTNAME=<name> Sends the specified hostname to the DHCP server.
NM_CONTROLLED=yes|no
  If set to 'no', NetworkManager will ignore this connection/device.
  Defaults to 'yes'.
ZONE=
  Network zone (trust level) of this connection.  If not set, default zone
  (specified in /etc/firewalld/firewalld.conf) is used. To see all available
  zones, run 'firewall-cmd --get-zones'.

ARPCHECKn=yes|no
  If set to 'no', ifup will not try to determine, if requested ip address
  is used by other machine in network.  Defaults to 'yes'.
  IPV4_FAILURE_FATAL=yes|no If set to yes, ifup-eth will end immediately
  after ipv4 dhclient fails.  Defaults to 'no'.
```

ifcfg-DEV i aliasy

Interfejsom można przypisywać dodatkowe numery poprzez tworzenie aliasów do plików ifcfg-DEV o nazwach ifcfg-DEV:0, ifcfg-DEV:1, itd. Pliki te zawierają wpisy postaci DEVICE=eth0:0, DEVICE=eth0:1, etc.[54]

ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=00:16:3E:05:02:37
ONBOOT=yes
```

ifcfg-eth0:0

```
DEVICE=eth0:0
BOOTPROTO=none
IPADDR=10.10.1.0
NETMASK=255.255.0.0
ONBOOT=yes
#PREFIX=16
```

[54] Poprzez pliki aliasowe nie można konfigurować interfejsu via DHCP.

ifcfg-DEV i aliasy

W nowszych dystrybucjach (np. CentOS 6 i 7) dodatkowe adresy można określać przy pomocy jednego pliku konfiguracyjnego, np.:

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
IPADDR1=10.10.2.0
#NETMASK1=255.255.0.0
PREFIX1=16
IPADDR2=10.10.3.0
#NETMASK2=255.255.0.0
PREFIX2=16
```

ifcfg-DEV i klony

Pliki konfiguracyjne interfejsów sieciowych mogą być klonowane, dzięki czemu możliwe jest uruchamianie tego samego interfejsu z różnymi opcjami.

ifcfg-dummy0

```
DEVICE=dummy0
BOOTPROTO=none
ONBOOT=yes
IPADDR=10.10.1.0
PREFIX=16
USERCTL=yes
NM_CONTROLLED=no
```

ifcfg-dummy0-2

```
DEVICE=dummy0
BOOTPROTO=none
ONBOOT=no
IPADDR=10.100.1.0
PREFIX=16
```

ifcfg-DEV i VLAN

Z danym urządzeniem fizycznym można związać interfejs pozwalający na komunikację z określoną wirtualna siecią lokalną (VLAN). Jądro musi zawierać moduł 8021q!

ifcfg-p4p1.70|vlan70

```
VLAN=yes
#VLAN_NAME_TYPE=VLAN_PLUS_VID_NO_PAD
#DEVICE=vlan70
DEVICE=p4p1.70
PHYSDEV=p4p1
BOOTPROTO=none
ONBOOT=no
TYPE=Ethernet
IPADDR=192.168.222.1
PREFIX=24
```

Komenda `ifup p4p1.70|vlan70` tworzy interfejs i go konfiguruje.

ifcfg-DEV i VLAN

Interfejsy sieciowe do obsługi VLAN-ów można tworzy/usuwać przy pomocy komendy vconfig (RHEL/CentOS 6).

```
# vconfig
...
Usage: add             [interface-name] [vlan_id]
       rem             [vlan-name]
       set_flag        [interface-name] [flag-num]       [0 | 1]
       set_egress_map  [vlan-name]      [skb_priority]   [vlan_qos]
       set_ingress_map [vlan-name]      [skb_priority]   [vlan_qos]
       set_name_type   [name-type]

* The vlan_id is the identifier (0-4095) of the VLAN you are operating on.
* skb_priority is the priority in the socket buffer (sk_buff).
* vlan_qos is the 3 bit priority in the VLAN header
* name-type:  VLAN_PLUS_VID (vlan0005), VLAN_PLUS_VID_NO_PAD (vlan5),
              DEV_PLUS_VID (eth0.0005), DEV_PLUS_VID_NO_PAD (eth0.5)
* FLAGS:  1 REORDER_HDR  When this is set, the VLAN device will move the
             ethernet header around to make it look exactly like a real
             ethernet device. [...] Default is OFF.
```

ifcfg-DEV i VLAN

Interfejsy sieciowe do obsługi VLAN-ów można tworzy/usuwać przy pomocy komendy `ip` (RHEL/CentOS 7).

```
ip link add link DEVICE name NAME type vlan [ protocol VLAN_PROTO ] id VLANID \
                    [ reorder_hdr { on | off } ] \
                    [ gvrp { on | off } ] \
                    [ mvrp { on | off } ] \
                    [ loose_binding { on | off } ] \
                    [ ingress-qos-map QOS-MAP ] [ egress-qos-map QOS-MAP ]
```

`VLAN_PROTO` – 802.1Q lub 802.1ad (QinQ)

`gvrp` – rejestracja poprzez GARP VLAN Registration Protocol

`mvrp` – rejestracja poprzez Multiple VLAN Registration Protocol[55]

```
# ip link add link eth0 name eth0.10 type vlan id 10
# ip -d link show eth0.10
  eth0.10@eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT
    link/ether 54:52:00:01:00:02 brd ff:ff:ff:ff:ff:ff
    vlan protocol 802.1Q id 10 <REORDER_HDR> addrgenmode eui64
```

[55] GARP – *Generic Attribute Registration Protocol*, dawniej *Group Address Registration Protocol*; powoli wypierany przez MVRP

Network trunking/bonding/teaming

Przełączniki ethernetowe umożliwiają łączenie wielu (2, 4, 8) portów, które są traktowane jako jeden kanał komunikacyjny (*bonding, Ethernet bonding, port trunking, channel teaming, NIC teaming, or link aggregation*).

Przyspieszenie przekazywania ramek między urządzeniami sieciowymi oraz między serwerem i przełącznikiem.

Po stronie serwera wymaga to odpowiedniego łączenia interfejsów (*bonding*). Potrzebny jest odpowiedni moduł jądra (bonding) oraz wsparcie sprzętowe ze strony interfejsów sieciowych i ich modułów obsługi.

ifcfg-bond0

```
DEVICE=bond0
BOOTPROTO=none
IPADDR=192.168.0.200
NETMASK=255.255.255.0
```

ifcfg-eth0|1

```
DEVICE=eth0|1
BOOTPROTO=none
MASTER=bond0
SLAVE=yes
```

Network bonding

Łączenie interfejsów po stronie serwera:

```
# ip address
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast ...
    link/ether 14:fe:b5:d8:ab:12 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast ...
    link/ether 14:fe:b5:d8:ab:14 brd ff:ff:ff:ff:ff:ff
...
8: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
    link/ether 14:fe:b5:d8:ab:12 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.200/24 brd 192.168.0.255 scope global bond0
```

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
bond0           8000.0cc47a4c62b1       no              eth0
                                                        eth1
```

Network teaming

CentOS 7 preferuje agregację interfejsów poprzez *network teaming*:[56]

- wsparcie dla LACP (*Link Aggregation Control Protocol*, IEEE 802.3ad)
- sterowanie ruchem (*runner: roundrobin*, *broadcast*, *activebackup*, *loadbalance*, *random*, *lacp*)
- monitorowanie łącza (*link-watcher: ethtool*, *arp_ping*, *nsna_ping* (pakiety NS/NA IPv6))
- *lockless Tx/Rx path*
- sterowanie i logika w przestrzeni użytkownika (*teamd* i *teamdctl*).
- interfejs D-Bus

[56] http://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-comparison_of_network_-teaming_to_bonding,
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/ch-configure_network_teaming

Network teaming

ifcfg-team0

```
DEVICE=team0
DEVICETYPE=Team
BOOTPROTO=dhcp
```

ifcfg-eth0|1

```
DEVICE=eth0|1
DEVICETYPE=TeamPort
TEAM_MASTER=team0
```

```
# teamdctl team0 state
setup:
  runner: roundrobin
ports:
  eth0
    link watches:
      link summary: up
      instance[link_watch_0]:
        name: ethtool
        link: up
        down count: 0
  eth1
    link watches:
      link summary: up
      ...
```

Przełącznik ethernetowy

Jądro systemu Linux dostarcza wsparcia do budowy przełączników (mostów) ethernetowych. W serwerach są one wykorzystywane (głównie) do obsługi komunikacji z maszynami wirtualnymi.

Tworzenie/usuwanie mostu z interfejsem o nazwie `<name>`:

```
# brctl addbr <name>
# brctl delbr <name>
```

Po utworzeniu mostu można do niego dodać/usunąć port poprzez dodanie/usunięcie interfejsu `<ifname>` veth1, ...:

```
# brctl addif <name> <ifname>
# brctl delif <name> <ifname>
```

Komenda `brctl show` pokazuje aktualną konfigurację urządzenia.

NetworkManager

Cele projektu:[57]

- automatyczne wykrywanie podstawowych urządzeń sieciowych i ich parametrów (jeśli tylko się da)
- dane gromadzone w tekstowych plikach konfiguracyjnych
- natychmiastowe powiadamianie użytkownika i programów o zmianie stanu urządzeń (D-Bus API)
- płynna zmiana połączeń, kiedy zachodzi taka potrzeba
- podstawowa konfiguracja via CLI i GUI

[57] http://fedoraproject.org/wiki/Tools/NetworkManager

NetworkManager

- konfiguracja poprzez *key file*, ifcfg-rh (Red Hat, Fedora), ifupdown (Debian, Ubuntu), ifcfg-suse (SUSE, OpenSUSE, *deprecated*)
- wsparcie dla IPv4 i IPv6
- interface D-Bus
- lokalny buforujący serwer nazw domenowych (dnsmasq, systemd-resolved, unbound)
- połączenia ethernetowe (802.3), InfiniBand
- połączenia Bluetooth, Wi-Fi (802.11)
- połączenia szerokopasmowe via USB lub Bluetooth
- połączenia VPN
- tworzenie mostów i scalanie interfejsów (*bonding/teaming*)
- obsługa urządzeń: tunel, macvlan, vxlan

NetworkManager (CentOS 7)

```
Usage: nmcli [OPTIONS] OBJECT { COMMAND | help }

OPTIONS
  -t[erse]                                    terse output
  -p[retty]                                   pretty output
  -m[ode] tabular|multiline                   output mode
  -f[ields] <field1,field2,...>|all|common    specify fields to output
  -e[scape] yes|no                            escape columns separators in values
  -n[ocheck]                                  don't check nmcli and NetworkManager versions
  -a[sk]                                      ask for missing parameters
  -w[ait] <seconds>                           set timeout waiting for finishing operations
  -v[ersion]                                  show program version
  -h[elp]                                     print this help

OBJECT
  g[eneral]       NetworkManager's general status and operations
  n[etworking]    overall networking control
  r[adio]         NetworkManager radio switches
  c[onnection]    NetworkManager's connections
  d[evice]        devices managed by NetworkManager
  a[gent]         NetworkManager secret agent or polkit agent
  m[onitor]       Watches for changes in connectivity state, devices, connection profiles
```

NetworkManager (CentOS 7)

```
nmcli general status|permissions|logging
nmcli networking on|off|connectivity
nmcli radio all|wifi|wwan|wimax on|off
nmcli connection down|up|show --active
nmcli conn edit [id|uuid|path] <ID>
nmcli device connect|disconnect|show p4p1|wlan0
nmcli agent secret|polkit|all
```

```
# nmcli con show
NAME              UUID                                  TYPE            DEVICE
second interface  9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04  802-3-ethernet  eth1
System eth0       5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  802-3-ethernet  eth0

# nmcli con down 9c92fad9-6ecb-3e6c-eb4d-8a47c6f50c04
Connection 'second interface' successfully deactivated (D-Bus active path: \
                        /org/freedesktop/NetworkManager/ActiveConnection/4)
```

Network Manager: dodatkowe czynności

- DHCP – czynności wykonywane przez dhclient-script są określane przez wartości odpowiednich zmiennych w plikach ifcfg-DEV. Dodatkowe czynności, które muszą być wykonane przed/po konfiguracji interfejsu mogą być realizowane przez zdefiniowane tzw. *enter/exit hooks* (`man dhclient-script`).
- NetworkManager – dodatkowe czynności są realizowane przez skrypty (w porządku alfabetycznym) umieszczone w katalogu /etc/NetworkManager/dispatcher.d po zajściu każdego zdarzenia sieciowego. Skrypty otrzymują dwa argumenty: nazwę interfejsu, którego stan uległ zmianie oraz rodzaj zmiany/akcji: [vpn-]pre-up|down, [vpn-]up|[vpn-]down, hostname, dhcp4|6-change. Dodatkowe informacje o połączeniu są dostępne przez zmienne środowiskowe `CONNECTION_UUID`, `CONNECTION_ID`, `CONNECTION_FILENAME`, `DEVICE_IFACE`, ...(`man NetworkManager`).

mDNS/DNS-SD[58]

- demon Avahi mDNS/DNS-SD jest realizacją architektury *Zeroconf* firmy Apple, zwanej także "Rendezvous" lub "Bonjour"
- demon rejestruje lokalne adresy IP i dostępne statyczne usługi używając protokołów mDNS/DNS-SD
- demon (poprzez API) umożliwia lokalnym programom dostęp do pamięci podręcznej tworzonych przez siebie rekordów mDNS
- zasoby mDNS są dostępne poprzez interfejs D-Bus
- autokonfiguracja hostów przy wykorzystaniu adresu rozgłoszenia grupowego 224.0.0.251 (IPv4) i ff02::fb (link-local IPv6);

  hosty otrzymują adresy z puli 169.254.0.0/16 (IPv4) i fe80::/10 (IPv6).

[58] *multicast Domain Name Service/DNS Service Discovery*
http://www.mactech.com/articles/mactech/Vol.19/19.05/Zeroconf/index.html,
http://zeroconf.org, http://dns-sd.org, http://multicastdns.org, http://avahi.org/

mDNS/DNS-SD[59]

- dostępne zasoby:[60]
  - `avahi-browse [-t] [-r] -a|--all`
  - `avahi-browse [-t] _services._dns-sd._udp`
  - `avahi-browse [-t] '_afpovertcp|ftp|ipp|http|https|libvirt._tcp'`
  - `avahi-browse [-t] '_printer|ssh|sftp-ssh|workstation|udisks-ssh._tcp'`
  - `avahi-discover`
- publikowanie dostępnych usług: `avahi-publish`
- rozwiązywanie nazw: `avahi-resolve [-4|6] -n|--name centos7-1.local`
- rozwiązywanie adresów: `avahi-resolve [-4|6] -a|--address 192.168.142.1`
- konfiguracja serwerów DNS: `avahi-dnsconfd`
- konfiguracja interfejsu iface: `avahi-autoip iface`

[59] Zapora nie może blokować adresu 224.0.0.251 oraz portów:5222 (xmmp client, XMPP Client Connection), 5298 (presence, XMPP Link-Local Messaging), 5353 (mdns, Multicast DNS); XMPP – Extensible Messaging and Presence Protocol.
[60] DNS SRV (RFC 2782) Service Types: https://www.dns-sd.org/ServiceTypes.html;
zob. także https://github.com/lathiat/avahi/blob/master/service-type-database/service-types

mDNS/DNS-SD

```
$ avahi-browse -at
+  wlan0 IPv6 scobie                                          SSH Remote Terminal  local
+  wlan0 IPv4 scobie                                          SSH Remote Terminal  local
+  wlan0 IPv6 scobie [1e:b6:78:0c:cb:1b]                      Workstation          local
+  wlan0 IPv4 scobie [1e:b6:78:0c:cb:1b]                      Workstation          local
+  wlan0 IPv6 PCP pmcd on scobie                              _pmcd._tcp           local
+  wlan0 IPv4 PCP pmcd on scobie                              _pmcd._tcp           local
+  wlan0 IPv4 Chromecast-fc1e904941bc3a80c72fceaeb29f4fd0     _googlecast._tcp     local
```

```
$ avahi-browse -r _googlecast._tcp
+  wlan0 IPv4 Chromecast-fc1e904941bc3a80c72fceaeb29f4fd0     _googlecast._tcp     local
=  wlan0 IPv4 Chromecast-fc1e904941bc3a80c72fceaeb29f4fd0     _googlecast._tcp     local
   hostname = [fc1e9049-41bc-3a80-c72f-ceaeb29f4fd0.local]
   address = [192.168.2.83]
   port = [8009]
   txt = ["rs=YouTube" "nf=1" "bs=FA8FCA5382AE" "st=1" "ca=4101" "fn=Kokocast" \
         "ic=/setup/icon.png" "md=Chromecast" "ve=05" "rm=" \
         "cd=79CED760DD7A895D9D69236038D1498F" "id=fc1e904941bc3a80c72fceaeb29f4fd0"]
```

DNS (*Domain Name System*)[61]

DNS wiąże z nazwami domenowymi szereg informacji, przede wszystkim zamienia nazwy hostów na ich adresy IP oraz przechowuje nazwy serwerów pocztowych właściwych dla poszczególnych domen.

- Do czasu wymyślenia i wdrożenia DNS przez P.Mockapetrisa w 1983 r. zamiana nazw na adresy IP odbywała się w oparciu o plik HOSTS.TXT (pobierany z SRI, teraz SRI International) przez każdego hosta w sieci. Rosnąca liczba hostów wymagała stworzenia bardziej skalowalnego systemu.
- Pierwotna specyfikacja DNS jest zawarta w dokumentach RFC 882 oraz 883, które zostały w 1987 r. zastąpione przez RFC 1034 i 1035 wraz z późniejszymi rozszerzeniami i uaktualnieniami.

  Dokumenty te definiują rozproszoną i hierarchiczną bazę danych.

[61] Configuring Domain Name System (DNS) servers
BIND 9 Administrator Reference Manual
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-81-2.pdf
http://www.zytrax.com/books/dns/
http://en.wikipedia.org/wiki/Root_name_server

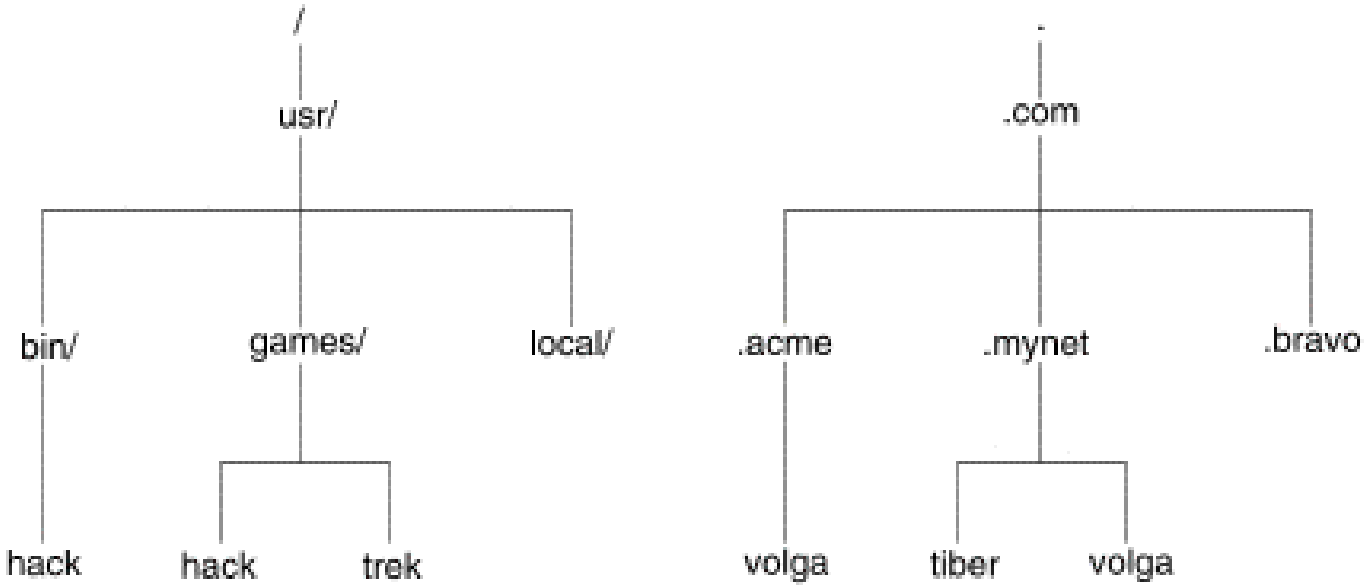
DNS

- 1984 – pierwsze uniksowe wdrożenie (Berkeley)
- 1985 – K. Dunlap (DEC) istotnie przerabia pierwotną wersję i tworzy program bind (*Berkeley Internet Name Domain*)
- 1990 – wersja systemu BIND na platformę Windows NT
- inne programy do tworzenia i odpytywania serwerów nazw: NSD (*Network Server Daemon*), djbdns, dnsmasq, Microsoft DNS, itp.
- DNS wykorzystuje UDP, a jeśli trzeba to TCP (DNSSEC, DNS dla IPv6, transfer plików strefowych)
- DNSSEC – bezpieczna wersja DNS[62]
- EDNS(0) (*Extended DNS*), GeoDNS[63]
  https://www.iplocation.net, https://www.ip2location.com

[62] https://medium.com/iocscan/how-dnssec-works-9c652257be0
[63] https://engineering.salesforce.com/why-is-edns-important-for-content-delivery-85f5690744ba
https://constellix.com/dns/geo-dns-services/geo-dns-explained/

DNS – hierarchia domen

`/usr/games/trek` `volga.mynet.com.`



Domeny i strefy

Domeny najwyższego poziomu (TLD, *Top Level Domain*):

- gTLD (*unsponsored generic TLD*): .biz, .com, .name, .net, .org, .pro (11/2016 – 1519, 12/2017 – 1541, 4/2020 –1930[64])

  Zarządzanie: ICANN (*Internet Corporation for Assigned Numbers and Names*), od 1998

- sTLD (*Sponsored TLD*): .aero, .coop, .edu, .gov, .int, .mil, .jobs, .museum, .travel

- ccTLD, *country code TLD* (ISO 3166): pl, es, gb (nie uk!), itd.

  Zarządzanie: organizacje/instytucje wyznaczone przez władze poszczególnych krajów

Czytanie adresu od prawej do lewej pozwala na prześledzenie procesu delegowania uprawnień do zarządzania daną domeną. Delegowanie uprawnień odbywa się w jednostkach zwanych strefami (*zones*).

[64] https://gtldresult.icann.org/applicationstatus/viewstatus

Domeny i strefy

- "." , czyli domena podstawowa (*root domain*) stanowi początek drzewa nazw
- domena jest częścią przestrzeni nazw domenowych
- poddomena (domena dziecko) jest domeną, która odchodzi (odgałęzia się) od danej domeny
- *strefa* (*zone*) to część przestrzeni nazw domenowych, która jest obsługiwana przez oddelegowany (serwer/serwery); delegowanie uprawnień odbywa w jednostkach-strefach (*zones*)

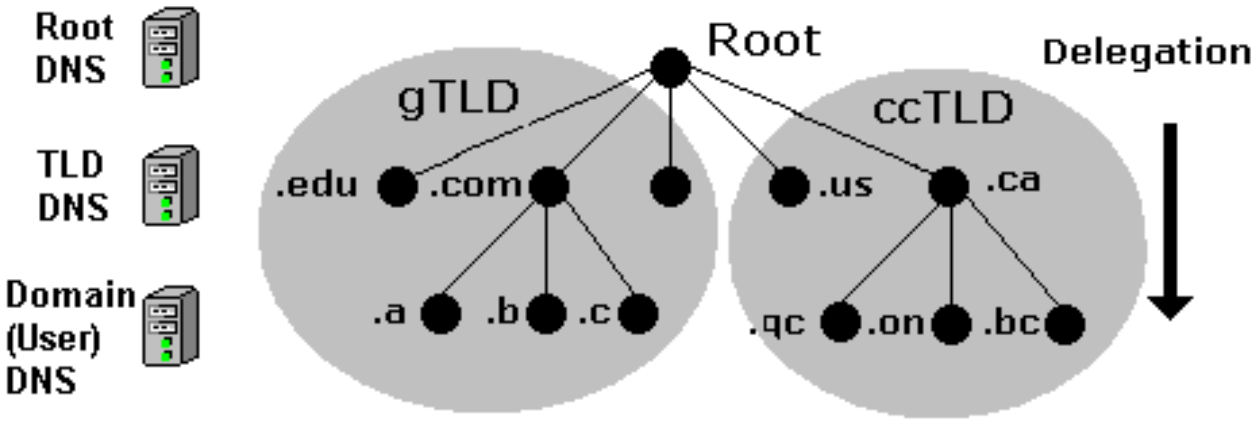
  W nazwach domenowych nie rozróżnia się dużych i małych liter. Nazwa pojedynczej (pod)domeny nie może mieć więcej niż 63 znaki, cała nazwa – 255 znaków. Liczba możliwych poziomów domen jest ograniczona do 127. Nazwy domenowe mogą zawierać znaki zapisane w unikodzie. Są one przechowywane w postaci znaków ASCII: konwersaja jest dokonywana przy pomocy schematu kodowania zwanego Punycode.[65]

[65] http://www.charset.org/punycode.php

Domeny i strefy

- Adresy www.umk.pl oraz www.fizyka.umk.pl pokazują, że umk jest domeną drugiego poziomu, z której wyodrębniono domenę trzeciego poziomu fizyka.umk.pl.
- FQDN (*Fully Qualified Domain Name*):
  www.umk.pl., www.fizyka.umk.pl.
- Zarządzanie nazwami w danej domenie może być w gestii jednego lub grupy serwerów nazw. Obsługują one wówczas jedną strefę. Delegowanie umożliwia rozproszenie administrowania nazwami na szereg serwerów (grup serwerów), które są odpowiedzialne za poszczególne strefy (autorytatywne dla tych stref).
- Zamiana adresów na nazwy odbywa się poprzez proces odwrotny kontrolowany przez tzw. odwrotne strefy (*reverse zones*) utrzymywane w tzw. *Internet Address and Routing Parameter Area*, czyli domenie arpa.

Domeny i serwery



Serwery domeny *root* dostępne via *anycast*:

- do 2002 – 13 serwerów dla domeny *root* w 4. krajach
- 12/2004 – 80 serwerów (wraz z kopiami) w 34. krajach
- 10/2015 – 632 serwery
- 04/2020 – 1099 jednostek zarządzanych przez 12 niezależnych operatorów[66]

[66] https://root-servers.org

Struktura DNS

Na system nazw domenowych składają się

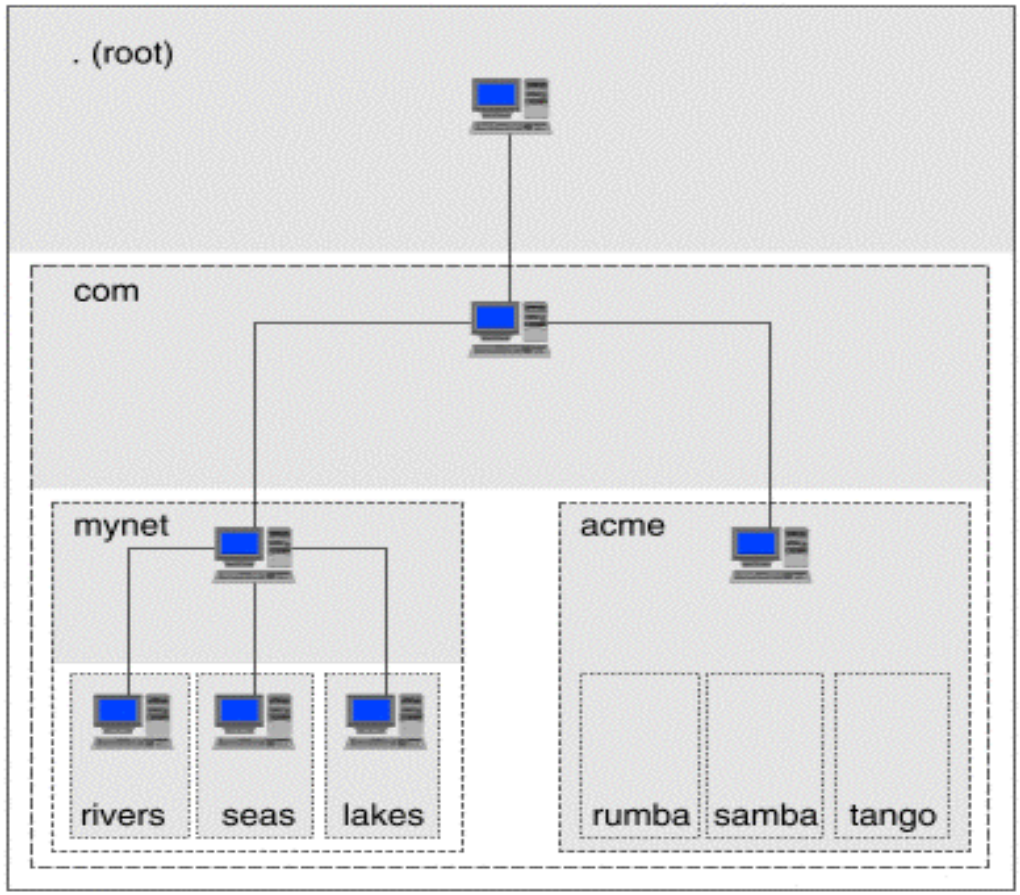
- dane opisujące jedną lub wiele domen; dane są przechowywane w postaci tekstowej w rekordach zasobów (*resource records*) gromadzonych w tzw. plikach stref (*zone files*)
- jeden lub więcej programów udostępniających dane; serwer działa w oparciu o wybraną konfigurację, czyta pliki stref, za które odpowiada oraz odpowiada na zapytania (kwerendy)
- program lub biblioteka (dostępne na każdym z hostów) umożliwiające użytkownikom rozwiązywanie nazw (*resolver*)

Rodzaje serwerów nazw

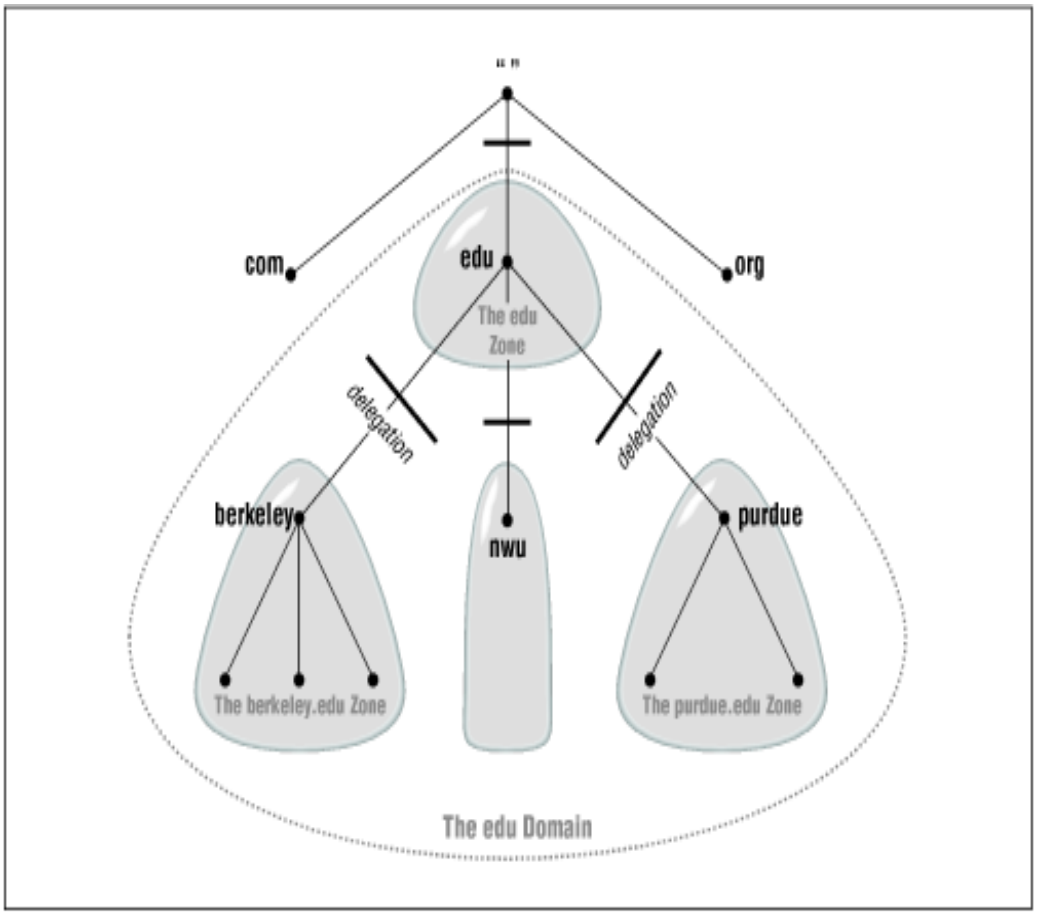
- serwery domeny głównej, serwery poddomen
  - dla każdej domeny musi być zdefiniowany tzw. serwer główny (*master*, *primary master*) oraz serwer zapasowy (*slave*, *secondary master*)
  - serwer główny i zapasowy są serwerami autorytatywnymi dla strefy

  Serwer może być serwerem głównym dla jednej strefy i zapasowym dla innej.
- serwery delegujące
- serwery buforujące
- serwery szczątkowe (*stub*) – serwery obsługujące tzw. strefy szczątkowe, czyli strefy opisywane tylko przez te rekordy zasobów, które są niezbędne dla określenia autorytatywnych serwerów nazw dla tych stref (cecha programu bind, nie element standardu DNS)

Domeny i strefy

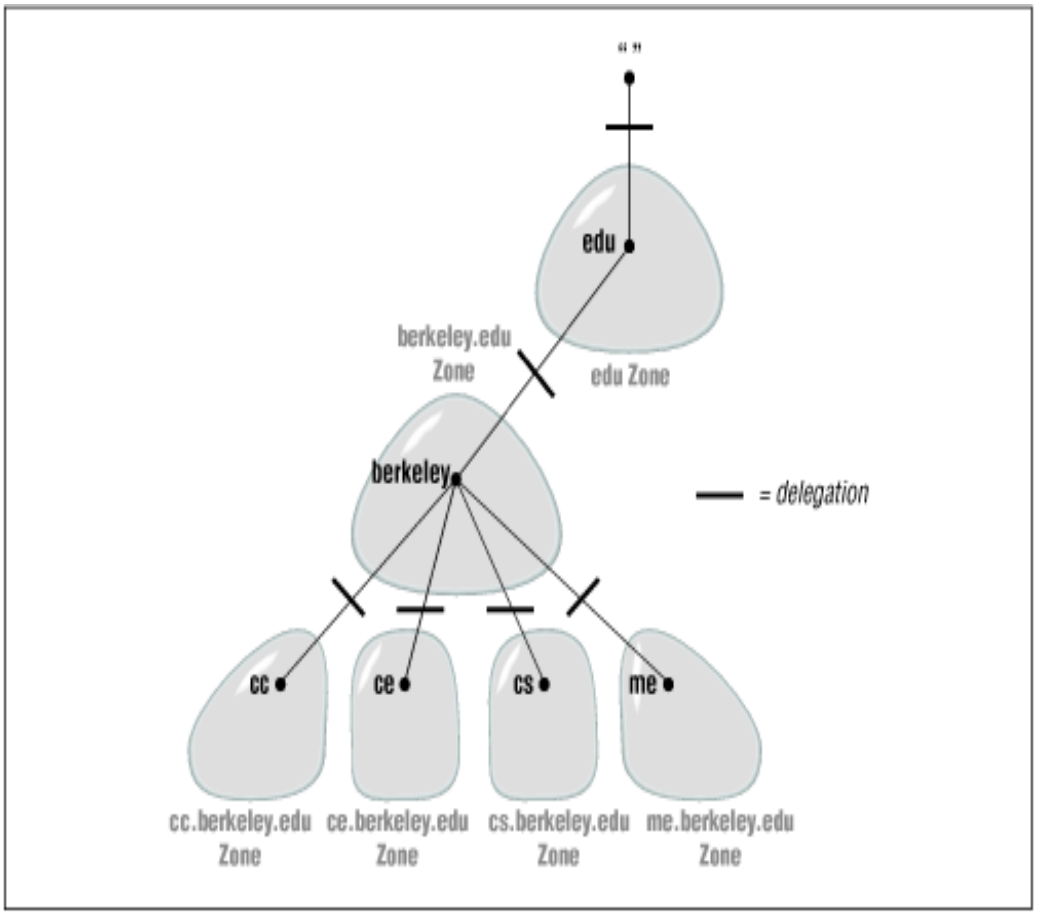


Domeny i strefy[67]



[67] http://oreilly.com/catalog/dns3/chapter/ch02.html#ch02_03.htm

Domeny i strefy[68]



[68] http://oreilly.com/catalog/dns3/chapter/ch02.html#ch02_03.htm

DNS – transakcje

- DNS kwerenda/odpowiedź
- transfer strefy: pobieranie wszystkich rekordów zasobów
- dynamiczne uaktualnienia
- powiadomienia DNS (*DNS NOTIFY*): wymuszenie przez serwer główny na serwerze zapasowym wykonania transferu strefy (jeśli uległ zmianie numer seryjny rekordu SOA)

DNS – zapytania/kwerendy

- głównym zadaniem serwera DNS jest odpowiadanie na zapytania kierowane od lokalnych lub odległych rezolwerów lub innych serwerów DNS, które działają na zlecenie rezolwerów
- zapytania mogą dotyczyć dowolnych domen
- serwer DNS może być serwerem autorytatywnym dla jednej lub wielu wskazanych domen

DNS – rodzaje kwerend

Jeśli zapytanie dotyczy domeny, dla której serwer nie ma plików stref, to odpowiedź zależy od rodzaju zapytania:

- kwerenda rekurencyjna (*recursive query*) – zwracana jest pełna odpowiedź na zapytanie; serwery nie muszą udzielać odpowiedzi na takie zapytania
- kwerenda iteracyjna (*iterative query*), nierekurencyjna – pełna odpowiedź może być udzielona lub może zostać wskazany inny serwer DNS, który potrafi udzielić odpowiedzi; wszystkie serwery muszą udzielać odpowiedzi na takie zapytania

Kwerenda rekurencyjna

W wyniku wysłania zapytania do serwera DNS kwerendy rekurencyjnej klient otrzymuje:

- odpowiedź (ew. wraz z rekordem CNAME) zawierającą status odpowiedzi: autorytatywna bądź pozytywna, tj. wzięta z pamięci podręcznej
- błąd (NXDOMAIN) wskazujący na brak poszukiwanej domeny lub hosta
- błąd wskazujący na chwilowe problemy z działaniem serwera DNS, ew. brak możliwości dostępu do innego serwera

Kwerenda iteracyjna

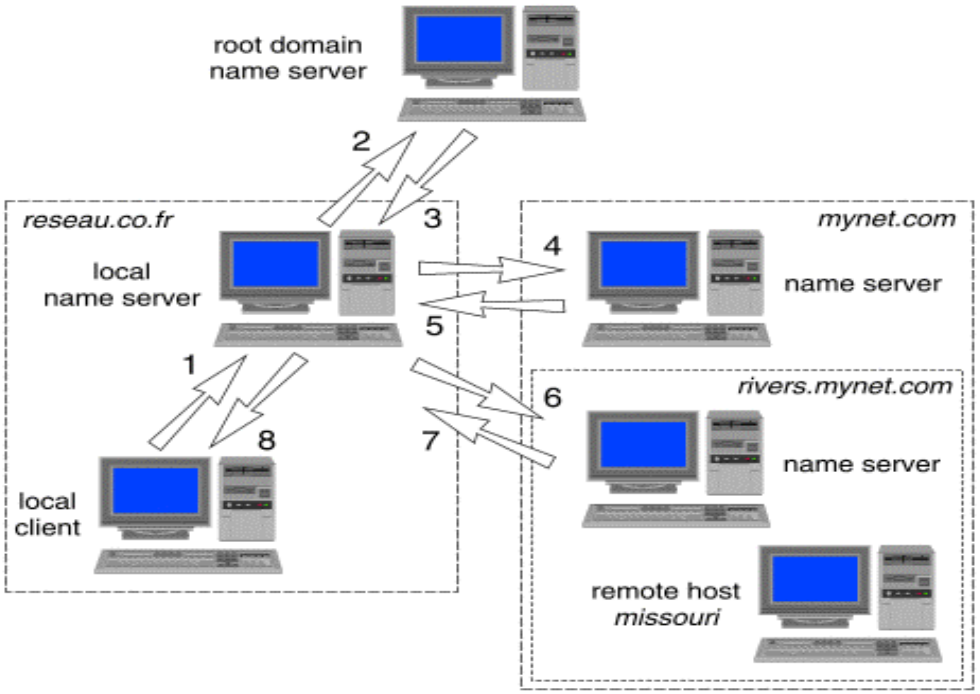
W wyniku wysłania zapytania do serwera DNS kwerendy iteracyjnej klient otrzymuje:

- odpowiedź (ew. wraz z rekordem CNAME) zawierającą status odpowiedzi: autorytatywna bądź pozytywna, tj. wzięta z pamięci podręcznej
- błąd (NXDOMAIN) wskazujący na brak poszukiwanej domeny lub hosta
- błąd wskazujący na chwilowe problemy z działaniem serwera DNS, ew. brak możliwości dostępu do innego serwera
- odpowiedź odsyłającą (*referal*), tj. nazwę i adres jednego lub więcej serwerów DNS, które są bliżej poszukiwanej nazwy domenowej

Rozwiązywanie nazw przez rekurencję

- klient wysyła do serwera zapytanie o adres IP związany z nazwą domenową, np. missouri.rivers.mynet.com (założenie: serwer nie jest serwerem autorytatywnym dla tej domeny)
- serwer DNS sprawdza, czy ta nazwa znajduje się w jego pamięci podręcznej – brak
- serwer DNS wysyła zapytanie o domenę missouri.rivers.mynet.com do serwera TLD
- serwer TLD odpowiada odsyłając adresy serwerów domeny com
- serwer DNS wysyła zapytanie o missouri.rivers.mynet.com do jednego z serwerów TLD dla domeny com
- serwer TLD odpowiada odsyłając adresy serwerów domeny mynet.com
- serwer DNS wysyła zapytanie o missouri.rivers.mynet.com do jednego z serwerów domeny mynet.com
- serwer odpowiada przesyłając adresy serwerów domeny rivers.mynet.com
- serwer DNS wysyła zapytanie o missouri.rivers.mynet.com do jednego z serwerów domeny rivers.mynet.com
- serwer odsyła odpowiedź w oparciu o plik strefy dla domeny rivers.mynet.com (być może wraz z rekordem CNAME)
- serwer DNS przekazuję tę informację do rezolera klienta

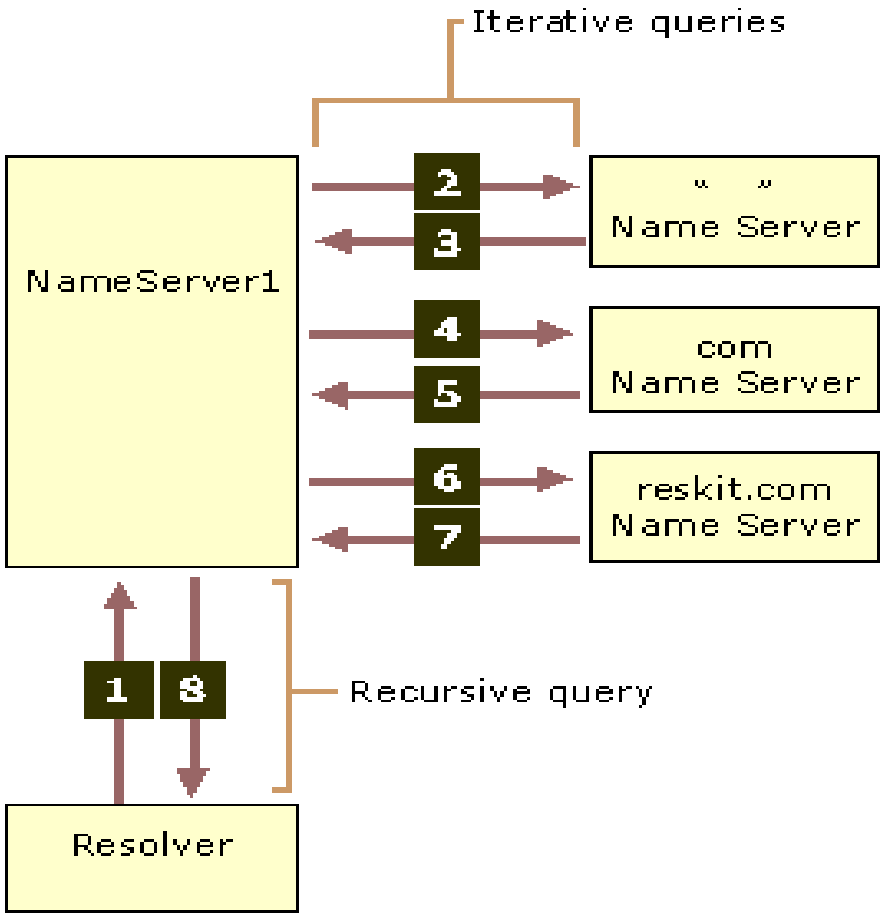
Rekurencyjne i iteracyjne rozwiązywanie nazw



Odpowiedzi otrzymywane w trakcie całego procesu są przechowywane w pamięci podręcznej serwera.

Klient korzysta z trybu rekurencyjnego, a serwer – iteracyjnego rozwiązywania nazw.

Iteracyjne i rekurencyjne rozwiązywanie nazw[69]



[69] http://technet.microsoft.com/en-us/library/cc961401.aspx

```
$ dig @hel www.fizyka.umk.pl any +trace

; <<>> DiG 9.9.4-RedHat-9.9.4-73.el7_6 <<>> @hel www.fizyka.umk.pl any +trace
; (1 server found)
;; global options: +cmd
.                       516348  IN      NS      e.root-servers.net.
.                       516348  IN      NS      a.root-servers.net.
.                       516348  IN      NS      b.root-servers.net.
.                       516348  IN      NS      d.root-servers.net.
.                       516348  IN      NS      k.root-servers.net.
.                       516348  IN      NS      m.root-servers.net.
.                       516348  IN      NS      h.root-servers.net.
.                       516348  IN      NS      l.root-servers.net.
.                       516348  IN      NS      c.root-servers.net.
.                       516348  IN      NS      j.root-servers.net.
.                       516348  IN      NS      i.root-servers.net.
.                       516348  IN      NS      f.root-servers.net.
.                       516348  IN      NS      g.root-servers.net.
.                       518028  IN      RRSIG   NS 8 0 518400 201906181700...
;; Received 525 bytes from 158.75.5.90#53(hel) in 751 ms
```

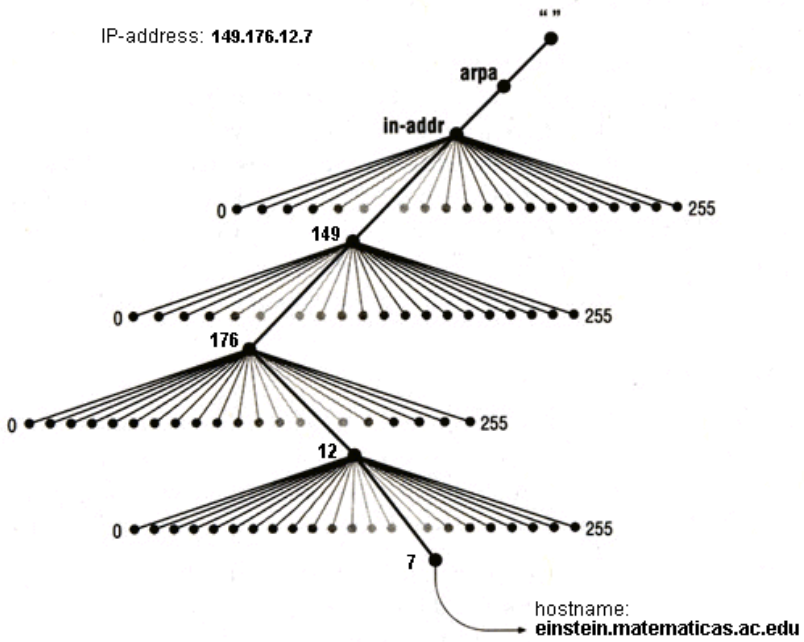
```
pl.                     172800  IN      NS      a-dns.pl.
pl.                     172800  IN      NS      b-dns.pl.
...
pl.                     172800  IN      NS      i-dns.pl.
pl.                     86400   IN      DS      25412 8 2 40FA53CB1DAFF433B7A...
pl.                     86400   IN      RRSIG   DS 8 1 86400 20190618170000 20190...
;; Received 901 bytes from 193.0.14.129#53(k.root-servers.net) in 147 ms

umk.pl.                 86400   IN      NS      ns2.man.torun.pl.
umk.pl.                 86400   IN      NS      koala.uci.umk.pl.
umk.pl.                 86400   IN      NS      blackbox.uci.umk.pl.
vv6ienn6...15o.pl.       3600   IN   NSEC3 1 1 12 01D5...56EEM NS SOA TXT RRSIG DNSKEY NSEC3PARAM
vv6ienn6...15o.pl.       3600   IN   RRSIG NSEC3 8 2 3600 20190704120000 20190604120000 ...
2qais3ov...jf1.pl.       3600   IN   NSEC3 1 1 12 01D5019B7312259D7574 2QBN...N05TQ NS DS RRSIG
2qais3ov...jf1.pl.       3600   IN   RRSIG NSEC3 8 2 3600 20190704120000 20190604120000 ...
;; Received 656 bytes from 77.79.212.238#53(f-dns.pl) in 48 ms

www.fizyka.umk.pl.      5       IN      A       158.75.1.45
www.fizyka.umk.pl.      5       IN      SSHFP   2 1 D8E75593ED11A3E24F9542AB0A9B9BDC9072A33E
www.fizyka.umk.pl.      5       IN      SSHFP   1 1 A0B07632948AE5EABE53E2EBD2F778445965717D
fizyka.umk.pl.          86400   IN      NS      hel.fizyka.umk.pl.
fizyka.umk.pl.          86400   IN      NS      ns1.fizyka.umk.pl.
fizyka.umk.pl.          86400   IN      NS      koala.uci.umk.pl.
;; Received 272 bytes from 158.75.1.4#53(koala.uci.umk.pl) in 1 ms
```

Rekurencyjne i iteracyjne rozwiązywanie nazw: *forwarder*

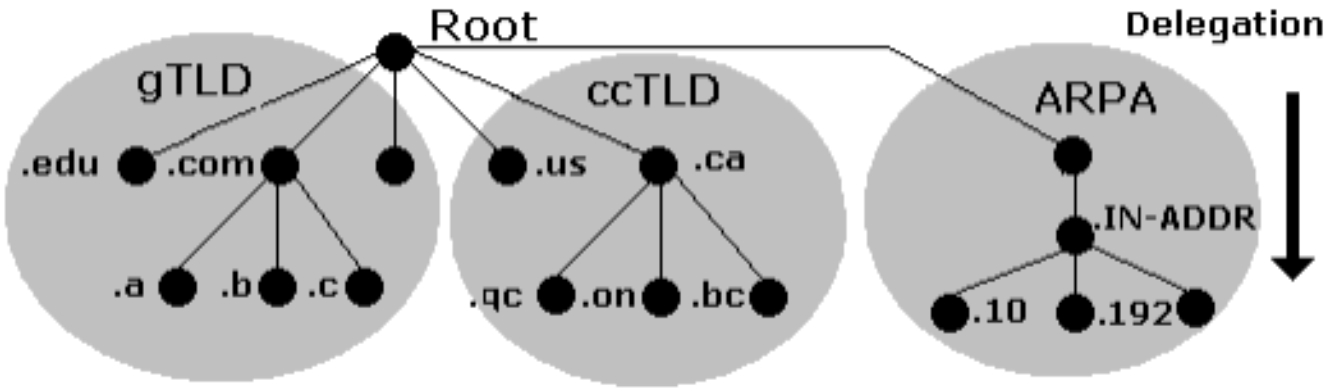
- serwer nazw może zachowywać się jak klient systemu DNS i zlecać innemu serwerowi nazw (tzw. *forwarder*) przeprowadzenie procesu rozwiązywania nazw
- w przypadku wystąpienia błędu zapytanie jest kierowane do innego serwera nazw lub uruchamiana jest iteracyjna procedura rozwiązywania nazw
- jeśli serwer nazw działa w trybie *forward-only*, to rozwiązywanie nazw może być przeprowadzane tylko przez wskazane serwery nazw

Domena in-addr.arpa (ip6.arpa)

Domena in-addr.arpa (ip6.arpa) jest wykorzystywana do zamiany adresów IPv4 (IPv6) na adresy domenowe.



Domena in-addr.arpa (ip6.arpa)



Domena in-addr.arpa (ip6.arpa)

- adresowi 158.75.5.90 odpowiada nazwa 90.5.75.158.in-addr.arpa, która jest odwzorowywana na hel.fizyka.umk.pl
- zamianę adresów IP na adresy domenowe wykonuje się przy pomocy zapytań odwrotnych (*reverse query/lookup*)
- współczesne systemy pocztowe używają odwrotnego rozwiązywania adresów jako prostego sposobu uwierzytelniania serwera pocztowego (w przypadku IPv6 mapowanie odwrotne jest obowiązkowe!)

Najważniejsze typy rekordów[70]

A address record
: Returns a 32-bit IPv4 address, most commonly used to map hostnames to an IP address of the host

AAAA IPv6 address record
: Returns a 128-bit IPv6 address, most commonly used to map hostnames to an IP address of the host.

CNAME Canonical name record
: Alias of one name to another: the DNS lookup will continue by retrying the lookup with the new name.

HINFO Host information record
: Allows definition of the hardware type and operating system of a host. For security reasons these records are rarely used on public servers.

MX mail exchange record
: Maps a domain name to a list of message transfer agents for that domain

NS name server record
: Delegates a DNS zone to use the given authoritative name servers

[70] http://en.wikipedia.org/wiki/List_of_DNS_record_types

Najważniejsze typy rekordów

PTR pointer record
: Pointer to a canonical name. Unlike a CNAME, DNS processing does NOT proceed, just the name is returned. The most common use is for implementing reverse DNS lookups, but other uses include such things as DNS-SD.

SOA start of [a zone of] authority record
: Specifies authoritative information about a DNS zone, including the primary name server, the email of the domain administrator, the domain serial number, and several timers relating to refreshing the zone.

SPF Sender Policy Framework
: Specified as part of the SPF protocol as an alternative to of storing SPF data in TXT records. Uses the same format as the earlier TXT record.

SRV Service locator
: Generalized service location record, used for newer protocols instead of creating protocol-specific records such as MX.

TXT Text record
: Originally for arbitrary human-readable text in a DNS record. Since the early 1990s, however, this record more often carries machine-readable data, such as specified by RFC 1464, opportunistic encryption, Sender Policy Framework, DKIM, DMARC DNS-SD, etc.

Extended DNS, GeoDNS

Usługi globalne wymagają utrzymywania wielu centrów danych zawierających wiele kopii tych samych danych i oferujących te same usługi chmurowe.

- Problem: w jaki sposób kierować zapytania/żądania klienta do najbliższego centrum danych/najbliższej chmury?
- Rozwiązanie: ENDS/GeoDNS
  - serwer DNS musi określić lokalizację klienta tylko w oparciu o IP: problem rekurencyjnego resolwera i zastosowanie przekierowań przez serwery brzegowe
  - zwiększenie rozmiaru pakietu poza 512 B i dołączanie do zapytań (kontrolowanego) fragmentu adresu IP klienta (*edns-client-subnet*)

CIDR i domena in-addr.arpa (ip6.arpa)[71]

W jaki sposób pogodzić taki system podziału przestrzeni nazw z bezklasowym trasowaniem międzydomenowym (CIDR)?

```
192.0.2.0/25   to organization A
192.0.2.128/26 to organization B
192.0.2.192/26 to organization C
```

```
$ORIGIN 2.0.192.in-addr.arpa.
;
1               PTR     host1.A.domain.
2               PTR     host2.A.domain.
3               PTR     host3.A.domain.
;
129             PTR     host1.B.domain.
130             PTR     host2.B.domain.
131             PTR     host3.B.domain.
;
193             PTR     host1.C.domain.
194             PTR     host2.C.domain.
195             PTR     host3.C.domain.
```

[71] Zob. RFC 2317

DDNS (Dynamic DNS)[72]

W jaki sposób łączyć się z prywatnym serwerem WWW, kamerą internetową, itp., których adresy IPv4 mogą ulegać zmianie?

Dzięki usłudze DDNS oraz odpowiedniemu oprogramowaniu zainstalowanemu na komputerze, który korzysta z usługi DDNS, można łączyć się z serwerem poprzez nazwę, np. odwoływać się poprzez adres typu *mojhost.dyndns.com lub mojhost.no-ip.com*.

[72] http://www.no-ip.com/, http://www.dyndns.com/

Diagnostyka

```
NAME
       dig - DNS lookup utility

SYNOPSIS
      dig [@server] [-b address] [-c class] [-f filename] [-k filename] [-m] \
           [-p port#] [-q name] [-t type] [-v] [-x addr] \
           [-y [hmac:]name:key] [[-4] | [-6]] [name] [type] [class] [queryopt...]

       dig [-h]

       dig [global-queryopt...] [query...]

DESCRIPTION
       dig (domain information groper) is a flexible tool for interrogating DNS
       name servers. It performs DNS lookups and displays the answers that are
       returned from the name server(s) that were queried.

       Unless it is told to query a specific name server, dig will try each of
       the servers listed in /etc/resolv.conf.

       When no command line arguments or options are given, will perform an NS
       query for "." (the root).
```

Zarządzanie demonem named

```
NAME
       rndc - name server control utility

SYNOPSIS
       rndc [-b source-address] [-c config-file] [-k key-file] [-s server] \
                                [-p port] [-q] [-r] [-V] [-y key_id] {command}

DESCRIPTION
       rndc controls the operation of a name server.

       rndc communicates with the name server over a TCP connection, sending
       commands authenticated with digital signatures. In the current versions
       of rndc and named, the only supported authentication algorithms are
       HMAC-MD5 (for compatibility), HMAC-SHA1, HMAC-SHA224, HMAC-SHA256 (default),
       HMAC-SHA384 and HMAC-SHA512. They use a shared secret on each end of the
       connection. This provides TSIG-style authentication for the command request
       and the name server's response. All commands sent over the channel must be
       signed by a key_id known to the server.

       rndc reads a configuration file to determine how to contact the
       name server and decide what algorithm and key it should use.
```

DNS – przykłady konfiguracji

- serwer buforujący
- serwer domeny labul.pl
- serwer domeny fizyka.umk.pl, is.umk.pl, osid.pl, etc

Bezpieczne zdalne administrowanie

Narzędzia zdalnego administrowania:

- niebezpieczne: Remote SHell (rlogin, rsh, rcp), telnet
- bezpieczne: Secure SHell (slogin=ssh, scp, sftp)

Zob.: strony podręcznika dla ssh, sshd, ssh_config, sshd_config oraz D. J. Barrett, R. E. Silverman, R. G. Byrnes, *SSH, the Secure Shell: The Definitive Guide* (O'Reilly, 2005)

Secure Shell: historia

- Tatu Ylönen (Helsinki University of Technology, 1995) tworzy narzędzie do przesyłania poprzez sieci TCP/IP szyfrowanych danych (w tym haseł) – Secure Shell version 1 (SSH1)

  Protokół SSH1 był oparty na chronionej (do września 2001) technologii zdalnego administrowania wykorzystującej szyfrowanie asymetryczne przy pomocy klucza publicznego wg algorytmu RSA (Rivest-Shamir-Adleman).

  SSH1 wykorzystywał do szyfrowania symetryczny algorytm 3DES, tj. każdy 64-bitowy blok danych był szyfrowany potrójnie przy pomocy standardowego algorytmu DES (*Data Encryption Standard*) wykorzystującego 56-bitowy klucz, który do początku 2000 r. był (jako amunicja) objęty obostrzeniami eksportowymi.

- SSH1 cieszył się wielką popularnością wśród administratorów

Secure Shell: historia

- sporządzenie dokumentacji SSH1 uświadamia, że niedoskonałości zastosowanych rozwiązań nie dają się usunąć przy zachowaniu wstecznej zgodności; dodatkowe problemy powodują ograniczenia patentowe i eksportowe
- Ylönen opuszcza HUT i wraz z grupą roboczą SECSH IETF tworzy nowy protokół, SSH2 (wersja wstępna protokołu została udostępniona w 1996 r.)
- Ylönen zakłada własną firmę SSH Communications Security (SCS), żeby rozwijać i sprzedawać wraz ze wsparciem technicznym rozwiązania oparte o Secure Shell użytkownikom komercyjnym; w 1996 pojawia się produkt oparty o SSH2, ale nie wspiera on SSH1[73]

[73] Obecnie firma SSH.COM oferuje *SSH Tectia*, https://www.ssh.com/ssh

Secure Shell: historia

- w 2000 SCS rozszerza prawo nieodpłatnego korzystania z SSH2 na systemach Linux i BSD (oraz pochodnych) oraz dla użytkowników niekomercyjnych
- Björn Grönvall (1999) opracowuje udoskonaloną wersję SSH (versja 1.2.12) o nazwie OSSH opartą na ostatniej (1.12) bezpłatnej wersji SSH1
- grupa programistów (m.in. Markus Friedl, Theo de Raadt, Aaron Campbell) pracująca nad włączeniem SSH do OpenBSD 2.6 wykorzystuje OSSH i tworzy (po dwóch miesiącach, grudzień 1999) OpenSSH 1.2.2, tj. wersję oprogramowania, która jest łatwo przenaszalna na inne platformy i nie zawiera chronionych prawem algorytmów
- w czerwcu 2000, wraz z wersją OpenBSD 2.7, pojawia się OpenSSH 2.0 całkowicie zgodny z SSH w wersji 2.0[74]

[74] http://www.openssh.org/

Secure Shell: historia

- 2005 – SCS wprowadza na rynek SSH G3, nowe wdrożenie SSH wchodzące w skład produktów SSH Tectia[75] (zachowana zgodność z SSH wersja 2, rozszerzona i zoptymalizowana architektura, poprawiona wydajność)
- 2006 – SSH osiąga status standardu (RFC 4250-4256, 4335, 4344,4345, 4419, 4432, 4462, 4716, 5656)

SSH wersja 1.x odnosi się do oprogramowania wykorzystującego protokół SSH w wersji 1 i obejmuje realizacje SSH oraz OpenBSD wykorzystujące algorytm RSA.

SSH wersja 2.x odnosi się do oprogramowania obsługującego zarówno RSA, jak i (EC)DSA *((Elliptic Curve) Digital Signature Algorithm)*

[75] http://www.ssh.com

Zalety SSH

SSH gwarantuje:

- prywatność danych poprzez silne szyfrowanie (`ssh -Q cipher`)[76]
- integralność komunikacji (`ssh -Q mac`)[77]
- uwierzytelnianie: hasła, podpisy oparte o klucz publiczny, certyfikaty OpenSSH, PAM, Kerberos, hasła jednorazowe (S/Key)
- autoryzacja, tj. kontrola dostępu do zasobów (kont, portów TCP, przekazywania sesji X)
- przekazywanie i tunelowanie *(forwarding and tunneling)* w celu szyfrowania sesji TCP/IP *(TCP port forwarding, X forwarding, agent forwarding)*

[76] Domyślnie są dostępne (CentOS 7): chacha20-poly1305@openssh.com, aes128-ctr,aes192-ctr,aes256-ctr, aes128-gcm@openssh.com,aes256-gcm@openssh.com, aes128-cbc,aes192-cbc,aes256-cbc

[77] Domyślnie są dostępne (CentOS 7): umac-64-etm@openssh.com, umac-128-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512-etm@openssh.com, hmac-sha1-etm@openssh.com, umac-64@openssh.com,umac-128@openssh.com, hmac-sha2-256, hmac-sha2-512, hmac-sha1

Rodzina protokołów SSH-2[78]

- SSH Transport Layer Protocol (SSH-TRANS)
  - algorithm negotiation
  - session key exchange
  - session ID
  - server authentication
  - privacy
  - integrity
  - data compression
- SSH Authentication Protocol (SSH-AUTH): client authentication
  - publickey
  - hostbased
  - password
  - gssapi (Generic Security Services API), gssapi-with-mic (Message Integrity Code)
  - keyboard-interactive (S/Key)

[78] https://www.ssh.com/ssh/protocol/

Rodzina protokołów SSH-2

- SSH Connection Protocol (SSH-CONN)
  - channel multiplexing
  - pseudo-terminals
  - flow control
  - signal propagation
  - remote program execution
  - authentication agent forwarding
  - TCP port and X forwarding
  - terminal handling
  - subsystems
- SSH File Transfer Protocol (SSH-SFTP)[79]
  - remote file access
  - file transfer

[79] SCP (*Secure copy*), rsync, FASP (*Fast and Secure Protocol*, także zwany Aspera, używa SSH do kontroli i UDP do przesyłania danych)

Konfiguracja ssh i sshd

Systemowe pliki konfiguracyjne dla klienta i serwera ssh oraz klucze znajdują się w katalogu `/etc/ssh`:

```
moduli
ssh_config
ssh_config.rpmnew
sshd_config
sshd_config.rpmnew
ssh_host_ecdsa_key
ssh_host_ecdsa_key.pub
ssh_host_ed25519_key
ssh_host_ed25519_key.pub
ssh_host_rsa_key
ssh_host_rsa_key.pub
```

Klucze i pliki konfiguracyjne użytkownika znajdują się w katalogu `~/.ssh/`:

```
authorized_keys
config
id_ed25519
id_ed25519.pub
known_hosts
rc
```

Programy: sshd, ssh, scp, sftp, ssh-keygen, ssh-copy-id, ssh-agent, ssh-add

Klucze RSA/DSA/ECDSA

nadawca A (k_S^A, k_P^A)	odbiorca B (k_S^B, k_P^B)	typ komunikacji
$E_{k_P^B}(m) = c$	$D_{k_S^B}(c) = m$	poufna
$E_{k_S^A}(m) = c$	$D_{k_P^A}(c) = m$	uwierzytelniona
$H(m) = N$ $E_{k_S^A}(N) = c_N$ $E_{k_P^B}(m + c_N) = c$	$D_{k_S^B}(c) = m + c_N$ $D_{k_P^A}(c_N) = N$ $H(m) = N$	poufna uwierzytelniona niezmieniona

k_S – klucz prywatny (sekretny), k_P – klucz publiczny (ogólnie dostępny)

E/D – funkcje szyfrujące/deszyfrujące, H – funkcja mieszająca

m – wiadomość, N – skrót wiadomości, c_N – podpis cyfrowy

Klucze ECDSA: zalety[80]

Ed25519 is immune to several side channel attacks:

- No secret array indices. The software never reads or writes data from secret addresses in RAM; the pattern of addresses is completely predictable. The software is therefore immune to cache-timing attacks, hyperthreading attacks, and other side-channel attacks that rely on leakage of addresses through the CPU cache.
- No secret branch conditions. The software never performs conditional branches based on secret data; the pattern of jumps is completely predictable. The software is therefore immune to side-channel attacks that rely on leakage of information through the branch-prediction unit.

Early public-key systems are secure assuming that it is difficult to factor a large integer composed of two or more large prime factors.

For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible: this is the ęlliptic curve discrete logarithm problem"(ECDLP).

The primary benefit promised by elliptic curve cryptography is a smaller key size, [...] a 256-bit elliptic curve public key should provide comparable security to a 3072-bit RSA public key.

[80] A relatively easy to understand primer on elliptic curve cryptography https://ed25519.cr.yp.to

Klucze: generowanie

```
# ssh-keygen -b 2048 -t ecdsa|ed25519|rsa
# ssh-keygen -p -f ~/.ssh/id_ecdsa|ed25519|rsa
```

```
# cat ~/.ssh/id_ecdsa
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEICRxf4Uir0LzDZzrBi49i4fx8eHfLH+nbxbQrJT7PmzLoAoGCCqGSM49
AwEHoUQDQgAEOziPZdmAznPOusFFqboyxIztuPguTfxbHogFoo8+hWD7W3UMTZp0
aeG5dazTSt4jtbZ+oSXIe2Ru5JNDNUnN+g==
-----END EC PRIVATE KEY-----

# cat ~/.ssh/id_ecdsa.pub
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAy\
NTYAAABBBDs4j2XZgM5zzrrBRam6MsSM7bj4Lk38Wx6IBaKPPoVg+1t1DE2adGnh\
uXWs00reI7W2fqElyHtkbuSTQzVJzfo= jkob@scobie.fizyka.umk.pl

# cat ~/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACDn4g9515Co2kMsIKpz9aTzvH69jC/T6R1Uw9NTY9/x/wAAAJDgcMI/4HDC
PwAAAAtzc2gtZWQyNTUxOQAAACDn4g9515Co2kMsIKpz9aTzvH69jC/T6R1Uw9NTY9/x/w
AAAEAxdPf6Rtg+e4Blx7Is9Wq3l6cuO8GIDSJ3zfEmknxHj+fiD3nXkKjaQywgqnP1pPO8
fr2ML9PpHVTDO1Nj3/H/AAAAC2prb2JAc2NvYmllAQI=
-----END OPENSSH PRIVATE KEY-----

# cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC11ZDI1NTE5AAAAIOfiD3nXkKjaQywgqnP1pPO8fr2ML9PpHVTDO1Nj3/H/ jkob@scobie
```

Klucze publiczne

Pliki `/etc/ssh/known_hosts` i `~/.ssh/known_hosts` zawierają klucze publiczne serwerów, z którymi nawiązane zostało połączenie:

```
wieslaw16,wieslaw16.fizyka.umk.pl,w16,w16.fizyka.umk.pl,158.75.5.140 ssh-rsa AAAAB3N\
zaC1yc2EAAAABIwAAAQEAsQ+V6lCwDmrBRr3WrlnsU3nonwBLFuNXHDBEOqJkA3oCyxNoSBux7DNuOOzCmH9U\
diQimL5tT1uNtVbJpJcHksjAYqJagtva/7SEwDwgsFkwnir9Q86f3TMHIYmFoGVvprQPWCkeFOgasjYKyC1hl\
aQab5cfn3wR3utV+HRsxwF19JFJrP5YH61yPZw9xy19RhxwlGyfQTTNQu2+ymoMtgQT6SpoZvWTdJOFtDTCHW\
Mu5B9cIBw4FRKCdAC1BzTZut2wDpzWSZ8XMcC1EC/KvU5inGNulwnxVZ+OdKLt5XnrFy60yPmtKcwYNTElPuM\
mJ7wsZ+PnVISKPr5UY+mqjw==
```

Plik `~/.ssh/authorized_keys` zawiera klucze publiczne użytkownika, który ma prawo rejestrować się/wykonywać komendy na serwerze bez podawania hasła:

```
ssh-dss AAAAB3NzaC1kc3MAAACBAJoDNiE+ePb1dj+jaEY4HiWeyGPswJtjYF4/lF5OU6qNX6TGUJN2/7rJf\
...
hQ4JbslJasdmg1hJJVHBpqG2M3VPpldbuMRCbvu1Nn5MzNK2qu96hN8b35R9t7iUI+5eI5ltovnQ==

command="/bin/ls" ssh-dss AAAAB3NzaC1kc3MAAACBAJoDNiE+ePb1dj+jaEY4HiWeyGPswJtjYF4/lF5\
...
ljaSdmg1hJJVHBpqG2M3VPpldbuMRCbvu1Nn2MzNK2qu96hN8b35R9t7iUI+5eI5ltovnQ== jkob@scobie
```

```
# ssh-copy-id -i ~/.ssh/id_rsa  host
```

StrictHostkeyChecking[81]

```
# ssh -o "StrictHostkeyChecking=ask|yes|no|off|accept-new" host
```

Key found?	Match?	Strict?	Action
no	–	yes	warn and fail
no	–	no	add key and connect
no	–	accept-new	add key and connect
no	–	ask	ask whether to add key and to connect
yes	yes	–	connect
yes	no	yes	warn and fail
yes	no	no	warn and connect
yes	yes	accept-new	connect
yes	no	accept-new	fail to connect
yes	no	ask	warn and ask whether to connect

```
# ssh -o "NoHostAuthenticationForLocalHost=yes" localhost

# grep NoHostAuthenticationForLocalHost ~/.ssh/config
NoHostAuthenticationForLocalHost yes
```

[81] Opcja `StrictHostkeyChecking=accept-new` dostępna w wersji OpenSSH_8.1p1 (Fedora 31).

CheckHostIP

```
# ssh -o "CheckHostIP=no" polon
# tail -1 ~/.ssh/known_hosts
polon ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYT...

# ssh -o "CheckHostIP=yes" polon
# tail -1 ~/.ssh/known_hosts
158.75.5.226 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYT...

# ssh -o "CheckHostIP=yes" polon.fizyka.umk.pl
# tail -1 ~/.ssh/known_hosts
polon.fizyka.umk.pl ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYT...


# rm ~/.ssh/known_hosts
# ssh polon.fizyka.umk.pl
# tail -1 ~/.ssh/known_hosts
polon.fizyka.umk.pl,158.75.5.226 ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYT...
```

VerifyHostKeyDNS

Hosty mogą być weryfikowane przez DNS, jeśli w bazie znajdują się rekordy IN SSHFP zawierające skróty kluczy publicznych wskazanych hostów:

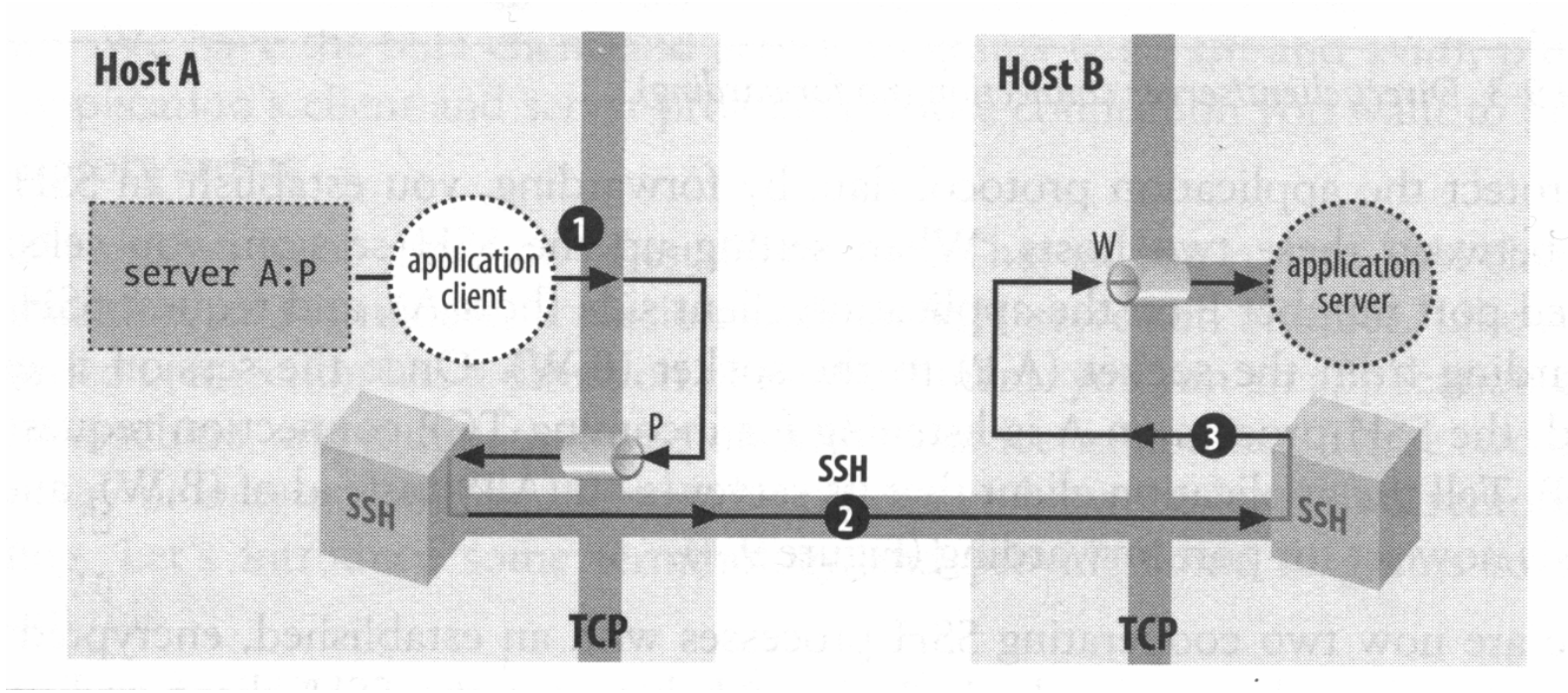
```
# ssh-keygen -r tor7 [-f /etc/ssh/ssh_host_rsa_key.pub]
tor7 IN SSHFP 1 1 9d27e33d92cf409e5f3b53c6f3c0ad16f171c74b
# ssh-keygen -r tor -f /etc/ssh/ssh_host_dsa_key.pub
tor7 IN SSHFP 2 1 94874da00a25a78f0dc0ce97044f6e48f9217861

# ssh -o "VerifyHostKeyDNS=yes" tor7.fizyka.umk.pl
The authenticity of host 'tor7.fizyka.umk.pl (158.75.5.91)' can't be established.
ECDSA key fingerprint is SHA256:h9CW8f24cYNXU3EJobzUmtPfufvmYUqwjHlFj31aLRw.
Matching host key fingerprint found in DNS.
ECDSA key fingerprint is MD5:d9:98:64:05:91:dd:7b:d7:41:2f:23:ec:e7:6d:a6:c2.
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?

# dig tor7.fizyka.umk.pl sshfp
;; ANSWER SECTION:
tor7.fizyka.umk.pl.     60      IN      SSHFP   4 2 ED9DF4E5C1306A24A839D33ABB8A64791A083347746C84DF
tor7.fizyka.umk.pl.     60      IN      SSHFP   3 1 9773DA8B7BFFF6BB6DC99201976D60EC8F0BD68B
tor7.fizyka.umk.pl.     60      IN      SSHFP   3 2 87D096F1FDB8718357537109A1BCD49AD3DFB9FBE6614AB0
tor7.fizyka.umk.pl.     60      IN      SSHFP   4 1 8CCF072291329F7A0D052553713E0942218C6A1C
```

SSH: dzielenie połączenia[82]

```
Tworzenie połączenia 'master':
# ssh -S /tmp/ssh-tor -M -f tor sleep 200

Tworzenie połączenia 'slave':
# ssh -S /tmp/ssh-tor tor

Jeśli ~/.ssh/config zawiera:

host tor-m
  hostname tor.fizyka.umk.pl
  Controlpath /tmp/ssh-tor
  ControlMaster yes

host tor-s
  hostname tor.fizyka.umk.pl
  Controlpath /tmp/ssh-tor

to tworzenie połączeń master/slave można uprościć:
# ssh -f -N tor-m
# ssh [-O check|exit] tor-s
```

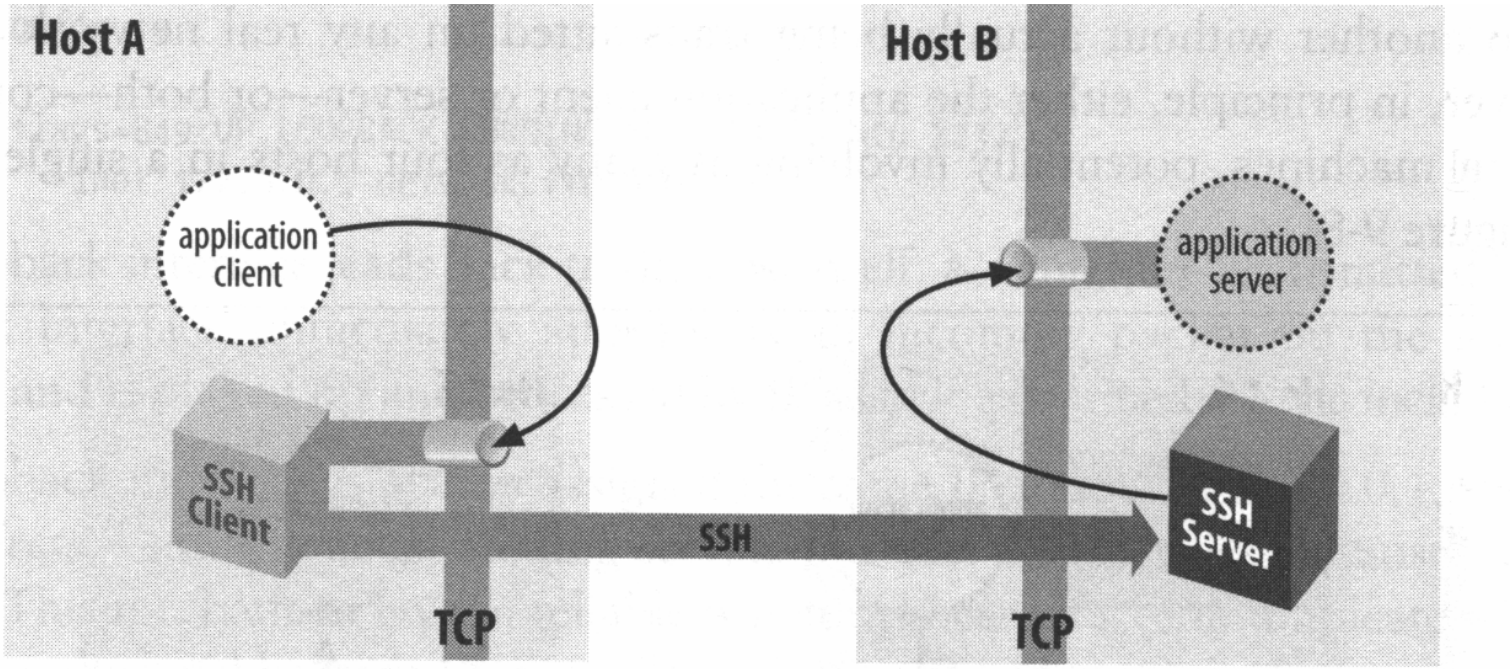
[82] Zob. http://jkob.fizyka.umk.pl/students/sa4k/cw/ssh.html, zad. 3-4.

SSH: Przekazywanie połączeń




```
ssh -L P:localhost:W  B
ssh -L P:B:W          B
```

SSH: lokalne przekazywanie połączeń



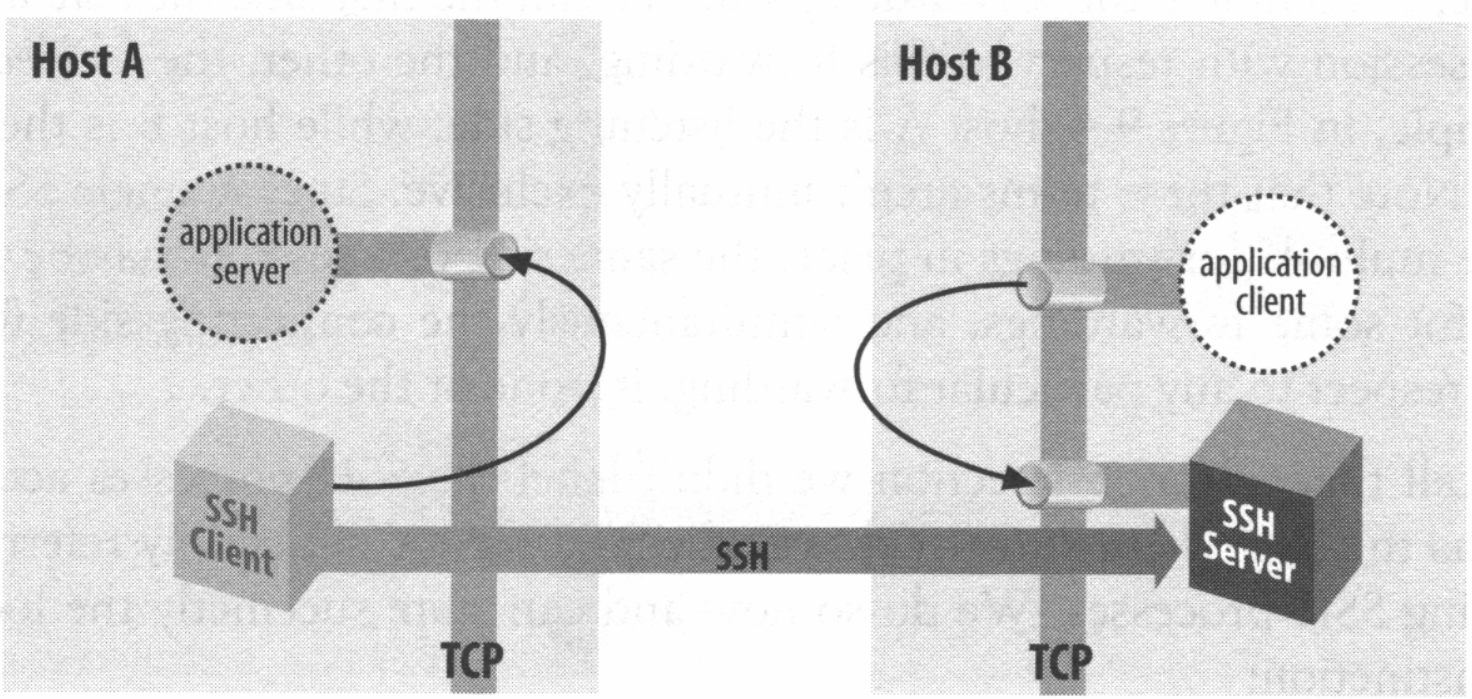
Lokalne przekazywanie połączeń charakteryzuje się tym, że klient usługi oraz strona nasłuchująca znajdują się po stronie klienta SSH.

SSH: lokalne przekazywanie połączeń – przykład[83]

```
Wykonanie na hoście 192.168.142.1 (klient) komendy
# ssh [-v] -N -L 2222:localhost:22 user@158.75.5.226
sprawia, że po stronie klienta
[root@centos7-1 ~]# ss -ntlp | grep 2222
LISTEN     0      128    127.0.0.1:2222          *:*                 users:(("ssh",pid=14204,fd=5))
LISTEN     0      128        ::1:2222         :::*                  users:(("ssh",pid=14204,fd=4))

[root@centos7-1 ~]# ss -ntp
ESTAB      0      0      192.168.142.1:55942   158.75.5.226:22       users:(("ssh",pid=14204,fd=3))

Po stronie serwera mamy
ESTAB      0      0      158.75.5.226:22       158.75.5.140:55942    users:(("sshd",pid=16440,fd=3),...

Wykonanie komendy
# ssh -p 2222 user@localhost
powoduje po stronie serwera pojawienie się dwóch dodatkowych połączeń
ESTAB      0      0          ::1:45832                ::1:22        users:(("sshd",pid=16440,fd=9))
ESTAB      0      0          ::1:22                   ::1:45832     users:(("sshd",pid=18026,fd=3),...
Pierwsze jest związane z obsługą tunelowania, a drugie -- z kolejną sesją ssh.

Komenda
# ssh [-v] -g -N -L 2222:localhost:22 user@158.75.5.226
powoduje udostępnienie portu 2222 nie tylko lokalnym procesom:
LISTEN     0      128          *:2222          *:*                 users:(("ssh",pid=12133,fd=4))
LISTEN     0      128         :::2222         :::*                 users:(("ssh",pid=12133,fd=5))
```

[83] Zob. http://jkob.fizyka.umk.pl/students/sa4k/cw/ssh.html, zad. 7-8.

SSH: zdalne przekazywanie połączeń



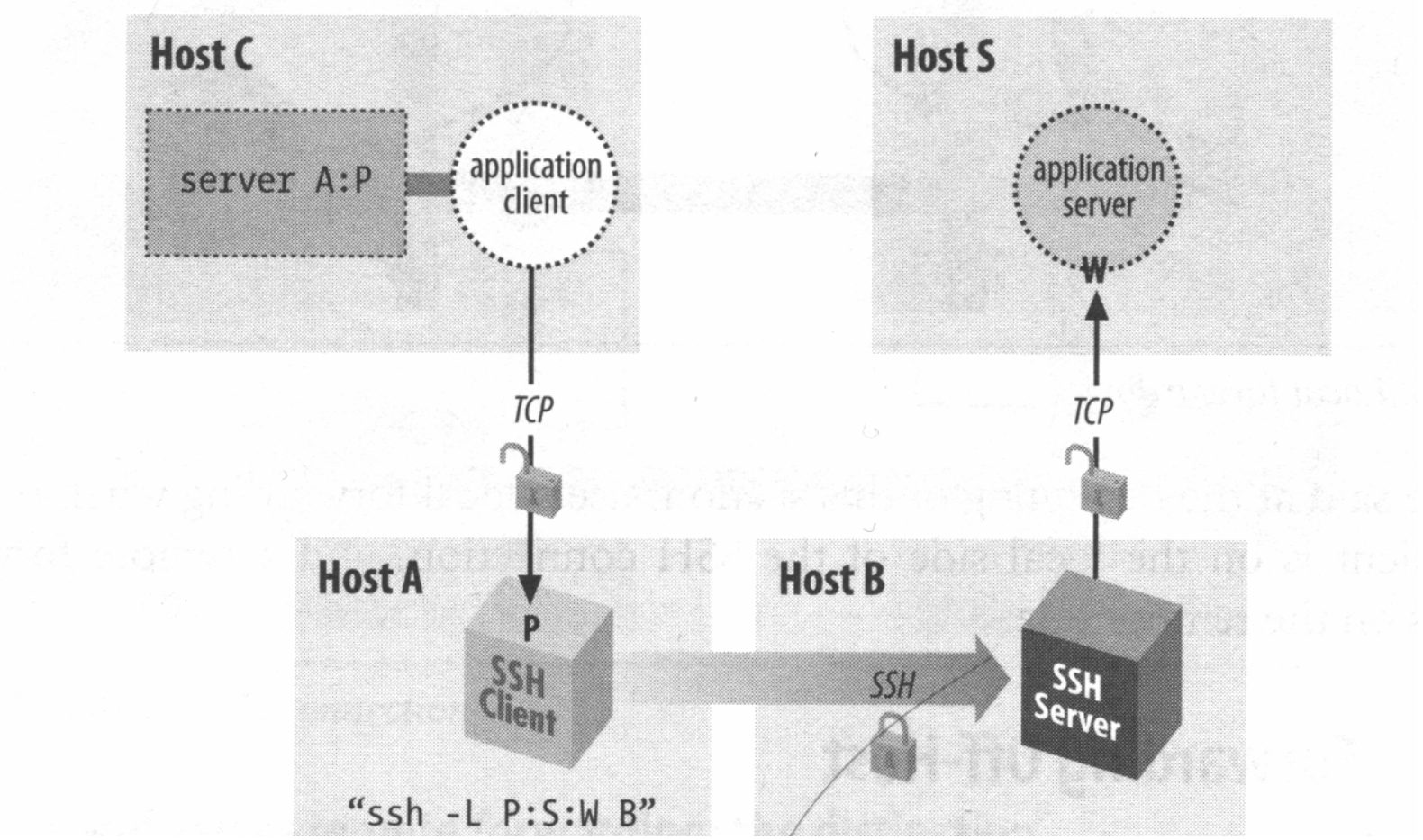
Zdalne przekazywanie połączeń charakteryzuje się tym, że klient usługi oraz strona nasłuchująca znajdują się po stronie serwera SSH.

SSH: zdalne przekazywanie połączeń – przykład[84]

```
Wykonanie na hoście 192.168.142.1 (klient) komendy
# ssh [-v] -N -R 2222:localhost:22 192.168.142.2

powoduje, że po stronie serwera 'ss -nltp' pokazuje
LISTEN     0      128    127.0.0.1:2222          *:*                  users:(("sshd",pid=20330,fd=10))
LISTEN     0      128        ::1:2222           :::*                  users:(("sshd",pid=20330,fd=9))

Po wykonaniu na serwerze (192.168.142.2)
# ssh -p 2222 labul@127.0.0.1

uzyskuje się dostęp do maszyny 192.168.142.1. Na maszynie lokalnej i zdalnej pojawiają się
nowe połączenia.
```

[84] Zob. http://jkob.fizyka.umk.pl/students/sa4k/cw/ssh.html, zad. 9.

SSH: przekazywanie portów poza hosta[85]



[85] *off-host port forwarding*, zob. http://jkob.fizyka.umk.pl/students/sa4k/cw/ssh.html, zad. 10-12.

SSH: przekazywanie portów poza hosta

Problem: Serwer WWW jest dostępny tylko dla hostów z wewnętrznej, wydzielonej sieci. Jak zapewnić do niego dostęp z zewnątrz sieci, jeśli użytkownicy mogą się logować przez SSH tylko na jeden, znajdujący się wewnątrz sieci serwer dostępowy proxyWWW?

Rozwiązania:

- `ssh [-g] -L 8080:www:80 proxyWWW` i http://localhost:8080

  Wady:

  - obsługa wirtualnych hostów (reakcja serwera www zależy od nazwy hosta)
  - odwołania bezwzględne, które zawierają adres hosta
  - odwołania do innego wewnętrznego serwera www lub strony dostępnej na innym porcie

- `ssh [-CN] -g -D 1080 proxyWWW` i rekonfiguracja przeglądarki (serwer proxy + protokół SOCKS (RFC 1928))

SSH: tunele tun/tap

OpenSSH wspiera tworzenie szyfrowanych tuneli pomiędzy dwoma hostami: lokalnym i zdalnym, jeśli serwer na to zezwala (`PermitTunnel yes`) i ma się uprawnienia użytkownika root.

Tunel może być typu *point-to-point* (tun, warstwa 3) lub *ethernet* (tap, warstwa 2).

```
# Komendy
ssh -w any rhost
ssh -w 0:0 rhost
ssh -w 10:10 rhost

# tworzą na lokalnym i zdalnym hoście interfejsy tun.
ip link show dev tun0
daje taki przykładowy wynik
7: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 500
    link/none

# Obu końcom tunelu należy nadać adresy IP oraz zmodyfikować tablice routingu na obu hostach.

# Komenda
ssh -o "Tunnel=Ethernet" -w any rhost
# tworzy na lokalnym i zdalnym hoście interfejsy tap:
6: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT qlen 1000
    link/ether 8a:a1:be:4f:df:d4 brd ff:ff:ff:ff:ff:ff
```

SSH: ProxyCommand i ProxyJump

Logowanie via jeden lub więcej serwerów ssh:

```
# ssh -t polon ssh tor
# ssh polon ssh -tt tor
# ssh -J polon,w16 tor
```

Uproszczenie: plik `/.ssh/config` i opcje `ProxyCommandi ProxyJump`:

```
Host tor-1
   Hostname tor.fizyka.umk.pl
   ProxyCommand ssh [-v] -W tor:22 polon.fizyka.umk.pl
Host tor-2
   Hostname tor.fizyka.umk.pl
   ProxyJump polon.fizyka.umk.pl
Host centos7-9
   Hostname centos7-9.fizyka.umk.pl
   User root
   IdentityFile /home/jkob/.ssh/id_4_centos
   ProxyCommand ssh -l jkob -W %h:%p w16.fizyka.umk.pl
```

Konfiguracja ssh i sshd

Analiza plików konfiguracyjnych:

- `sshd_config`
  - X11Forwarding
  - AllowTcpForwarding
  - GatewayPorts
  - Subsystem
  - AllowUsers|DenyUsers
  - Match User|Group|Host|Address
- `ssh_config`
  - ForwardX11
  - ForwardAgent
  - ProxyCommand

Sekwencje unikowe SSH

Domyślnie znak tyldy (~) jest traktowany jako początek sekwencji unikowej.[86]

Lista dostępnych sekwencji unikowych:

```
~.   - zakończ połączenia
~B   - wyślij BREAK do zdalnego systemu
~C   - otwórz linię komend, żeby dodawać/usuwać przekazywanie portów
~R   - żądanie natychmiastowej zmiany klucza
~^Z - zawieś połączenia
~#   - wypisz listę wszystkich przekazywanych połączeń
~&   - przenieś sesje w tło (w czasie czekania na zakończenie połączenie)
~?   - wypisz wiadomośc pomocy
~~   - prześlij znak unikowy
```

[86] Komenda `ssh -e% serwer` zmienia tyldę na znak procentów.