KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Administrowanie sieciami lokalnymi i serwerami

Jacek Kobus

Wydział Fizyki, Astronomii i Informatyki Stosowanej UMK (2015/2016)

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

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 LILO i GRUB/GRUB2
   - protokoły BOOTP (ARP/RARP), DHCP, TFTP, PXE
3. Wykrywanie i dynamiczna konfiguracja urządzeń (udev)
4. Konfiguracja urządzeń sieciowych
5. Bezpieczne rejestrowanie się, budowa bezpiecznych tuneli (SSH)
6. Usługa nazw domenowych (DNS)
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. *Modern Operating Systems*. Prentice-Hall International, Upper Saddle River, 1992.

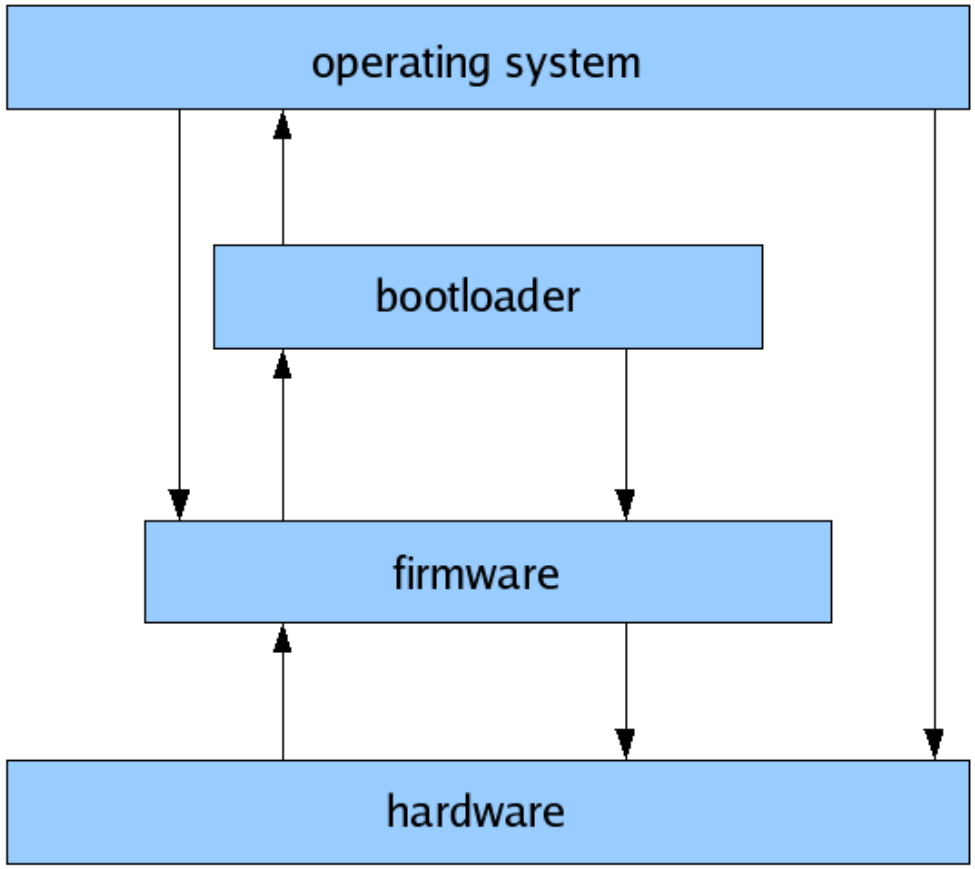
Uruchamiania systemu operacyjnego[1]

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 your situation by your 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 FFF:0000h i wykonuje skok do wskazanego adresu, gdzie jest BIOS; rozpoczyna się POST
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 000:0472 zawiera liczbę 1234h.

9. Pobranie ustawień z CMOS
10. Poszukiwanie systemu do uruchomienia (kolejność bootowania); poszukiwany jest sektor, który zawiera program ładujący fazy pierwszej (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)

Uruchamianie systemu operacyjnego:
http://www.ibm.com/developerworks/linux/library/l-linuxboot/
http://www.almesberger.net/cv/papers/ols2k-9.ps.gz
http://www.fizyka.umk.pl/~jkob/ssk-admin/bootinglinux-current.pdf

Uruchamianie systemu komputerowego: faza II

- Ładowanie do pamięci i przekazywanie sterowania do głównego kodu startowego zapisanego w pierwszym bloku (sektorze, C=0, S=1, H=0) dysku, tzw. MBR-e (*Master Boot Record*) przy pomocy funkcji przerwania INT 19:[3]

  ```
  INT 19: 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; problemy:

  - *1024 limit* (504 MiB): *enhanced* BIOS wspierający translację
  - *8 GB limit*: funkcja przerwania INT 13h zastąpiona przez funkcję *extended* INT 13h)[4]

- kopiowanie instrukcji z pierwszego sektora aktywnej partycji (*volume boot sector*) i ich wykonanie

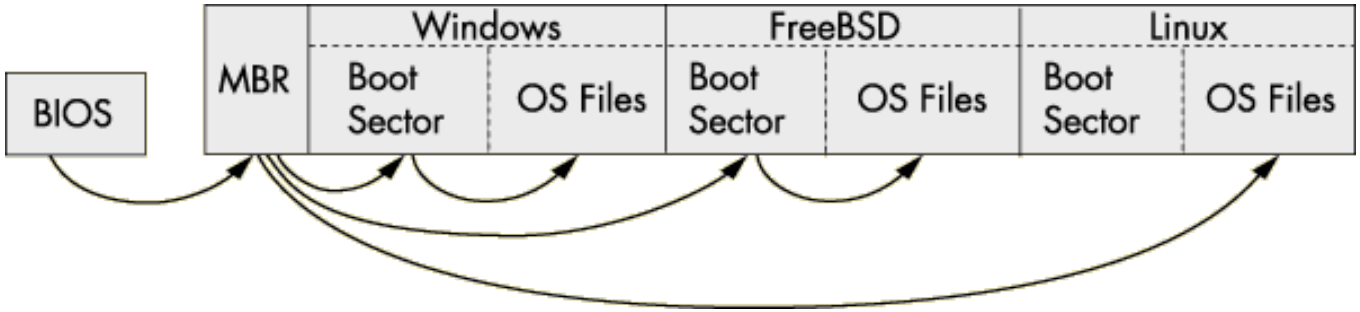
[3] http://www.uv.tietgen.dk/staff/mlha/pc/Prog/ASM/INT/
[4] http://www.pcguide.com/ref/hdd/bios/sizeMB504-c.html

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[5]



[5]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)
- 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 – implementacje standardu Open Firmware (IEEE 1275-1994)
- SLOF (*Slimline*) – Open Firmware wytwarzany przez IBM, wspiera architekturę PowerPC

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 and Windows Server 2008 R2

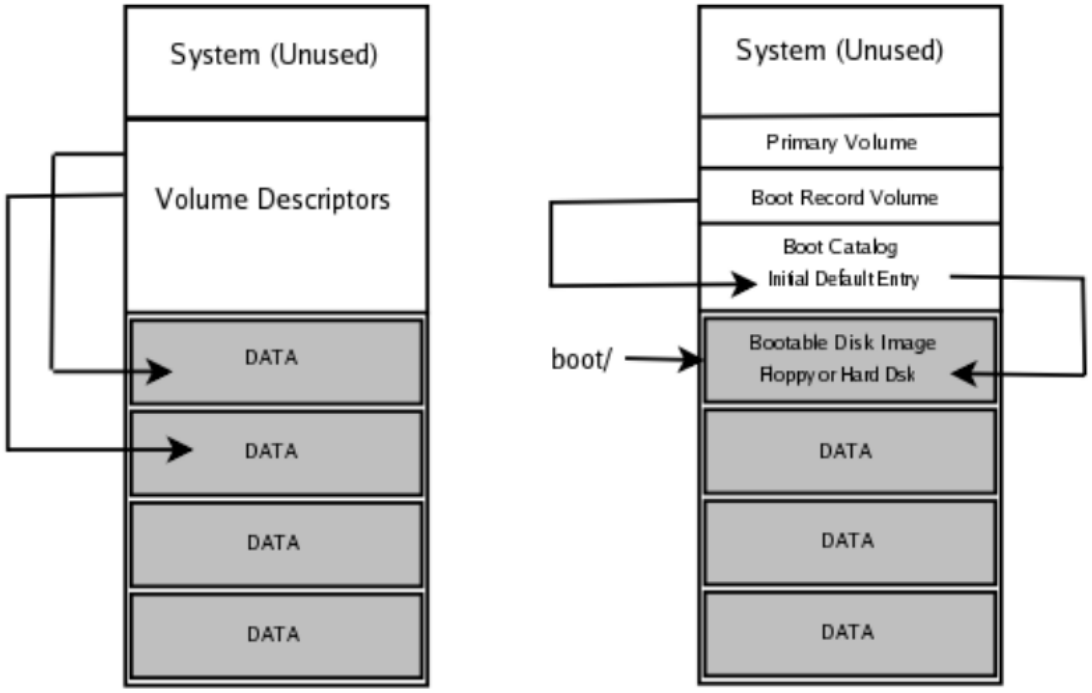
Programy ładujące *LINUX fazy drugiej[6]

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

  Standard EI 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 tego 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 (Basic Input/Optput System)

- 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)

Coreboot

- 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 dostarcza tych wywołań (nowoczesne systemy operacyjne z nich nie korzystają)
- wersja x86 wykonuje się w trybie 32-bitowym po wykonaniu 10 instrukcji

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 w połowie lat 1990 i rozwijanej przez Unified EFI Forum. UEFI wspiera szyfrowanie, uwierzytelnianie sieciowe, *User Interface Architecture (Human Interface Infrastructure)*. Najnowsza wersja: 2.5 (04/2015).

Zalety UEFI

- 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

- 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

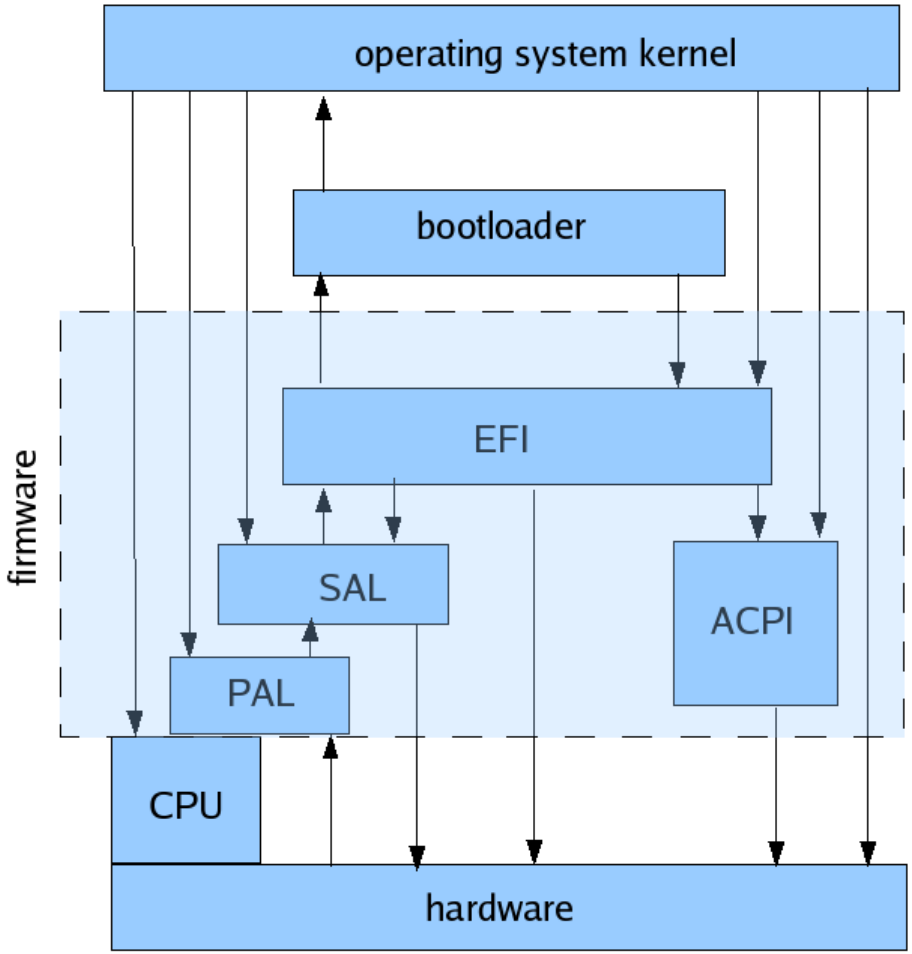
- moduły obsługi urządzeń – oprócz modułów zależnych od architektury specyfikacja EFI określa środowisko programów obsługi urządzeń niezależnych od architektury
- *boot manager* – menadżer odpowiedzialny za wybr i uruchamienie SO
- dyski – wsparcie dla tablic partycji MSDOS oraz GPT (*GUID Partition 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*)

Programy ładujące EFI

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; `CONFIG_EFI_STUB=y`[8]

[8] https://wiki.archlinux.org/index.php/UEFI_Bootloaders; zob. /boot/config-3.19.8-.fc20.x86_64



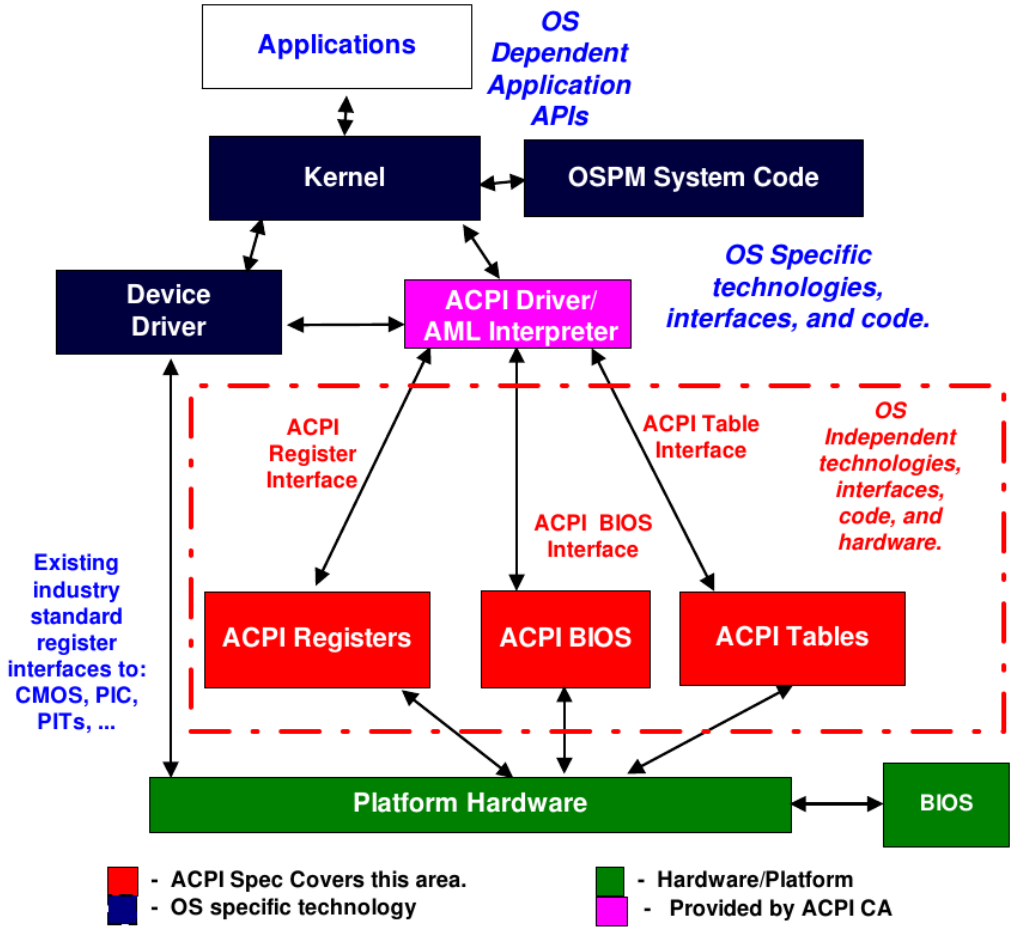
IA-64 firmware components

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 (implementowanych 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*)
- patrz: `/proc/acpi` oraz `/sys/firmware/acpi`

ACPI – schemat[9]



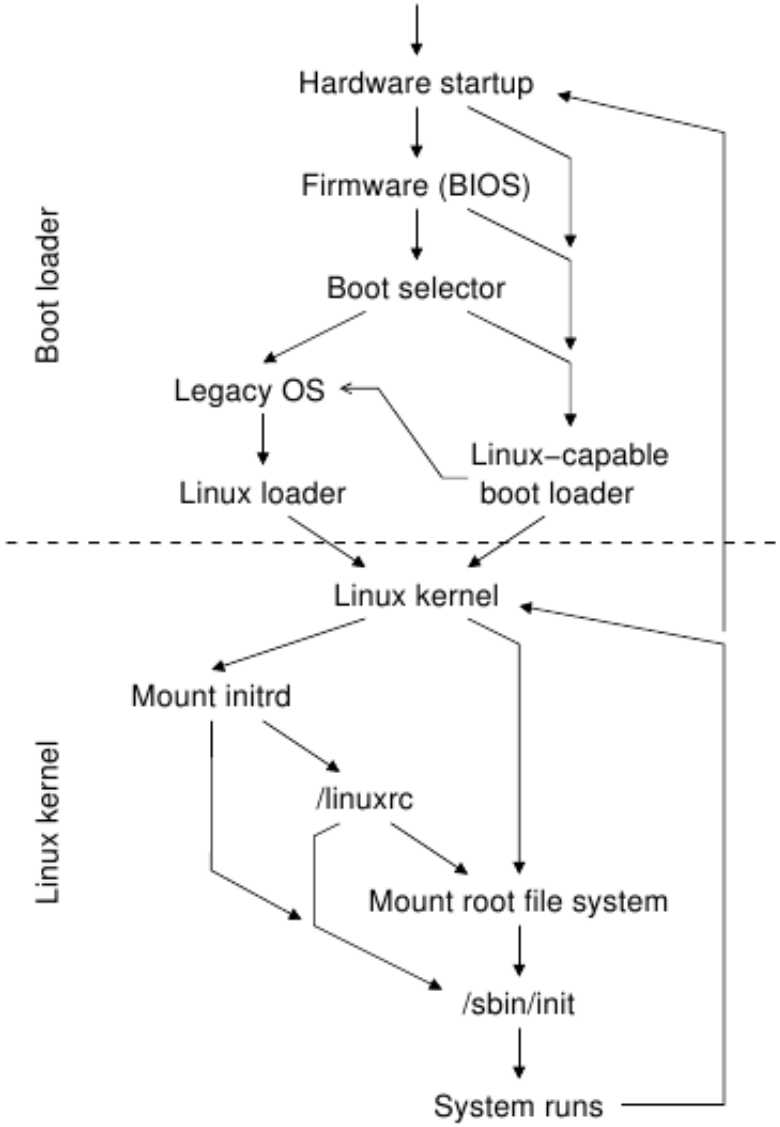
[9] http://www.acpi.info/presentations/ACPI_Overview.pdf

ACPI

Zalety ACPI:

- 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)

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


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

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

`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` (inicjacja pozostałych podsystemów jądra, zwolnienie pamięci używanej przy uruchamianiu jądra)
- pojawia się znak zachęty (tekstowy lub graficzny)

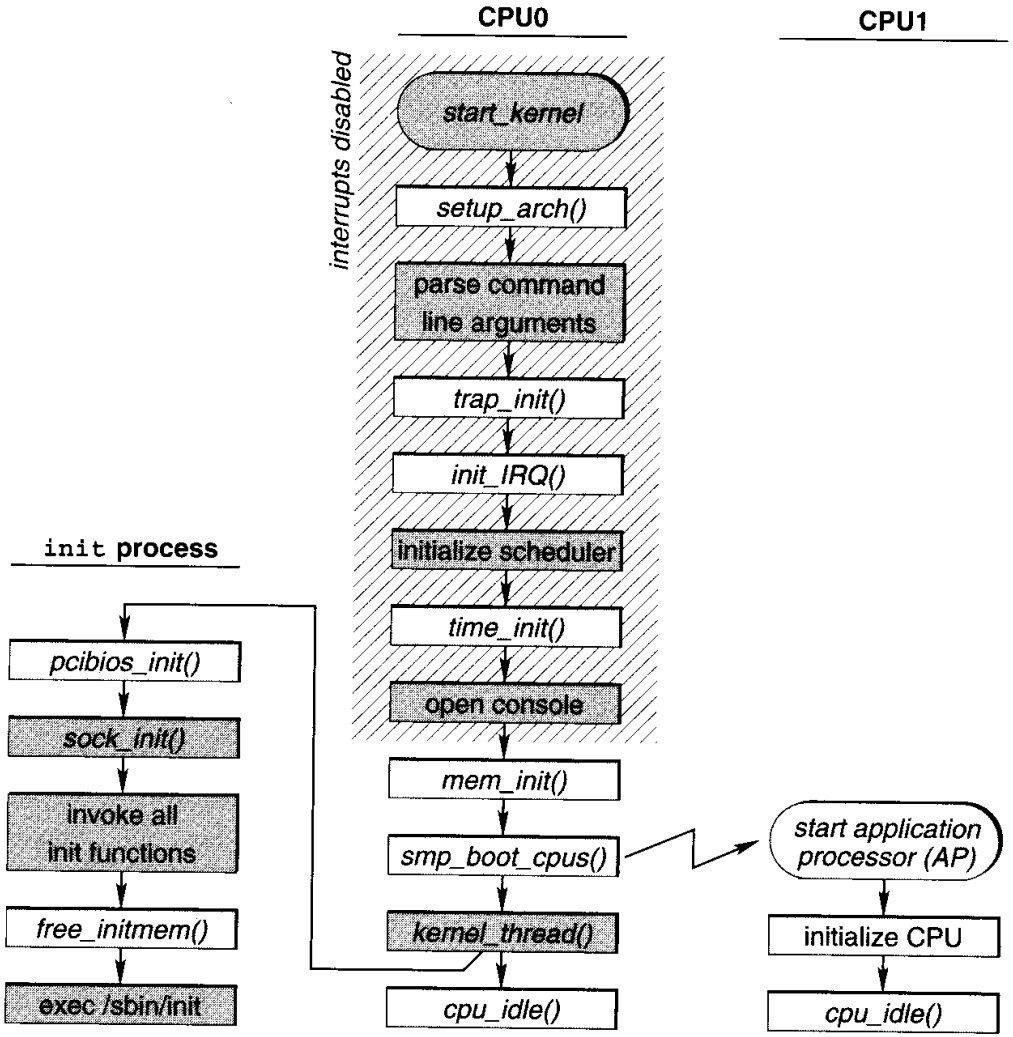


Figure 10.16. Overview of the Linux bootstrap procedure.

Inicjowanie pracy systemu (`/sbin/init`)

- `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.

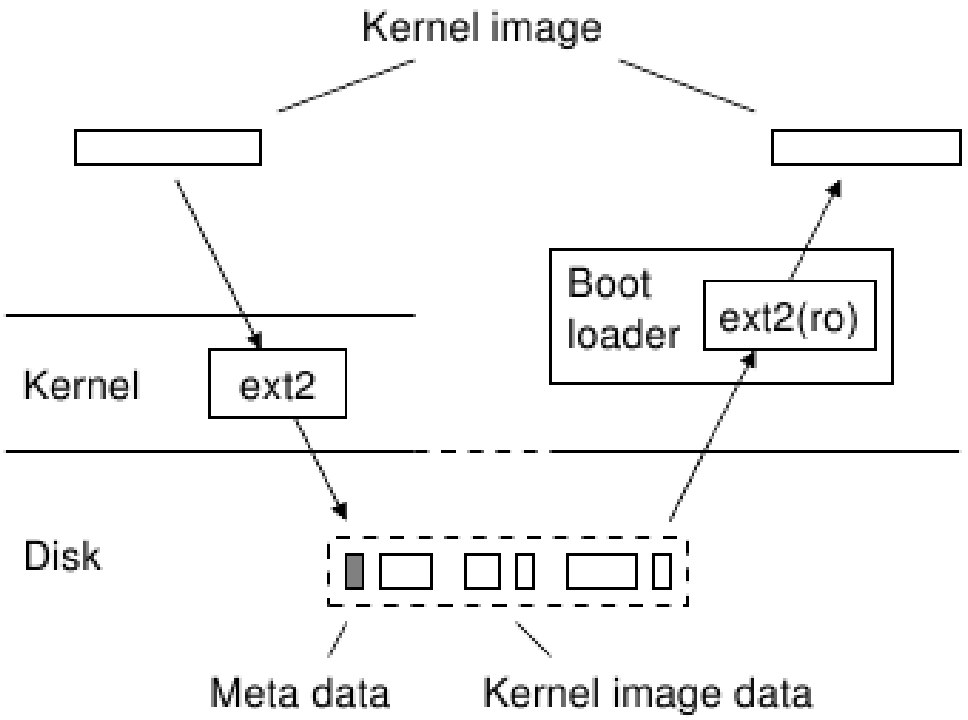
Patrz: http://www-128.ibm.com/developerworks/library/l-linuxboot/index.html

Rodzaje programów uruchomieniowych

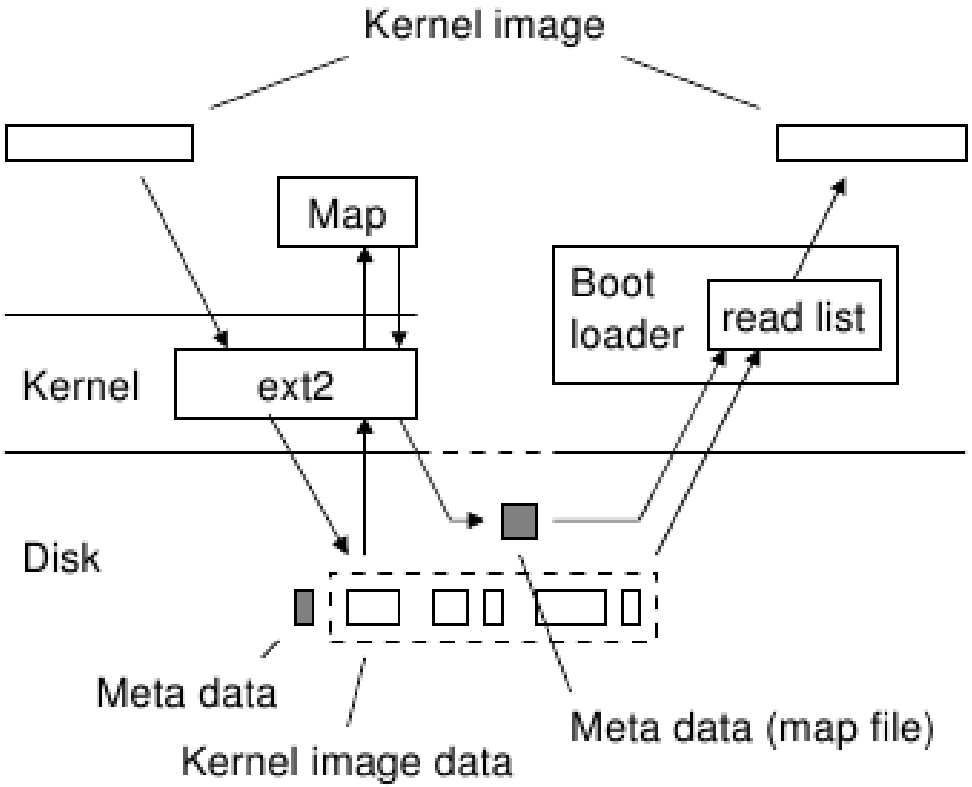
- programy specjalizowane przystosowane do uruchamiania systemu z konkretnego nośnika, np. syslinux, isolinux
- programy ogólnego przeznaczenia pracujące pod kontrolą innego systemu operacyjnego (systemu operacyjnego gospodarza), np. loadlin
- programy ogólnego przeznaczenia zależne od systemu plików,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

/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 – nowoczesny i wszechstronny program ładujący

GRUB *GNU GRand Unified Bootloader* – 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, NTFS, ReiserFS, UDF, XFS), wspiera cpio, tar
- wspiera automatyczną dekompresję plików (gzip, xz)
- niezależny od sposobu translacji geometrii dysku
- prawidłowo rozpoznaje wielkość dostępnej pamięci RAM
- wsparcie dla obsługi trybu LBA (*Logical Block Addressing*)
- może modyfikować tablicę partycji i zmieniać kolejność partycji, które widzi system operacyjny
- umożliwia przekazywanie parametrów do jądra uruchamianego systemu

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[10]

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. dzięki temu 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

[10] 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

```
...
menuentry 'Fedora (3.2.5-3.fc16.i686.PAE)' \
              --class fedora --class gnu-linux --class gnu --class os {
   load_video
   set gfxpayload=keep
   insmod gzio
   insmod lvm
   insmod part_msdos
   insmod ext2
   set root='(system-slash)'
   search --no-floppy --fs-uuid --set=root c939145c-41eb-4397-b798-c46182e2c0db
   echo 'Loading Fedora (3.2.5-3.fc16.i686.PAE)'
   linux /boot/vmlinuz-3.2.5-3.fc16.i686.PAE root=LABEL=slash ro rd.md=0 rd.dm=0\
       KEYTABLE=us quiet rd.lvm.lv=system/slash rd.lvm.lv=system/swap \
       rhgb rd.luks=0 SYSFONT=latarcyrheb-sun16 LANG=en_US.UTF-8
   echo 'Loading initial ramdisk ...'
   initrd /boot/initramfs-3.2.5-3.fc16.i686.PAE.img
...
}
```

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

GRUB2 – własności[11]

- GRUB2 jest modularny i nie potrzebuje już *stage 1.5*, moduły są ładowane w razie potrzeby (obsługa LVM, RAID)
- partycje są numerowane od 1 i są poprzedzone nazwą typu tablicy partycji
- urządzenia blokowe są numerowane od 0
- instalacja GRUB2 na dysku z tablicą GPT wymaga utworzenia oddzielnej partycji (2 MB) typu bios_grub (*BIOS boot partition*, EF02), gdzie instalowany jest program rozruchowy (ściślej, plik core.img)
- instalacja GRUB2 na dysku z tablicą MSDOS wymaga wolnego miejsca za MBR-m; pierwsza partycja powinna zaczynać się sektora 2048, tzw. problem *MBR gap*[12]

[11] https://docs.fedoraproject.org/en-US/Fedora/22/pdf/Multiboot_Guide/Fedora-22-Multiboot_Guide-en-US.pdf
[12] 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=158.75.5.97:/arc-data/images/diskless/i386/CentOS56
```

Lista parametrów jądra: http://www.kernel.org/doc/Documentation/kernel-parameters.txt
Narzędzia: pxeos, pxeboot

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 -
```

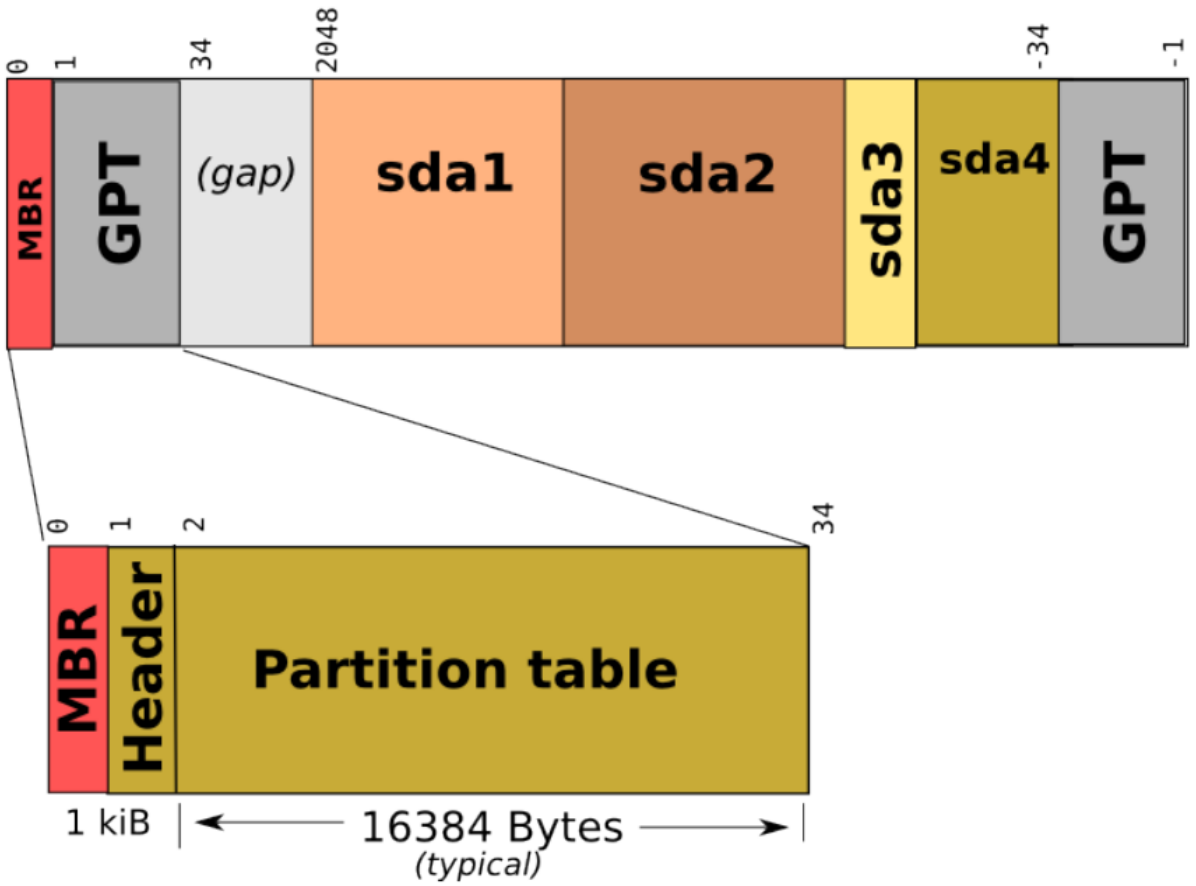
Tablica partycji GPT

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

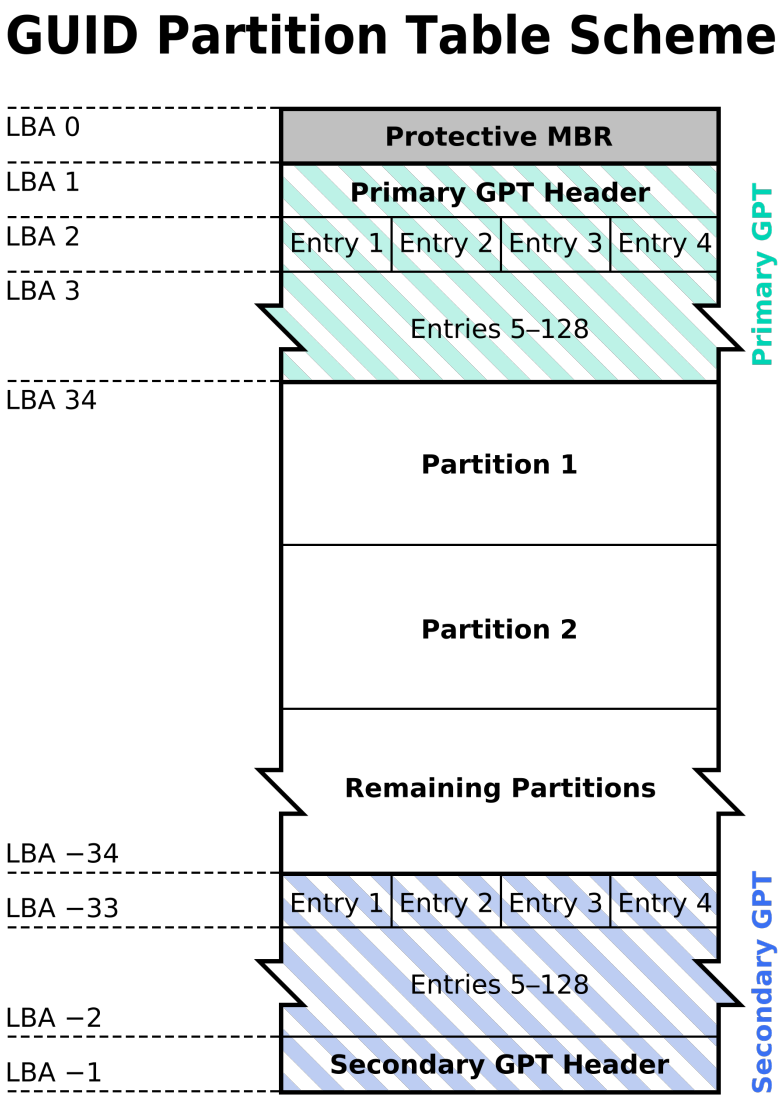
Zalety:

- 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[13]



[13] 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

Narzędzia: parted, gdisk (gpt+inne), blkid

Generowanie UUID: uuidgen [-rt]

Czy GUID jest identyczny z UUID?

Kiedy bootowanie z dysku GPT wymaga systemu wspierającego UEFI?
Zobacz: http://www.rodsbooks.com/gdisk/bios.html

Więcej:
http://www.ata-atapi.com/hiwtab.htm
http://en.wikipedia.org/wiki/GUID_Partition_Table
http://stackoverflow.com/questions/246930/is-there-any-difference-between-a-guid-and-a-uuid
http://www.rodsbooks.com/gdisk/index.html

Praktyczne uwagi dotyczące używania EFI/GPT:
http://www.rodsbooks.com/gdisk/booting.html

Pliki obrazu initrd/initramfs[14]

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.

[14] 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:[15]

```
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>
```

[15]Skrypt ułatwiający modyfikowanie plików initramfs.img: http://www.fizyka.umk.pl/~jkob/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[16]

```
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).
```

[16] 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-item  12 root root 1024 Mar 10 19:20 .
dr-xr-xr-item.  6 root root 3072 Mar 10 19:20 ..
lrwxrwxrwx      1 root root    7 Mar 10 19:20 bin -> usr/bin
drwxr-xr-item   2 root root 1024 Mar 10 19:20 dev
drwxr-xr-item  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-item   2 root root 1024 Mar 10 19:20 proc
drwxr-xr-item   2 root root 1024 Mar 10 19:20 root
drwxr-xr-item   2 root root 1024 Mar 10 19:20 run
lrwxrwxrwx      1 root root    8 Mar 10 19:20 sbin -> usr/sbin
-rwxr-xr-item   1 root root 3017 Mar 10 19:20 shutdown
drwxr-xr-item   2 root root 1024 Mar 10 19:20 sys
drwxr-xr-item   2 root root 1024 Mar 10 19:20 sysroot
drwxr-xr-item   2 root root 1024 Mar 10 19:20 tmp
drwxr-xr-item   7 root root 1024 Mar 10 19:20 usr
drwxr-xr-item   2 root root 1024 Mar 10 19:20 var
```

Zawartość /boot/initramfs-2.6.32-573.7.1.el6.i686

```
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 bin
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 cmdline
drwxr-xr-x  3 jkob jkob 4096 10-02 13:38 dev
-rw-r--r--  1 jkob jkob   19 10-02 13:38 dracut-004-388.el6
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 emergency
drwxr-xr-x  8 jkob jkob 4096 10-02 13:38 etc
-rwxr-xr-x  1 jkob jkob 9007 10-02 19:07 init
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 initqueue
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 initqueue-finished
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 initqueue-settled
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 initqueue-timeout
drwxr-xr-x  9 jkob jkob 4096 10-02 13:38 lib
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 mount
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 netroot
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 pre-mount
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 pre-pivot
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 pre-trigger
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 pre-udev
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 proc
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 sbin
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 sys
drwxr-xr-x  2 jkob jkob 4096 10-02 13:38 sysroot
drwxrwxrwt  2 jkob jkob 4096 10-02 13:38 tmp
drwxr-xr-x  7 jkob jkob 4096 10-02 13:38 usr
drwxr-xr-x  4 jkob jkob 4096 10-02 13:38 var
```

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

```
drwxr-xr-item 12 root root 4,0K 10-21 20:55 .
drwxrwxr-item  4 jkob jkob 4,0K 10-21 20:55 ..
lrwxrwxrwx     1 root root    7 10-21 20:55 bin -> usr/bin
drwxr-xr-item  2 root root 4,0K 10-21 20:55 dev
drwxr-xr-item 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-item  2 root root 4,0K 10-21 20:55 proc
drwxr-xr-item  2 root root 4,0K 10-21 20:55 root
drwxr-xr-item  4 root root 4,0K 10-21 20:55 run
lrwxrwxrwx     1 root root    8 10-21 20:55 sbin -> usr/sbin
-rwxr-xr-item  1 root root 3,0K 10-21 20:55 shutdown
drwxr-xr-item  2 root root 4,0K 10-21 20:55 sys
drwxr-xr-item  2 root root 4,0K 10-21 20:55 sysroot
drwxr-xr-item  2 root root 4,0K 10-21 20:55 tmp
drwxr-xr-item  7 root root 4,0K 10-21 20:55 usr
drwxr-xr-item  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:[17]

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

[17] 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[18]

- 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

[18] 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 ]
             [ -cf config-file ] [ -lf lease-file ] [ -pf pid-file ]
             [ -tf trace-output-file ] [ -play trace-playback-file ]
             [ if0 [ ...ifN ] ]
```

DHCP: działanie

DHCP Lease Stages[19]

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.

[19] 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`

```
authoritative;

ddns-update-style none;
#ddns-update-style interim;
#ddns-update-style ad-hoc;      # deprecated

# 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 158.75.5.250, 158.75.5.252, 158.75.1.4;
    default-lease-time 15000;
    max-lease-time 20000;
    filename "centos-5.7-i386-install/pxelinux.0";
    next-server 158.75.5.97;

    pool {
            range 158.75.105.243 158.75.105.253;
             max-lease-time 1200;
        }
}
```

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     158.75.5.250, 158.75.1.4;
  option domain-name             "fizyka.umk.pl";
  option root-path               "192.168.2.241:/images/xterminals/i386";
  next-server                    192.168.2.241;
  option routers                 192.168.2.241;
  option log-servers             192.168.2.241;
  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
```

PXELINUX

Działanie PXELINUXa wymaga[20]

- 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 /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

[20] 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

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

PXELINUX

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
```

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:[21]

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.

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

Uruchamianie usług: upstart[22]

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.

[22] 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[23]

W nowszych wersjach Linuksa (Debian, Fedora, Gentoo, Ubuntu (>3/2015)), /sbin/init jest dowiązaniem symbolicznym do /usr/lib/systemd/systemd, który przejął kontrolę nad uruchamianiem, wstrzymywaniem i zarządzaniem procesami systemu i usługami.

systemd is a system and service manager for Linux, compatible with SysV and LSB 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
- supports snapshotting and restoring of the system state
- maintains mount and automount points
- implements an elaborate transactional dependency-based service control logic
- it can work as a drop-in replacement for sysvinit

[23] http://fedoraproject.org/wiki/Systemd, http://linuxconfau.blip.tv/file/4696791

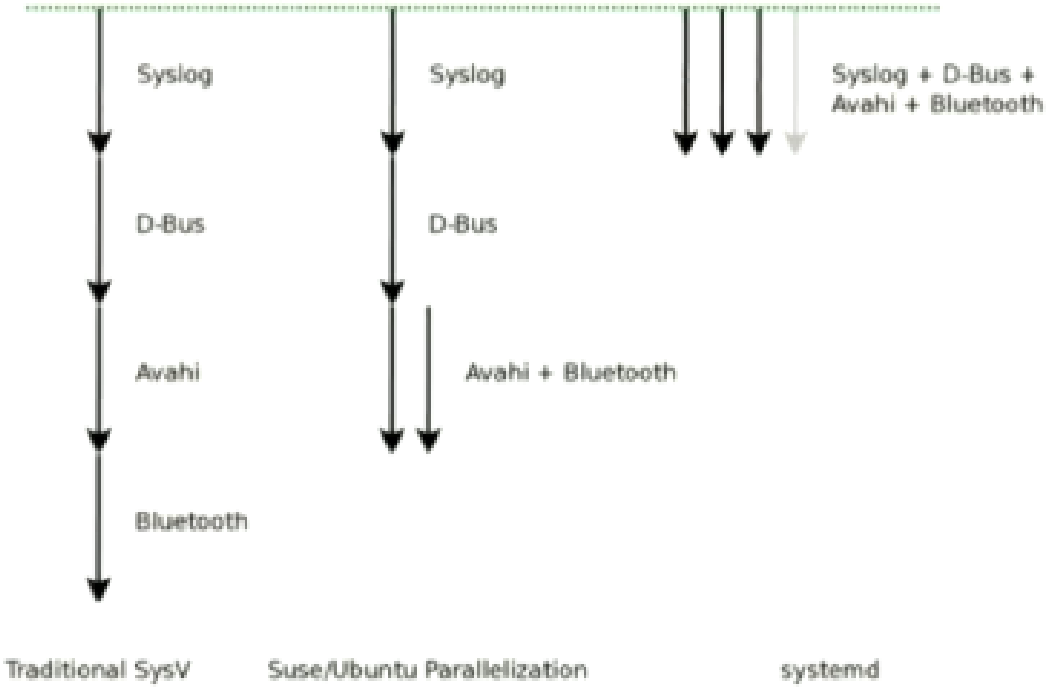
Uruchamianie usług: systemd – gniazda[24]

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.

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

Uruchamianie usług: systemd – zrównoleglenie



Uruchamianie usług: systemd – zalety[25]

- 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*)

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

Uruchamianie usług: systemd – schemat

systemd starts up and supervises the entire system and is based around the notion of units composed of a name and a type and matching a configuration file with the same name and type (e.g. a unit avahi.service has a configuration file with the same name and is a unit encapsulating the Avahi daemon). There are seven different types of units:

- service: these are the most obvious kind of unit: daemons that can be started, stopped, restarted, reloaded.
- socket: this unit encapsulates a socket in the file-system or on the Internet. systemd currently support AF_INET, AF_INET6, AF_UNIX sockets of the types stream, datagram, and sequential packet. It can also support classic FIFOs as transport. Each socket unit has a matching service unit, that is started if the first connection comes in on the socket or FIFO (e.g. nscd.socket starts nscd.service on an incoming connection).
- device: this unit encapsulates a device in the Linux device tree. If a device is marked for this via udev rules, it will be exposed as a device unit in systemd. Properties set with udev can be used as configuration source to set dependencies for device units.

systemd – schemat działania

- mount: this unit encapsulates a mount point in the file system hierarchy.
- automount: this unit type encapsulates an automount point in the file system hierarchy. Each automount unit has a matching mount unit, which is started (i.e. mounted) as soon as the automount directory is accessed.
- target: this unit type is used for logical grouping of units: instead of actually doing anything by itself it simply references other units, which thereby can be controlled together. (e.g. multi-user.target, which is a target that basically plays the role of run-level 5 on classic SysV system; or bluetooth.target which is requested as soon as a bluetooth dongle becomes available and which simply pulls in bluetooth related services that otherwise would not need to be started: bluetoothd and obexd and suchlike).
- snapshot: similar to target units snapshots do not actually do anything themselves and their only purpose is to reference other units.

systemd: configuracja

/usr/lib/systemd/system/vmstat.service:

```
[Unit]
Description=vmstat

[Service]
Type=notify
Environment=LANG=C

ExecStart=/usr/bin/vmstat 1
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGKILL
PrivateTmp=true

StandardOutput=syslog
#StandardOutput=tty
TTYPath=/dev/pts/5

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

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
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
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
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 list-units --type=target


halt -p   |   init 0   |   reboot   |   shutdown -P now
halt|poweroff|reboot|shutdown -f
```

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
```

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

udev – własności[26]

- 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

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

[26] 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

```
NAME
    udev - Linux 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, udevd(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 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 znikają po wyłączeniu/włączeniu systemu)

Katalog /dev nie zawiera plików (typowych) urządzeń sieciowych (np. związanego z obsługą interfejsu eth0).

udev: numeracja urządzeń

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).

Wykrywanie i konfiguracja urządzeń: udev

```
  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

Interfejs SCSI

```
# 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
```

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
```

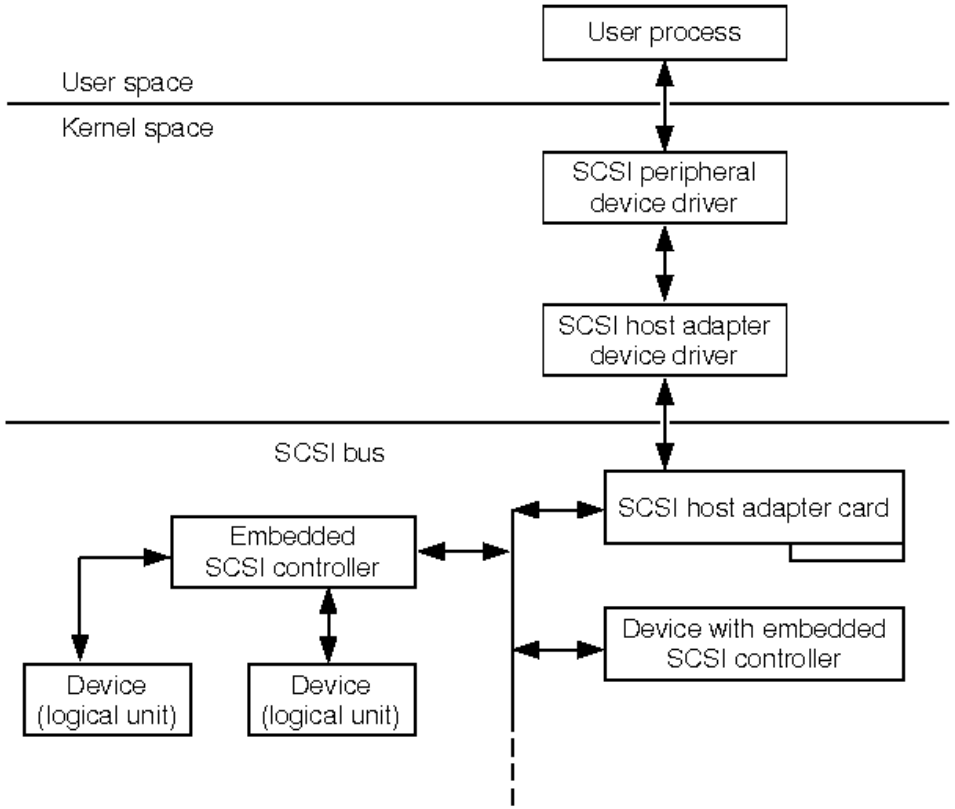
Interfejs SCSI

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 -t
[0:0:0:0]    disk    ata:                                /dev/sda
[1:0:0:0]    cd/dvd  ata:                                /dev/sr0
```

- h=SCSI host adapter number
- c=channel/bus number
- i=id number (target)
- l=lun (logical unit number)

Interfejs SCSI[27]



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

Interfejsy SCSI i USB

Oznaczenia urządzeń USB:

```
/sys/class/scsi_device/48:0:0:0/device/block/sdb/sdb1
/sys/devices/pci...00:1d.7/usb1/1-4/1-4:1.0/host48/scsi_host/host48
/sys/devices/pci...00:1d.7/usb1/b-p/b-p:c.i/host48/scsi_host/host48

# lsscsi -t
[48:0:0:0]   disk    usb: 1-4:1.0                    /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

# lsscsi -t
[51:0:0:0]   disk    usb: 3-2.2:1.0                  /dev/sdb
```

- 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]
```

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 + uevents

- wykrycie urządzenia powoduje udostępnienie przez jądro informacji o tym urządzeniu via sysfs
- tworzony jest oddzielny katalog zawierający pliki zawierające różne atrybuty urządzenia
- dodanie/usuniecie urządzenia powoduje wysłanie przez jądro u-zdarzenia (uevent), które jest odbierane przez udev
- udev przetwarza zdarzenie wg reguł umieszczonych w /etc/udev/rules.d/ (ew. /lib/udev/rules.d i /etc/udev/rules.d/); pliki reguł muszą być postaci $*$.rules

udev – reguły[28]

Konfiguracja urządzeń odbywa się wg reguł umieszczonych w /lib/udev/rules.d oraz /etc/udev/rules.d.

Ładowanie potrzebnych modułów można osiągnąć przy pomocy jednej reguły!

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

plik: `/etc/udev/rules.d/70-persistent-net.rules`

```
# PCI device 0x8086:0x4222 (iwl3945)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="00:18:de:39:06:9b", \
                  ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="wlan*", NAME="wlan0"
```

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

udev – reguły

```
find /sys -name p4p1
udevadm  info --attribute-walk --path /sys/class/net/p4p1

  KERNEL=="p4p1"
    SUBSYSTEM=="net"
    DRIVER==""
    ATTR{addr_assign_type}=="0"
    ...
    ATTR{type}=="1"
    ATTR{link_mode}=="0"
    ATTR{address}=="00:15:c5:b1:9a:8a"
    ATTR{broadcast}=="ff:ff:ff:ff:ff:ff"
    ATTR{operstate}=="down"
    ATTR{mtu}=="1500"
    ATTR{flags}=="0x1002"
    ATTR{tx_queue_len}=="1000"
    ATTR{netdev_group}=="0"


SUBSYSTEM=="net", ACTION=="add", DRIVERS=="tg3", ATTR{address}=="00:15:c5:b1:9a:8a", \
                                                 NAME:="eth0"
```

udev – ładowanie modułów

```
# udevadm info --attribute-walk --path /sys/class/net/p4p1/

# cat /sys/class/net/p4p1/device/modalias
pci:v000014E4d00001600sv00001028sd000001CCbc02sc00i00

# MODALIAS=pci:v000014E4d00001600sv00001028sd000001CCbc02sc00i00

# modprobe -r tg3
# modprobe $MODALIAS
# lsmod | grep tg3
```

Aliasy modułów są przechowywane w /lib/modules/$(uname -r)/modules.alias (?).

udev – reguły[29]

plik: `/etc/udev/rules.d/00-usbkey.rules`

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

#SUBSYSTEMS=="usb", ATTRS{serial}=="FCE7073C4301C68A", KERNEL="sd?1", SYMLINK+="usb256m", \
                        GROUP="usbstorage", RUN+="/home/jkob/bin/backup2usbkey256m.sh"
```

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

udev – przydatne komendy

```
# udevadm info --attribute-walk --path  $(udevadm info --query=path --name /dev/sdb)
# udevadm info --attribute-walk --name /dev/sdb
# 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://www.fizyka.umk/pl/~jkob/labul/scripts/showudev.sh

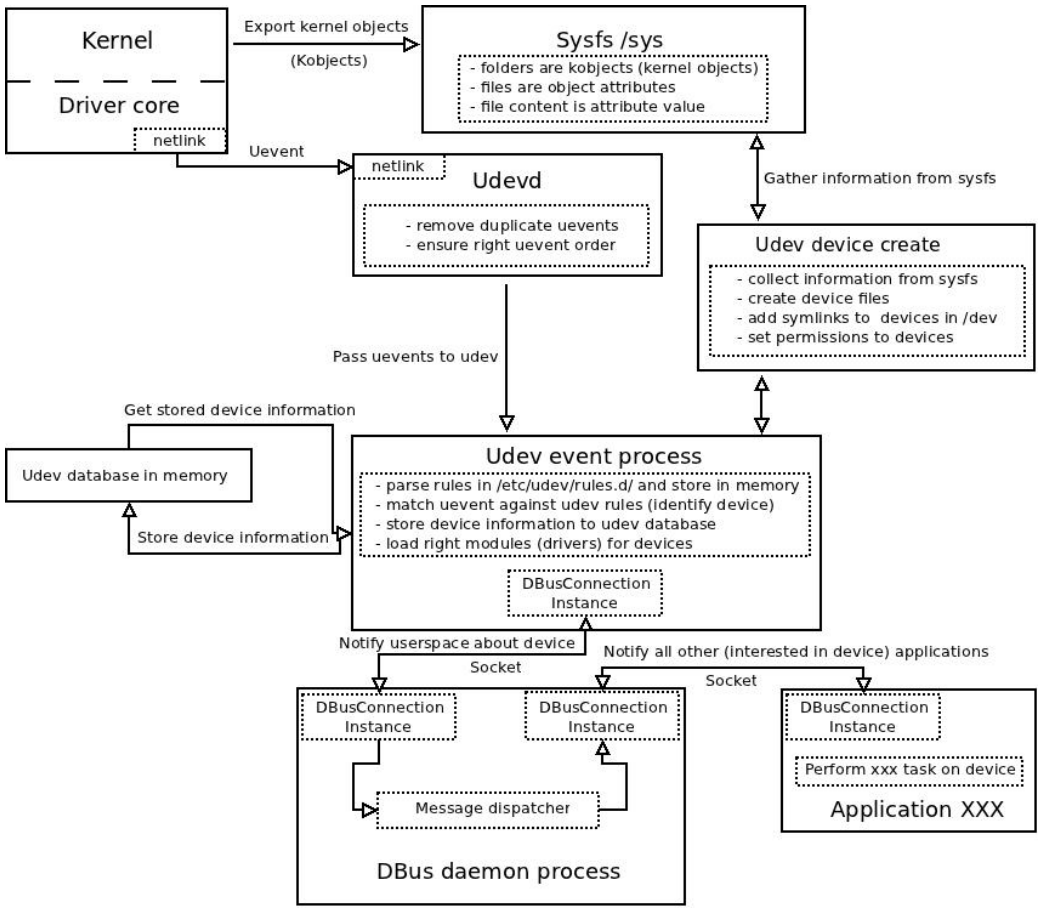
dbus-daemon

```
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`

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

udev + d-bus[30]



[30] 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:

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

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

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.[31]

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

hald

Demon hald (skrypt haldaemon) jest odpowiedzialny za zbieranie informacji o urządzeniach i monitorowanie zmiany stanu tych urządzeń.

Komenda `lshal` pokazuje aktualną listę urządzeń pod nadzorem demona:

```
udi = '/org/freedesktop/Hal/devices/pci_14e4_1677'
  info.bus = 'pci'  (string)
  info.linux.driver = 'tg3'  (string)
  info.parent = '/org/freedesktop/Hal/devices/pci_8086_2660'  (string)
  info.product = 'NetXtreme BCM5751 Gigabit Ethernet PCI Express' (string)
  info.subsystem = 'pci'  (string)
  info.udi = '/org/freedesktop/Hal/devices/pci_14e4_1677'  (string)
  info.vendor = 'Broadcom Corporation'  (string)
  linux.hotplug_type = 2  (0x2)  (int)
  linux.subsystem = 'pci'  (string)
  linux.sysfs_path ='/sys/devices/pci0000:00/0000:00:1c.0/0000:02:00.0'  (string)
  pci.device_class = 2  (0x2)  (int)
  ...
```

Wykrywanie i konfiguracja urządzeń: hald

`lshal -m` pozwala na obserwację pracy demona hald i śledzenie zmian w konfiguracji sprzętowej oraz zmian stanu śledzonych urządzeń. W ten sposób można śledzić komunikaty pochodzące od zasilacza/baterii, karty radiowej (włączanie/wyłączanie), podłączanie/odłączanie myszki lub zewnętrznego dysku, wkładanie płyty CDROM, itp.

```
22:44:23.738: usb_device_46d_c052_noserial_0 added
22:44:23.816: usb_device_46d_c052_noserial_0_if0 added
22:44:23.861: usb_device_46d_c052_noserial_0_usbraw added
22:44:23.938: usb_device_46d_c052_noserial_0_if0_logicaldev_input added
22:44:34.082: acpi_AC property ac_adapter.present = false
22:44:34.165: acpi_BAT0 property battery.charge_level.rate = 0 (0x0)
22:44:34.168: acpi_BAT0 property battery.voltage.current = 12572 (0x311c)
22:44:34.170: acpi_BAT0 property battery.reporting.rate = 0 (0x0)
22:54:24.819: storage_model_CDRW/DVD_GCC4244 property storage.removable.media_available = true
22:54:25.058: volume_label_SAGA_RODU_F_CD1 added
22:55:34.727: volume_label_SAGA_RODU_F_CD1 property volume.mount_point = ''
22:55:34.732: volume_label_SAGA_RODU_F_CD1 property volume.is_mounted_read_only = false
22:55:34.735: volume_label_SAGA_RODU_F_CD1 property volume.is_mounted = false
22:55:44.837: storage_model_CDRW/DVD_GCC4244 property storage.removable.media_available = false
22:55:44.889: volume_label_SAGA_RODU_F_CD1 removed
```

udisks i upower

```
# udisks --enumerate
# udisks --monitor|monitor-detail
# udisks --show-info /dev/sda
# udisks --set-spindown /dev/sdb --spindown-timeout 5
# udisks --detach /dev/sdb
# udisks --[un]mount /dev/sdc


# upower --enumerate
# upower --monitor
# upower --monitor-detail
```

Interfejs sieciowy: zmiana nazwy

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

Przykład:

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

Interfejs sieciowy: zmiana nazwy

```
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

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 (patrz także `ifrename -p`).

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

Konfiguracja interfejsów sieciowych: /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

Uruchamianie/zatrzymywanie wszystkich interfejsów:

```
service network [start|stop]
```

Uruchamianie/zatrzymywanie pojedynczego interfejsu DEV:

```
ifup|ifdown DEV
/etc/sysconfig/network-scripts/ifup|ifdown-ipv6 DEV
```

Przykładowe nazwy interfejsu sieciowego: DEV=eth1,em2,p4p1,wlan0,vlan70, ...

Konfiguracja interfejsów sieciowych: /etc/sysconfig/network

```
HOSTNAME=licserv.fizyka.umk.pl
NETWORKING=yes|no
GATEWAY=<gateway IP>
#NETWORKING_IPV6=no  + net.ipv6.conf.all.disable_ipv6=1 (/etc/sysctl.conf)
NETWORKING_IPV6=yes
IPV6_AUTOCONF=yes
#IPV6_ROUTER=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.

Konfiguracja interfejsów sieciowych: ifcfg-DEV[32]

ifcfg-eth0: statyczna konfiguracja interfejsu eth0:

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

ifcfg-eth0: dynamiczna konfiguracja interfejsu eth0 via DHCP:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
DOMAINNAME=fizyka.umk.pl
PEERDNS=yes
DNS1=8.8.8.8
DNS2=8.8.4.4
```

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

`PEERDNS=yes` powoduje użycie wartości zmiennej `DOMAINNAME` z ifcfg-DEV oraz dopisanie wartości pobranych z serwera DHCP.

Konfiguracja interfejsów sieciowych: 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.

Poprzez pliki aliasowe nie można konfigurować interfejsu via DHCP!

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

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

Konfiguracja interfejsów sieciowych: 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
```

Konfiguracja interfejsów sieciowych: ifcfg-DEV i aliasy

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

```
# cat ifcfg-eth0
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
```

Konfiguracja interfejsów sieciowych: ifcfg-DEV i klony

Pliki konfiguracyjne interfejsów sieciowych mogą być klonowane, dzięki czemu możliwe jest uruchamianie tego samego interfejsu z różnymi opcjami. Np. interfejs dummy może być zarządzany przez użytkownika, jeśli utworzymy pliki ifcfg-dummy0 oraz ifcfg-dummy-user postaci:

```
# cat ifcfg-dummy0
DEVICE=dummy0
BOOTPROTO=none
ONBOOT=yes
IPADDR=10.10.1.0
PREFIX=16
USERCTL=yes
NM_CONTROLLED=no
```

```
# cat ifcfg-dummy0-user
```

Plik ifcfg-dummy0-user może być pusty!

```
# cat ifcfg-dummy0-2
DEVICE=dummy0
BOOTPROTO=none
ONBOOT=no
IPADDR=10.100.1.0
PREFIX=16
```

Konfiguracja interfejsów sieciowych: ifcfg-DEV i VLAN

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

```
# cat 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.

Konfiguracja interfejsów sieciowych: ifcfg-DEV i VLAN

Interfejsy sieciowe do obsługi VLAN-ów można tworzy/usuwać przy pomocy komendy vconfig.

```
# 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 [interface-name] is the name of the ethernet card that hosts
  the VLAN you are talking about.
* 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)
...
```

Konfiguracja interfejsów sieciowych: bonding

Nowoczesne przełączniki ethernetowe umożliwiają łączenie wielu (2, 4, 8) portów, które są traktowane jako jeden kanał komunikacyjny (*trunking*), dzieki czemu można poprawić przekazywanie ramek nie tylko między urządzeniami sieciowymi, ale także serwerem i przełącznikiem. Po stronie serwera wymaga to odpowiedniego łączenia interfejsów (*bonding*). Potrzebny jest moduł jądra bonding oraz wsparcie sprzętowe ze strony interfejsów sieciowych i ich modułów obsługi.

```
# cat ifcfg-bond0
DEVICE=bond0
BOOTPROTO=none
IPADDR=192.168.0.200
NETMASK=255.255.255.0
```

```
# cat ifcfg-eth0|1
DEVICE=eth0|1
BOOTPROTO=none
MASTER=bond0
SLAVE=yes
```

```
# ip address
2: eth0: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master bond0 qlen 1000
    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 master bond0 qlen 1000
    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
```

Konfiguracja interfejsów sieciowych: NetworkManager

Cele projektu:[33]

- 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

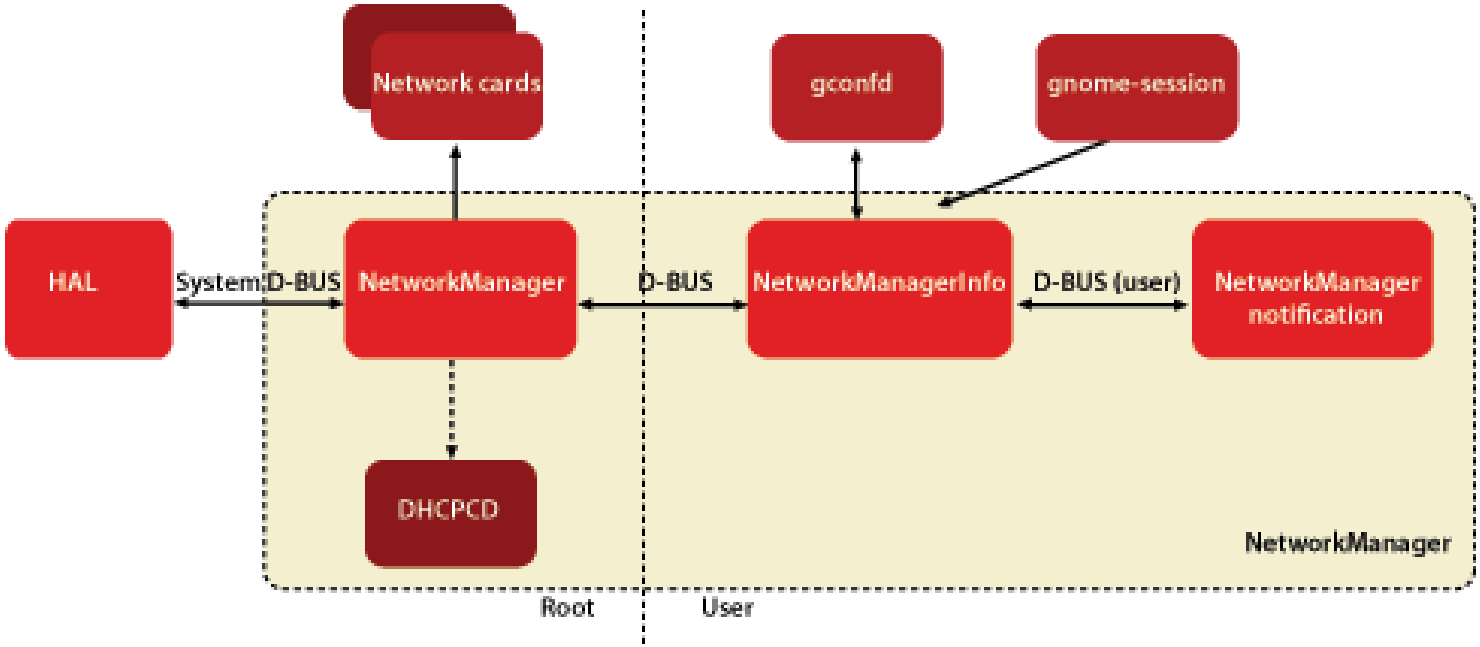
[33] http://fedoraproject.org/wiki/Tools/NetworkManager

Konfiguracja interfejsów sieciowych: NetworkManager

Własności:

- konfiguracja via keyfile, ifcfg-rh (Red Hat, Fedora), ifupdown (Debian, Ubuntu), ifcfg-suse (SUSE, OpenSUSE)
- dobre wsparcie dla IPv4 (konfiguracja statyczna i dynamiczna)
- interface D-Bus
- lokalny buforujący serwer nazw domenowych (dnsmasq)
- połączenia ethernetowe (802.3)
- połączenia WiFi (802.11)
- połączenia VPN
- połączenia szerokopasmowe via USB lub Bluetooth

Konfiguracja interfejsów sieciowych: NetworkManager[34]



[34] http://www.redhat.com/magazine/003jan05/features/networkmanager/

Konfiguracja interfejsów sieciowych: NetworkManager

Centos 6.4:

```
# cat ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="none"
DNS1="158.75.5.250"
GATEWAY="158.75.5.254"
HWADDR="00:16:3e:05:02:49"
IPADDR="158.75.5.90"
NETMASK="255.255.254.0"
NM_CONTROLLED="yes"
ONBOOT="yes"
TYPE="Ethernet"
```

Konfiguracja interfejsów: NetworkManager (CentOS 6.x)

```
NAME
       nmcli - command-line tool for controlling NetworkManager
SYNOPSIS
       nmcli  [ OPTIONS ] OBJECT { COMMAND | help }
       OBJECT := { nm | con | dev }
DESCRIPTION
       nmcli is a command-line tool for controlling NetworkManager and getting its status.
       It is not meant as a replacement of nm-applet or other similar clients.  Rather
       it's a complementary utility to these programs.  The main nmcli's usage is on
       servers, headless machines or just for power users who prefer the command line.
       The use cases comprise:
       - Initscripts: ifup/ifdown can utilize NetworkManager via nmcli instead of having
         to manage connections itself and possibly interfere with Network-Manager.
       - Servers, headless machines: No GUI is available; then nmcli can be used to
         activate/deactivate connections.  However, if a connection requires a secret to
         activate and if that secret is not stored at the system level, nmcli will not be
         able to activate it; it is currently unable to supply the needed secrets to
         NetworkManager.
       - User sessions: nmcli can be used activate/deactivate connections from the command
         line, but a full NetworkManager client (like nm-applet) is used for supplying
         secrets not stored at the system level.  Keyring dialogs and password prompts may
         appear if this happens.
```

Konfiguracja interfejsów: NetworkManager (CentOS 7)

```
NAME
       nmcli - command-line tool for controlling NetworkManager
SYNOPSIS
       nmcli  [ OPTIONS ] OBJECT { COMMAND | help }
       OBJECT := { general | networking | radio | connection | device | agent  }
       ...
DESCRIPTION
       nmcli is a command-line tool for controlling NetworkManager and reporting network
       status. It can be utilized as a replacement for nm-applet or other graphical
       clients. nmcli is used to create, display, edit, delete, activate, and deactivate
       network connections, as well as control and display network device status.
       Typical uses include:
       - Scripts: utilize NetworkManager via nmcli instead of managing network
         connections manually. nmcli supports a terse output format which is better
         suited for script processing. Note that NetworkManager can also execute scripts,
         called "dispatcher scripts", in response to network events. See NetworkManager
         for details about these dispatcher scripts.
       - Servers, headless machines, and terminals:  nmcli can be used to control
         NetworkManager without a GUI, including creating, editing, starting and stopping
         network connections and viewing network status.
```

Konfiguracja interfejsów: NetworkManager (CentOS 6.x)

```
nmcli nm status
nmcli nm wifi|wwan on|off
nmcli con status
nmcli con list id Capehorn
nmcli dev status
nmcli dev list iface p4p1|wlan0
```

```
# nmcli con
NAZWA                   UUID                                   TYP              RZECZYWISTY-OKRES-C
neostrada_d9e4          4324592f-8e7a-4be3-b64f-9e731aa88c96   802-11-wireless  sob, 24 mar 2012, 2
kis                     ae37c4b3-b9ee-4048-8fe1-dcbc0007b7e3   802-11-wireless  pon, 12 mar 2012, 1
eduroam                 ba719f52-6a35-4d33-ad6d-1c987f8400fc   802-11-wireless  nigdy
Capehorn                b1984dce-e7cd-42e8-9c72-7200201752dd   802-11-wireless  śro, 28 mar 2012, 1
Play Online Domyślne    87325929-f032-43e5-9c20-5ff9f27eaf18   gsm              śro, 28 mar 2012, 1
```

Konfiguracja interfejsów: NetworkManager (CentOS 7)

```
nmcli general status|permissions|logging
nmcli networking on|off|connectivity
nmcli radio all|wifi|wwan|wimax on|off
nmcli connection start|stop|show active
nmcli conn show configured eduroam
nmcli device show p4p1|wlan0
nmcli agent secret|polkit|all
```

Konfiguracja interfejsów sieciowych: najważniejsze zmienne[35]

```
/etc/sysconfig/network-scripts/ifcfg-<interface-name> and
/etc/sysconfig/network-scripts/ifcfg-<interface-name>:<alias-name>:

  The first defines an interface, and the second contains only the parts of the definition
  that are different in a "alias" (or alternative) interface.  For example, the network
  numbers might be different, but everything else might be the same, so only the network
  numbers would be in the alias file, but all the device information would be in the base
  ifcfg file.

 DEVICE=<name of physical device>
 IPADDRn=
 PREFIXn= Network prefix. It is used for all configurations except aliases and ippp
   devices. It takes precedence over NETMASK when both PREFIX and NETMASK are set.
 NETMASKn= Subnet mask; just useful for aliases and ippp devices. For all other
   configurations, use PREFIX instead.

The "n" is expected to be consecutive positive integers starting from 0.
It can be omitted if there is only one address being configured.

 ONBOOT=yes|no (not valid for alias devices; use ONPARENT)
```

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

```
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.

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'.
```

```
If BOOTPROTO is not "none", then the only other item that must be set is the DEVICE
item; all the rest will be determined by the boot protocol.  No "dummy" entries
need to be created.

Base items being deprecated:
 NETWORK=<will be calculated automatically with ipcalc>
 BROADCAST=<will be calculated automatically with ipcalc>


/etc/sysconfig/network-scripts/route-<interface-name>

Contains lines that specify additional routes that should be added when the
associated interface is brought up.

The files are processed by the ifup-routes script and uses the /sbin/ipcalc
utility for all network masks and numbers. Routes are specified using the
syntax:

 ADDRESSn=<network>
   NETMASKn=<network/prefix mask>
   GATEWAYn=<next-hop router/gateway IP address>

 The "n" is expected to be consecutive positive integers starting from 0.
```

```
Ethernet 802.1q VLAN items:
  DEVICE=eth0.42
       Initscripts use the device name for VLAN devices.
              Example: eth0.42 for vlan 42 on device eth0.
       Valid VLAN ID range is 0-4095. Most ethernet switches reserve
       VLAN ID 1 to be used as management VLAN; starting from VLAN
       ID 2 is recommended.
  REORDER_HDR=yes|no
       When enabled the VLAN device will move the ethernet header
       around to make it look exactly like a real ethernet device.
       This may help programs such as ISC dhcpd which read the raw
       ethernet packet and make assumptions about the location of
       bytes. If you don't need it turn it off because there
       is a small performance penalty. Default is on.
  GVRP=yes|no
       When enabled, this will announce new vlan creation to a GVRP
       enabled trunk port on a switch. Default is off.

Bonding-specific items
  SLAVE=yes
    Specifies device as a slave
  MASTER=bondXX
    Specifies master device to bind to
```

```
IPv6-only items for real interfaces:
  IPV6INIT=yes|no
      Enable or disable IPv6 configuration for this interface
      Default: no
  IPV6FORWARDING=yes|no
      Enable or disable global forwarding of incoming IPv6 packets
      Note: Obsolete in interface specification!
      Default: no
  IPV6ADDR=<IPv6 address>[/<prefix length>]
      Specify a primary static IPv6 address here
      Optional, if normal host and a router advertisement daemon is on local link
      Required, if node is a router and interface should route packets
      Note: if prefix length is omitted, 64 is assumed

  Obsoleted values from earlier releases:

  FORWARD_IPV4=yes|no
      This setting has been moved into net.ipv4.ip_forward setting
      in /etc/sysctl.conf. Setting it to 1 there enables IP forwarding,
      setting it to 0 disables it (which is the default for RFC compliance).
  NETWORKING_IPV6=yes|no
      Enable or disable global IPv6 initialization
```

Konfiguracja interfejsów sieciowych: 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 – wykonania dodatkowych czynności może zostać przeprowadzone przez skrypty umieszczone w katalogu /etc/NetworkManager/dispatcher.d. Są one wykonywane w porządku alfabetycznym po zajściu każdego zdarzenia sieciowego i otrzymują dwa argumenty: nazwę interfejsu, którego stan uległ zmianie oraz rodzaj zmiany/akcji: pre-up|down, up|down, vpn-pre-up|down, vpn-up|down, hostname, dhcp4|6-change.

Konfiguracja interfejsów sieciowych: 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.

Konfiguracja interfejsów sieciowych: avahi-daemon[36]

The Avahi mDNS/DNS-SD daemon implements Apple's Zeroconf architecture (also known as "Rendezvous" or "Bonjour"). The daemon registers local IP addresses and static services using mDNS/DNS-SD and provides two IPC APIs for local programs to make use of the mDNS record cache the avahi-daemon maintains. . . . there is the D-Bus interface which provides a rich object oriented interface to D-Bus enabled applications.

Autokonfiguracja hostów jest wykonywana w oparciu o (otwarty) protokół mDNS/DNS-SD (multicast Domain Name Service/DNS Service Discovery), który korzysta z 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).

[36] 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/

Konfiguracja interfejsów sieciowych: avahi-daemon

- dostępne zasoby:
  - `avahi-browse [-t] [-r] --all`
  - `avahi-browse _services._dns-sd._udp`
  - `avahi-browse '_afpovertcp|ftp|ipp|http|libvirt|phone._tcp'`
  - `avahi-browse '_printer|ssh|sftp-ssh|telnet|udisks-ssh._tcp'`
  - `avahi-browse '_services._dns-sd._tcp'`
  - `avahi-discover`
  - `avahi-publish`
  - `/etc/avahi/hosts`, `/etc/avahi/services`
- rozwiązywanie nazw: `avahi-resolve --name|address`
- DNS: `avahi-dnsconfd`
- konfiguracja interfejsu iface: `avahi-autoip iface`
- iptables nie może blokować adresu 224.0.0.251 oraz portów:[37]
  - 5222 – xmmp client (XMPP Client Connection)
  - 5298 – presence (XMPP Link-Local Messaging)
  - 5353 – mdns (Multicast DNS)

[37] XMPP – Extensible Messaging and Presence Protocol

DNS (*Domain Name System*)[38]

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 zaimplementowania 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.

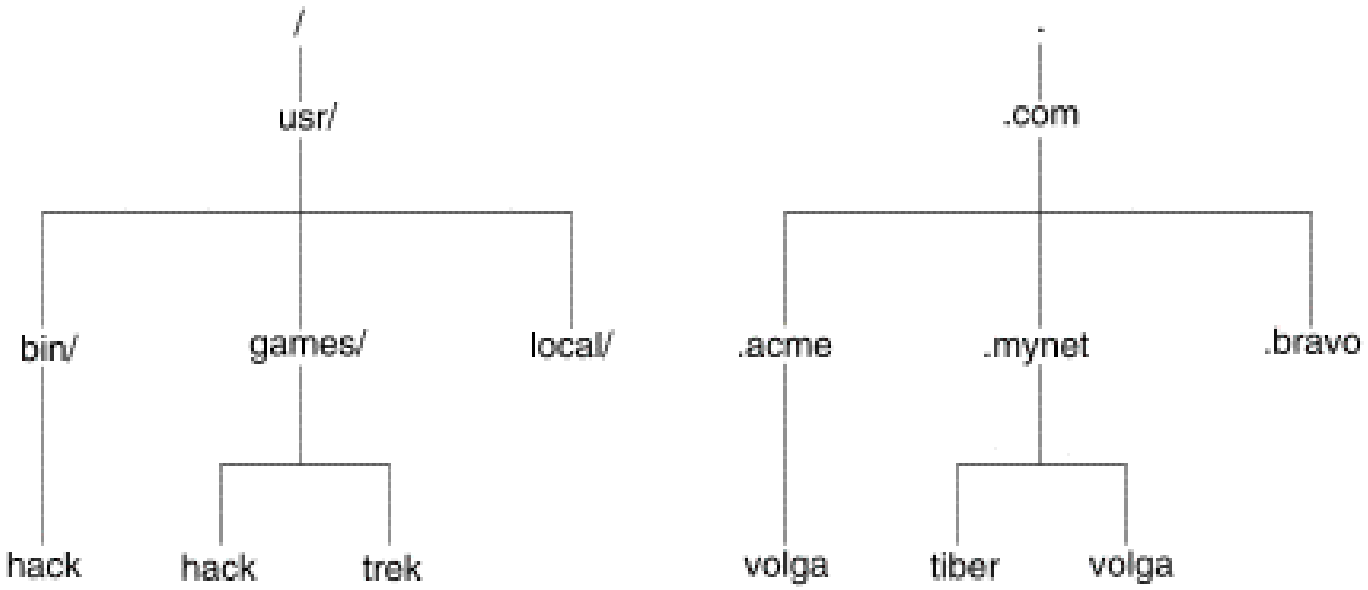
[38] Configuring Domain Name System (DNS) servers, BIND 9 Administrator Reference Manual, http://www.zytrax.com/books/dns/, http://en.wikipedia.org/wiki/Root_name_server

DNS

- Pierwsza uniksowa implementacja była dziełem czterech studentów z Berkeley (1984). W 1985 K. Dunlap (DEC) istotnie przerabia tę implementację i tworzy program o nazwie BIND (*Berkeley Internet Name Domain*), który do chwili obecnej jest utrzymywany przez M. Karelsa, P. Almquista i P. Vixie'go.
- Na początku 1990 pojawia się wersja systemu BIND na platformę Windows NT.
- Dostępnych jest szereg innych programów do tworzenia i odpytywania serwerów nazw (oprócz bind8 i bind9): NSD, djbdns, dnsmasq, Microsoft DNS, itp.

DNS – hierarchia domen

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



Domeny i strefy

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

- gTLD, *generic TLD*: biz, com, edu, gov, mil, net, org, int

  W 2004 dodano do gTLD podkategorię sTLDs (*Sponsored TLD*), np. .aero, .museum, .travel, and .jobs; charakteryzujące się ograniczoną (a nie otwartą) rejestracją.

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

- 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*).

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)

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 unikodzie. Są one przechowywane w postaci znaków ASCII: konwersaja jest dokonywana przy pomocy schematu kodowania zwanego Punycode.[39]

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

Domeny i strefy

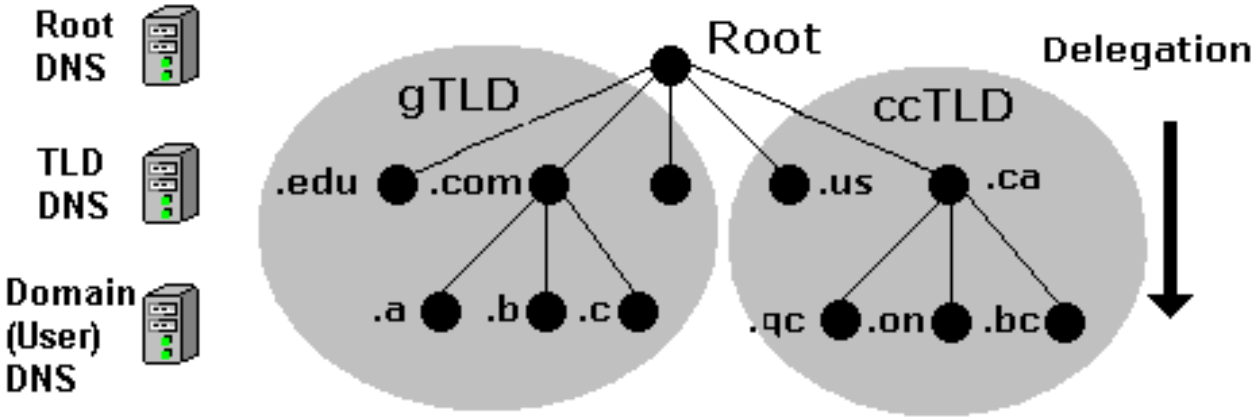
Adresy www.umk.pl oraz www.fizyka.umk.pl pokazują, że umk jest domeną drugiego poziomu, z której wyodrębniono domenę trzeciego poziomu fizyka.umk.pl.

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 zone*) utrzymywane w tzw. *Internet Address and Routing Parameter Area*, czyli domenie arpa.

Domeny i serwery



Do 2002 było 13 serwerów dla domeny *root* zlokalizowanych w 4. krajach. W grudniu 2004 było ich (wraz kopiami zapasowymi) 80 w 34. krajach (w 2014 – 504); dostępność via *anycast*.

Struktura DNS

Na system nazw domenowych składają się

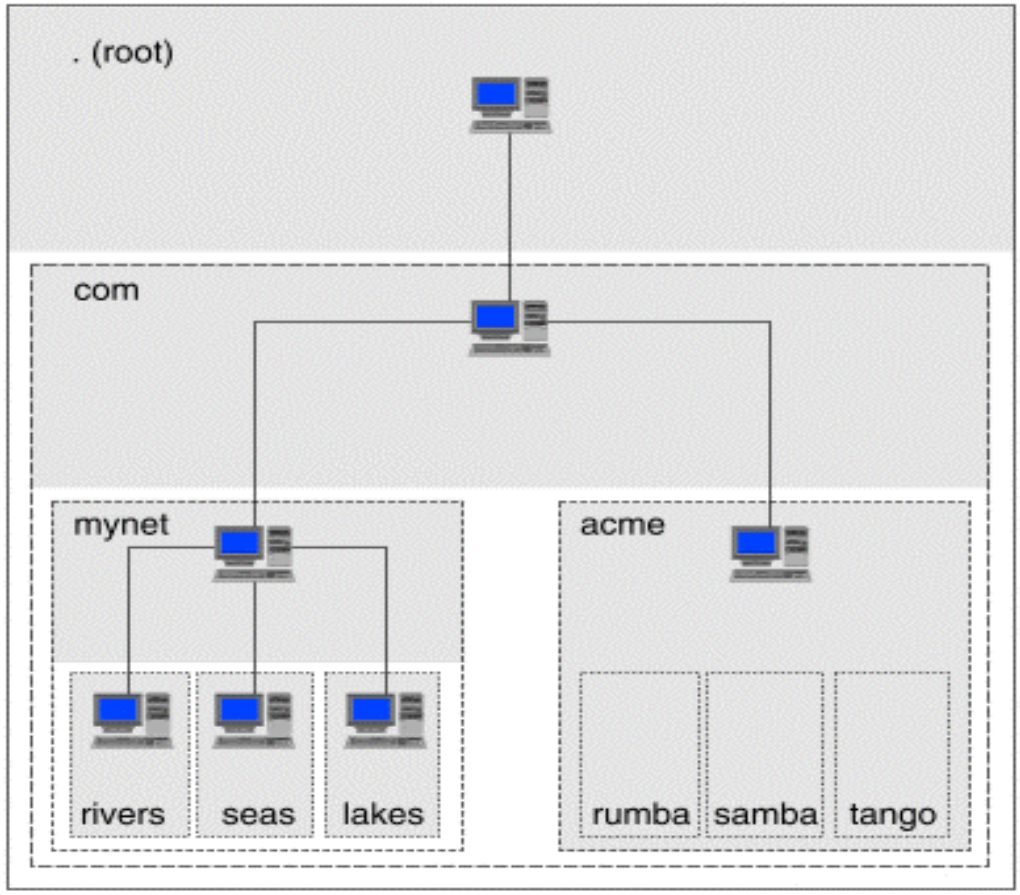
- dane opisujące jedną lub wiele domen; dane są przechowywane w postaci tekstowej w rekordach zasobów (*resource records*) gromadzonych w tzw. plikach stref (*zone files*)
- jeden lub więcej programów udostępniających dane; serwer działa w oparciu o wybraną konfigurację, czyta pliki stref, za które odpowiada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

Rodzaje serwerów nazw

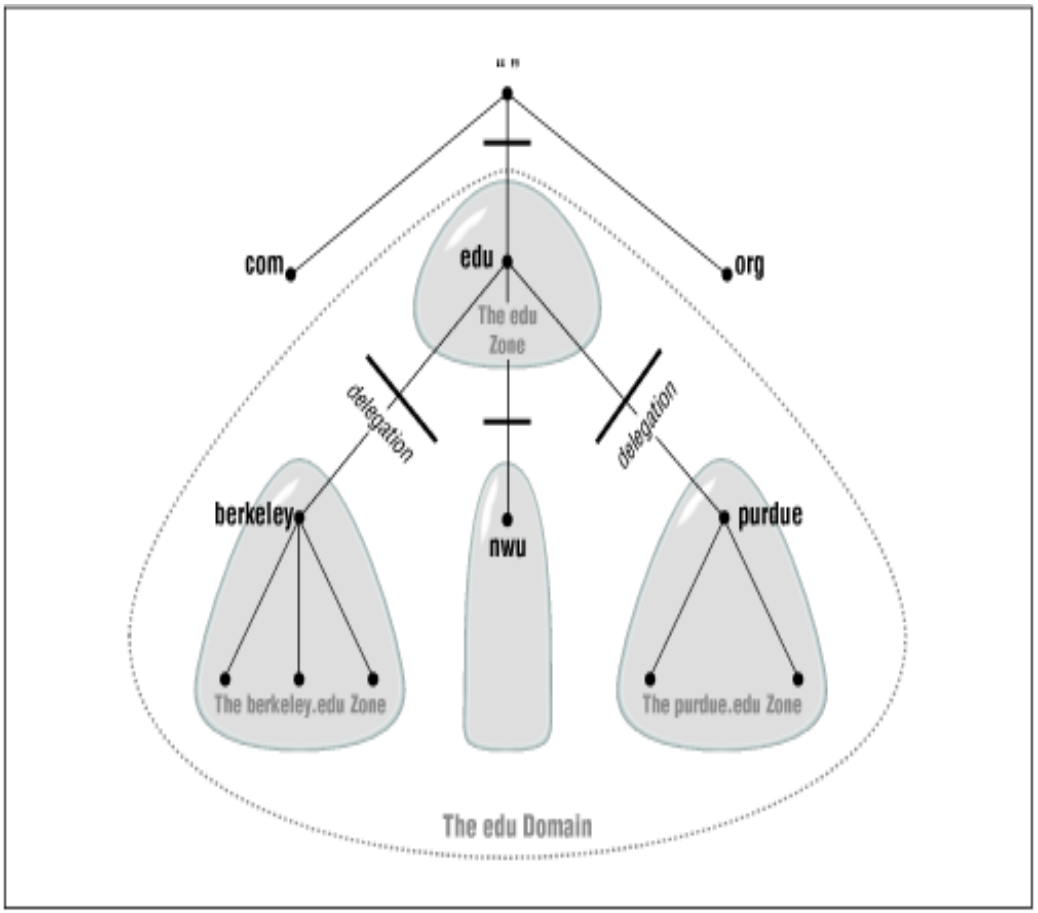
- serwery domeny głównej, serwery poddomen
  - dla każdej domeny musi być zdefiniowany tzw. serwer główny (*master*, *primary master*) oraz serwer zapasowy (*slave*, *secondary master*)
  - serwer główny i zapasowy są serwerami autorytatywnymi dla strefy

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

Domeny i strefy

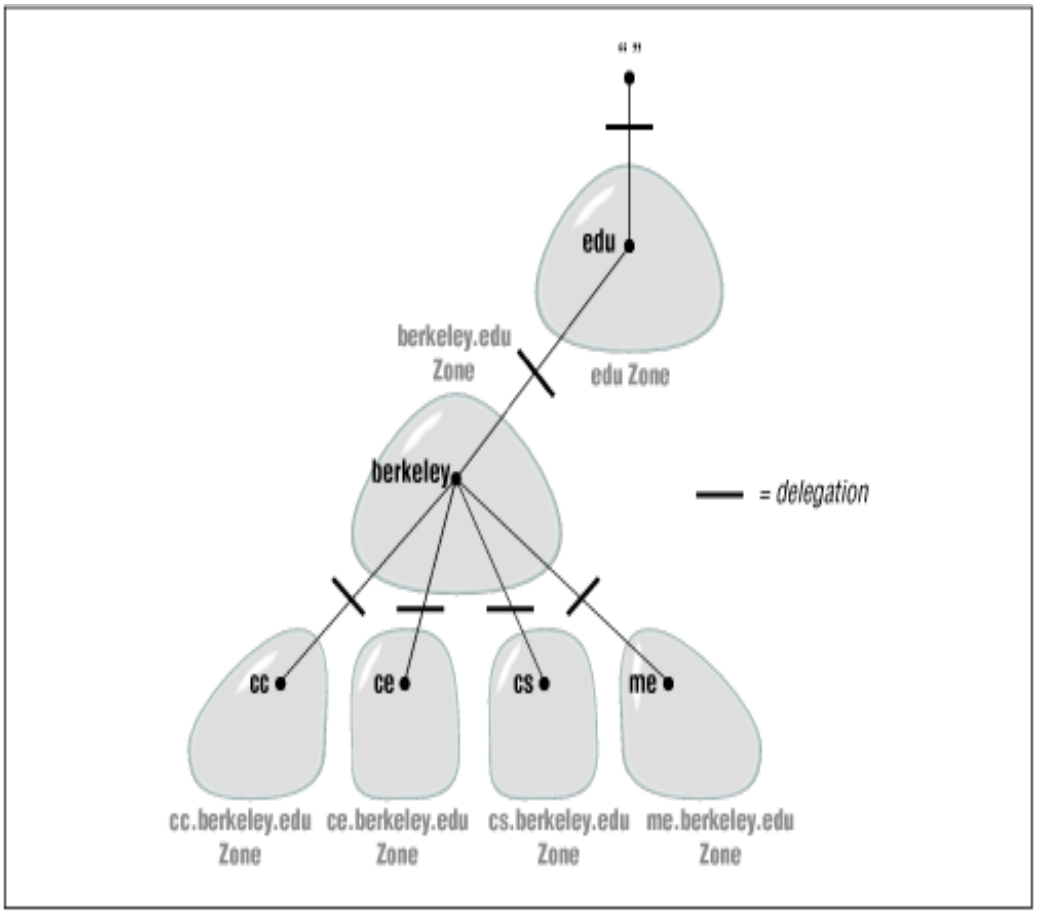


Domeny i strefy[40]



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

Domeny i strefy[41]



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

Zapytania

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. W zależności od konfiguracji serwer może być serwerem autorytatywnym dla jednej lub wielu wskazanych domen, może być serwerem zapasowym, może być odpowiedzialny za przekazywanie zapytań do innych serwerów, itp.

Zapytania

Do serwera DNS mogą być kierowane zapytania dotyczące różnych domen. 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 (być może 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 (brak możliwości dostępu do innego serwera)

Kwerenda iteracyjna

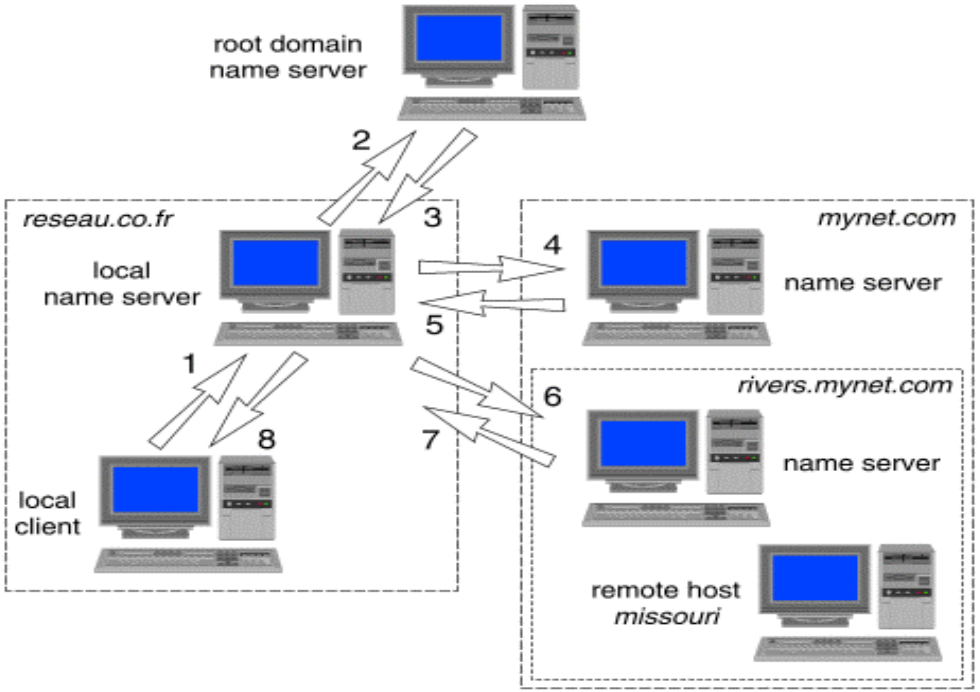
W wyniku wysłania zapytania do serwera DNS kwerendy iteracyjnej klient otrzymuje:

- odpowiedź (być może 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 (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

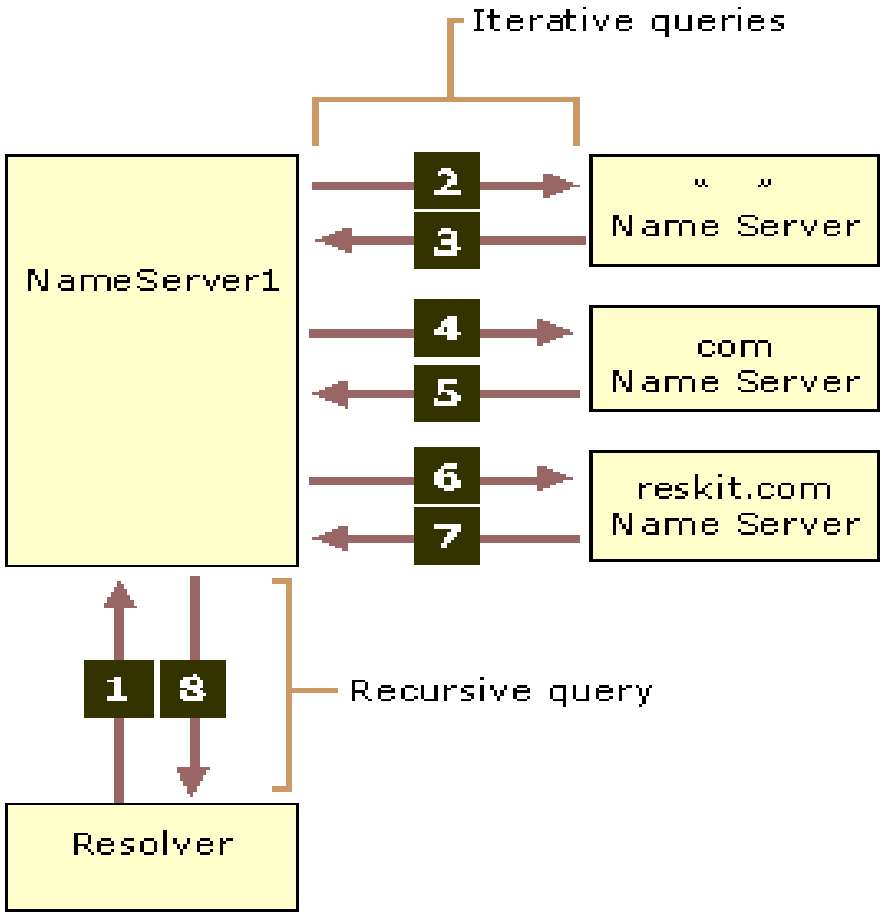
Rekurencyjne i iteracyjne rozwiązywanie nazw



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[42]



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

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

; <<>> DiG 9.8.2rc2-RedHat-9.8.2-0.4.rc2.fc16 <<>> @hel www.fizyka.umk.pl any +trace
; (1 server found)
;; global options: +cmd
.                       484190  IN      NS      k.root-servers.net.
.                       484190  IN      NS      a.root-servers.net.
.                       484190  IN      NS      d.root-servers.net.
.                       484190  IN      NS      c.root-servers.net.
.                       484190  IN      NS      m.root-servers.net.
.                       484190  IN      NS      i.root-servers.net.
.                       484190  IN      NS      h.root-servers.net.
.                       484190  IN      NS      g.root-servers.net.
.                       484190  IN      NS      j.root-servers.net.
.                       484190  IN      NS      b.root-servers.net.
.                       484190  IN      NS      l.root-servers.net.
.                       484190  IN      NS      e.root-servers.net.
.                       484190  IN      NS      f.root-servers.net.
;; Received 512 bytes from 158.75.5.90#53(158.75.5.90) in 682 ms
```

```
pl.                     172800  IN      NS      e-dns.pl.
pl.                     172800  IN      NS      f-dns.pl.
pl.                     172800  IN      NS      c-dns.pl.
pl.                     172800  IN      NS      g-dns.pl.
pl.                     172800  IN      NS      d-dns.pl.
pl.                     172800  IN      NS      a-dns.pl.
pl.                     172800  IN      NS      i-dns.pl.
pl.                     172800  IN      NS      h-dns.pl.
;; Received 407 bytes from 192.203.230.10#53(192.203.230.10) in 698 ms

umk.pl.                 86400   IN      NS      koala.uci.uni.torun.pl.
umk.pl.                 86400   IN      NS      blackbox.uci.uni.torun.pl.
umk.pl.                 86400   IN      NS      flis.man.torun.pl.
;; Received 163 bytes from 149.156.1.6#53(149.156.1.6) in 422 ms

fizyka.umk.pl.          10800   IN      NS      dns1.fizyka.umk.pl.
fizyka.umk.pl.          10800   IN      NS      koala.uci.umk.pl.
fizyka.umk.pl.          10800   IN      NS      dns2.fizyka.umk.pl.
fizyka.umk.pl.          10800   IN      NS      hel.fizyka.umk.pl.
;; Received 179 bytes from 158.75.1.5#53(158.75.1.5) in 496 ms
```

```
www.fizyka.umk.pl.      600     IN      A       158.75.5.251
fizyka.umk.pl.          60      IN      NS      dns1.fizyka.umk.pl.
fizyka.umk.pl.          60      IN      NS      dns2.fizyka.umk.pl.
fizyka.umk.pl.          60      IN      NS      koala.uci.umk.pl.
fizyka.umk.pl.          60      IN      NS      hel.fizyka.umk.pl.
;; Received 163 bytes from 158.75.5.250#53(158.75.5.250) in 42 ms
```

Rekurencyjne i iteracyjne rozwiązywanie nazw

Serwer nazw może jednak zachowywać się jak klient systemu DNS i zlecać innemu serwerowi nazw (zwanemu *forwarder*) przeprowadzenie procesu rozwiązywania nazw. W wyniku otrzymuje poszukiwaną odpowiedź, albo komunikat o błędzie. 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.

Najważniejsze typy rekordów[43]

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. If a space exists in the field it must be enclosed in quotes

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

[43] 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.

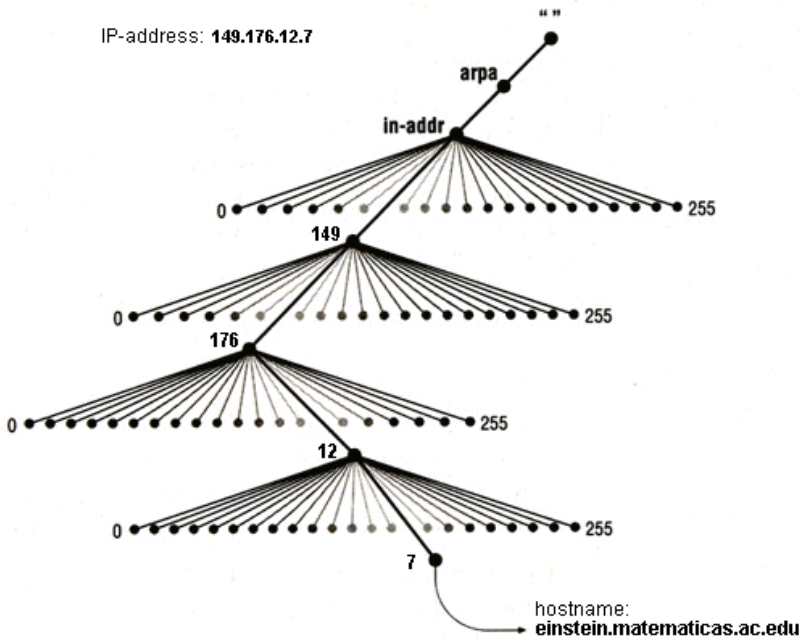
SPF Sender Policy Framework
: Specified as part of the SPF protocol as an alternative to of storing SPF data in TXT records. Uses the same format as the earlier TXT record.

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

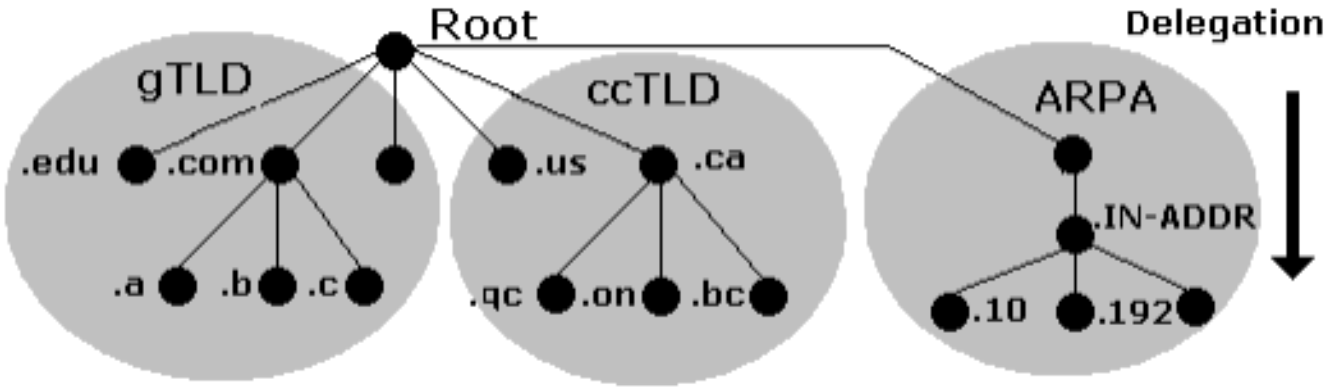
TXT Text record
: Originally for arbitrary human-readable text in a DNS record. Since the early 1990s, however, this record more often carries machine-readable data, such as specified by RFC 1464, opportunistic encryption, Sender Policy Framework, DKIM, DMARC DNS-SD, etc.

Domena in-addr.arpa (ip6.arpa)

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



Domena in-addr.arpa (ip6.arpa)



Domena in-addr.arpa (ip6.arpa)

Np. adresowi 158.75.5.251 odpowiada nazwa 251.5.75.158.in-addr.arpa, która jest odwzorowywana na www.fizyka.umk.pl. Zamianę adresów IP na adresy domenowe wykonuje się przy pomocy zapytań odwrotnych (*reverse query/lookup*).

W jaki sposób pogodzić taki system podziału przestrzeni nazw z bezklasowym trasowaniem międzydomenowym (CIDR)? Zobacz RFC 2317.

Współczesne systemy pocztowe używają odwrotnego rozwiązywania adresów jako prostego sposobu uwierzytelniania serwera pocztowego. W przypadku IPv6 mapowanie odwrotne jest obowiązkowe.

DDNS (Dynamic DNS)[44]

W jaki sposób łączyć się z prywatnym serwerem WWW, kamerą internetową, itp., których IP się zmienia?

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

[44] 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]
           [-p port#] [-t type] [-x addr] [-y name:key] [-4] [-6] [name] [type]
           [class] [queryopt...]

       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).
```

dig – przykłady zapytań

```
dig  [ @dnsserver ]
dig  [ @dnsserver ] fizyka.umk.pl
dig  [ @dnsserver ] -t any fizyka.umk.pl
dig  [ @dnsserver ] fizyka.umk.pl any
dig  [ @dnsserver ] fizyka.umk.pl soa +[no]multiline
dig  [ @dnsserver ] fizyka.umk.pl mx
dig  [ @dnsserver ] fizyka.umk.pl txt
dig  [ @dnsserver ] mail.fizyka.umk.pl a [in]
dig  [ @dnsserver ] mail.fizyka.umk.pl mx +[no]recurse
dig  [ @dnsserver ] mail.fizyka.umk.pl mx +trace
dig  [ @dnsserver ] fizyka.umk.pl  +nssearch
dig  [ @dnsserver ] fizyka.umk.pl  +nostats
dig  [ @dnsserver ] fizyka.umk.pl  +[no]question +[no]answer
dig  [ @dnsserver ] -x 158.75.5.90
dig  [ @dnsserver ] .com +trace +multiline
dig  [ @dnsserver ] fizyka.umk.pl soa rp.pl mx
dig  [ @dnsserver ] www.fizyka.umk.pl a +noall +answer
```

Zarządzanie demonem named

```
NAME
       rndc - name server control utility

SYNOPSIS
       rndc [-c config-file] [-k key-file] [-s server] [-p port] [-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 named the only supported
       authentication algorithm is HMAC-MD5, which uses 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.
```

rndc: przykłady użycia

Przy pomocy rndc można lokalnie i zdalnie zarządzać serwerem DNS:

```
controls {
        inet 127.0.0.1 port 953 allow { localhost; } keys { rndckey4tal; };
        inet 158.75.5.51 port 953 allow { 158.75.5.90; } keys { rndckey4tal; };
};
include ‘‘/etc/rndc.key’’;
```

```
rndc
rndc -s localhost status|reload|stop|restart
rndc -s localhost querylog
rndc -s serwer komenda
```

DNS – przykłady konfiguracji

- serwer domeny labul.pl
- serwer buforujący
- serwer domeny fizyka.umk.pl, is.umk.pl, romp.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)

Historia Secure Shell

- 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

Historia Secure Shell

- 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

Historia Secure Shell

- 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. implementację, 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[45]

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

Historia Secure Shell

- 2005 – SCS wprowadza na rynek SSH G3, nową implementację SSH wchodząca w skład produktów SSH Tectia[46] (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 implementacje 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)*

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

Zalety SSH

SSH gwarantuje:

- prywatność danych via silne szyfrowanie (AES, ARCFOUR, Blowfish, Twofish, IDEA, DES, 3DES)
- integralność komunikacji, tj. zapewnienie, że przesyłane dane nie ulegają modyfikacji (MD5, SHA-1)
- uwierzytelnianie, tj. potwierdzanie tożsamości nadawców i odbiorców (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)*

Rodzina protokołów SSH-2

- 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
  - keyboard-interactive (S/Key)

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)
  - remote file access
  - file transfer

Konfiguracja ssh i sshd

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

```
/etc/ssh/moduli
/etc/ssh/ssh_config
/etc/ssh/sshd_config
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/ssh/sshrc
```

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

```
authorized_keys
config
id_rsa
id_rsa.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

Generowanie kluczy:

```
$ ssh-keygen -b 2048 -t rsa|dsa|ecdsa
$ ssh-keygen -p -f ~/.ssh/id_rsa|id_dsa|id_ecdsa
```

Zawartość `~/.ssh/id_dsa`

```
-----BEGIN DSA PRIVATE KEY-----
MIIBuwIBAAKBgQCaAzYhPnj29XY/o2hGOB4lnshj7MCbY2BeP5ReTlOqjV+kxlCT
...
iZP8ku33hsnMmK+M97XE
-----END DSA PRIVATE KEY-----
```

Zawartość `~/.ssh/id_rsa`

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,7B402913AEB8307C62EDEC5F2F098C72

STqnxl2RXGi+0qH+kt3N0vfej6+/JNLINSM4+ytx6jCMMiw4dzVG9a0aaZfz/NI5
...
25/C2aPtG0hROcdCyxfNrCC+qamGbHoi2poSRNWWhqw=
-----END RSA PRIVATE KEY-----
```

Zawartość: `~/.ssh/id_rsa.pub`

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAsmp2ExEWQzywTC7oTLH9BwIBUp9f3qJC1pc/kDqxa\
lwUKv3GvngeMWUTvvVVErmaBEnI8wHjAN88VpUORKlBOoeBTH8tr2yAcc5Ptm0ieRXBaJFIs3+0RV\
d2ZhWjxqUZDvuXgyYd8Bzb8QYGOeP37H4omH8QwyBNzlJSAlAppYM=jkob@localhost.localdomain
```

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+OdKLt5XnrFy6OyPmtKcwYNTElPuM\
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

```
$ ssh -o "StrictHostkeyChecking=ask|yes|no" host
```

Key found?	Match?	Strict?	Action
no	–	yes	warn and fail
no	–	no	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	no	ask	warn and ask whether to connect

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

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

CheckHostIP

```
$ ssh -o "CheckHostIP=no" uran

# w known_hosts pojawia się wpis:
$ uran ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtEz8woL5h7yx/anG/Qc3...


$ ssh -o "CheckHostIP=yes" uran

# w known_hosts pojawia się wpis:
uran,158.75.5.91 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtEz8woL5h7yx/anG/Qc3...


$ ssh -o "CheckHostIP=yes" uran.fizyka.umk.pl

# w known_hosts pojawia się wpis:
uran.fizyka.umk.pl ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAtEz8woL5h7yx/anG/Qc3...
```

VerifyHostKeyDNS

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

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

$ ssh -o "VerifyHostKeyDNS=yes" tor.fizyka.umk.pl
The authenticity of host 'tor.fizyka.umk.pl (158.75.5.68)' can't be established.
RSA key fingerprint is 60:4b:dc:81:fb:9a:12:df:f1:28:d7:12:4d:2e:c6:e1.
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?

$ dig tor.fizyka.umk.pl sshfp
;; ANSWER SECTION:
tor.fizyka.umk.pl.      86400   IN      SSHFP   1 1 BCA7E7DEA6A7C720E3EF3093339C645D84EFFCAF
tor.fizyka.umk.pl.      86400   IN      SSHFP   2 1 40F7438997B2B0AEDA1BF0E78C13218C133EE58D

# /etc/ssh/ssh_config
Host *.fizyka.umk.pl
  VerifyHostKeyDNS yes                  # domyślnie  VerifyHostKeyDNS no
```

Dzielenie połączenia

```
# tworzenie połączenia 'master'
$ ssh -S /tmp/ssh-tor -M -f tor sleep 20

# tworzenie połączenia 'slave'
# ssh -S /tmp/ssh-tor tor


# ~/.ssh/config

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

host tor-slave
  hostname tor.fizyka.umk.pl
  Controlpath /tmp/ssh-tor
#  ControlMaster no

$ ssh -f tor-master sleep 20
$ ssh tor-slave
```

ssh -vv

```
jkob@scobie# ssh -vv -X uran

# Wersja oprogramowania używanego przez klienta
OpenSSH_5.8p2, OpenSSL 1.0.0i-fips 19 Apr 2012

# klient ssh pobiera konfigurację z plików:
debug1: Reading configuration data /home/jkob/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config

# Nawiązuje połączenie TCP
debug2: ssh_connect: needpriv 0
debug1: Connecting to uran [158.75.5.68] port 22.
debug1: Connection established.

# Szuka pliku, gdzie znajduje się prywatny klucz SSH-2
# próbuje załadować informację o certyfikacie z pliku *-cert.pub
# w celu identyfikacji nazw
debug1: identity file /home/jkob/.ssh/id_rsa type 1
debug1: identity file /home/jkob/.ssh/id_rsa-cert type -1
debug1: identity file /home/jkob/.ssh/id_dsa type -1
debug1: identity file /home/jkob/.ssh/id_dsa-cert type -1
debug1: identity file /home/jkob/.ssh/id_ecdsa type -1
```

```
debug1: identity file /home/jkob/.ssh/id_ecdsa-cert type -1

# Serwer ogłasza używaną przez siebie wersję protokołu (i wersję programu)
debug1: Remote protocol version 1.99, remote software version OpenSSH_4.3

# Klient wykrywa problem z OpenSSH_4.3 i stosuje odpowiednią łatę
debug1: match: OpenSSH_4.3 pat OpenSSH_4*

# Klient włącza obsługę SSH-2
debug1: Enabling compatibility mode for protocol 2.0

# Klient zgłasza serwerowi używaną przez siebie wersje protokołu i programu
debug1: Local version string SSH-2.0-OpenSSH_5.8

# Uzgadnianie parametrów sesji
# Klient wysyła wiadomość inicjującą wymianę klucza (Key EXchange INITialization)
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received

# Klient oferuje następujące algorytmy wymiany klucza
debug2: kex_parse_kexinit:
diffie-hellman-group-exchange-sha256,diffie-hellman-group-exchange-sha1,\
                        diffie-hellman-group14-sha1,diffie-hellman-group1-sha1
```

```
# Klient ogłasza wspierane formy certyfikowania kluczy
debug2: kex_parse_kexinit:
ssh-rsa-cert-v01@openssh.com,ssh-rsa-cert-v00@openssh.com,ssh-rsa,\
                  ssh-dss-cert-v01@openssh.com,ssh-dss-cert-v00@openssh.com,ssh-dss

# Klient określa rodzaje wspieranych szybkich algorytmów szyfrowania danych (bulk ciphers)
debug2: kex_parse_kexinit:
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc,\
         blowfish-cbc,cast128-cbc,aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.se

# Klient określa używane funkcje haszujące. Każda wiadomość słana przsz SSH, wraz z
# numerem sekwencyjnym i identyfikatorem sesji, jest haszowana; podpis jest dołączany do
# wiadomości (jako MAC, Message Authentication Code)
debug2: kex_parse_kexinit:
hmac-md5,hmac-sha1,umac-64@openssh.com,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-

# Klient wskazuje metody kompresji, które wspiera
debug2: kex_parse_kexinit: none,zlib@openssh.com,zlib

# Klient otrzymuje od serwera analogiczną listę parametrów komunikacji
debug2: kex_parse_kexinit:
diffie-hellman-group-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1
debug2: kex_parse_kexinit: ssh-rsa,ssh-dss
debug2: kex_parse_kexinit:
```

```
aes128-ctr,aes192-ctr,aes256-ctr,arcfour256,arcfour128,aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc
                                              aes192-cbc,aes256-cbc,arcfour,rijndael-cbc@lysator.liu.s
debug2: kex_parse_kexinit:
hmac-md5,hmac-sha1,hmac-ripemd160,hmac-ripemd160@openssh.com,hmac-sha1-96,hmac-md5-96
debug2: kex_parse_kexinit: none,zlib@openssh.com

# Serwer ogłasza swój wybór parametrów

debug2: mac_setup: found hmac-md5
debug1: kex: server->client aes128-ctr hmac-md5 none
# Jeśli klient żąda kompresji, to powyżej mamy
#       kex: server->client aes128-ctr hmac-md5 zlib@openssh.com

# Klient ogłasza swój wybór parametrów
debug2: mac_setup: found hmac-md5
debug1: kex: client->server aes128-ctr hmac-md5 none

# Następuje wymiana kluczy
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug2: dh_gen_key: priv key bits set: 129/256
debug2: bits set: 497/1024
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
```

```
# W tej fazie ma także miejsce proces uwierzytelniania serwera (np. poprzez dane
# zawarte w pliku ~/.ssh/known_hosts), żeby chronić się przed atakami typu spoofing
# i man-in-the-middle
debug1: Server host key: RSA 60:4b:dc:81:fb:9a:12:df:f1:28:d7:12:4d:2e:c6:e1
debug2: key_type_from_name: unknown key type '2048'
debug2: key_type_from_name: unknown key type '2048'
debug1: Host 'uran' is known and matches the RSA host key.
debug1: Found key in /home/jkob/.ssh/known_hosts:139
debug2: bits set: 521/1024
debug1: ssh_rsa_verify: signature correct

# W oparciu o wymienione wiadomości klient i serwer (osobno) tworzą główny klucz sesji
# (master key) oraz hash wymiany. Klucz sesji nie jest przesyłany i powinien być trudny
# do odgadnięcia. Standard SSH-2 wymaga, aby klucz był regenerowany co najmniej raz na
# godzinę lub po przesłaniu każdego 1~GB danych.

# Hash klucza głównego jest używany jako identyfikator sesji (session identifier).

# Główny klucz sesji stanowi podstawę uzyskania dodatkowych kluczy
# potrzebnych do szyfrowania i sprawdzania integralności danych

debug2: kex_derive_keys
```

```
debug2: set_newkeys: mode 1

# Klient i serwer powiadamiają się wzajemnie o rozpoczęciu używania nowych kluczy
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug2: set_newkeys: mode 0
debug1: SSH2_MSG_NEWKEYS received

# Serwer jest gotowy na przyjęcie pierwszego zlecenia. Klient określa dostępny tryb
# uwierzytelnienia dla danego użytkownika oraz dostęp do wskazanych zasobów
# (authentication/authorization)

# Opcja ForwardAgent=no|yes pozwala na określenie, czy możliwe jest przekazywanie danych,
# którymi dysponuje agent
debug1: Roaming not allowed by server

debug1: SSH2_MSG_SERVICE_REQUEST sent
debug2: service_accept: ssh-userauth
debug1: SSH2_MSG_SERVICE_ACCEPT received

debug2: key: /home/jkob/.ssh/id_rsa (0xb7b9f198)
debug2: key: /home/jkob/.ssh/id_dsa ((nil))
debug2: key: /home/jkob/.ssh/id_ecdsa ((nil))
```

```
# Klient pyta serwer o dostępne metody uwierzytelniania
debug1: Authentications that can continue: publickey,gssapi-with-mic,password

# Klient dokonuje wyboru metody uwierzytelniania (kolejność zależy od klienta).
debug1: Next authentication method: gssapi-with-mic
debug1: Unspecified GSS failure.  Minor code may provide more information
Credentials cache file '/tmp/krb5cc_1000' not found

# Druga próba uwierzytelnienia oparta jest o klucz publiczny użytkownika
# i kończy się sukcesem.
debug1: Next authentication method: publickey
debug1: Offering RSA public key: /home/jkob/.ssh/id_rsa
debug2: we sent a publickey packet, wait for reply
debug1: Server accepts key: pkalg ssh-rsa blen 149
debug2: input_userauth_pk_ok: fp ff:68:fd:c2:86:f6:28:60:1f:e4:1c:aa:a2:b5:3b:42
debug1: read PEM private key done: type RSA
debug1: Authentication succeeded (publickey).
Authenticated to uran ([158.75.5.68]:22).
debug1: channel 0: new [client-session]
debug2: channel 0: send open
debug1: Entering interactive session.
debug2: callback start
debug2: x11_get_proto: /usr/bin/xauth  list :0 2>/dev/null
```

```
debug1: Requesting X11 forwarding with authentication spoofing.
debug2: channel 0: request x11-req confirm 0
debug2: client_session2_setup: id 0
debug2: fd 3 setting TCP_NODELAY
debug2: channel 0: request pty-req confirm 1
debug1: Sending environment.
debug1: Sending env XMODIFIERS = @im=none
debug2: channel 0: request env confirm 0
debug1: Sending env LANG = pl_PL.UTF-8
debug2: channel 0: request env confirm 0
debug1: Sending env LANGUAGE =
debug2: channel 0: request env confirm 0
debug2: channel 0: request shell confirm 1
debug2: callback done
debug2: channel 0: open confirm rwindow 0 rmax 32768
debug2: channel_input_status_confirm: type 99 id 0

debug2: PTY allocation request accepted on channel 0
debug2: channel 0: rcvd adjust 2097152
debug2: channel_input_status_confirm: type 99 id 0
debug2: shell request accepted on channel 0
Last login: Mon May  7 21:48:52 2012 from ldap.fizyka.umk.pl
```

```
# Po stronie serwera także można śledzić połączenie. W tym celu trzeba uruchomić demona
# sshd z opcją -d (lub -ddd) lub dodać opcję LOGLEVEL DEBUG w pliku /etc/ssh/sshd_config.

# W tym miejscu po stronie serwera pojawia się komunikat
May  7 22:29:03 uran sshd[1728]: Connection from 158.75.5.250 port 60839
May  7 22:29:03 uran sshd[1728]: debug1: Client protocol version 2.0; client software version OpenS
May  7 22:29:03 uran sshd[1728]: debug1: match: OpenSSH_5.8 pat OpenSSH*
May  7 22:29:03 uran sshd[1728]: debug1: Enabling compatibility mode for protocol 2.0
May  7 22:29:03 uran sshd[1728]: debug1: Local version string SSH-2.0-OpenSSH_4.3
May  7 22:29:03 uran sshd[1729]: debug1: permanently_set_uid: 74/74
May  7 22:29:03 uran sshd[1729]: debug1: list_hostkey_types: ssh-rsa,ssh-dss
May  7 22:29:03 uran sshd[1729]: debug1: SSH2_MSG_KEXINIT sent
May  7 22:29:03 uran sshd[1729]: debug1: SSH2_MSG_KEXINIT received
May  7 22:29:03 uran sshd[1729]: debug1: kex: client->server aes128-ctr hmac-md5 none
May  7 22:29:03 uran sshd[1729]: debug1: kex: server->client aes128-ctr hmac-md5 none
May  7 22:29:04 uran sshd[1729]: debug1: SSH2_MSG_KEX_DH_GEX_REQUEST received
May  7 22:29:04 uran sshd[1729]: debug1: SSH2_MSG_KEX_DH_GEX_GROUP sent
May  7 22:29:04 uran sshd[1729]: debug1: expecting SSH2_MSG_KEX_DH_GEX_INIT
May  7 22:29:04 uran sshd[1729]: debug1: SSH2_MSG_KEX_DH_GEX_REPLY sent
May  7 22:29:04 uran sshd[1729]: debug1: SSH2_MSG_NEWKEYS sent
May  7 22:29:04 uran sshd[1729]: debug1: expecting SSH2_MSG_NEWKEYS
May  7 22:29:04 uran sshd[1729]: debug1: SSH2_MSG_NEWKEYS received
May  7 22:29:04 uran sshd[1729]: debug1: KEX done
```

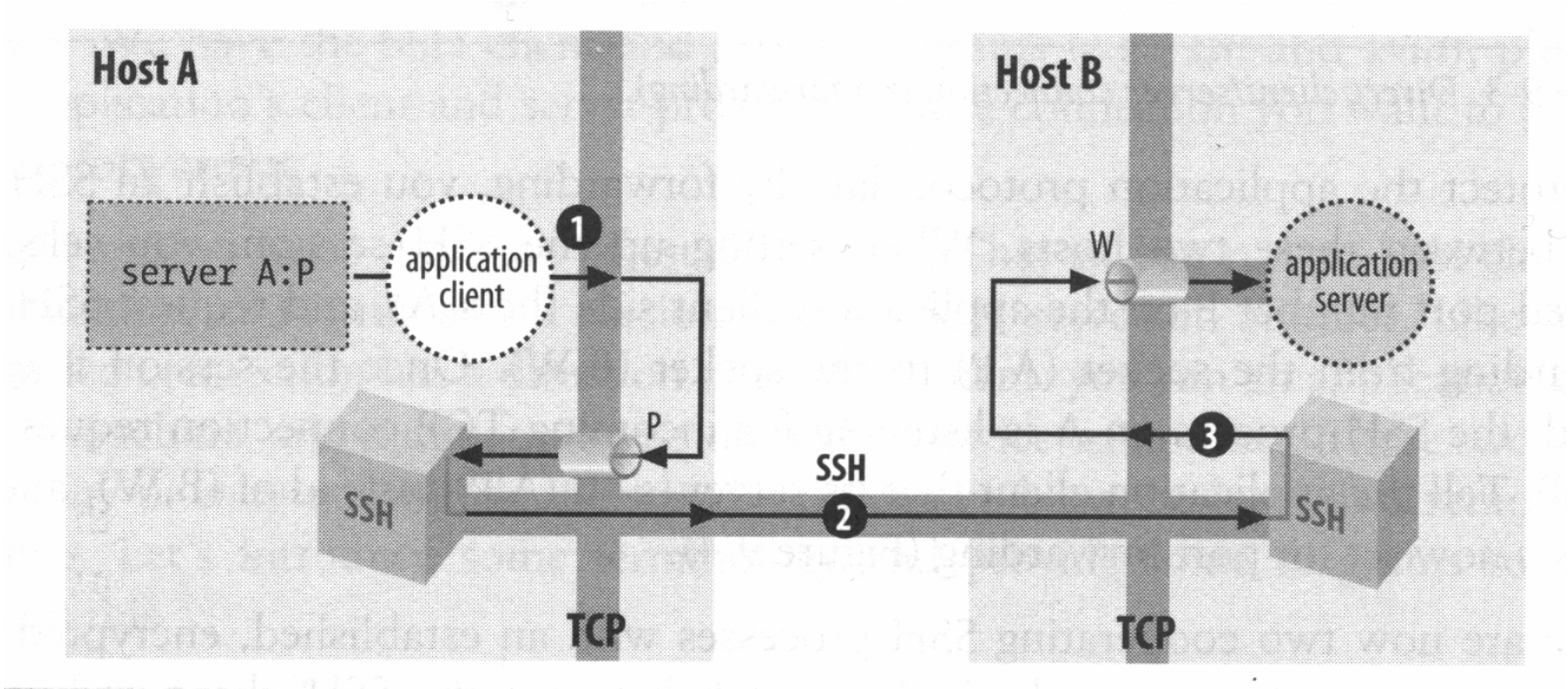
```
May  7 22:29:04 uran sshd[1729]: debug1: userauth-request for user jkob service ssh-connection meth
May  7 22:29:04 uran sshd[1729]: debug1: attempt 0 failures 0
May  7 22:29:04 uran sshd[1728]: debug1: PAM: initializing for "jkob"
May  7 22:29:04 uran sshd[1728]: debug1: PAM: setting PAM_RHOST to "ldap.fizyka.umk.pl"
May  7 22:29:04 uran sshd[1728]: debug1: PAM: setting PAM_TTY to "ssh"
May  7 22:29:04 uran sshd[1729]: debug1: userauth-request for user jkob service ssh-connection meth
May  7 22:29:04 uran sshd[1729]: debug1: attempt 1 failures 1
May  7 22:29:04 uran sshd[1729]: debug1: test whether pkalg/pkblob are acceptable
May  7 22:29:04 uran sshd[1728]: debug1: temporarily_use_uid: 1001/1001 (e=0/0)
May  7 22:29:04 uran sshd[1728]: debug1: trying public key file /home/jkob/.ssh/authorized_keys
May  7 22:29:04 uran sshd[1728]: debug1: matching key found: file /home/jkob/.ssh/authorized_keys,
May  7 22:29:04 uran sshd[1728]: Found matching RSA key: ff:68:fd:c2:86:f6:28:60:1f:e4:1c:aa:a2:b5:
May  7 22:29:04 uran sshd[1728]: debug1: restore_uid: 0/0
May  7 22:29:04 uran sshd[1729]: Postponed publickey for jkob from 158.75.5.250 port 60839 ssh2
May  7 22:29:04 uran sshd[1729]: debug1: userauth-request for user jkob service ssh-connection meth
May  7 22:29:04 uran sshd[1729]: debug1: attempt 2 failures 1
May  7 22:29:04 uran sshd[1728]: debug1: temporarily_use_uid: 1001/1001 (e=0/0)
May  7 22:29:04 uran sshd[1728]: debug1: trying public key file /home/jkob/.ssh/authorized_keys
May  7 22:29:04 uran sshd[1728]: debug1: matching key found: file /home/jkob/.ssh/authorized_keys,
May  7 22:29:04 uran sshd[1728]: Found matching RSA key: ff:68:fd:c2:86:f6:28:60:1f:e4:1c:aa:a2:b5:
May  7 22:29:04 uran sshd[1728]: debug1: restore_uid: 0/0
May  7 22:29:04 uran sshd[1728]: debug1: ssh_rsa_verify: signature correct
May  7 22:29:04 uran sshd[1728]: debug1: do_pam_account: called
May  7 22:29:04 uran sshd[1728]: Accepted publickey for ob from 158.75.5.250 port 60839 ssh2
```

```
May  7 22:36:52 uran sshd[2128]: debug1: monitor_child_preauth: jkob has been authenticated by priv

# Klient został uwierzytelniony, więc serwer zaczyna działać jako serwer SSH i kontrolę
# przejmuje protokół SSH-CONN.

# SSH-CONN oferuje przede wszystkim multipleksowanie, tj. pozwala na dynamiczne tworzenie
# w ramach jednego, bezpiecznego, w pełni dupleksowego strumienia bajtów (dostarczanego
# przez SSH_TRANS), szeregu logicznych kanałów SSH-CONN (identyfikowanych przez numery
# kanałów). Kanały mogą być tworzone i niszczone przez obie strony transmisji. Każdy
# kanał zapewnia sterowanie przepływem (flow-control), a jego typ określa sposób
# wykorzystania. Możliwe typy to: session, x11, forward-tcpip, direct-tcpip.

May  7 22:54:00 uran sshd[2958]: debug1: monitor_child_preauth: jkob has been authenticated by priv
May  7 22:54:01 uran sshd[2958]: debug1: temporarily_use_uid: 1001/1001 (e=0/0)
May  7 22:54:01 uran sshd[2958]: debug1: ssh_gssapi_storecreds: Not a GSSAPI mechanism
May  7 22:54:01 uran sshd[2958]: debug1: restore_uid: 0/0
May  7 22:54:01 uran sshd[2958]: debug1: PAM: establishing credentials
May  7 22:54:01 uran sshd[2958]: pam_unix(sshd:session): session opened for user jkob by (uid=0)
May  7 22:54:01 uran sshd[2960]: debug1: PAM: reinitializing credentials
May  7 22:54:01 uran sshd[2960]: debug1: permanently_set_uid: 1001/1001
May  7 22:54:01 uran sshd[2960]: debug1: Entering interactive session for SSH2.
May  7 22:54:01 uran sshd[2960]: debug1: server_init_dispatch_20
May  7 22:54:01 uran sshd[2960]: debug1: server_input_channel_open: ctype session rchan 0 win 10485
May  7 22:54:01 uran sshd[2960]: debug1: input_session_request
```

```
May  7 22:54:01 uran sshd[2960]: debug1: channel 0: new [server-session]
May  7 22:54:01 uran sshd[2960]: debug1: session_new: init
May  7 22:54:01 uran sshd[2960]: debug1: session_new: session 0
May  7 22:54:01 uran sshd[2960]: debug1: session_open: channel 0
May  7 22:54:01 uran sshd[2960]: debug1: session_open: session 0: link with channel 0
May  7 22:54:01 uran sshd[2960]: debug1: server_input_channel_open: confirm session

# Klient żąda sesji, która zapewnie przekazywanie danych serwera X11 do klienta
# bezpiecznym kanałem, po ustawieniu zmiennej DISPLAY oraz praw dostępu.

May  7 22:54:01 uran sshd[2960]: debug1: server_input_channel_req: channel 0 request x11-req reply
May  7 22:54:01 uran sshd[2960]: debug1: session_by_channel: session 0 channel 0
May  7 22:54:01 uran sshd[2960]: debug1: session_input_channel_req: session 0 req x11-req
May  7 22:54:01 uran sshd[2960]: debug1: channel 1: new [X11 inet listener]
May  7 22:54:01 uran sshd[2960]: debug1: channel 2: new [X11 inet listener]


# Klient żąda utworzenia pseudo-terminala (pty) dla danego kanału, który jest
# potrzebny do pracy interaktywnej.

May  7 22:54:01 uran sshd[2960]: debug1: server_input_channel_req: channel 0 request pty-req reply
May  7 22:54:01 uran sshd[2960]: debug1: session_by_channel: session 0 channel 0
May  7 22:54:01 uran sshd[2960]: debug1: session_input_channel_req: session 0 req pty-req
May  7 22:54:01 uran sshd[2960]: debug1: Allocating pty.
```

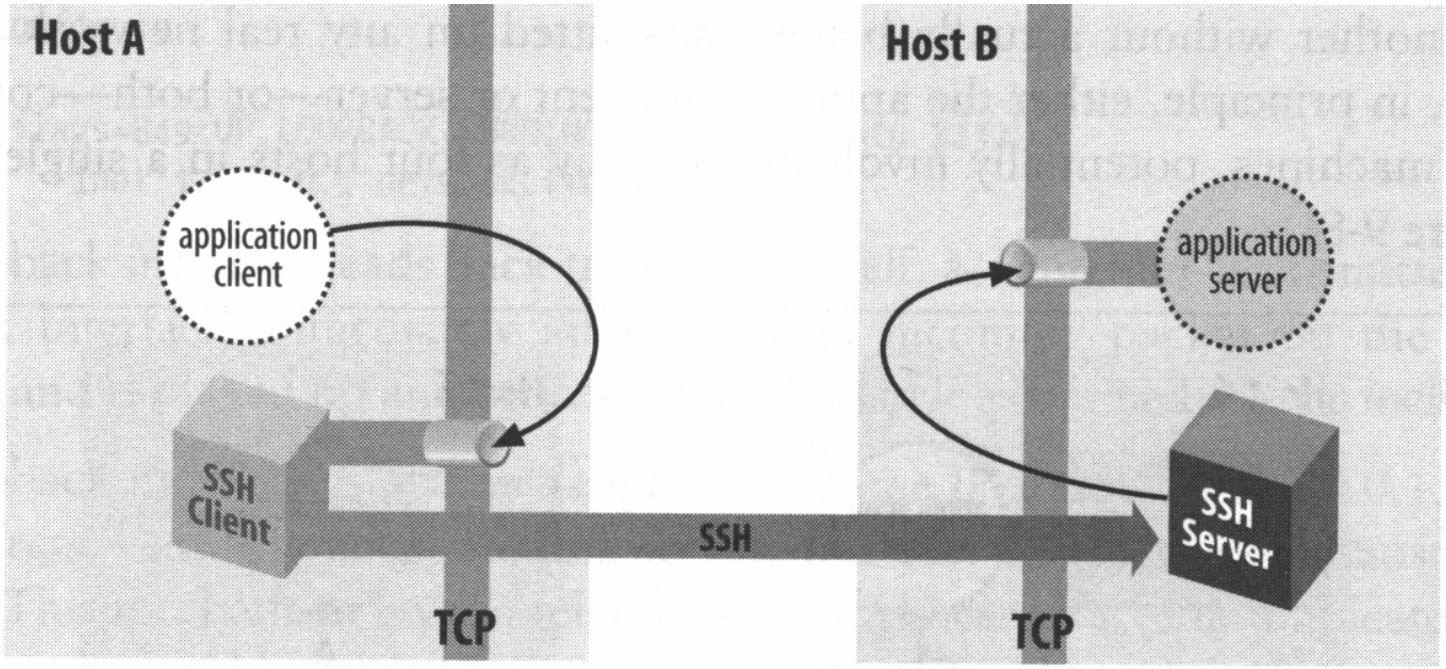
```
May  7 22:54:01 uran sshd[2958]: debug1: session_new: init
May  7 22:54:01 uran sshd[2958]: debug1: session_new: session 0
May  7 22:54:01 uran sshd[2960]: debug1: session_pty_req: session 0 alloc /dev/pts/1
May  7 22:54:01 uran sshd[2960]: debug1: server_input_channel_req: channel 0 request env reply 0


# Klient żąda uruchomienia po stronie serwera domyślnej powłoki (może także zażądać
# uruchomienia dowolnego programu (exec) lub abstrakcyjnego serwisu (subsystem)

May  7 22:54:02 uran sshd[2960]: debug1: server_input_channel_req: channel 0 request shell reply 1
May  7 22:54:02 uran sshd[2960]: debug1: session_by_channel: session 0 channel 0
May  7 22:54:02 uran sshd[2960]: debug1: session_input_channel_req: session 0 req shell
May  7 22:54:02 uran sshd[2961]: debug1: Setting controlling tty using TIOCSCTTY.
```

SSH: Przekazywanie połączeń




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

SSH: lokalne przekazywanie połączeń

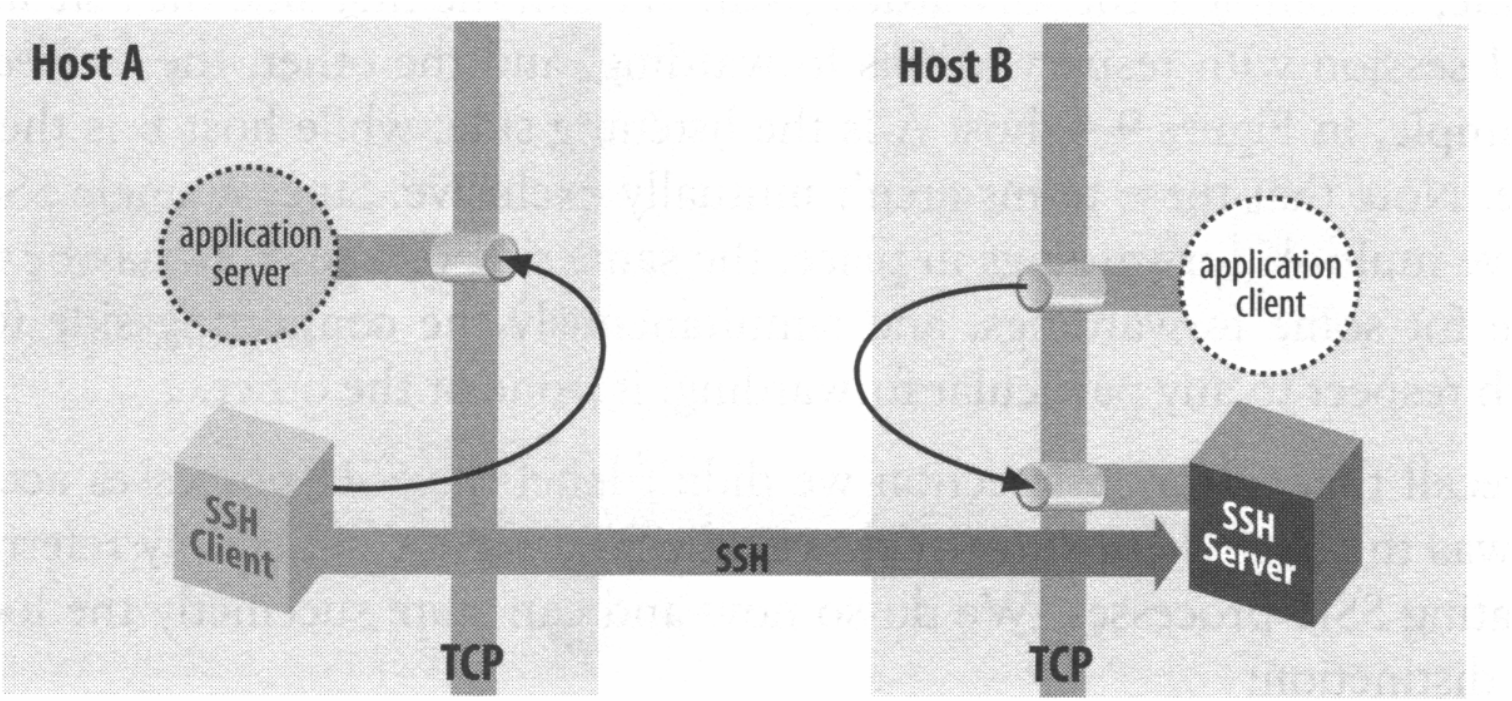


Lokalne przekazywanie połączeń charakteryzuje się tym, że aplikacja klienta oraz strona nasłuchująca znajdują się po stronie klienta SSH.

SSH: lokalne przekazywanie połączeń – przykład

```
# Wykonanie na hoście 158.75.5.2 (klient) komend
ssh [-v] -N -L 2222:localhost:22 158.75.5.91

# sprawia, że po stronie klienta 'netstat -nltp' oraz 'netstat -ntp' pokazują
# tcp        0      0 127.0.0.1:2222              0.0.0.0:*                   LISTEN 2320/ssh
# tcp        0      0 ::1:2222                    :::*                        LISTEN 2320/ssh

# tcp        0      0 158.75.5.2:55420            158.75.5.91:22              ESTABLISHED 1581/ssh

# Po stronie serwera mamy
# tcp        0      0 ::ffff:158.75.5.91:22       ::ffff:158.75.5.2:45257     ESTABLISHED 27807/sshd: jkob [p

# Wykonanie komendy
ssh -p 2222 localhost

# powoduje po stronie serwera pojawienie się dwóch dodatkowych połączeń
# tcp        0      0 127.0.0.1:56530             127.0.0.1:22                ESTABLISHED 27809/sshd: jkob
# tcp        0      0 ::ffff:127.0.0.1:22         ::ffff:127.0.0.1:56530      ESTABLISHED 27834/sshd: jkob [p

# Pierwsze jest związane z obsługą tunelowania, a drugie -- z kolejną sesja ssh.

# Komenda
ssh [-v] -g -N -L 2222:localhost:22 158.75.5.91

# powoduje udostępnienie portu 2222 nie tylko lokalnym procesom. 'netstat -nltp' pokazuje
tcp        0      0 0.0.0.0:2222                0.0.0.0:*                   LISTEN 23674/ssh
tcp        0      0 :::2222                     :::*                        LISTEN 23674/ssh
```

SSH: zdalne przekazywanie połączeń

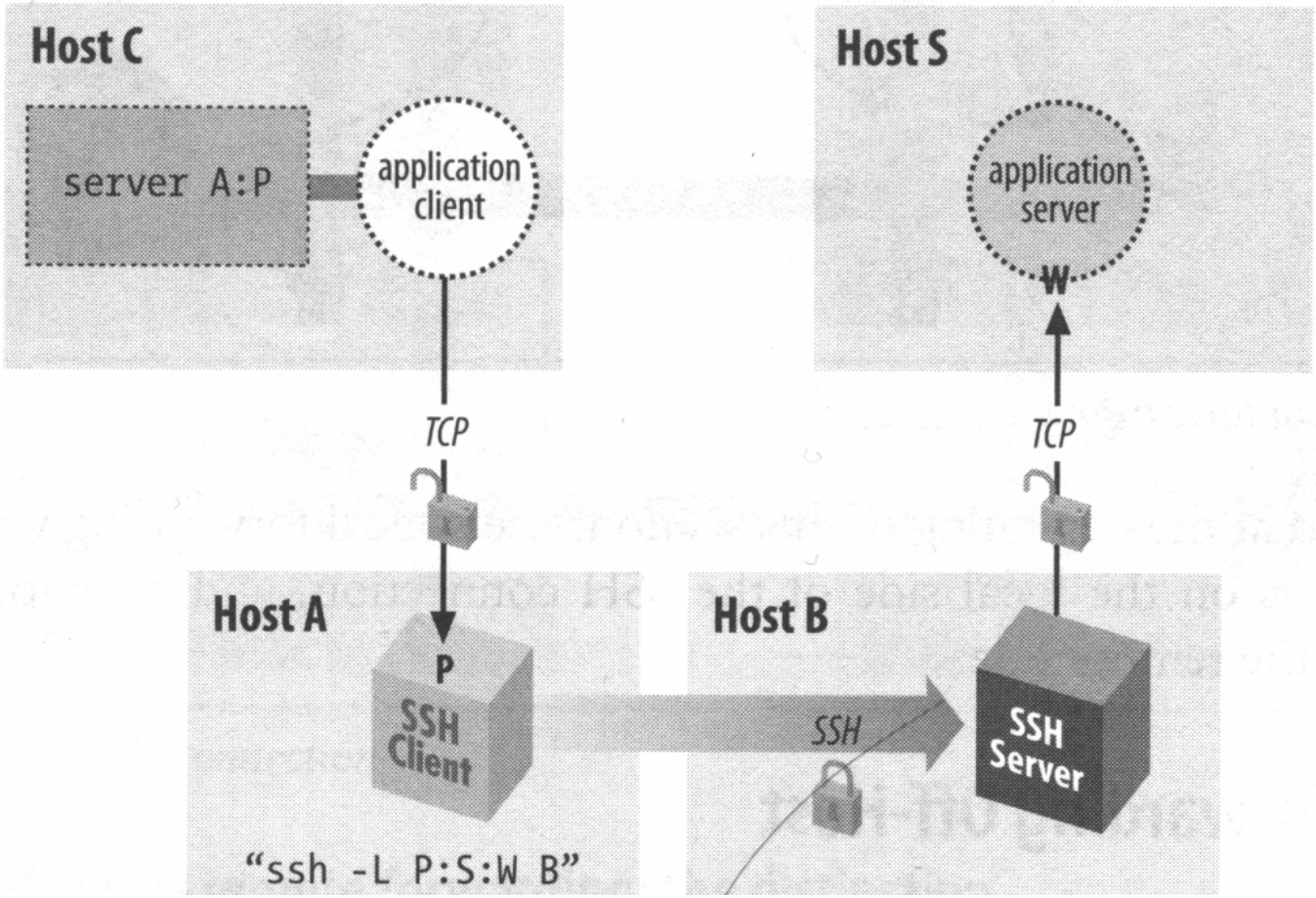


Zdalne przekazywanie połączeń charakteryzuje się tym, że aplikacja klienta oraz strona nasłuchująca znajdują się po stronie serwera SSH.

SSH: zdalne przekazywanie połączeń – przykład

```
# Wykonanie na maszynie 158.75.5.2 (klient) komedy
ssh [-v] -N -R 2222:localhost:22 158.75.5.91

# powoduje, że po stronie serwera 'netstat -nltp' pokazuje
# tcp        0      0 127.0.0.1:2222              0.0.0.0:*                   LISTEN 21554/sshd: jkob
# tcp        0      0 ::1:2222                    :::*                        LISTEN 21554/sshd: jkob

# Po wykonaniu na serwerze (zdalna maszyna)
ssh -p 2222 root@127.0.0.1

# uzyskuje się dostęp do maszyny 158.75.5.2. Na maszynie lokalnej i zdalnej pojawiają się nowe połączenia.
```

SSH: przekazywanie portów poza hostem[47]



[47] *off-host port forwarding*

SSH: przekazywanie portów poza hostem

```
# Załóżmy, że tylko z komputera 158.75.105.253 można się połączyć via ssh na serwer
# 158.75.5.91.  W jaki sposób użytkownicy sieci 158.75.105.224/27 mogą uzyskać dostęp
# np. do serwera WWW (o adresie 158.75.5.251) pracującego w sieci 158.75.4.0/23?

# Na maszynie 158.75.105.253 należy wykonać komendę:
ssh -g -L 8080:158.75.5.251:80 158.75.5.91

# Wtedy na maszynach w sieci 158.75.105.224/27 dostęp do serwera WWW będzie możliwy
# poprzez adres http://158.75.105.254:8080.
```

Wady powyższego rozwiązania:

- 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: dynamiczne przekazywanie portów

OpenSSH wspiera użycie protokołu SOCKS (RFC 1928) do rozwiązania w/w problemów.

Na maszynie 158.75.105.253 wystarczy wykonać komendę `ssh [-CN] -g -D 1080 158.75.5.91` i w przeglądarkach używanych na maszynach w lokalnej sieci włączyć obsługę proxy poprzez protokół SOCKS kierując ruch pod adres 158.75.105.253 i na port 1080.

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. Komenda 'ip link show dev tun0' pokazuje:
# 69: tun0: <POINTOPOINT,MULTICAST,NOARP> mtu 1500 qdisc noop state DOWN 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:
# 83: tap0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 500
#   link/ether 9e:d4:4b:a0:7b:6c brd ff:ff:ff:ff:ff:ff
```

SSH: ProxyCommand

Jak wykonać logowanie na zdalny sewer przechodząc przez jeden (lub więcej) innych serwerów?

Porównać

```
# ssh    polon ssh tor
# ssh -t polon ssh tor
```

Operację łączenia można uprościć przez zastosowanie opcji `ProxyCommand`.

```
# fragment pliku ~/.ssh/config

Host tor1
   hostname tor
   ProxyCommand ssh -e none polon.fizyka.umk.pl nc %h %p

Host tor2
   hostname tor
   ProxyCommand ssh [-v] -W tor:22 polon.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.[48]

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
```

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