

# UNIX

## Programování a správa systému II

Jan Kasprzak  
<kas@fi.muni.cz>  
<https://www.fi.muni.cz/~kas/>

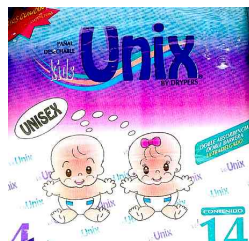
*Motto: UNIX je user-friendly,  
ale své přátele si vybírá.*

# Kapitola 1

## Úvod

# Předpoklady

- C a POSIX.1 – na úrovni PV065
- UNIX z uživatelského hlediska – shell, základní příkazy
- Síť TCP/IP z uživatelského hlediska



# Cíle kursu

- Správa systému (subsystémy, adresářový strom, ...)
- Architektura TCP/IP
- Programování síťových aplikací
- Konfigurace síťových protokolů a služeb

# Ukončení předmětu

- Průběžné odpovědníky
  - 5 dní na splnění (otevřeno 12+ dní, případně neschopenka na posledních 7 dní)
  - 4 otázky, +2 body/-1 bod 4 body = splněno
  - nejvýše tři nesplněné
- Závěrečný test: výběr právě jedné možnosti, 15 otázek, +8/-4 body
- Zkontrolujte si zapsané ukončení

## Upozornění:

Naučit se z paměti slidy nestačí!

# Ukončení předmětu


- Průběžné odpovědníky
  - 5 dní na splnění (otevřeno 12+ dní, případně neschopenka na posledních 7 dní)
  - 4 otázky, +2 body/-1 bod 4 body = splněno
  - nejvýše tři nesplněné
- Závěrečný test: výběr právě jedné možnosti, 15 otázek, +8/-4 body
- Zkontrolujte si zapsané ukončení



## Upozornění:

Naučit se z paměti slidy nestačí!

# Obsah: Správa systému


- Instalace, zálohování, disky
- Systém souborů a adresářů – rozložení v systému
- Start systému – `init`, `systemd`
- Uživatelé a skupiny – `data`, `PAM`, `nsswitch`, programování
- Linux a desktop – když nechceme statický `/dev` 
- Lokální subsystemy – `syslog`, `cron`, tiskárny, diskové kvóty
- Virtualizace – virtuální stroje, kontejnery

# Obsah: Síť TCP/IP

- Architektura sítě TCP/IP – vrstvy sítě, formáty packetů
- Programování síťových aplikací – rozhraní BSD sockets



# Obsah: Síťové aplikace

- Konfigurace sítě – ARP, přidělení adres, směrování
- DNS – architektura, typy záznamů, konfigurace
- RPC služby – RPC a XDR, portmapper, NFS, NIS/YP
- Uživatelské informace po síti – LDAP, Kerberos
- Elektronická pošta – formát zpráv, SMTP, POP-3, IMAP
- Firewally – packetové filtry, aplikační brány, netfilter 

# Typografické konvence

- Specifická vlastnost pro Linux: 
- ... pro BSD , Solaris 
- ... pro IRIX , Red Hat/Fedoru 
- ... pro GNU nástroje: 



## Úkol:

Doma se zkuste zamyslet a vyřešit.



## Příklad: Tohle rozhodně nezkoušejte ^\_~

```
root@eva01# rm -rf /
```

# Další zdroje informací

- Tato prezentace:  
<https://www.fi.muni.cz/~kas/pv077/>
- PV090 UNIX – Seminář ze správy systému
- Linux Weekly – <https://lwn.net/>
- Experimentujte! – fakultní linuxové počítače, vlastní stroj s Linuxem, <https://stratus.fi.muni.cz/>, ...

# Další zdroje informací

- Tato prezentace:  
<https://www.fi.muni.cz/~kas/pv077/>
- **PV090** UNIX – Seminář ze správy systému
- Linux Weekly – <https://lwn.net/>
- Experimentujte! – fakultní linuxové počítače, vlastní stroj s Linuxem, <https://stratus.fi.muni.cz/>, ...

# Další zdroje informací

- Tato prezentace:  
<https://www.fi.muni.cz/~kas/pv077/>
- PV090 UNIX – Seminář ze správy systému
- **Linux Weekly** – <https://lwn.net/>
- Experimentujte! – fakultní linuxové počítače, vlastní stroj s Linuxem, <https://stratus.fi.muni.cz/>, ...

# Další zdroje informací

- Tato prezentace:  
<https://www.fi.muni.cz/~kas/pv077/>
- PV090 UNIX – Seminář ze správy systému
- Linux Weekly – <https://lwn.net/>
- **Experimentujte!** – fakultní linuxové počítače, vlastní stroj s Linuxem, <https://stratus.fi.muni.cz/>, ...

# Kapitola 2

## Instalace systému

# Instalace systému - I.

Není standardizováno, jen obecná fakta:

- Start jádra z instalačního média
- Kořenový svazek: ramdisk, instalační médium, miniroot , NFS
- Rozdělení disků na oblasti - fdisk(8), parted(8) , gdisk(8), divvy(8), disklabel(8) 
- Vytvoření souborových systémů - mkfs(8), newfs(8) 
- Inicializace odkládacího prostoru - mkswap(8) 

## Otázka:

Co vlastně mkswap(8) na disk zapisuje?



# Instalace systému - I.

Není standardizováno, jen obecná fakta:



- Start jádra z instalačního média
- Kořenový svazek: ramdisk, instalační médium, miniroot , NFS
- Rozdělení disků na oblasti - fdisk(8), parted(8) , gdisk(8), divvy(8), disklabel(8) 
- Vytvoření souborových systémů - mkfs(8), newfs(8) 
- Inicializace odkládacího prostoru - mkswap(8) 



## Otázka:

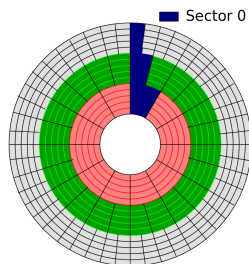
Co vlastně mkswap(8) na disk zapisuje?

# Instalace systému - II.

- Počáteční konfigurace hardware - vytvoření jádra pro nový systém (nebo *initial ramdisk* )
- Instalace jednotlivých částí systému
- Post-instalační konfigurace systému - doménové jméno, konfigurace sítě, časové zóny, systémového hesla a podobně
- Restart nainstalovaného systému
- Post-instalační konfigurace - další nastavení na živém systému (firstboot )

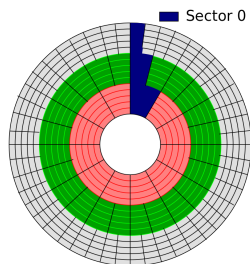
# Rotační magnetické disky

- **Geometrie:** sektory, stopy, povrchy
- Latence:
  - rotační
  - změna pozice hlavy
- Zónový zápis
  - vnější stopy jsou rychlejší!
- Sekvenční vs. náhodný přístup
  - stovky MB/s
  - stovky IOPS
- Mechanická životnost
  - ložiska, uložení hlavy
  - badblocks(8), smartctl(8)



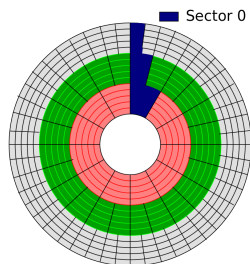
# Rotační magnetické disky

- Geometrie: sektory, stopy, povrchy
- **Latence:**
  - rotační
  - změna pozice hlavy
- Zónový zápis
  - vnější stopy jsou rychlejší!
- Sekvenční vs. náhodný přístup
  - stovky MB/s
  - stovky IOPS
- Mechanická životnost
  - ložiska, uložení hlavy
  - badblocks(8), smartctl(8)



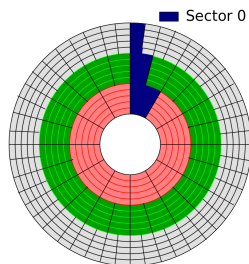
# Rotační magnetické disky

- Geometrie: sektory, stopy, povrchy
- Latence:
  - rotační
  - změna pozice hlavy
- **Zónový** zápis
  - vnější stopy jsou rychlejší!
- Sekvenční vs. náhodný přístup
  - stovky MB/s
  - stovky IOPS
- Mechanická životnost
  - ložiska, uložení hlavy
  - badblocks(8), smartctl(8)



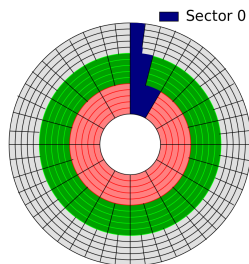
# Rotační magnetické disky

- Geometrie: sektory, stopy, povrchy
- Latence:
  - rotační
  - změna pozice hlavy
- Zónový zápis
  - vnější stopy jsou rychlejší!
- Sekvenční vs. náhodný přístup
  - stovky MB/s
  - stovky IOPS
- Mechanická životnost
  - ložiska, uložení hlavy
  - badblocks(8), smartctl(8)



# Rotační magnetické disky

- Geometrie: sektory, stopy, povrchy
- Latence:
  - rotační
  - změna pozice hlavy
- Zónový zápis
  - vnější stopy jsou rychlejší!
- Sekvenční vs. náhodný přístup
  - stovky MB/s
  - stovky IOPS
- Mechanická životnost
  - ložiska, uložení hlavy
  - badblocks(8), smartctl(8)



# Solid-state disk

- Flash paměti (NOR, NAND)
- Větší bloky než u disků (i 64 KB); erase-blocks
- Operace: čtení, mazání, zápis
- Omezená životnost: počet zápisových operací
  - SW nebo HW vyrovnávání opotřebení
  - dodatečné bity v erase-blocku
  - někdy dost primitivní
- Rychlý náhodný i sekvenční přístup
- Nelze zápis části erase-blocku
- Speciální souborové systémy
- Podpora trim? `fstrim(8)`, `blkdiscard(8)`



# Další typy úložišť

- **NVMe** disky (XIP?)
- Non-volatile RAM (memristory, ...)
- SMR disky

## Čtení na dobrou noc

Optimizing Linux with cheap flash drives  
<http://lwn.net/Articles/428584/>

# Další typy úložišť

- NVMe disky (XIP?)
- **Non-volatile RAM** (memristory, ...)
- SMR disky

## Čtení na dobrou noc

Optimizing Linux with cheap flash drives  
<http://lwn.net/Articles/428584/>

# Další typy úložišť

- NVMe disky (XIP?)
- Non-volatile RAM (memristory, ...)
- **SMR disky**

## Čtení na dobrou noc

Optimizing Linux with cheap flash drives  
<http://lwn.net/Articles/428584/>

# Další typy úložišť

- NVMe disky (XIP?)
- Non-volatile RAM (memristory, ...)
- SMR disky



## Čtení na dobrou noc

Optimizing Linux with cheap flash drives  
<http://lwn.net/Articles/428584/>

# Diskové oddíly (partition)

- **Disk** – blokové zařízení
- Využití disků – souborový systém, swapovací oblast, databáze, ...
- Disková oblast – souvislá část disku. Blokové zařízení.
- BIOS/DOS MBR – 4 oblasti, 2 TB, extended oblast, 8 bitů identifikátor typu
- BSD disklabel – rozdělení jedné MBR oblasti
- GPT – 128 oblastí, 64-bitové ID bloku, UUID typu oblasti, UUID identifikátor oblasti, protective MBR

# Diskové oddíly (partition)

- Disk – blokové zařízení
- **Využití disků** – souborový systém, swapovací oblast, databáze, ...
- Disková oblast – souvislá část disku. Blokové zařízení.
- BIOS/DOS MBR – 4 oblasti, 2 TB, extended oblast, 8 bitů identifikátor typu
- BSD disklabel – rozdělení jedné MBR oblasti
- GPT – 128 oblastí, 64-bitové ID bloku, UUID typu oblasti, UUID identifikátor oblasti, protective MBR

# Diskové oddíly (partition)

- Disk – blokové zařízení
- Využití disků – souborový systém, swapovací oblast, databáze, ...
- **Disková oblast** – souvislá část disku. Blokové zařízení.
- BIOS/DOS MBR – 4 oblasti, 2 TB, extended oblast, 8 bitů identifikátor typu
- BSD disklabel – rozdělení jedné MBR oblasti
- GPT – 128 oblastí, 64-bitové ID bloku, UUID typu oblasti, UUID identifikátor oblasti, protective MBR

# Diskové oddíly (partition)

- Disk – blokové zařízení
- Využití disků – souborový systém, swapovací oblast, databáze, ...
- Disková oblast – souvislá část disku. Blokové zařízení.
- **BIOS/DOS MBR** – 4 oblasti, 2 TB, extended oblast, 8 bitů identifikátor typu
- BSD disklabel – rozdělení jedné MBR oblasti
- GPT – 128 oblastí, 64-bitové ID bloku, UUID typu oblasti, UUID identifikátor oblasti, protective MBR



# Diskové oddíly (partition)

- Disk – blokové zařízení
- Využití disků – souborový systém, swapovací oblast, databáze, ...
- Disková oblast – souvislá část disku. Blokové zařízení.
- BIOS/DOS MBR – 4 oblasti, 2 TB, extended oblast, 8 bitů identifikátor typu
- **BSD disklabel** – rozdělení jedné MBR oblasti
- GPT – 128 oblastí, 64-bitové ID bloku, UUID typu oblasti, UUID identifikátor oblasti, protective MBR

# Diskové oddíly (partition)

- Disk – blokové zařízení
- Využití disků – souborový systém, swapovací oblast, databáze, ...
- Disková oblast – souvislá část disku. Blokové zařízení.
- BIOS/DOS MBR – 4 oblasti, 2 TB, extended oblast, 8 bitů identifikátor typu
- BSD disklabel – rozdělení jedné MBR oblasti
- **GPT** – 128 oblastí, 64-bitové ID bloku, UUID typu oblasti, UUID identifikátor oblasti, protective MBR

# Správa logických svazků

- Logical Volume Manager (lvm)
- Spojení více fyzických zařízení do jednoho

# Struktura LVM

- **Physical volume (pv)** – disk, disková oblast. Skládá se z
- **Physical extent (pe)** – část diskové oblasti, pevná délka (např. 4 MB).
- **Volume group (vg)** – obsahuje několik PV, jejichž PE jsou v ní zpřístupněny jako
- **Logical extent (le)** – odpovídá příslušnému PE.
- **Logical volume (lv)** – odpovídá blokovému zařízení. Skládá se z několika LE v rámci jedné VG. Na LV se vytvoří souborový systém a používá se.

# Další služby LVM

- Změna velikosti VG – přidání/odebrání několika PV.
- Změna velikosti LV – přidání/odebrání několika LE.  
Musí navazovat změna velikosti souborového systému.
- Odebrání PV – transparentní.
- Klon LV – atomický snímek, nezabírá mnoho místa, copy-on-write. `fsfreeze(8)`
- Thin provisioning – alokace až při zápisu. Trim?

# Device mapper



- Vrstva v jádře - přemapování blokových zařízení
- LVM - user-space detekce VG, konfigurace jaderného DM
- dm-crypt - šifrování disku (userspace: LUKS)
- dm-verity - ochrana proti *evil maid attack*
- dm-flakey, dm-dust, dm-delay
- dm-raid



# RAID-0

Disk1	Disk2	Disk3
1	2	3
4	5	6
...	...	...

- Prokládání disků
- Dva nebo více disků
- Stejně velké disky
- **Není redundantní!**

# RAID-1

Disk1	Disk2	Disk3
1	1	1
2	2	2
...	...	...

- Zrcadlení disků.
- Dva nebo více disků
  - Trik: mít všechny disky bootovatelné
- Větší propustnost čtení.

## Otázka:

Jaká je náročnost zápisu na RAID-1?



# RAID-1

Disk1	Disk2	Disk3
1	1	1
2	2	2
...	...	...

- Zrcadlení disků.
- Dva nebo více disků
  - Trik: mít všechny disky bootovatelné
- Větší propustnost čtení.



## Otázka:

Jaká je náročnost zápisu na RAID-1?

# RAID-5

Disk1	Disk2	Disk3
1	2	P12
3	P34	4
P56	5	6
...	...	...

- Paritní bloky
  - RAID-4 - paritní disk
- Tři nebo více disků
- Větší náročnost zápisu

## Otázka:

Jaký má vliv velikost bloku RAID-5 na jeho výkonnost?

# RAID-5

Disk1	Disk2	Disk3
1	2	P12
3	P34	4
P56	5	6
...	...	...

- Paritní bloky
  - RAID-4 - paritní disk
- Tři nebo více disků
- Větší náročnost zápisu



## Otázka:

Jaký má vliv velikost bloku RAID-5 na jeho výkonnost?

# RAID-6

Disk1	Disk2	Disk3	Disk4
1	2	P12	Q12
3	P34	Q34	4
P56	Q56	5	6
Q78	7	8	P78
...	...	...	...

- Dva paritní bloky
  - Různé funkce P a Q
- Čtyři nebo více disků
- Redundance i při rekonstrukci

## Otázka:

Porovnejte výkon degradovaného RAID-5 a RAID-6

# RAID-6

Disk1	Disk2	Disk3	Disk4
1	2	P12	Q12
3	P34	Q34	4
P56	Q56	5	6
Q78	7	8	P78
...	...	...	...

- Dva paritní bloky
  - Různé funkce P a Q
- Čtyři nebo více disků
- Redundance i při rekonstrukci



## Otázka:

Porovnejte výkon degradovaného RAID-5 a RAID-6

# RAID-10

Disk1	Disk2	Disk3	Disk4
1	1	2	2
3	3	4	4
5	5	6	6
...	...	...	...

- RAID-0 nad RAID-1 částmi

## Otázka:

Lze mít RAID-10 nad lichým počtem disků?

# RAID-10

Disk1	Disk2	Disk3	Disk4
1	1	2	2
3	3	4	4
5	5	6	6
...	...	...	...

- RAID-0 nad RAID-1 částmi



## Otázka:

Lze mít RAID-10 nad lichým počtem disků?

# RAID-10 near/far


## ■ Alternativní rozložení RAID-10

Disk1	Disk2	Disk3	Disk4
1	1	2	2
3	3	4	4
5	5	6	6
...	...	...	...

Disk1	Disk2	Disk3	Disk4
1	2	3	4
5	6	7	8
...	...	...	...
3	4	1	2
7	8	5	6
...	...	...	...




# SW RAID nebo HW RAID?

- RAID write hole
  - Write-intent-bitmap 
- Cache, scheduling
  - Bufferbloat?
- Disaster recovery
  - Chyby ve firmwaru?

## Doporučení:

V Linuxu Použijte SW RAID (md).


# SW RAID nebo HW RAID?

- RAID write hole
  - Write-intent-bitmap 
- Cache, scheduling
  - Bufferbloat?
- Disaster recovery
  - Chyby ve firmwaru?

## Doporučení:

V Linuxu Použijte SW RAID (md).


# SW RAID nebo HW RAID?

- RAID write hole
  - Write-intent-bitmap 
- Cache, scheduling
  - Bufferbloat?
- **Disaster recovery**
  - Chyby ve firmwaru?

## Doporučení:

V Linuxu Použijte SW RAID (md).

# SW RAID nebo HW RAID?

- RAID write hole
  - Write-intent-bitmap 
- Cache, scheduling
  - Bufferbloat?
- Disaster recovery
  - Chyby ve firmwaru?



## Doporučení:

V Linuxu Použijte SW RAID (md).



# Kapitola 3

## Souborové systémy

# Vlastnosti souborových systémů

System souborů musí zajišťovat:

- **Efektivní práce s metadaty** – adresářové operace (vyhledání souboru, přejmenování, mnoho souborů v adresáři, atd.).
- **Efektivní operace s daty** – čtení/zápis (malá fragmentace etc.)
- **Spolehlivé zotavení po havárii.**
- **Co nejmenší prostor na režii** – velikost metadat.

# Svazky

- **Svazek**: systém souborů, **volume**.
- Uložen na blokovém zařízení (disková oblast).
  - Ale: síťové svazky nemají blokové zařízení
- **Připojení svazku** – na existující adresář.
  - ztotožnění s kořenem připojovaného svazku
  - **mountpoint**

# Struktura souborového systému

- **Boot block** je první blok svazku. Zavádí se z něj operační systém, nebo je prázdný.
- **Super block** – další blok svazku. Obsahuje sumární informace o svazku.
- **Tabulka i-uzlů** – informace o souborech.
- **Datové bloky**



# Zotavení po havárii

- Možné nekonzistence:
  - pořadí zápisových operací
  - write-back cache
  - změny dat/metadat
  - chyby HW nebo OS
- Kontrola konzistence fsck(8). Časově náročné.
- Synchronní zápis metadat? – problémy se starými daty v souborech (bezpečnost!).

# BSD Soft updates

- Závislosti mezi diskovými operacemi.
- Omezení počtu typů nekonzistencí (rychlejší `fsck(8)`).
- Ale: problém pořadí data versus metadata.
- Neřeší se chyba OS nebo HW.
- Komplikovaná implementace.

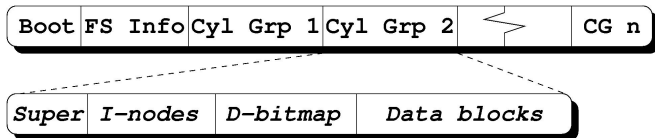
# Žurnálované souborové systémy

- Transakční přístup.
- Změny nejprve zapsány do logu (žurnálu) a pak provedeny.
- Po havárii – přehrání celých transakcí.
- Některé operace – i rychlejší než nežurnálovaný FS.
- Celkově o něco pomalejší.
- Žurnál jen metadat nebo i dat.
- Chyba OS nebo HW se řeší pomocí fsck(8).
- Jen transakce z jádra (ne user-space).

# FAT

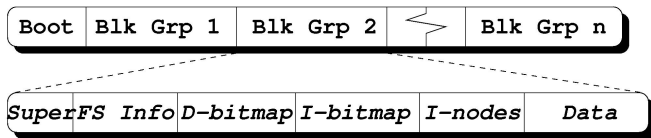
- Nemá i-uzly (nelze mít soubor ve více adresářích, nemá UNIXová přístupová práva).
- Pomalý přímý přístup k souboru (sekvenční procházení přes FAT).
- Fragmentace už při současném zápisu do dvou souborů.
- Fragmentace při rušení souboru.
- Na větších FS velká délka bloku → špatné využití místa.
- Výhody - na menších FS malá režie, jednoduchá implementace.

# UFS



- **FFS, EFS, UFS** - původně v 4.x BSD.
- Cylinder groups. Nutná znalost geometrie disku.
- Snížení fragmentace, 4-8 KB bloky.
- **Fragmenty** - lepší využití místa na disku.
- Kopie superbloku.
- Rezervované místo pro superuživatele
- Původně: synchronní zápis metadat.
- FreeBSD: soft updates.
- Kontrola disku na pozadí.
- \*BSD, Solaris (+ žurnálování), Linux.

# Ext2 filesystem



- Skupiny bloků (block groups). Není nutná znalost geometrie disku. Jednodušší implementace, využití celých bloků.
- Alokační strategie: Předalokované bloky, alokace dat poblíž příslušných metadat, zamezení zaplnění jedné skupiny bloků.
- Obvykle 1 KB (až 4 KB) bloky - rychlejší než FFS s 4 KB bloky.
- Bitmapa volných i-uzlů.

# Ext2FS - pokračování

- Asynchronní zápis metadat; na požádání umí i synchronní.
- Velikost až do 4 TB dat.
- Rychlé symbolické linky.
- No-atime, relatime.
- Maximum mount count. tune2fs (8).
- Možnosti při chybě – panic, remount r-only, ignore.
- libe2fs – knihovna pro přístup k e2fs. e2defrag.

# Ext3 filesystem



- Struktury na disku - zpětně kompatibilní s ext2.
- **Žurnálování** - změny zapisovány přes transakční log.
- **Žurnálování dat** - journal, ordered, writeback.
  - **0\_PONIES**
- **Rozšířené atributy** - další metadata (např. security context).
- **Access control lists**
- **Adresáře** - lineární struktura nebo strom (HTree; bezpečnost!)



# Ext4 filesystem



- Limit velikosti 1 EiB
- Extent-based adresace
- Předalokace místa - `fallocate(2)`
- Časová razítka - rozlišení 1 ns, `statx(2)`
- Allocate on flush (viz XFS)
- Kvóty na projekty (32-bitové ID)
- Zápisové bariéry (`barrier=0`, viz `0_PONIES`)



- Všechna data v jednom B+ stromu.
- Alokace místa – i menší kousky než jeden sektor.
- I-uzly – alokace podle potřeby.
- Efektivní i při velkém množství souborů v adresáři nebo velkém množství malých souborů.



- **Plug-iny** souborového systému (např. vyhledávání/indexace).
- **Soubory s více proudy dat** (např. metadata) – každý soubor je také adresář.
- **Transakce** – více datových operací může být spojeno do jedné atomické transakce.





- Rozdělení svazku - allocation groups velikosti 0.5 až 4 GB.
- Organizace dat - B+ strom
- DMAPI - data manipulation API - zpřístupnění vlastností B-stromu (vkládání/rušení dat uprostřed souboru).
- Real-time extenze - možnost alokace šířky pásma; garantovaná propustnost.
- O\_DIRECT - přístup bez cachování.
- Allocate on flush - další snížení fragmentace.
- XFS on RAID - podpora paralelizace.

- Zettabyte File System
- Interně podobný jako Slab alokátor v paměti.
- RAID-Z – jednotlivé slaby s různou úrovní redundance.
- Kontrolní součty dat
- Self-healing (automatické opravy chyb).
- Copy-on-write: sjednocení duplicitních bloků

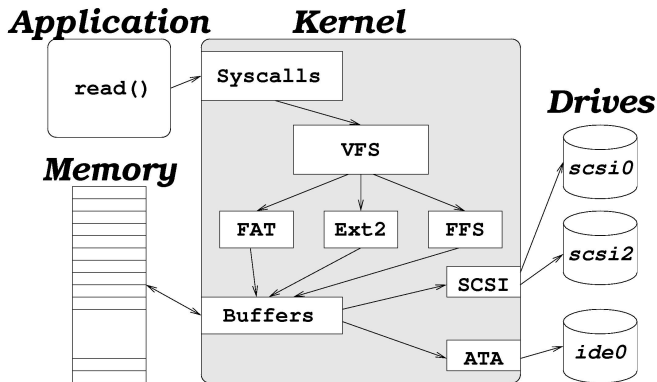


- Copy-on-write B-stromy
  - top-down B-tree
  - závislosti operací jen v jedné datové struktuře (na rozdíl od soft-updates)
  - evidence volného místa
- Zapisovatelné snímky FS.
- Subvolumes
- Kontrolní součty metadat (volitelně i dat).
- Interní RAID
- In-place konverze z ext4
- Deduplikace

# Další služby FS

- Komprese dat – celý FS nebo jen určité soubory.
- Obnova smazaných souborů.
- Nepřemistitelné soubory – ext[234]fs.
- Soubory, umožňující pouze přidávat data – append-only.
- Změna velikosti svazku za běhu – AIX jfs, Tru64 advfs, ext[34]fs, BTRFS, ...
- Transparentní šifrování
- Steganografie – StegFS , Rubberhose 

# Virtual file system





# Zálohování

## Proč zálohovat?

- Ochrana dat před nechtěným smazáním
- Ochrana dat před **výpadkem** hardwaru
- Sledování změn v datech
- Obnovení dat po bezpečnostním incidentu



# Problémy zálohování

- Nízká rychlost zálohovacích médií  
- nejde o snímek systému
- Malá kapacita médií - nelze každý den zálohovat všechno
- Nespolehlivost médií - je nutno mít několik sad záloh



# Víceúrovňové zálohování

- Řeší problém rychlosti a velikosti zálohovacího média
- Záloha úrovně 0 – kompletní svazek nebo adresář
- Záloha úrovně  $n+1$  – soubory a adresáře, modifikované od začátku zálohy úrovně  $n$
- Rozpoznat i smazané soubory
- Čas vytvoření zálohy musí být uložen na zálohovacím médiu, nikoli na disku

# Víceúrovňové zálohování

- Řeší problém rychlosti a velikosti zálohovacího média
- Záloha úrovně 0 – kompletní svazek nebo adresář
- Záloha úrovně  $n+1$  – soubory a adresáře, modifikované od začátku zálohy úrovně  $n$
- Rozpoznat i **smazané soubory**
- Čas vytvoření zálohy musí být uložen na zálohovacím médiu, nikoli na disku

# Víceúrovňové zálohování

- Řeší problém rychlosti a velikosti zálohovacího média
- Záloha úrovně 0 – kompletní svazek nebo adresář
- Záloha úrovně  $n+1$  – soubory a adresáře, modifikované od začátku zálohy úrovně  $n$
- Rozpoznat i smazané soubory
- Čas vytvoření zálohy musí být uložen na zálohovacím médiu, nikoli na disku

# Formát záloh

- **Vlastnický formát** - nevýhoda - nelze zálohu rozbalit kdekoli
- **tar(1)** - neumožňuje zabalit jen některé soubory (GNU tar ano 🤪)
- **cpio(1)** - pozor na zabalení s absolutní cestou
- **dump(8)** - formát příslušný určitému typu souborového systému. Odpovídající restore(8) obvykle umí běžet nad libovolným FS
- **Zálohy databází** - nutná spolupráce DB stroje

# Centrální zálohovací systémy

- Bacula ([bacula.org](http://bacula.org))
  - centrální server s databází
  - zálohované stroje
  - uzly pro ukládání záloh
- Restic ([restic.net](http://restic.net))
  - zálohovaný stroj
  - úložiště dat (SSH, REST, S3/Ceph, ...)
  - šifrování, komprese, deduplikace, kontrolní součty
  - repozitář s více zálohami (viz Git)
  - pozor na útoky typu ransomware


# Co vzít v úvahu

- **On-line replika** – standby databáze, zrcadlení přes DRBD, atd.
- **Off-site backup** – proti živelným pohromám a krádeži
- **Zabezpečení** – šifrovaná nebo zamčená záloha
- **Uchovávat i hodně staré zálohy**



## DRBD

Distributed Replicated Block Device

<http://www.drbd.org/> 

## Upozornění:

RAID není záloha!




# Co vzít v úvahu

- **On-line replika** – standby databáze, zrcadlení přes DRBD, atd.
- **Off-site backup** – proti živelným pohromám a krádeži
- **Zabezpečení** – šifrovaná nebo zamčená záloha
- **Uchovávat i hodně staré zálohy**



## DRBD

Distributed Replicated Block Device

<http://www.drbd.org/> 



## Upozornění:

RAID není záloha!

# Zálohování na disk

- Rychlejší – přímý přístup versus převíjení pásky
- Nebezpečnější – `rm -rf`, elektrický výboj
- Cenově dostupnější – není třeba zvláštní mechaniku a řadič
- `rsync(1)` – synchronizace dvou adresářů; přenáší se jen rozdíly
- `cp -l` – kopie podstromu, běžné soubory jen hardlinkované
- Lze vylepšovat – odmontování a uspání disku, šifrování, atd.


# Zálohování na disk

- Rychlejší – přímý přístup versus převíjení pásky
- Nebezpečnější – `rm -rf`, elektrický výboj
- Cenově dostupnější – není třeba zvláštní mechaniku a řadič
- `rsync(1)` – synchronizace dvou adresářů; přenáší se jen rozdíly
- `cp -l` – kopie podstromu, běžné soubory jen hardlinkované 🐮.
- Lze vylepšovat – odmontování a uspání disku, šifrování, atd.

# Kontrolujte funkčnost záloh



# Rozložení adresářů v systému

- **Tradiční** – není specifikováno žádnou de iure normou
- **Rozdíly** – BSD versus System V, modifikace od jednotlivých výrobců
- **Linux** – FileSystem Hierarchy standard (FHS) 



# Kořenový svazek - I.

- Malý svazek, není sdílen mezi více stroji.
- Programy by neměly vyžadovat vytváření dalších souborů nebo adresářů přímo pod /.

`/bin` - uživatelské programy, nezbytné k jednouživatelskému běhu systému a k nastartování sítě

`/boot` - soubory zavaděče systému a jádro (někdy samostatný svazek; měl by být dostupný firmwaru počítače)

`/dev` - speciální soubory. Obvykle obsahuje program MAKEDEV(8)

## Kořenový svazek - II.

`/etc` - konfigurační soubory. Nelze sdílet mezi počítači. Na systémech blízkých SVR3 navíc spustitelné soubory pro správce systému (např. `mount(8)`); jinak viz `/sbin`).

`/home` - domovské adresáře uživatelů. Obvykle samostatný svazek. Někdy `/usr`, `/usr/home` nebo jiný.

`/lib` - sdílené knihovny, nezbytné pro jednu uživatelský běh systému. Plug-iny, moduly jádra a další data.

`/lib32`, `/lib64` - alternativní adresáře pro multilib systémy.

## Kořenový svazek - III.

- `/media` – pro automaticky připojované svazky.
- `/mnt` – pro dočasně připojované svazky.
- `/opt` – přidané větší softwarové balíky. Obvykle samostatný svazek.
- `/root` – domovský adresář superuživatele. Někdy též `/`.
- `/sbin` – systémové programy (programy, které používá jen systém sám nebo správce systému). Na některých systémech chybí a tyto programy jsou v `/etc`.
- `/tmp` – dočasné soubory. Adresář, přístupný všem uživatelům (vyžaduje sticky bit). Některé systémy promazávají `/tmp` při startu systému.



# Adresář /usr - I.

- **Sdílitelná data**, přístupná v běžném případě pouze pro čtení.
- Subsystémy by neměly vytvářet další adresáře pod /usr - historickou výjimkou je X11.

**X11** - X Window System (často též X11R6).  
Obsahuje mj. i podadresáře bin, lib a include s odkazy z adresářů /usr/bin, /usr/lib a /usr/include.

**bin** - uživatelské programy, které nejsou nezbytné v jednouživatelském režimu. Také zde jsou interpretery.

**doc** - dokumentace (někdy share/doc).

## Adresář /usr - II.

`games` - hry a vzdělávací programy `^_~`

`include` - hlavičkové soubory pro jazyk C.

`lib` - knihovny, které nejsou nezbytně nutné pro jednu uživatelský běh systému. Read-only data aplikací, závislá na platformě (například moduly pro Perl a podobně).

`lib32`, `lib64` - podobně jako `/lib32`, `/lib64`.

`libexec` - programy, které nejsou určeny ke spouštění uživatelem.

`local` - adresář pro lokálně instalovaný software. Obsahuje podadresáře `bin`, `games`, `include`, `lib`, `sbin`, `share` a `src`. BSD sem dává i porty 🐉.

## Adresář /usr - III.

`man` - manuálové stránky (někdy `share/man`).


`sbin` - systémové programy, které nejsou nezbytně nutné pro běh systému (síťové služby, tiskový démon a podobně).

`share` - data nezávislá na architektuře (informace o časových zónách, `terminfo` a podobně).

`src` - zdrojové texty od systémových komponent.

`spool`, `tmp` - symbolické linky do `/var` z důvodu zpětné kompatibility.

# Adresář /var – I.

- Data, která se mohou měnit (tiskové fronty, mailboxy, různé cache).
- Není sdílitelný mezi počítači.
  - `adm` – administrativní data. Často obsahuje systémové logy .
  - `cache` – generovaná data (cache) subsystémů.
  - `lock` – aplikační zámky – například zámky na sériové linky.
  - `log` – systémové logy.
  - `mail` – poštovní schránky uživatelů (někdy ve `pool/mail`).

# Adresář /var – II.

`opt` – modifikovatelná data pro balíky v /opt.

`run` – soubory, vztahující se k běžícím programům.

`spool` – fronty (tiskové, poštovní a další).

`tmp` – dočasné soubory.

## Otázka:

Do kterého adresáře byste umístili PID soubor běžícího procesu?

A kam pomocný program, který je spouštěn z vámi implementovaného démona?

# Adresář /var – II.

`opt` – modifikovatelná data pro balíky v /opt.

`run` – soubory, vztahující se k běžícím programům.

`spool` – fronty (tiskové, poštovní a další).

`tmp` – dočasné soubory.



## Otázka:

Do kterého adresáře byste umístili PID soubor běžícího procesu?

A kam pomocný program, který je spouštěn z vámi implementovaného démona?

# Aktuální vývoj

- `usrmove` – přesun read-mostly dat z / do /usr.

## Otázka:

Proč ne naopak /usr/bin do /bin?

- Stateless systémy – pro cloud, bez lokálních dat (CoreOS, Project Atomic, etcd).
- /run – pokud možno před připojením /var.

## Čtení na dobrou noc

Factory Reset, Stateless Systems,  
Reproducible Systems & Verifiable Systems  
[https://0pointer.de/blog/projects/  
stateless.html](https://0pointer.de/blog/projects/stateless.html)

# Aktuální vývoj

- `usrmove` – přesun read-mostly dat z / do /usr.



## Otázka:

Proč ne naopak /usr/bin do /bin?

- Stateless systémy – pro cloud, bez lokálních dat (CoreOS, Project Atomic, etcd).
- /run – pokud možno před připojením /var.

## Čtení na dobrou noc

Factory Reset, Stateless Systems,  
Reproducible Systems & Verifiable Systems  
[https://0pointer.de/blog/projects/  
stateless.html](https://0pointer.de/blog/projects/stateless.html)



# Aktuální vývoj

- `usrmove` – přesun read-mostly dat z / do /usr.



## Otázka:

Proč ne naopak /usr/bin do /bin?

- **Stateless systémy** – pro cloud, bez lokálních dat (CoreOS, Project Atomic, etcd).
- /run – pokud možno před připojením /var.

## Čtení na dobrou noc

Factory Reset, Stateless Systems,  
Reproducible Systems & Verifiable Systems  
[https://0pointer.de/blog/projects/  
stateless.html](https://0pointer.de/blog/projects/stateless.html)

# Aktuální vývoj

- `usrmove` – přesun read-mostly dat z / do /usr.



## Otázka:

Proč ne naopak /usr/bin do /bin?

- **Stateless systémy** – pro cloud, bez lokálních dat (CoreOS, Project Atomic, etcd).
- `/run` – pokud možno před připojením `/var`.

## Čtení na dobrou noc

Factory Reset, Stateless Systems,  
Reproducible Systems & Verifiable Systems  
[https://0pointer.de/blog/projects/  
stateless.html](https://0pointer.de/blog/projects/stateless.html)

# Aktuální vývoj

- `usrmove` – přesun read-mostly dat z / do /usr.

## ? Otázka:

Proč ne naopak /usr/bin do /bin?

- **Stateless systémy** – pro cloud, bez lokálních dat (CoreOS, Project Atomic, etcd).
- `/run` – pokud možno před připojením /var.

## i Čtení na dobrou noc

Factory Reset, Stateless Systems,  
Reproducible Systems & Verifiable Systems  
[https://0pointer.de/blog/projects/  
stateless.html](https://0pointer.de/blog/projects/stateless.html)




# Kapitola 4

## Start systému

# Start systému - init

- Program - /sbin/init
- Proces číslo 1.

## Varianty:


- BSD init - soubory /etc/gettytab a /etc/rc 
- System V init - řídicí soubor /etc/inittab
- Systemd 
- Další - OpenRC, Upstart, runit, SMF , ...



## Hard-coded paths

/sbin/init, někdy ještě /bin/sh.  
Ostatní je nezávislé na kernelu.

# SystemV init: Úrovně běhu systému

- Runlevels - číslo od 0 do 6.
- Určuje, které subsystemy jsou aktivní.
- 0 - Halt - zastavení systému
- 1 - Single - jednouživatelský běh systému
- 2 - Multi - víceživatelský běh systému
- 3 - Remote FS - obvykle 2 + sdílení disků
- 4 - Free
- 5 - Free - Red Hat zde má 3 + X-Window system 
- 6 - Reboot - restart systému

# System V init

## init(8), telinit(8)


```
# init [0123456aAbBcCsSqQ]
```

- 0-6 Přejít na příslušnou úroveň chodu systému.
- a-c,A-C Nastartování jednorázových činností, stav se neeviduje.
- sS Totéž co init 1, jen konzolou se stane současný terminál.
- qQ Způsobí znovunačtení souboru /etc/inittab.

# System V startovací skripty

- Startovací skripty v `/etc/init.d` – pro každý subsystém.
- Adresáře `/etc/rc[0-6].d`:
- Symbolické linky `[SK][0-9][0-9]skript` (například `K56syslog` nebo `S60sshd`) do `../init.d`.
- Startovací skripty se spouštějí s parametrem `start` nebo `stop`.

## Red Hat/Fedora:

- Adresář `/etc/sysconfig`.
- Další parametry: `restart`, `reload`, `condrestart` a `status`.
- Program `chkconfig(8)`. Také na IRIXu .



## 📄 Příklad: Soubor /etc/inittab

```
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
...
l6:6:wait:/etc/rc.d/rc 6
ud::once:/sbin/update
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
pf::powerfail:/sbin/shutdown -h 'Power fail'
pr:12345:powerokwait:/sbin/shutdown -c \
    'Power restored'
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
4:2345:off:/sbin/mingetty tty4
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

# /etc/inittab

## Příklad: Syntaxe inittab

```
id:2345:respawn:/sbin/mingetty tty1
```

- **Identifikace úlohy** – pozor, v některých systémech může být nejvýše dvouznaková.
- Runlevel
- Způsob spouštění:
  - sysinit
  - once, wait
  - powerfail, powerok, powerokwait
  - respawn
  - off
- Příkaz + argumenty

# /etc/inittab

## 📄 Příklad: Syntaxe inittab

```
id:2345:respawn:/sbin/mingetty tty1
```

- **Identifikace úlohy** – pozor, v některých systémech může být nejvýše dvouznaková.
- **Runlevel**
- **Způsob spouštění:**
  - sysinit
  - once, wait
  - powerfail, powerok, powerokwait
  - respawn
  - off
- **Příkaz + argumenty**

# /etc/inittab

## 📄 Příklad: Syntaxe inittab

```
id:2345:respawn:/sbin/mingetty tty1
```

- Identifikace úlohy – pozor, v některých systémech může být nejvýše dvouznaková.
- Runlevel
- Způsob spouštění:
  - sysinit
  - once, wait
  - powerfail, powerok, powerokwait
  - respawn
  - off
- Příkaz + argumenty

# /etc/inittab

## ■ Příklad: Syntaxe inittab

```
id:2345:respawn:/sbin/mingetty tty1
```

- Identifikace úlohy – pozor, v některých systémech může být nejvýše dvouznaková.
- Runlevel
- Způsob spouštění:
  - sysinit
  - once, wait
  - powerfail, powerok, powerokwait
  - respawn
  - off
- Příkaz + argumenty

# Identifikace úlohy v souboru `inittab`



## Otázka:

K čemu slouží první sloupec v souboru `inittab`?

```
id:5:initdefault:
si::sysinit:/etc/rc.d/rc.sysinit
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
...
l6:6:wait:/etc/rc.d/rc 6
ud::once:/sbin/update
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
4:2345:off:/sbin/mingetty tty4
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

# Problémy System V initu

- Přístup k mnoha souborům (zpomaluje boot).
- Možné nekonzistence linků v /etc/rc.d.
- Jen priority, ne závislosti:
  - „Když restartuji X, mělo by se automaticky restartovat Y.“
- Nemožnost paralelizace.
- Chybí podrobnější konfigurace
  - respawn interval apod.
- Init neví, jestli služba (ještě) běží.

# Problémy System V initu

- Přístup k mnoha souborům (zpomaluje boot).
- Možné **nekonzistence linků** v /etc/rc.d.
- Jen priority, ne závislosti:
  - „Když restartuji X, mělo by se automaticky restartovat Y.“
- Nemožnost paralelizace.
- Chybí podrobnější konfigurace
  - respawn interval apod.
- Init neví, jestli služba (ještě) běží.



# Problémy System V initu

- Přístup k mnoha souborům (zpomaluje boot).
- Možné nekonzistence linků v /etc/rc.d.
- Jen priority, ne **závislosti**:
  - „Když restartuji X, mělo by se automaticky restartovat Y.“
- Nemožnost paralelizace.
- Chybí podrobnější konfigurace
  - respawn interval apod.
- Init neví, jestli služba (ještě) běží.

# Problémy System V initu

- Přístup k mnoha souborům (zpomaluje boot).
- Možné nekonzistence linků v /etc/rc.d.
- Jen priority, ne závislosti:
  - „Když restartuji X, mělo by se automaticky restartovat Y.“
- Nemožnost **paralelizace**.
- Chybí podrobnější konfigurace
  - respawn interval apod.
- Init neví, jestli služba (ještě) běží.

# Problémy System V initu

- Přístup k mnoha souborům (zpomaluje boot).
- Možné nekonzistence linků v /etc/rc.d.
- Jen priority, ne závislosti:
  - „Když restartuji X, mělo by se automaticky restartovat Y.“
- Nemožnost paralelizace.
- Chybí **podrobnější konfigurace**
  - respawn interval apod.
- Init neví, jestli služba (ještě) běží.

# Problémy System V initu

- Přístup k mnoha souborům (zpomaluje boot).
- Možné nekonzistence linků v /etc/rc.d.
- Jen priority, ne závislosti:
  - „Když restartuji X, mělo by se automaticky restartovat Y.“
- Nemožnost paralelizace.
- Chybí podrobnější konfigurace
  - respawn interval apod.
- Init neví, jestli služba (ještě) běží.

# Systemd

- Start úloh podle potřeby
- Předem otevřené sockety, automount body, atd.
- Aktivace/serializace až při použití
- Sledování procesů přes control groups
- Snaha vyhnout se shellu
  - Ale: zpětně kompatibilní se SystemV skripty
- Konfigurace: **Unity** různých typů
- Periodické spouštění úloh a další typy činností

## Čtení na dobrou noc

Lennart Poettering: Rethinking PID 1

<http://0pointer.de/blog/projects/systemd.html>

# Systemd

- Start úloh podle potřeby
- Předem otevřené sockety, automount body, atd.
- Aktivace/serializace až při použití
- Sledování procesů přes control groups
- Snaha vyhnout se shellu
  - Ale: zpětně kompatibilní se SystemV skripty
- Konfigurace: **Unity** různých typů
- Periodické spouštění úloh a další typy činností



## Čtení na dobrou noc

Lennart Poettering: Rethinking PID 1

<http://0pointer.de/blog/projects/systemd.html>

# Systemd unit

- formát `.ini`
- název: *identifikátor.typ-jednotky*, např. `sshd.service`
- umístění: `/etc/systemd`, `/lib/systemd`, `/run/systemd ...`
- sekce: `[Unit]`, `[Install]`

## Příklad: `multi-user.target`

```
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

# Typy jednotek

- target
- service
- socket
- device
- mount
- automount
- timer
- swap
- slice – hierarchické řízení zdrojů přes CGroups
- scope – procesy nespouštěné přes systemd



## 📄 Příklad: sshd.service

### [Unit]

Description=OpenSSH server daemon

After=network.target sshd-keygen.service

Wants=sshd-keygen.service

### [Service]

Type=forking

PIDFile=/var/run/sshd.pid

EnvironmentFile=-/etc/sysconfig/ssh

ExecStart=/usr/sbin/sshd \$OPTIONS

ExecReload=/bin/kill -HUP \$MAINPID

KillMode=process

Restart=on-failure

RestartSec=42s

### [Install]

WantedBy=multi-user.target

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`



# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Systemd: ovládání

- `systemctl enable [--now] sshd.service`
  - `systemctl enable getty@ttyS1.service`
  - `/usr/lib/systemd/system/getty@.service`
  - `/etc/systemd/system/  
multi-user.target.wants/`
- `systemctl daemon-reload`
- `systemctl disable [--now] sshd.service`
- `systemctl start sshd.service`
- `systemctl stop sshd.service`
- `systemctl status sshd.service`
- `systemctl list-units`
- `systemctl reset-failed sshd.service`
- `systemctl is-system-running`

# Service units

- Type=simple, exec, forking, oneshot, dbus, notify, notify-reload, idle
- ExecStart=, ExecStop=, ExecReload=
- Restart=, RestartSec=
- Parametry spouštění - `systemd.exec(5)`
  - User=, Group=, DynamicUser=
  - WorkingDirectory=, RootDirectory=
  - BindPaths=, BindReadOnlyPaths=
  - NoNewPrivileges=, RestrictSUIDSGID=
  - ProtectSystem=, ProtectHome=
  - ReadWritePaths=
  - RuntimeDirectory=
  - SystemCallFilter=

# Service units

- Type=simple, exec, forking, oneshot, dbus, notify, notify-reload, idle
- ExecStart=, ExecStop=, ExecReload=
- Restart=, RestartSec=
- Parametry spouštění - `systemd.exec(5)`
  - User=, Group=, DynamicUser=
  - WorkingDirectory=, RootDirectory=
  - BindPaths=, BindReadOnlyPaths=
  - NoNewPrivileges=, RestrictSUIDSGID=
  - ProtectSystem=, ProtectHome=
  - ReadWritePaths=
  - RuntimeDirectory=
  - SystemCallFilter=

# Service units

- Type=simple, exec, forking, oneshot, dbus, notify, notify-reload, idle
- ExecStart=, ExecStop=, ExecReload=
- Restart=, RestartSec=
- Parametry spouštění - `systemd.exec(5)`
  - User=, Group=, DynamicUser=
  - WorkingDirectory=, RootDirectory=
  - BindPaths=, BindReadOnlyPaths=
  - NoNewPrivileges=, RestrictSUIDSGID=
  - ProtectSystem=, ProtectHome=
  - ReadWritePaths=
  - RuntimeDirectory=
  - SystemCallFilter=

# Service units

- Type=simple, exec, forking, oneshot, dbus, notify, notify-reload, idle
- ExecStart=, ExecStop=, ExecReload=
- Restart=, RestartSec=
- Parametry spouštění - `systemd.exec(5)`
  - User=, Group=, DynamicUser=
  - WorkingDirectory=, RootDirectory=
  - BindPaths=, BindReadOnlyPaths=
  - NoNewPrivileges=, RestrictSUIDSGID=
  - ProtectSystem=, ProtectHome=
  - ReadWritePaths=
  - RuntimeDirectory=
  - SystemCallFilter=

# Systemd - další vlastnosti

- **systemd-tmpfiles(8)**
- systemd-oomd(8)
- systemd-logind(8)
- systemd-coredumpd(8)
- systemd-networkd(8)
- systemd-resolved(8)
- systemd-run(8) a `systemctl --user`

# Systemd - další vlastnosti

- `systemd-tmpfiles(8)`
- `systemd-oomd(8)`
- `systemd-logind(8)`
- `systemd-coredumpd(8)`
- `systemd-networkd(8)`
- `systemd-resolved(8)`
- `systemd-run(8)` a `systemctl --user`



# Systemd - další vlastnosti

- `systemd-tmpfiles(8)`
- `systemd-oomd(8)`
- `systemd-logind(8)`
- `systemd-coredumpd(8)`
- `systemd-networkd(8)`
- `systemd-resolved(8)`
- `systemd-run(8)` a `systemctl --user`

# Systemd - další vlastnosti

- `systemd-tmpfiles(8)`
- `systemd-oomd(8)`
- `systemd-logind(8)`
- `systemd-coredumpd(8)`
- `systemd-networkd(8)`
- `systemd-resolved(8)`
- `systemd-run(8)` a `systemctl --user`

# Systemd - další vlastnosti

- `systemd-tmpfiles(8)`
- `systemd-oomd(8)`
- `systemd-logind(8)`
- `systemd-coredumpd(8)`
- `systemd-networkd(8)`
- `systemd-resolved(8)`
- `systemd-run(8)` a `systemctl --user`

# Systemd - další vlastnosti

- `systemd-tmpfiles(8)`
- `systemd-oomd(8)`
- `systemd-logind(8)`
- `systemd-coredumpd(8)`
- `systemd-networkd(8)`
- `systemd-resolved(8)`
- `systemd-run(8)` a `systemctl --user`

# Systemd - další vlastnosti

- `systemd-tmpfiles(8)`
- `systemd-oemd(8)`
- `systemd-logind(8)`
- `systemd-coredumpd(8)`
- `systemd-networkd(8)`
- `systemd-resolved(8)`
- `systemd-run(8)` a `systemctl --user`

## 📄 Příklad: Program s plovoucí licencí

```
[Service]
Type=exec
DynamicUser=yes
ProtectSystem=strict
PrivateDevices=yes
ProtectKernelTunables=yes
ProtectKernelModules=yes
# SystemCallFilter=@system-service
NoNewPrivileges=yes
PrivateTmp=yes
LimitCPU=5m
ExecStart=/bin/sh -c 'exec /ten/program \
    < /run/program/to-program \
    > /run/program/from-program 2>&1'
KillMode=process
Restart=always
```

## 📄 Příklad: Služba z více komponent

```
# redis-@.service
[Unit]
Description=Redis cluster member on port %i
After=network.target
PartOf=redis.target
Wants=redis.target
[Service]
ExecStart=/usr/bin/redis-server \
    /etc/redis-%i.conf
ExecStop=/usr/libexec/redis-shutdown \
    /etc/redis-%i.conf
Type=notify
User=redis
Group=redis
[Install]
WantedBy=redis-user.target
```

## Příklad: Služba z více komponent

```
# redis.target
[Unit]
Description=Redis database cluster
After=network.target

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




## Příklad: Časovač

```
# fulltextidx.timer
[Unit]
Description=IS fulltext indexing timer

[Timer]
# OnActiveSec=0
OnUnitInactiveSec=420
RandomizedDelaySec=42

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


# Svazky - I.

- `/etc/fstab`
  - automaticky připojované svazky
  - swapovací oblasti 
  - manuálně připojované svazky

## Příklad: `/etc/fstab`

```
/dev/sda1 /          ext3    defaults    1 1
tmpfs     /dev/shm tmpfs    defaults    0 0
devpts    /dev/pts devpts   gid=5,mode=620 0 0
sysfs     /sys      sysfs    defaults    0 0
proc      /proc     proc     defaults    0 0
/dev/sda2 none      swap     pri=10      0 0
```






# Svazky - I.

- `/etc/fstab`
  - automaticky připojované svazky
  - swapovací oblasti 
  - manuálně připojované svazky


## Příklad: `/etc/fstab`

```
/dev/sda1 /          ext3    defaults    1 1
tmpfs     /dev/shm tmpfs    defaults    0 0
devpts    /dev/pts devpts   gid=5,mode=620 0 0
sysfs     /sys      sysfs    defaults    0 0
proc      /proc    proc     defaults    0 0
/dev/sda2 none      swap     pri=10      0 0
```

# Svazky - II.

- `/etc/mtab` - aktuálně připojené svazky (také v `/proc/mounts` ).
- Program `mount(8)` - připojení svazku.
- Program `umount(8)` - odpojení svazku.
- `Bind-mount` - připojení existujícího adresáře jako svazku .
- `Vícenásobné připojení` téhož svazku .
- `Loop device` - vytvoření blokového zařízení ze souboru. Možnost připojení souboru jako svazku (např. ISO 9660 obraz CD).  .

# Odkládací prostor

- **Disková oblast.** Někdy možnost swapovat do souboru.
- **Vzdálený odkládací prostor** – těžké implementovat (out-of-memory deadlock).
- **Seznam** – obvykle v /etc/fstab
- **Aktivace/deaktivace** – swapon(8), swapoff(8).
- **Další informace** – swap(8), /proc/swaps 



## Čtení na dobrou noc

In defence of swap: common misconceptions

<https://chrisdown.name/2018/01/02/in-defence-of-swap.html>

# Automounter

- Závislosti mezi počítači
  - např. při výpadku napájení
  - vzájemné sdílení svazků: problematické
- Řešení: připojování svazků podle potřeby
- Implementace: virtuální souborový systém
  - Automount point - detekce přístupu k adresáři
  - Mapování podadresářů na souborové systémy
  - Příklad: aisa:/home
  - Nebo přímá náhrada automount pointu
- Implementace:
  - autofs
  - automount
  - amd(8) - user-space

# Automounter



- Závislosti mezi počítači
  - např. při výpadku napájení
  - vzájemné sdílení svazků: problematické
- Řešení: připojování svazků podle potřeby
- Implementace: virtuální souborový systém
  - Automount point - detekce přístupu k adresáři
  - Mapování podadresářů na souborové systémy
  - Příklad: aisa:/home
  - Nebo přímá náhrada automount pointu
- Implementace:
  - autofs
  - automount
  - amd(8) - user-space

# Automounter

- Závislosti mezi počítači
  - např. při výpadku napájení
  - vzájemné sdílení svazků: problematické
- Řešení: připojování svazků podle potřeby
- Implementace: virtuální souborový systém
  - Automount point – detekce přístupu k adresáři
  - Mapování podadresářů na souborové systémy
  - Příklad: aisa:/home
  - Nebo přímá náhrada automount pointu
- Implementace:
  - autofs
  - automount
  - amd(8) – user-space



# Automounter

- Závislosti mezi počítači
  - např. při výpadku napájení
  - vzájemné sdílení svazků: problematické
- Řešení: připojování svazků podle potřeby
- Implementace: virtuální souborový systém
  - **Automount point** – detekce přístupu k adresáři
  - Mapování podadresářů na souborové systémy
  - Příklad: `aisa:/home`
  - Nebo přímá náhrada automount pointu
- Implementace:
  - autofs 
  - automount 
  - amd(8) – user-space



# Příklad: Konfigurace autofs



```
/etc/auto.master:
```

```
/ftp    /etc/auto.ftp  
/home   /etc/auto.home
```

```
/etc/auto.ftp:
```

```
pub ftp.fi.muni.cz:/export/ftp/ftp/pub
```

```
/etc/auto.home:
```

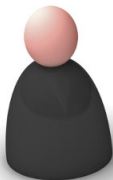
```
*      home.fi.muni.cz:/export/home/&
```

# Kapitola 5

## Uživatelé a skupiny

# Uživatelé a skupiny

- **UID/GID** – identifikace uživatele/skupiny z hlediska jádra systému.
- **Jméno uživatele** – používá se při přihlašování a u logování do souborů.
- **Základní databáze** – /etc/passwd, /etc/group.



# Soubor /etc/passwd

## Příklad: Řádek z /etc/passwd

```
ayanami:x:1999:2014:Rei Ayanami:/home/ayanami\  
:/bin/bash
```

- **Jméno uživatele** – klíč v /etc/passwd.
- **Heslo** – v zašifrované podobě.
- **UID** – ne nutně jedinečné.
- **GID** – reálné/efektivní GID, které mají procesy po přihlášení.
- **GCOS** – komentář, obvykle celé jméno uživatele.
- **Domovský adresář** – pracovní adresář shellu po přihlášení.
- **Shell** – je spuštěn po přihlášení.

# Pole GCOS v /etc/passwd

- Některé systémy – **strukturované pole GCOS**.
- Několik záznamů oddělených čárkou (místnost, tel. číslo, atd).
- Využívá např. `finger(1)`.
- GCOS - **General Electric Comprehensive Operating System**



## Dennis Ritchie píše:

*„Sometimes we sent printer output or batch jobs to the GCOS machine. The GCOS field in the password file was a place to stash the information for the \$IDENTcard. Not elegant.“*

# Šifrování hesel

- **Standardně** - 25-krokový DES, 2 znaky sůl, zbytek heslo.
- **Knihovní funkce** - crypt(3).
- **Jiné metody** - MD5, SHA-1, SHA-512, ... (+ sůl)

## Příklad: Standardní šifrování hesla

```
$ perl -e 'print crypt("jezek", "42"), "\n"'  
42uresi6Z/E/w
```

## Příklad: SHA-512 heslo

```
$6$pQ50iSwS$0WAzrqjLC4rkfafAsPohh/6HvmieN6\  
jcYxEiAotx84wpaG1Wrgvj/CJbGfRXGzlg48zErbYE\  
DIWvFzDi7UxxZ/
```

# Šifrování hesel

- Standardně - 25-krokový DES, 2 znaky sůl, zbytek heslo.
- Knihovní funkce - crypt(3).
- Jiné metody - MD5, SHA-1, SHA-512, ... (+ sůl)



## Příklad: Standardní šifrování hesla

```
$ perl -e 'print crypt("jezek", "42"), "\n"'  
42uresi6Z/E/w
```

## Příklad: SHA-512 heslo

```
$6$pQ50iSwS$0WAzrqjLC4rkfafAsPohh/6HvmieN6\  
jcYxEiAotx84wpaG1Wrgvj/CJbGfRXGzlg48zErbYE\  
DIWvFzDi7UxxZ/
```



# Šifrování hesel

- **Standardně** - 25-krokový DES, 2 znaky sůl, zbytek heslo.
- **Knihovní funkce** - crypt(3).
- **Jiné metody** - MD5, SHA-1, SHA-512, ... (+ sůl)

## **Příklad: Standardní šifrování hesla**

```
$ perl -e 'print crypt("jezek", "42"), "\n"'  
42uresi6Z/E/w
```

## **Příklad: SHA-512 heslo**


```
$6$pQ50iSwS$0WAzrqjLC4rkfafAsPohh/6HvmieN6\  
jcYxEiAotx84wpaG1Wrgvj/CJbGfRXGzlg48zErbYE\  
DIWvFzDi7UxxZ/
```

# Ukládání hesel

- **Standardní UNIX** – hesla jsou vystavena útoku hrubou silou (John the Ripper) a slovníkovému útoku (crack(8)).
- **Shadow passwords** – hesla a další údaje jsou uloženy v souboru /etc/shadow. Omezení hesla na určitý čas, omezení frekvence změny hesla. Nutnost set-uid/gid u programů, pracujících s hesly.

## **Příklad: Záznam v /etc/shadow**

```
ayanami:$6$p...xZ/:14224:0:99999:7:::
```

- **BSD** – /etc/master.passwd – analogie shadow 
- **Trusted control base**: umožňuje zakázat recyklaci hesla, změnu hesla, volbu vlastního hesla, atd.

# Formát souboru /etc/group

## Příklad: Záznam v /etc/group

```
nerv:x:2014:ayanami,asuka,shinji
```

- **Jméno skupiny** - identifikace pro logování do souboru a pro přepínání GID pomocí `newgrp(1)`.
- **Heslo skupiny** - obvykle nepoužito. Případně i v `/etc/gshadow`. Skupiny bez hesla přidány při přihlášení.
- **Číslo skupiny** - identifikace pro systém.
- **Seznam uživatelů** - jména oddělená čárkami. Primární skupina je implicitně, uživatel zde nemusí být uveden.

# Modifikace tabulky uživatelů

- Speciální programy - vipw(8) - je-li databáze uživatelů uložena i jinde (shadow, master.passwd).
- Změna uživatelských informací - chfn(8).
- Dávkové přidávání - useradd, groupadd, userdel, groupdel - vytváří domovský adresář, alokuje volné UID, kopíruje soubory z /etc/skel.
- pwconv(8) - převod hesel do shadow.

## Otázka:

K čemu je dobré mít i skupinu pro každého uživatele?

# Modifikace tabulky uživatelů

- Speciální programy - vipw(8) - je-li databáze uživatelů uložena i jinde (shadow, master.passwd).
- Změna uživatelských informací - chfn(8).
- Dávkové přidávání - useradd, groupadd, userdel, groupdel - vytváří domovský adresář, alokuje volné UID, kopíruje soubory z /etc/skel.
- pwconv(8) - převod hesel do shadow.



## Otázka:

K čemu je dobré mít i skupinu pro každého uživatele?

# Uživatelé, skupiny a systemd(8)

## ■ Přidělování čísel:

<https://systemd.io/UIDS-GIDS/>

■ Deklarativní vytváření: `/usr/lib/sysusers.d`

■ `systemd-sysusers(8)`

### Příklad: `sysusers.d/basic.conf`

```
u root      0           "Super User"  /root
g adm       4           -             -
g nobody    65534          -             -
u nobody    65534:65534   "Nobody"     -
[... ]
g users     100          -             -
```

# Uživatelé, skupiny a systemd(8)

- Přidělování čísel:  
`https://systemd.io/UIDS-GIDS/`
- Deklarativní vytváření: `/usr/lib/sysusers.d`
- `systemd-sysusers(8)`

## Příklad: `sysusers.d/basic.conf`

```
u root 0 "Super User" /root
g adm 4 - -
g nobody 65534 - -
u nobody 65534:65534 "Nobody" -
[... ]
g users 100 - -
```

# Uživatelé, skupiny a systemd(8)

- Přidělování čísel:  
`https://systemd.io/UIDS-GIDS/`
- Deklarativní vytváření: `/usr/lib/sysusers.d`
- **systemd-sysusers(8)**

## Příklad: `sysusers.d/basic.conf`

```
u root 0 "Super User" /root
g adm 4 - -
g nobody 65534 - -
u nobody 65534:65534 "Nobody" -
[... ]
g users 100 - -
```



# Uživatelé, skupiny a systemd(8)

- Přidělování čísel:  
`https://systemd.io/UIDS-GIDS/`
- Deklarativní vytváření: `/usr/lib/sysusers.d`
- `systemd-sysusers(8)`

## Příklad: `sysusers.d/basic.conf`

```
u root 0 "Super User" /root
g adm 4 - -
g nobody 65534 - -
u nobody 65534:65534 "Nobody" -
[... ]
g users 100 - -
```

# Soubor /etc/shells

## ■ Příklad: Soubor /etc/shells

```
/bin/sh  
/bin/bash  
/bin/tcsh
```

- Změna shellu pomocí `chsh(1)`.
- Některé služby jen pro uživatele s platným shellem.
- `/sbin/nologin` - shell pro pseudouživatele.



# Name Service Switch

- Alternativní zdroje dat pro systémové tabulky (passwd, group, hosts, ...).
- Implementace - plug-iny do libc.
  - /lib/libnss\_*služba*.so.X



## Příklad: Soubor /etc/nsswitch.conf

```
passwd:    files sss systemd
group:     files sss systemd
hosts:     myhostname \
           mdns4_minimal [NOTFOUND=return] \
           dns [!UNAVAIL=return] files
networks: nis [NOTFOUND=return] files
```

# NSSwitch - konfigurace

- Formát souboru - databáze, mezera, popis služeb.
- Podrobnější specifikace - `[ [!]STATUS=AKCE ...]`

## Akce:

**RETURN** - vrácení právě nalezené hodnoty nebo chyby.

**CONTINUE** - pokračování použitím další služby.

# NSSwitch - návratové stavy

- SUCCESS** - záznam nalezen, nedošlo k chybě. Implicitní akce je RETURN.
- NOTFOUND** - vyhledávání proběhlo bez chyby, ale záznam se nenašel. Implicitní akce je CONTINUE.
- UNAVAIL** - služba není trvale dostupná (např. nezkonfigurovaná). Implicitně CONTINUE.
- TRYAGAIN** - dočasná chyba (timeout, vyčerpání prostředků, atd.). Implicitně CONTINUE.

# Uživatelé a skupiny - programování

## getpwnam(3), getpwuid(3) Databáze uživatelů

```
#include <pwd.h>
#include <sys/types.h>
struct passwd *getpwnam(const char *name);
struct passwd *getpwuid(uid_t uid);

struct passwd {
    char *pw_name;
    char *pw_passwd;
    uid_t pw_uid;
    gid_t pw_gid;
    char *pw_gecos;
    char *pw_dir;
    char *pw_shell;
};
```

# Poznámky k `getpw*` ( )

- Vrací ukazatel na strukturu, popisující záznam daného uživatele.
- Pozor – funkce nejsou reentrantní.
- Seznam všech uživatelů:
  - `getpwent(3)`, `setpwent(3)`, `endpwent(3)`
- NSSwitch a příkazová řádka:
  - `getent(1)`
  - `$ getent passwd ayanami`

# Poznámky k `getpw*` ( )

- Vrací ukazatel na strukturu, popisující záznam daného uživatele.
- **Pozor – funkce nejsou reentrantní.**
- Seznam všech uživatelů:
  - `getpwent(3)`, `setpwent(3)`, `endpwent(3)`
- NSSwitch a příkazová řádka:
  - `getent(1)`
  - `$ getent passwd ayanami`



# Poznámky k `getpw*` ( )

- Vrací ukazatel na strukturu, popisující záznam daného uživatele.
- Pozor – funkce nejsou reentrantní.
- Seznam všech uživatelů:
  - `getpwent(3)`, `setpwent(3)`, `endpwent(3)`
- NSSwitch a příkazová řádka:
  - `getent(1)`
  - `$ getent passwd ayanami`

# Poznámky k `getpw*()`

- Vrací ukazatel na strukturu, popisující záznam daného uživatele.
- Pozor – funkce nejsou reentrantní.
- Seznam všech uživatelů:
  - `getpwent(3)`, `setpwent(3)`, `endpwent(3)`
- NSSwitch a příkazová řádka:
  - `getent(1)`
  - `$ getent passwd ayanami`

# Databáze skupin

## getgrnam(3), getgrgid(3)

```
#include <grp.h>
#include <sys/types.h>
struct group *getgrnam(const char *name);
struct group *getgrgid(gid_t gid);

struct group {
    char *gr_name;
    char *gr_passwd;
    gid_t gr_gid;
    char **gr_mem;
};
```

Seznam všech skupin – getgrent(3), setgrent(3), endgrent(3).

# Pluggable Authentication Modules

- **PAM** – Sun Microsystems, nyní GPL nebo BSD. Hlavní vývoj nyní Red Hat.
- **Téměř všechny UNIXy** – distribuce Linuxu, Solaris, HP-UX. IRIX a některé BSD nikoliv. Různé stupně vývoje.
- **Modulární přístup k autentizaci** – čipové karty, hesla, biometriky, síťové databáze (Kerberos, LDAP, NIS), atd.
- **Architektura** – knihovna `libpam`, plug-iny v `/lib/security`, konfigurace v `/etc/pam.conf` a `/etc/pam.d/*`.

# PAM - fáze autentizace

**account** - jestli vůbec člověk má účet, nevyexpirované heslo, může k dané službě přistupovat?

**auth** - vlastní autentizace - ověření identity žadatele (heslo, jednorázové heslo, biometriky, čipové karty + PIN, atd.).

**password** - změny autentizačních mechanismů (změna hesla apod.).

**session** - akce před zpřístupněním služby a po ukončení (audit, připojení domovského adresáře, nastavení uživatelských limitů, atd).

## 📄 Příklad: PAM - formát konfigurace

```
auth      required      pam_securetty.so
auth      required      pam_env.so
auth      required      pam_nologin.so
auth      sufficient   pam_unix.so nullok
auth      required      pam_deny.so
account   required      pam_unix.so
password  required      pam_cracklib.so retry=3
password  sufficient   pam_unix.so nullok shadow
password  required      pam_deny.so
session   required      pam_limits.so
session   required      pam_unix.so
session   optional     pam_console.so
```

# PAM - řídicí hodnoty

**required** - pokud selže, selže i celý autentizační proces.

**requisite** - totéž, ale skončí se hned.

**sufficient** - stačí k autentizaci bez ohledu na výsledek následujících modulů.

**optional** - spustí se, ale výsledek se použije pouze pokud jde o jediný modul daného typu.

# Soubor utmp

- `/var/run/utmp`
- Seznam právě přihlášených uživatelů.
- Pole struktur utmp.
- POSIX.1 rozhraní utmpx.
- **Programy** - `who(1)`, `w(1)`.



# Rozhraní utmpx

## getutxent(3)

## Práce s utmpx

```
#include <utmpx.h>
```

```
struct utmpx *getutxent(void);  
struct utmpx *getutxid(struct utmpx *);  
struct utmpx *getutxline(struct utmpx *);  
struct utmpx *pututxline(struct utmpx *);  
void setutxent(void);  
void endutxent(void);
```

# Soubor wtmp

- `/var/log/wtmp`
- Záznam o přihlášeních a odhlášeních uživatelů.
- Stejný formát jako u `utmp`, uživatel `NULL` značí odhlášení na daném terminálu.
- **Speciální záznamy** – start a ukončení systému, změna úrovně běhu systému. Změna systémového data.
- **Neexistující wtmp** – zákaz vedení záznamů. Při rotování `wtmp` nutno vždy vytvořit nový soubor.
- **Soubor btmp** – záznamy o chybných přihlášeních.
- **Programy** – `last(8)`, `lastb(8)`.

# Terminálové procesy

`getty` - inicializace linky, výpis zprávy, čekání na vstup.

`login` - načtení hesla, zápis do wtmp a utmp.

`shell` - uživatelský program.



oldcomputr.com

# Kapitola 6

## Správa zařízení

# Zařízení v UNIXu

```
$ ls -l /dev
brw-r----- 1 root disk 8, 16\
             Oct 15 20:56 /dev/sdb
...
crw-rw---- 1 kas  root 5, 1 \
             Oct 15 20:58 /dev/console
```

- **zařízení** – speciální soubor v /dev
- typ: bloková, znaková
- hlavní číslo – číslo ovladače v jádře
- vedlejší číslo – interní ID pro ovladač

# Zařízení v UNIXu

```
$ ls -l /dev  
brw-r----- 1 root disk 8, 16\  
    Oct 15 20:56 /dev/sdb  
...  
crw-rw---- 1 kas  root 5, 1 \  
    Oct 15 20:58 /dev/console
```

- **zařízení** – speciální soubor v /dev
- **typ**: bloková, znaková
- **hlavní číslo** – číslo ovladače v jádře
- **vedlejší číslo** – interní ID pro ovladač

# Zařízení v UNIXu

```
$ ls -l /dev
brw-r----- 1 root disk 8, 16\
    Oct 15 20:56 /dev/sdb
...
crw-rw---- 1 kas  root 5, 1 \
    Oct 15 20:58 /dev/console
```

- **zařízení** – speciální soubor v /dev
- **typ**: bloková, znaková
- **hlavní číslo** – číslo ovladače v jádře
- **vedlejší číslo** – interní ID pro ovladač

# Zařízení v UNIXu

```
$ ls -l /dev
brw-r----- 1 root disk 8, 16\
    Oct 15 20:56 /dev/sdb
...
crw-rw---- 1 kas  root 5, 1 \
    Oct 15 20:58 /dev/console
```

- **zařízení** – speciální soubor v /dev
- **typ**: bloková, znaková
- **hlavní číslo** – číslo ovladače v jádře
- **vedlejší číslo** – interní ID pro ovladač



# Zařízení v C

- Datový typ `dev_t`:
  - `dev_t` `makedev(unsigned maj, unsigned min);`
  - `unsigned` `major(dev_t dev);`
  - `unsigned` `minor(dev_t dev);`
- `dev_t` musí být numerický typ!
- Použití: `stat(2)`, `mknod(2)`, ...

# Zařízení v C

- Datový typ `dev_t`:
  - `dev_t` `makedev(unsigned maj, unsigned min);`
  - `unsigned major(dev_t dev);`
  - `unsigned minor(dev_t dev);`
- `dev_t` musí být numerický typ!
- Použití: `stat(2)`, `mknod(2)`, ...

# Zařízení v C

- Datový typ `dev_t`:
  - `dev_t` `makedev(unsigned maj, unsigned min);`
  - `unsigned major(dev_t dev);`
  - `unsigned minor(dev_t dev);`
- `dev_t` musí být numerický typ!
- Použití: `stat(2)`, `mknod(2)`, ...

# Problémy

Kolik bitů na vedlejší číslo? SCSI: kanál, target, LUN, partition.

Příliš velké /dev:

```
$ ls /dev | wc -l  
1431
```

Dynamicky vznikající zařízení

A v neposlední řadě ...

# Problémy

Kolik bitů na vedlejší číslo? SCSI: kanál, target, LUN, partition.

Příliš velké /dev:

```
$ ls /dev | wc -l  
1431
```

Dynamicky vznikající zařízení

A v neposlední řadě ...

# Problémy

Kolik bitů na vedlejší číslo? SCSI: kanál, target, LUN, partition.

Příliš velké /dev:

```
$ ls /dev | wc -l  
1431
```

Dynamicky vznikající zařízení



A v neposlední řadě ...

# Problémy

Kolik bitů na vedlejší číslo? SCSI: kanál, target, LUN, partition.

Příliš velké /dev:

```
$ ls /dev | wc -l  
1431
```

Dynamicky vznikající zařízení

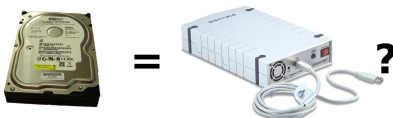


A v neposlední řadě ...

# ... pojmenování zařízení:

Podle topologie (`/dev/dsk/c0t3d1s8`) 

- přesun disku na jiný řadič
- přesun disku na jiný ovladač



Podle ovladače a pořadí (`/dev/sda1`)

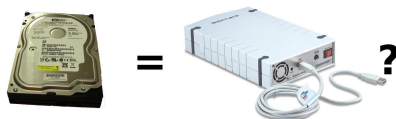
- výpadek disku: ostatní se přejmenují



# ... pojmenování zařízení:

Podle topologie (`/dev/dsk/c0t3d1s8`) 

- přesun disku na jiný řadič
- přesun disku na jiný ovladač



Podle ovladače a pořadí (`/dev/sda1`) 

- výpadek disku: ostatní se přejmenují

# Jak pojmenovávat?

Podle topologie: eth0 je ta **v tomto PCI slotu**.

Podle výrobce: tento fotoaparát vždy jako /dev/fotak

Podle pořadí: nějaká myš jako /dev/mouse0.

Podle výrobního čísla: pouze můj mobil jako  
/dev/mobil.

Nebo úplně jinak: label filesystemu, UUID, ...

# Jak pojmenovávat?

Podle topologie: eth0 je ta v tomto PCI slotu.

Podle výrobce: **tento fotoaparát** vždy jako /dev/fotak

Podle pořadí: nějaká myš jako /dev/mouse0.

Podle výrobního čísla: pouze můj mobil jako  
/dev/mobil.

Nebo úplně jinak: label filesystemu, UUID, ...



# Jak pojmenovávat?

Podle topologie: eth0 je ta v tomto PCI slotu.

Podle výrobce: tento fotoaparát vždy jako /dev/fotak

Podle pořadí: nějaká myš jako /dev/mouse0.

Podle výrobního čísla: pouze můj mobil jako  
/dev/mobil.

Nebo úplně jinak: label filesystemu, UUID, ...



# Jak pojmenovávat?

Podle topologie: `eth0` je ta v tomto PCI slotu.

Podle výrobce: tento fotoaparát vždy jako `/dev/fotak`

Podle pořadí: nějaká myš jako `/dev/mouse0`.

Podle výrobního čísla: pouze **můj mobil** jako  
`/dev/mobil`.

Nebo úplně jinak: label filesystemu, UUID, ...



# Jak pojmenovávat?

Podle topologie: `eth0` je ta v tomto PCI slotu.

Podle výrobce: tento fotoaparát vždy jako `/dev/fotak`

Podle pořadí: nějaká myš jako `/dev/mouse0`.

Podle výrobního čísla: pouze můj mobil jako  
`/dev/mobil`.

Nebo úplně jinak: **label filesystemu, UUID, ...**



# DevFS

- virtuální souborový systém
- idea ze Solarisu
- ovladače samy registrují soubory
- „nějaká“ výchozí přístupová práva
- pojmenování: jako na Solarisu, symlinky pro kompatibilitu

## Problémy DevFS

- politika uvnitř jádra
- není perzistentní nastavení
- race conditions

# DevFS

- virtuální souborový systém
- idea ze Solarisu
- ovladače samy registrují soubory
- „nějaká“ výchozí přístupová práva
- pojmenování: jako na Solarisu, symlinky pro kompatibilitu



## Problémy DevFS

- politika uvnitř jádra
- není perzistentní nastavení
- race conditions





- virtuální souborový systém
- obvykle jako /sys
- obraz subsystému ovladačů v jádře
- adresáře podle topologie, tříd zařízení, ovladačů, ...
- inventář hardwaru
- dynamická alokace hlavních čísel



# Příklad: SysFS



```
# echo 0 0 0 > /sys/class/scsi_host/host0/scan
```

```
# echo 1 > /sys/block/sdb/device/delete
```

```
# echo 45 > /sys/devices/platform/\  
w83627hf/temp_max
```

```
# cat /sys/class/input/input0/name  
Power Button (FF)
```

```
# cat /sys/bus/pci/devices/0000 00 1b.0/vendor  
0x8086
```



# Příklad: SysFS



```
# echo 0 0 0 > /sys/class/scsi_host/host0/scan
```

```
# echo 1 > /sys/block/sdb/device/delete
```

```
# echo 45 > /sys/devices/platform/\  
w83627hf/temp_max
```

```
# cat /sys/class/input/input0/name  
Power Button (FF)
```

```
# cat /sys/bus/pci/devices/0000 00 1b.0/vendor  
0x8086
```



# Příklad: SysFS



```
# echo 0 0 0 > /sys/class/scsi_host/host0/scan

# echo 1 > /sys/block/sdb/device/delete

# echo 45 > /sys/devices/platform/\
w83627hf/temp_max

# cat /sys/class/input/input0/name
Power Button (FF)

# cat /sys/bus/pci/devices/0000:00:1b.0/vendor
0x8086
```



# Příklad: SysFS



```
# echo 0 0 0 > /sys/class/scsi_host/host0/scan

# echo 1 > /sys/block/sdb/device/delete

# echo 45 > /sys/devices/platform/\
w83627hf/temp_max

# cat /sys/class/input/input0/name
Power Button (FF)

# cat /sys/bus/pci/devices/0000:00:1b.0/vendor
0x8086
```



# Příklad: SysFS



```
# echo 0 0 0 > /sys/class/scsi_host/host0/scan

# echo 1 > /sys/block/sdb/device/delete



# echo 45 > /sys/devices/platform/\
w83627hf/temp_max

# cat /sys/class/input/input0/name
Power Button (FF)

# cat /sys/bus/pci/devices/0000 00 1b.0/vendor
0x8086
```

# Hotplug



- Reakce na události ovladačů, sběrnic, ...
- /sbin/hotplug - notifikace spuštěním programu 
- AF\_NETLINK - notifikační socket 
- *Coldplug* - inventarizace po startu



- Správa /dev v uživatelském prostoru
- Politika mimo jádro
- Využívá SysFS a hotplug notifikaci
- /dev na disku nebo na ramdisku
- démon udevd(8)

## Konfigurace udev

- adresář /etc/udev
- pravidla v /etc/udev/rules.d
- příkaz udevadm(8)
  - udevadm trigger
  - udevadm info





- Správa /dev v uživatelském prostoru
- Politika mimo jádro
- Využívá SysFS a hotplug notifikaci
- /dev na disku nebo na ramdisku
- démon udevd(8)

## Konfigurace udev

- adresář /etc/udev
- pravidla v /etc/udev/rules.d
- příkaz udevadm(8)
  - udevadm trigger
  - udevadm info

# 📄 Příklad: Teploměr



```
KERNEL=="ttyUSB*", \  
  ATTRS{product}=="Papouch TMU Thermometer", \  
  ATTRS{serial}=="PPQ3NTMG", \  
  SYMLINK+="tmu0"
```

## 📄 Příklad: Spuštění programu



```
KERNEL=="sd*1", \
  SYSFS{model}=="G3", \
  SYSFS{vendor}=="M-System", \
  RUN+="/usr/local/sbin/podcast-to-player"
```

# Psaní vlastních pravidel

## Příklad: Výpis atributů

```
$ udevadm info -q path -n /dev/ttyACM0  
/devices/pci0000:00/0000:00:13.2/usb2/2-3/  
  /2-3.2/2-3.2:1.0/tty/ttyACM0  
$ udevadm info -q all -a -n /dev/sda
```

# udev a disky

Podle výrobního čísla

```
/dev/disk/by-id/scsi-SATA_HDS724040KLAT80_
```

Podle topologie

```
/dev/disk/by-path/pci-0000:00:0f.0-scsi-0_
```

Podle UUID filesystemu

```
/dev/disk/by-uuid/1ffe43cc-5ca6-45d5-80df-
```

# udev a disky

Podle výrobního čísla

```
/dev/disk/by-id/scsi-SATA_HDS724040KLAT80_
```

Podle topologie

```
/dev/disk/by-path/pci-0000:00:0f.0-scsi-0_
```

Podle UUID filesystemu

```
/dev/disk/by-uuid/1ffe43cc-5ca6-45d5-80df-
```

# udev a disky

Podle výrobního čísla

```
/dev/disk/by-id/scsi-SATA_HDS724040KLAT80_
```

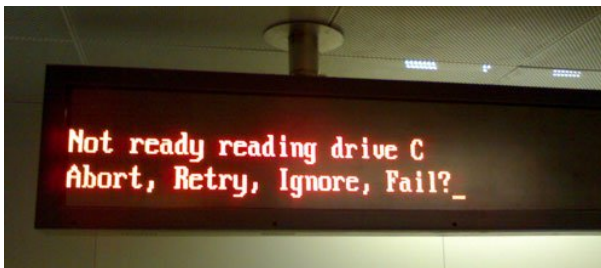
Podle topologie

```
/dev/disk/by-path/pci-0000:00:0f.0-scsi-0_
```

Podle UUID filesystemu

```
/dev/disk/by-uuid/1ffe43cc-5ca6-45d5-80df-
```

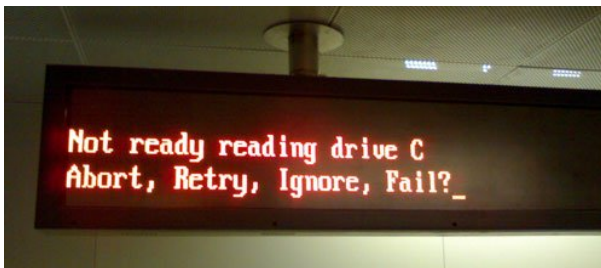
# Jádro versus aplikace



- Modularita UNIXu - jádro je daleko od aplikací
- Není jak se dovědět: vznik zařízení, plný disk, aktivace síťové karty, ...
- I na desktopu: příchod VoIP volání, full-screen aplikace, ...

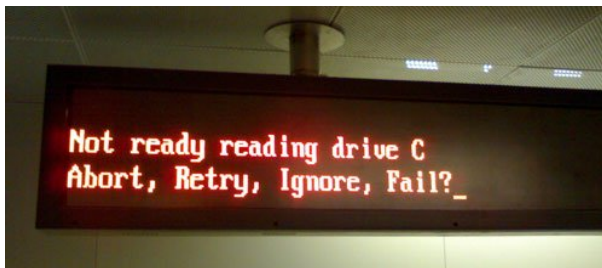


# Jádro versus aplikace



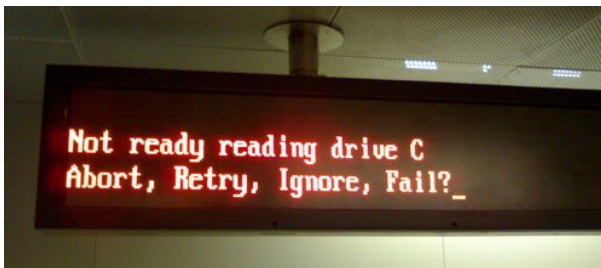
- Modularita UNIXu – jádro je daleko od aplikací
- Není jak se dovědět: vznik zařízení, plný disk, aktivace síťové karty, ...
- I na desktopu: příchod VoIP volání, full-screen aplikace, ...

# Jádro versus aplikace



- Modularita UNIXu – jádro je daleko od aplikací
- Není jak se dovědět: vznik zařízení, plný disk, aktivace síťové karty, ...
- I na desktopu: příchod VoIP volání, full-screen aplikace, ...

# Jádro versus aplikace



- Modularita UNIXu – jádro je daleko od aplikací
- Není jak se dovědět: vznik zařízení, plný disk, aktivace síťové karty, ...
- I na desktopu: příchod VoIP volání, full-screen aplikace, ...

# D-Bus

- Desktop Bus
- Zasílání zpráv
- Broadcast – subscribe
- System bus, session bus
- Vzdálené volání objektů
- `dbus-monitor(8)`

# Kapitola 7

# Logování

# Syslog

- Démon `syslogd(8)`.
- Zpracovává hlášení o událostech.
- Socket `/dev/log` (AF\_UNIX, SOCK\_DGRAM).
- Typ zprávy (facility): kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, authpriv, local0 - local7.
- Priorita zprávy (priority): emerg, alert, crit, err, warning, notice, info, debug.

## Příklad: Zprávy v syslogu

```
Mar 11 20:16:42 yurika ntpd[1314]: \
synchronized to 147.251.48.140, stratum 2
Mar 11 20:20:55 yurika ntpd[1314]: \
time reset -2.348625 s
```

# Syslog

- Démon `syslogd(8)`.
- Zpracovává hlášení o událostech.
- Socket `/dev/log` (AF\_UNIX, SOCK\_DGRAM).
- Typ zprávy (facility): kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, authpriv, local0 - local7.
- Priorita zprávy (priority): emerg, alert, crit, err, warning, notice, info, debug.

## Příklad: Zprávy v syslogu

```
Mar 11 20:16:42 yurika ntpd[1314]: \  
synchronized to 147.251.48.140, stratum 2  
Mar 11 20:20:55 yurika ntpd[1314]: \  
time reset -2.348625 s
```

# Syslog z programu v C - I.

## openlog(3) Otevření systémového logu

```
#include <syslog.h>
void openlog(char *id, int opt, int facility);
```

- Řetězec id je připojen před každou zprávou.
- Parametr opt - logický součet z následujícího:

**LOG\_CONS** - pokud se nepodaří odeslat zprávu, píše přímo na systémovou konzolu.

**LOG\_NDELAY** - otevřít spojení ihned (jinak až při první zprávě).

**LOG\_PERROR** - psát také na stderr.

**LOG\_PID** - do zprávy zahrnout PID procesu.



# Syslog z programu v C - II.

## closelog(3)

## Uzavření logu

```
#include <syslog.h>
void closelog();
```

Ukončí zasílání zpráv (uvolní deskriptor).

## syslog(3)

## Zápis zprávy do logu

```
#include <syslog.h>
void syslog(int priority, char *fmt, ...);
```

Řetězec fmt má podobný význam jako v printf(3).

# Syslog z programu v C - II.

## closelog(3)

## Uzavření logu

```
#include <syslog.h>
void closelog();
```

Ukončí zasílání zpráv (uvolní deskriptor).

## syslog(3)

## Zápis zprávy do logu

```
#include <syslog.h>
void syslog(int priority, char *fmt, ...);
```

Řetězec fmt má podobný význam jako v printf(3).

# Syslog z příkazové řádky

## logger(1)

## Zápis do syslogu

```
$ logger [-is] [-p pri] [message ...]
```

### Příklad: logger(1)

```
$ logger -p lpr.notice -i Printer lp0 on fire!
```

# Konfigurace syslogu

- Konfigurace - /etc/syslog.conf.
- Syntaxe: *typ/priorita zprávy tabulátor soubor.*

Poslední položka se rozlišuje podle prvního znaku:

/ - běžný soubor

- - totéž, nevolá se fsync(2).

| - logování rourou do programu.

\* - všem nalogovaným uživatelům.

@ - logování po síti na jiný stroj.

*ostatní* - seznam uživatelů.

## Příklad: /etc/syslog.conf

```
kern.* /dev/console
*.info;mail.none;authpriv.none \
/var/log/messages
authpriv.* /var/log/secure
mail.* /var/log/maillog
*.emerg *
uucp,news.crit /var/log/spooler
#*.debug -/var/log/debug
local0.info /var/log/ppp
local2.=info |/usr/bin/log-parser
*.notice @loghost.domena.cz
```

# Syslog - alternativy

- Společné vlastnosti – stejné rozhraní pro C, stejný síťový protokol.

## rsyslog

- Modulární
- Vstupní moduly – socket, kernel, UDP, TCP, ...
- Výstupní moduly – např. MySQL.
- Filtrovací pravidla
- Zpětně kompatibilní `syslog.conf` (5).


## syslog-ng

- Podepisované zprávy, časová razítka.
- Šifrovaný přenos zpráv (TLS).
- Filtrovací a třídící pravidla.

# Syslog - alternativy

- Společné vlastnosti – stejné rozhraní pro C, stejný síťový protokol.

## rsyslog

- Modulární
- Vstupní moduly – socket, kernel , TCP, ...
- Výstupní moduly – např. MySQL.
- Filtrovací pravidla
- Zpětně kompatibilní `syslog.conf(5)`.


## syslog-ng

- Podepisované zprávy, časová razítka.
- Šifrovaný přenos zpráv (TLS).
- Filtrovací a třídící pravidla.

# Syslog - alternativy

- Společné vlastnosti – stejné rozhraní pro C, stejný síťový protokol.

## rsyslog

- Modulární
- Vstupní moduly – socket, kernel , TCP, ...
- Výstupní moduly – např. MySQL.
- Filtrovací pravidla
- Zpětně kompatibilní `syslog.conf(5)`.

## syslog-ng

- Podepisované zprávy, časová razítka.
- Šifrovaný přenos zpráv (TLS).
- Filtrovací a třídící pravidla.





- Logovací subsystém v systemd.
- Indexované ukládání zpráv.
- Rozšířená množina metadat.
- Důvěryhodné atributy.
- Ochrana před manipulováním logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.

Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>



- Logovací subsystém v systemd.
- **Indexované** ukládání zpráv.
- Rozšířená množina metadat.
- Důvěryhodné atributy.
- Ochrana před manipulováním logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.

Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>



- Logovací subsystém v systemd.
- Indexované ukládání zpráv.
- Rozšířená množina **metadat**.
- Důvěryhodné atributy.
- Ochrana před manipulováním logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.

Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>



- Logovací subsystém v systemd.
- Indexované ukládání zpráv.
- Rozšířená množina metadat.
- **Důvěryhodné atributy.**
- Ochrana před manipulováním logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.

Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>



- Logovací subsystém v systemd.
- Indexované ukládání zpráv.
- Rozšířená množina metadat.
- Důvěryhodné atributy.
- Ochrana před **manipulováním** logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.

Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>



- Logovací subsystém v systemd.
- Indexované ukládání zpráv.
- Rozšířená množina metadat.
- Důvěryhodné atributy.
- Ochrana před manipulováním logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.

Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>



- Logovací subsystém v systemd.
- Indexované ukládání zpráv.
- Rozšířená množina metadat.
- Důvěryhodné atributy.
- Ochrana před manipulováním logu.
- Viz též `journalctl(8)`, `sd_journal_print(3)`.



## Čtení na dobrou noc

<http://0pointer.de/blog/projects/the-journal.html>

## 📄 Příklad: Journal - atributy zprávy

```
_SERVICE=systemd-logind.service  
MESSAGE=User ayanami logged in  
MESSAGE_ID=422bc3d271414bc8bc9570f222f24a9  
_EXE=/lib/systemd/systemd-logind  
_COMM=systemd-logind  
_CMDLINE=/lib/systemd/systemd-logind  
_PID=1999 _UID=0 _GID=0  
_SYSTEMD_CGROUP=/system/systemd-logind.service  
_CGROUPS=cpu:/system/systemd-logind.service  
PRIORITY=6  
_BOOT_ID=422bc3d271414bc8bc95870f222f24a9  
_MACHINE_ID=c686f3b205dd48e0b43ceb6eda479721  
_HOSTNAME=eva00.nerv.gov.jp  
LOGIN_USER=2014
```



# Odstraňování chyb

*[...] phenomena like **heisenbugs** (errors that vanish when you try to debug them), **schrödingbugs** (errors that manifest only when you're trying to debug something else), and **mandelbugs** (complex errors that seem to fluctuate more and more chaotically, the closer you look at them).*

*[...] If there's anything less enjoyable than beating your head against a bug for several hours, it's finally discovering that your debugging print statement was itself buggy, and the problem isn't anywhere near where you thought it was. This is presumably a **Homerbug**.*

-Damian Conway: Perl Best Practices

# Odstraňování chyb

- Modularita UNIXu – zjednodušuje určení místa výskytu problému.
- Kde chyba vzniká? – najít konkrétní program/knihovnu.
- Čas přístupu k souboru (ale: pozor na noatime/relatime).

- Spouští se vůbec tento program?
- Načítá se vůbec tento konfigurační soubor?

# Odstraňování chyb

- Modularita UNIXu – zjednodušuje určení místa výskytu problému.
- Kde chyba vzniká? – najít konkrétní program/knihovnu.
- Čas přístupu k souboru (ale: pozor na noatime/relatime).

- Spouští se vůbec tento program?
- Načítá se vůbec tento konfigurační soubor?

# Odstraňování chyb

- Modularita UNIXu – zjednodušuje určení místa výskytu problému.
- Kde chyba vzniká? – najít konkrétní program/knihovnu.
- Čas přístupu k souboru (ale: **pozor na noatime/relatime**).

- Spouští se vůbec tento program?
- Načítá se vůbec tento konfigurační soubor?

# Odstraňování chyb

- Modularita UNIXu – zjednodušuje určení místa výskytu problému.
- Kde chyba vzniká? – najít konkrétní program/knihovnu.
- Čas přístupu k souboru (ale: pozor na noatime/relatime).



- Spouští se vůbec tento program?
- Načítá se vůbec tento konfigurační soubor?

# Chybová hlášení

- Zjistit, co chybové hlášení znamená
- Není-li to jasné - přečíst dokumentaci, případně zdrojové texty programu.

## Doporučení pro programátory

Vypisujte chyby z pohledu volajícího.  
(ale: obecné chyby versus interní chyby)

- Pokud není chybové hlášení - zjistit, do kterého logovacího souboru se zapisuje.
- Vyžádat si podrobnější informace - zapnout podrobné výpisy v konfiguračním souboru nebo na příkazové řádce.
- Zachytit podrobnější informace - zapnout sledování zpráv priority debug v `syslog.conf`.

# Chybová hlášení

- Zjistit, co chybové hlášení znamená
- Není-li to jasné - přečíst dokumentaci, případně zdrojové texty programu.



## Doporučení pro programátory

Vypisujte chyby z pohledu volajícího.  
(ale: obecné chyby versus interní chyby)

- Pokud není chybové hlášení - zjistit, do kterého logovacího souboru se zapisuje.
- Vyžádat si podrobnější informace - zapnout podrobné výpisy v konfiguračním souboru nebo na příkazové řádce.
- Zachytit podrobnější informace - zapnout sledování zpráv priority debug v `syslog.conf`.

# Chybová hlášení

- Zjistit, co chybové hlášení znamená
- **Není-li to jasné** – přečíst dokumentaci, případně zdrojové texty programu.



## Doporučení pro programátory

Vypisujte chyby z pohledu volajícího.  
(ale: obecné chyby versus interní chyby)

- **Pokud není chybové hlášení** – zjistit, do kterého logovacího souboru se zapisuje.
- Vyžádat si **podrobnější informace** – zapnout podrobné výpisy v konfiguračním souboru nebo na příkazové řádce.
- **Zachytit podrobnější informace** – zapnout sledování zpráv priority debug v `syslog.conf`.



# Chybová hlášení

- Zjistit, co chybové hlášení znamená
- **Není-li to jasné** – přečíst dokumentaci, případně zdrojové texty programu.






## Doporučení pro programátory

Vypisujte chyby z pohledu volajícího.  
(ale: obecné chyby versus interní chyby)

- **Pokud není chybové hlášení** – zjistit, do kterého logovacího souboru se zapisuje.
- **Vyžádat si podrobnější informace** – zapnout podrobné výpisy v konfiguračním souboru nebo na příkazové řádce.
- **Zachytit podrobnější informace** – zapnout sledování zpráv priority debug v `syslog.conf`.

# Sledování procesu

- Služby jádra, které proces postupně vykonává.
- `strace(1)` – Linux a další systémy 
- `par(1)` – IRIX 
- `truss(1)` – Solaris/SunOS 
- Zjištění chyby služby jádra, která vedla k ukončení procesu.
- Podobné nástroje – `ltrace(1)`, `valgrind(1)`.



# Trace toolkits

- Sledování systému jako celku.
- Na živém produkčním systému.
- Nemělo by ovlivnit systém (ale heisenbugs).
- Nemělo by zpomalit systém, není-li zapnuto.




Příklady:

- DTrace
- SystemTap
- BPFTrace, BCC

# Trace toolkits

- Sledování systému jako celku.
- Na živém produkčním systému.
- Nemělo by ovlivnit systém (ale heisenbugs).
- Nemělo by zpomalit systém, není-li zapnuto.

Příklady:

- DTrace  
- SystemTap 
- BPFTrace, BCC 

# Sondy

- **Sonda (probe)** – místo v programu, které lze sledovat (např. vstup do jisté funkce).
- **Atributy** – např. parametry funkce, PID procesu, ...
- **Implementace** – prázdné instrukce, v případě potřeby se doplní odskok na ladící kód.
- **Dynamické sondy** – umístit odskok kamkoli do kódu.

# Sondy

- **Sonda (probe)** – místo v programu, které lze sledovat (např. vstup do jisté funkce).
- **Atributy** – např. parametry funkce, PID procesu, ...
- **Implementace** – prázdné instrukce, v případě potřeby se doplní odskok na ladící kód.
- **Dynamické sondy** – umístit odskok kamkoli do kódu.

# Trasovací nástroje v Linuxu



- **Statické**, předdefinované
  - Definované člověkem
  - Stablnější v čase
  - Možnost vyšší úrovně abstrakce (TCP spojení, SQL příkaz)
  - Linux kernel: [tracepoints](#) (ftrace)
  - Linux userland: [USDT probes](#), kompatibilní s DTrace probes
- Dynamické
  - Složitější (co když instrukci zrovna někdo vykonává?)
  - Linux kernel: [kprobes](#)
  - Linux userland: [uprobes](#)

# Trasovací nástroje v Linuxu




- Statické, předdefinované
  - Definované člověkem
  - Stabilnější v čase
  - Možnost vyšší úrovně abstrakce (TCP spojení, SQL příkaz)
  - Linux kernel: [tracepoints](#) (ftrace)
  - Linux userland: [USDT probes](#), kompatibilní s DTrace probes
- **Dynamické**
  - Složitější (co když instrukci zrovna někdo vykonává?)
  - Linux kernel: [kprobes](#)
  - Linux userland: [uprobes](#)




# Berkeley Packet Filter



- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
- Není turingovsky kompletní
- Verifikovatelný; nelze zacyklit
- Ukládání dat do asociativních polí
  - Histogramy, per-proces statistiky, ...
- JIT kompilace v kernelu
- Linux eBPF: nástroj na vymístění politiky z jádra
  - Alternativní přístupová práva
  - Bezpečnostní moduly
  - Programy pro trasování
  - ... (programování Linuxu v BPF ^\_~)


# Berkeley Packet Filter



- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
  - Není turingovsky kompletní
  - Verifikovatelný; nelze zacyklit
  - Ukládání dat do asociativních polí
    - Histogramy, per-proces statistiky, ...
  - JIT kompilace v kernelu
  - Linux eBPF: nástroj na vymístění politiky z jádra
    - Alternativní přístupová práva
    - Bezpečnostní moduly
    - Programy pro trasování
    - ... (programování Linuxu v BPF ^\_~)


# Berkeley Packet Filter



- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
- Není turingovsky kompletní
- Verifikovatelný; nelze zacyklit
- Ukládání dat do asociativních polí
  - Histogramy, per-proces statistiky, ...
- JIT kompilace v kernelu
- Linux eBPF: nástroj na vymístění politiky z jádra
  - Alternativní přístupová práva
  - Bezpečnostní moduly
  - Programy pro trasování
  - ... (programování Linuxu v BPF ^\_~)


# Berkeley Packet Filter



- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
- Není turingovsky kompletní
- Verifikovatelný; nelze zacyklit
- Ukládání dat do asociativních polí
  - Histogramy, per-proces statistiky, ...
- JIT kompilace v kernelu
- Linux eBPF: nástroj na vymístění politiky z jádra
  - Alternativní přístupová práva
  - Bezpečnostní moduly
  - Programy pro trasování
  - ... (programování Linuxu v BPF ^\_~)


# Berkeley Packet Filter



- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
- Není turingovsky kompletní
- Verifikovatelný; nelze zacyklit
- Ukládání dat do asociativních polí
  - Histogramy, per-proces statistiky, ...
- JIT kompilace v kernelu
- Linux eBPF: nástroj na vymístění politiky z jádra
  - Alternativní přístupová práva
  - Bezpečnostní moduly
  - Programy pro trasování
  - ... (programování Linuxu v BPF ^\_~)


# Berkeley Packet Filter




- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
- Není turingovsky kompletní
- Verifikovatelný; nelze zacyklit
- Ukládání dat do asociativních polí
  - Histogramy, per-proces statistiky, ...
- JIT kompilace v kernelu
- Linux eBPF: nástroj na vymístění politiky z jádra
  - Alternativní přístupová práva
  - Bezpečnostní moduly
  - Programy pro trasování
  - ... (programování Linuxu v BPF ^\_~)

# Berkeley Packet Filter



- Původně nástroj pro urychlení tcpdump (8), pcap (3)  

- Bytekód pro spuštění uvnitř jádra
- Není turingovsky kompletní
- Verifikovatelný; nelze zacyklit
- Ukládání dat do asociativních polí
  - Histogramy, per-proces statistiky, ...
- JIT kompilace v kernelu
- Linux eBPF: nástroj na vymístění politiky z jádra
  - Alternativní přístupová práva
  - Bezpečnostní moduly
  - Programy pro trasování
  - ... (programování Linuxu v BPF ^\_~)

# Programovací jazyky pro trasování

- Popis toho, co chceme sledovat.
- DTrace – jazyk D, podobný AWK.
- Kompilace do mezikódu.
- Jádro kontroluje korektnost.
- bpf-tools – nad BCC.
  - Kód v Pythonu + C
  - Relativně komplikované
- bpftrace(8) : kompilace do eBPF
  - Jazyk podobný D-Trace (a AWK)
  - Název sondy, podmínka, programový kód





# Příklad: BPF-Tools

```
# /usr/share/bcc/tools/tcpconnect  
# /usr/share/bcc/tools/hardirqs  
# /usr/share/bcc/tools/perlcalls
```

# BPFTrace

## Příklad: Velikost read(2)

```
kr:vfs_read { @ = hist(retval); }
```

## Příklad: Doba trvání read(2)

```
kprobe:vfs_read { @start[tid] = nsecs; }  
kretprobe:vfs_read /@start[tid]/ {  
    @ns[comm] = hist(nsecs - @start[tid]);  
    delete(@start[tid]);  
}
```

## Příklad: Volání readline(3)

```
uretprobe:bash:readline {  
    printf("%s\n", str(retval))  
}
```

# BPFTrace

## 📄 Příklad: Velikost read(2)

```
kr:vfs_read { @ = hist(retval); }
```

## 📄 Příklad: Doba trvání read(2)

```
kprobe:vfs_read { @start[tid] = nsecs; }  
kretprobe:vfs_read /@start[tid]/ {  
    @ns[comm] = hist(nsecs - @start[tid]);  
    delete(@start[tid]);  
}
```

## Příklad: Volání readline(3)

```
uretprobe:bash:readline {  
    printf("%s\n", str(retval))  
}
```

# BPFTrace

## 📄 Příklad: Velikost read(2)

```
kr:vfs_read { @ = hist(retval); }
```

## 📄 Příklad: Doba trvání read(2)

```
kprobe:vfs_read { @start[tid] = nsecs; }  
kretprobe:vfs_read /@start[tid]/ {  
    @ns[comm] = hist(nsecs - @start[tid]);  
    delete(@start[tid]);  
}
```

## 📄 Příklad: Volání readline(3)

```
uretprobe:bash:readline {  
    printf("%s\n", str(retval))  
}
```

# Trasování: další odkazy

- <http://www.brendangregg.com/linuxperf.html>
- <https://www.youtube.com/watch?v=16slh29iN1g>
- <https://github.com/mmisono/bpftrace-tetris>

# Dobrá rada na závěr

## Rada nad zlato :-)



Po nějaké době řešení problému je dobré si znovu přečíst chybové hlášení a dokumentaci.



# Kapitola 8

# Subsystemy

# Cron

- Vykonávání prací v zadaném čase.
- Úlohy běží pod UID/GID toho, kdo tuto úlohu požadoval.
- Standardní výstup a chybový výstup je zaslán uživateli e-mailem.
- **Typy úloh:** jednorázové, pravidelné, dávkové.
- Implementace: vixie-cron ,  
cronie .





# Cron - pravidelné úlohy

## crontab(1) Pravidelně vykonávané úlohy

```
$ crontab [-u user] soubor.crontab
```

```
$ crontab [-u user] [-l|-r|-e]
```

-l vypíše tabulku.

-e editace (\$EDITOR, \$VISUAL).

-r smaže tabulku.

-u *login* - nastavení pro jiného uživatele.

*bez parametru* - bere tabulku **ze std. vstupu**.

# Formát crontabulky

## Příklad: řádek crontabulky

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/
*/5 8-16 * * 1-5    /usr/bin/kdo-zrovna-pracuje
2,22,42 * * * *    /usr/local/bin/20mins
```

- **Položky** - minuta, hodina, den v měsíci, měsíc, den v týdnu, příkaz + argumenty.
- Logický **součin** podmínek.
- Pokud den v měsíci a v týdnu není \*, bere se jejich logický součet.

# Systémové pravidelné úlohy

- „Rozumné“ periody (jak často, nikoliv kdy přesně):
  - /etc/cron.hourly
  - /etc/cron.daily
  - /etc/cron.weekly
  - /etc/cron.monthly
- `run-parts(1)` - spouštění všech skriptů v adresáři.
- Crontabulky systémových služeb: `/etc/cron.d`
  - Navíc identifikace uživatele.
- Někdy také `/etc/crontab`.

**Příklad:** `/etc/cron.d/0hourly`

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
```

```
MAILTO=root
```

```
01 * * * * root run-parts /etc/cron.hourly
```

# Systémové pravidelné úlohy

- „Rozumné“ periody (jak často, nikoliv kdy přesně):
  - /etc/cron.hourly
  - /etc/cron.daily
  - /etc/cron.weekly
  - /etc/cron.monthly
- `run-parts(1)` - spouštění všech skriptů v adresáři.
- Crontabulky systémových služeb: `/etc/cron.d`
  - Navíc identifikace `uživatele`.
- Někdy také `/etc/crontab`.

Příklad: `/etc/cron.d/0hourly`

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
```

```
MAILTO=root
```

```
01 * * * * root run-parts /etc/cron.hourly
```

# Systemové pravidelné úlohy

- „Rozumné“ periody (jak často, nikoliv kdy přesně):
  - /etc/cron.hourly
  - /etc/cron.daily
  - /etc/cron.weekly
  - /etc/cron.monthly
- `run-parts(1)` - spouštění všech skriptů v adresáři.
- Crontabulky systémových služeb: `/etc/cron.d`
  - Navíc identifikace `uživatele`.
- Někdy také `/etc/crontab`.



## Příklad: `/etc/cron.d/0hourly`

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
01 * * * * root run-parts /etc/cron.hourly
```

# Anacron



## Problémy nočních úloh:

- Co dělat, když je počítač přes noc vypnutý?
  - Jak nevyčerpat baterii laptopu?
  - Jak nepřetížit virtualizační server?
- anacron(8) - pravidelné úlohy s denní a delší periodou.
- /etc/anacrontab
- Časové razítko každé úlohy.

# Anacron



## Problémy nočních úloh:

- Co dělat, když je počítač přes noc vypnutý?
  - Jak nevyčerpat baterii laptopu?
  - Jak nepřetížit virtualizační server?
- 
- anacron(8) - pravidelné úlohy s denní a delší periodou.
  - /etc/anacrontab
  - Časové razítko každé úlohy.

# Anacron



## Problémy nočních úloh:

- Co dělat, když je počítač přes noc vypnutý?
  - Jak nevyčerpat baterii laptopu?
  - Jak nepřetížit virtualizační server?
- 
- `anacron(8)` - pravidelné úlohy s denní a delší periodou.
  - `/etc/anacrontab`
  - Časové razítko každé úlohy.



# Anacron



## Problémy nočních úloh:

- Co dělat, když je počítač přes noc vypnutý?
  - Jak nevyčerpat baterii laptopu?
  - Jak nepřetížit virtualizační server?
- 
- `anacron(8)` - pravidelné úlohy s denní a delší periodou.
  - `/etc/anacrontab`
  - Časové razítko každé úlohy.

## Příklad: /etc/anacrontab

```
RANDOM_DELAY=45  
START_HOURS_RANGE=3-22
```

```
# period[days] delay[mins] job-id command  
1          5    cron.daily    nice run-parts \  
            /etc/cron.daily  
7          25   cron.weekly   nice run-parts \  
            /etc/cron.weekly  
@monthly  45   cron.monthly  nice run-parts \  
            /etc/cron.monthly
```

### Otázka:

K čemu je omezení pomocí START\_HOURS\_RANGE?

## Příklad: /etc/anacrontab

```
RANDOM_DELAY=45  
START_HOURS_RANGE=3-22
```

```
# period[days] delay[mins] job-id command  
1          5    cron.daily    nice run-parts \  
           /etc/cron.daily  
7          25   cron.weekly  nice run-parts \  
           /etc/cron.weekly  
@monthly  45   cron.monthly nice run-parts \  
           /etc/cron.monthly
```



### Otázka:

K čemu je omezení pomocí START\_HOURS\_RANGE?

# Jednorázové úlohy

- `at(1)` - vykonání práce v zadaném čase.
  - Pamatuje si proměnné prostředí.
- `atq(1)` - výpis fronty příkazů `at(1)`.
- `atrm(1)` - smazání úlohy z fronty.
- `batch(1)` - odložené vykonávání (např. až `load average` klesne pod 1).



## Dávkové zpracování

Lépe použít samostatné dávkové frontové systémy.

- `Implementace` - `crond(8)` nebo samostatný `atd(8)`.

# Recyklace logovacích souborů

- **Problém** – logovací soubory mohou zaplnit disk.
- **Rotování logovacích souborů** – přejmenování, komprese, zachovat několik posledních.
- **Notifikace** zapisujícímu procesu (např. SIGHUP pro `syslogd(8)`).

## logrotate(8)

```
# logrotate soubor  
# logrotate /etc/logrotate.conf
```

- Spouštět například z `/etc/cron.daily`.

# Recyklace logovacích souborů

- **Problém** – logovací soubory mohou zaplnit disk.
- **Rotování logovacích souborů** – přejmenování, komprese, zachovat několik posledních.
- **Notifikace** zapisujícímu procesu (např. SIGHUP pro `syslogd(8)`).

## logrotate(8)

```
# logrotate soubor  
# logrotate /etc/logrotate.conf
```

- Spouštět například z `/etc/cron.daily`.

## Příklad: /etc/logrotate.conf

```
weekly
rotate 4
create
dateext
compress

include /etc/logrotate.d

/var/log/wtmp {
    monthly
    create 0664 root utmp
        minsize 1M
    rotate 1
}
```

## 📄 Příklad: /etc/logrotate.d/syslog

```
/var/log/messages /var/log/secure \  
    /var/log/maillog /var/log/spooler \  
    /var/log/boot.log /var/log/cron {  
sharedscripts  
postrotate  
    /bin/kill -HUP \  
        'cat /var/run/syslogd.pid'  
endscript  
}
```

### Otázka:

Jaký problém by způsobilo uvedení cesty ve tvaru  
„/var/log/\*“?



## 📄 Příklad: /etc/logrotate.d/syslog

```
/var/log/messages /var/log/secure \  
    /var/log/maillog /var/log/spooler \  
    /var/log/boot.log /var/log/cron {  
sharedscripts  
postrotate  
    /bin/kill -HUP \  
        'cat /var/run/syslogd.pid'  
endscript  
}
```



### Otázka:

Jaký problém by způsobilo uvedení cesty ve tvaru „/var/log/\*“?

# Promazávání dočasných souborů

- Možnost přepnutí /tmp.
- **Pozor** na symbolické linky!

## tmpwatch(8)



```
# tmpwatch [-u|-m|-c] [-x soubor] [-X glob] \  
[-f] čas adresář
```

Příklad: /etc/cron.daily/tmpwatch

```
/usr/sbin/tmpwatch -umc -x /tmp/.X11-unix \  
-x /tmp/.XIM-unix -x /tmp/.font-unix \  
-x /tmp/.ICE-unix -x /tmp/.Test-unix \  
-X '/tmp/hsperfdata_*' 10d /tmp  
/usr/sbin/tmpwatch -umc 30d /var/tmp
```

# Promazávání dočasných souborů

- Možnost přepnutí /tmp.
- **Pozor** na symbolické linky!

## tmpwatch(8)



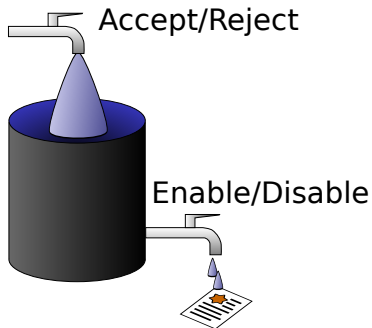
```
# tmpwatch [-u|-m|-c] [-x soubor] [-X glob] \  
[-f] čas adresář
```

### Příklad: /etc/cron.daily/tmpwatch

```
/usr/sbin/tmpwatch -umc -x /tmp/.X11-unix \  
-x /tmp/.XIM-unix -x /tmp/.font-unix \  
-x /tmp/.ICE-unix -x /tmp/.Test-unix \  
-X '/tmp/hsperfdata_*' 10d /tmp  
/usr/sbin/tmpwatch -umc 30d /var/tmp
```

# Tiskárny

- **Spooling** - tisková fronta.
- Více front na jednu tiskárnu (např. oboustranný tisk).
- **Filtry** - konverze do jazyka tiskárny.



# BSD tiskový subsystém

- Konfigurace: /etc/printcap.
- Server: lpd(8), /etc/hosts.lpd.
- Filtry: vstupní, datový, výstupní.
- Řízení - příkaz lpc(8).
- Uživatelé: lpr(1), lpq(1), lprm(1).

 **Příklad: /etc/printcap**

```
dj|lp|HP DeskJet 540 – raw device:\
:sd=/var/spool/lpd/dj:\
:sh:\
:sf:\
:mx#10000:\
:lf=/var/log/lpd-errs:\
:pl#60:\
:lp=/dev/lp1:
```

*nebo*

```
:rm=geofront.nervhq.com:\
:rp=lj4:
```

# System V Tiskárny

- **Démon** `lpsched(8)`.
- **Administrace:** `lpadm(8)`, `lpshut(8)`.
- **Řízení přístupu** - `accept(8)`, `reject(8)`, `enable(8)`, `disable(8)`.
- `lpmove(8)`.
- **Uživatelé:** `lp(1)`, `cancel(1)`, `lpstat(1)`.

# Common UNIX Printing System

- Nový síťový protokol (nad HTTP, port 631).
- Administrace – WWW rozhraní.
- Rozšiřitelné **atributy** tiskáren:
  - duplex
  - ekonomický režim
  - zásobníky papíru
- **Autokonfigurace klientů** – nemusí být tiskové fronty i na klientech.
- **Popis tiskárny** – soubor **PPD**.  
Obsahuje PostScriptové příkazy pro příslušné atributy.





# Čtení na dobrou noc



## Komiks na dobrou noc

Why I Believe Printers Were Sent From Hell  
To Make Us Miserable

<https://theoatmeal.com/comics/printers>

# Diskové kvóty

- Sledování místa, obsazeného uživatelem.
- V rámci jednoho svazku.
- Kvóta na počet **i-uzlů** a **bloků**.
- **Měkká/tvrdá** kvóta.
- **Časový limit**.
- Soubor `/quota`, `/quota.user`, `/quota.group` nebo přímo v metadatech FS.

# Získání informací o kvótách

## quota(1)

```
$ quota [-v] [-u user]
```

- v - informace o všech kvótách (jinak jen o překročených).
- u - jen superuživatel.

### Příklad: Formát výpisu kvót

```
# quota -v -u fordprefect
Disk quotas for user fordprefect (uid 42):
Filesystem blocks      quota      limit grace \
/thewguide  8570400 10000000 12000000      \
              files      quota      limit grace
              66980    180000    200000
```

# Získání informací o kvótách

## quota(1)

```
$ quota [-v] [-u user]
```

- v - informace o všech kvótách (jinak jen o překročených).
- u - jen superuživatel.

## Příklad: Formát výpisu kvót

```
# quota -v -u fordperfect
```

```
Disk quotas for user fordperfect (uid 42):  
Filesystem blocks      quota      limit grace \  
/theguide  8570400 10000000 12000000  \  
            files      quota      limit grace  
            66980   180000   200000
```

# Kvóty - administrace

## quotaon(8)

## Administrace kvót

```
# quotaon [-a] [filesystem]
# quotaoff [-a] [filesystem]
# quotacheck [-a] [filesystem]
```



### Konzistence kvót

Je-li souborový systém používán s vypnutými kvótami, je nutné před víceuživatelským použitím znovu přepočítat obsazení disku.

# Nastavení kvót uživatelům

## edquota(8)

## Editace kvót

```
# edquota uživatel ...  
# edquota [-p vzor] uživatel ...
```

## setquota(8)

## Nastavení kvót

```
# setquota [-u|g] uživatel \  
    block-soft block-hard \  
    inode-soft inode-hard \  
    -a|filesystem  
# setquota -t \  
    block-grace inode-grace \  
    -a|filesystem  
# setquota -b -a|filesystem
```

# Nastavení kvót uživatelům

## edquota(8)

## Editace kvót

```
# edquota uživatel ...  
# edquota [-p vzor] uživatel ...
```

## setquota(8)

## Nastavení kvót

```
# setquota [-u|g] uživatel \  
    block-soft block-hard \  
    inode-soft inode-hard \  
    -a|filesystem  
# setquota -t \  
    block-grace inode-grace \  
    -a|filesystem  
# setquota -b -a|filesystem
```

# Hlášení překročených kvót

```
repquota(8), warnquota(8)
```

```
# repquota -a|filesystem  
# warnquota [-c configfile]
```

## Příklad: repquota(8)

```
*** Report for user quotas on device /theguide  
Block grace time: 7days; Inode grace time: 7days  
                Block limits      ...  
User           used  soft  hard grace ...  
-----  
root           -- 443708    0    0      ...  
zaphod         +- 217614 200000 300000  none ...  
fordprefect    --  61028 500000 600000      ...  
arthurdent     +- 175334 150000 180000 5days ...
```



# Kapitola 9

# Virtualizace

# Virtualizace

Co je virtualizace?

- **Zapouzdření SW**

- bez vyhrazeného HW
- procesu, aplikace, celého OS
- hypervizor/hostitel: poskytuje virtuální HW
- host (ang. guest), instance, virtuální stroj: klient

- Virtualizace je také

- UNIXový proces
- chroot(2)
- jail(2)
- SELinux sandbox

# Virtualizace

## Co je virtualizace?

- Zapouzdření SW
  - bez vyhrazeného HW
  - procesu, aplikace, celého OS
  - hypervizor/hostitel: poskytuje virtuální HW
  - host (ang. guest), instance, virtuální stroj: klient
- Virtualizace je také
  - UNIXový proces
  - chroot(2)
  - jail(2)
  - SELinux sandbox

# Virtualizace

## Co je virtualizace?

- Zapouzdření SW
  - bez vyhrazeného HW
  - procesu, aplikace, celého OS
  - hypervizor/hostitel: poskytuje virtuální HW
  - host (ang. guest), instance, virtuální stroj: klient
- Virtualizace je také
  - UNIXový proces
    - `chroot(2)`
    - `jail(2)`
    - SELinux sandbox

# Virtualizace

## Co je virtualizace?

- Zapouzdření SW
  - bez vyhrazeného HW
  - procesu, aplikace, celého OS
  - hypervizor/hostitel: poskytuje virtuální HW
  - host (ang. guest), instance, virtuální stroj: klient
- Virtualizace je také
  - UNIXový proces
  - chroot(2)
  - jail(2)
  - SELinux sandbox

# Virtualizace

## Co je virtualizace?

- Zapouzdření SW
  - bez vyhrazeného HW
  - procesu, aplikace, celého OS
  - hypervizor/hostitel: poskytuje virtuální HW
  - host (ang. guest), instance, virtuální stroj: klient
- Virtualizace je také
  - UNIXový proces
  - chroot(2)
  - jail(2)
  - SELinux sandbox

# Virtualizace

## Co je virtualizace?

- Zapouzdření SW
  - bez vyhrazeného HW
  - procesu, aplikace, celého OS
  - hypervizor/hostitel: poskytuje virtuální HW
  - host (ang. guest), instance, virtuální stroj: klient
- Virtualizace je také
  - UNIXový proces
  - chroot(2)
  - jail(2)
  - SELinux sandbox

# Co virtualizujeme?

- Virtualizace OS
- Virtualizace CPU
- Virtualizace RAM
- Virtualizace periférií

## Upozornění:

UNIX je vlastně také virtualizační prostředí!



# Co virtualizujeme?

- Virtualizace OS
- Virtualizace CPU
- Virtualizace RAM
- Virtualizace periférií



## Upozornění:

UNIX je vlastně také virtualizační prostředí!

# Emulace

- Interpretace instrukcí virtuálním procesorem
- JIT kompilace
- Interpretace I/O příkazů hardwaru
- Cíl: virtualizovat HW, který nemám

# Nativní virtualizace

- Přímé spouštění nativních instrukcí
- Musí podporovat HW
- Vnořená virtualizace?
  - IBM VM/OS, cca 1970

# Paravirtualizace

- Přizpůsobení virtualizovaného SW
- Např. samostatná CPU platforma
- Jiné operace pro změnu mapování stránek, přerušení, ...
- Paravirtualizované periferie

## Otázka:

Který typ virtualizace je nejrychlejší?

# Paravirtualizace

- Přizpůsobení virtualizovaného SW
- Např. samostatná CPU platforma
- Jiné operace pro změnu mapování stránek, přerušení, ...
- Paravirtualizované periferie



## Otázka:

Který typ virtualizace je nejrychlejší?

# Virtualizace periferií

## ■ Emulace

- Zachycování I/O instrukcí
- Generování virtuálních přerušení
- Virtuální DMA

## ■ HW virtualizace

- Síťové karty: samostatné HW fronty
- Disky: samostatné LUN
- Hypervizor je jen arbitr

## ■ Paravirtualizace

- Přizpůsobit komunikaci: SW - SW
- Virtio
- VMware tools, ...
- *Jak si tvůrci OS představují, že by měl HW komunikovat*

# Virtualizace periferií

## ■ Emulace

- Zachycování I/O instrukcí
- Generování virtuálních přerušení
- Virtuální DMA


## ■ HW virtualizace

- Síťové karty: samostatné HW fronty
- Disky: samostatné LUN
- Hypervizor je jen arbitr

## ■ Paravirtualizace

- Přizpůsobit komunikaci: SW - SW
- Virtio
- VMware tools, ...
- *Jak si tvůrci OS představují, že by měl HW komunikovat*

# Virtualizace periferií

- Emulace
  - Zachycování I/O instrukcí
  - Generování virtuálních přerušení
  - Virtuální DMA
- HW virtualizace
  - Síťové karty: samostatné HW fronty
  - Disky: samostatné LUN
  - Hypervizor je jen arbitr
- Paravirtualizace
  - Přizpůsobit komunikaci: SW - SW
  - Virtio 
  - VMware tools, ...
  - *Jak si tvůrci OS představují, že by měl HW komunikovat*



# Virtualizace disků

- Soubor/blokové zařízení: disk uvnitř VM
- Formáty souborů (image)
  - RAW
    - Například LV
  - QCOW2: thin provisioning, snímky
  - VMDK: z VMware
- Další vlastnosti
  - Někdy možnost uložit i stav RAM
  - Snímek živého OS
  - Migrace na jiný HW: distribuované úložiště?
    - CEPH, GlusterFS, ...


# Virtualizace disků

- Soubor/blokové zařízení: disk uvnitř VM
- Formáty souborů (image)
  - RAW
    - Například LV
  - QCOW2: thin provisioning, snímky
  - VMDK: z VMware
- Další vlastnosti
  - Někdy možnost uložit i stav RAM
  - Snímek živého OS
  - Migrace na jiný HW: distribuované úložiště?
    - CEPH, GlusterFS, ...


# Virtualizace disků

- Soubor/blokové zařízení: disk uvnitř VM
- Formáty souborů (image)
  - RAW
    - Například LV
  - QCOW2: thin provisioning, snímky
  - VMDK: z VMware
- Další vlastnosti
  - Někdy možnost uložit i stav RAM
  - Snímek živého OS
  - Migrace na jiný HW: distribuované úložiště?
    - CEPH, GlusterFS, ...


# Virtualizace sítě

- TUN/TAP device 
- Čtení/zápis řídicího souboru = vysílání/příjem paketů
- Hypervizor překládá I/O operace TUN/TAP souboru
  - virtuální síťová karta
- Připojení VM do sítě
  - TUN/TAP je běžné rozhraní
  - Standardní směrování
  - + NAT
  - Bridge
  - SDN/VXLAN


# Virtualizace sítě

- TUN/TAP device 
- Čtení/zápis řídicího souboru = vysílání/příjem paketů
- Hypervizor překládá I/O operace TUN/TAP souboru
  - virtuální síťová karta
- Připojení VM do sítě
  - TUN/TAP je běžné rozhraní
  - Standardní směrování
  - + NAT
  - Bridge
  - SDN/VXLAN

# Virtualizace sítě

- TUN/TAP device 
- Čtení/zápis řídicího souboru = vysílání/příjem paketů
- Hypervizor překládá I/O operace TUN/TAP souboru
  - virtuální síťová karta
- Připojení VM do sítě
  - TUN/TAP je běžné rozhraní
  - Standardní směrování
  - + NAT
  - Bridge
  - SDN/VXLAN

# Virtualizace sítě

- TUN/TAP device 
- Čtení/zápis řídicího souboru = vysílání/příjem paketů
- Hypervizor překládá I/O operace TUN/TAP souboru
  - virtuální síťová karta
- Připojení VM do sítě
  - TUN/TAP je běžné rozhraní
  - Standardní směrování
  - + NAT
  - Bridge
  - SDN/VXLAN

# Virtualizace v Linuxu

## ■ QEMU

- emulátor (x86, x86-64, ARM, ARM64, celé PC, ...)

## ■ KVM

- virtualizace CPU (+ QEMU na virtualizaci celého PC)
- hypervizor = kernel Linuxu

## ■ XEN

- původně paravirtualizace
- nyní i plná virtualizace CPU
- malý hypervizor
- Dom0 Linux pro správu většiny HW
- DomU virtuální stroje

## ■ VirtualBox

- nativní běh user-space
- původně emulace a JIT kompilace privilegovaných instrukcí
- nyní HW virtualizace CPU



# Virtualizace v Linuxu

## ■ QEMU

- emulátor (x86, x86-64, ARM, ARM64, celé PC, ...)

## ■ KVM

- virtualizace CPU (+ QEMU na virtualizaci celého PC)
- hypervizor = kernel Linuxu

## ■ XEN

- původně paravirtualizace
- nyní i plná virtualizace CPU
- malý hypervizor
- Dom0 Linux pro správu většiny HW
- DomU virtuální stroje

## ■ VirtualBox

- nativní běh user-space
- původně emulace a JIT kompilace privilegovaných instrukcí
- nyní HW virtualizace CPU

# Virtualizace v Linuxu

## ■ QEMU

- emulátor (x86, x86-64, ARM, ARM64, celé PC, ...)

## ■ KVM



- virtualizace CPU (+ QEMU na virtualizaci celého PC)
- hypervizor = kernel Linuxu


## ■ XEN

- původně paravirtualizace
- nyní i plná virtualizace CPU
- malý hypervizor
- Dom0 Linux pro správu většiny HW
- DomU virtuální stroje

## ■ VirtualBox

- nativní běh user-space
- původně emulace a JIT kompilace privilegovaných instrukcí
- nyní HW virtualizace CPU

# Virtualizace v Linuxu

- QEMU
  - emulátor (x86, x86-64, ARM, ARM64, celé PC, ...)
- KVM 
  - virtualizace CPU (+ QEMU na virtualizaci celého PC)
  - hypervizor = kernel Linuxu
- XEN
  - původně paravirtualizace
  - nyní i plná virtualizace CPU
  - malý hypervizor
  - Dom0 Linux pro správu většiny HW
  - DomU virtuální stroje
- VirtualBox
  - nativní běh user-space
  - původně emulace a JIT kompilace privilegovaných instrukcí
  - nyní HW virtualizace CPU

# Libvirt

- Správce virtuálních strojů
- Nezávislý na hypervizoru
  - XEN
  - KVM
  - LXC
  - ...
- XML definice VM, VNet, disku, ...
- Z příkazové řádky: `virsh(8)`
- GUI: `virt-manager(1)`
  - včetně vzdáleného připojení
  - nad SSH

# Libvirt

- Správce virtuálních strojů
- Nezávislý na hypervizoru
  - XEN
  - KVM
  - LXC
  - ...
- XML definice VM, VNet, disku, ...
- Z příkazové řádky: `virsh(8)`
- GUI: `virt-manager(1)`
  - včetně vzdáleného připojení
  - nad SSH

# Libvirt

- Správce virtuálních strojů
- Nezávislý na hypervizoru
  - XEN
  - KVM
  - LXC
  - ...
- XML definice VM, VNet, disku, ...
- Z příkazové řádky: `virsh(8)`
- GUI: `virt-manager(1)`
  - včetně vzdáleného připojení
  - nad SSH

# Libvirt

- Správce virtuálních strojů
- Nezávislý na hypervizoru
  - XEN
  - KVM
  - LXC
  - ...
- XML definice VM, VNet, disku, ...
- Z příkazové řádky: `virsh(8)`
- GUI: `virt-manager(1)`
  - včetně vzdáleného připojení
  - nad SSH

# Libvirt

- Správce virtuálních strojů
- Nezávislý na hypervizoru
  - XEN
  - KVM
  - LXC
  - ...
- XML definice VM, VNet, disku, ...
- Z příkazové řádky: `virsh(8)`
- GUI: `virt-manager(1)`
  - včetně vzdáleného připojení
  - nad SSH



# A co takhle cloud?

## There is no cloud



...just other people's computers



Privátní cloud:

- OpenNebula
- OpenStack
- oVirt

Nové slovo: Layer bloat  
Hybridní cloud?

Tip:

Vyzkoušejte <https://stratus.fi.muni.cz/>.

# A co takhle cloud?

## There is no cloud



...just other people's computers



Privátní cloud:

- OpenNebula
- OpenStack
- oVirt

Nové slovo: Layer bloat  
Hybridní cloud?

Tip:

Vyzkoušejte <https://stratus.fi.muni.cz/>.

# A co takhle cloud?

## There is no cloud



...just other people's computers



Privátní cloud:

- OpenNebula
- OpenStack
- oVirt

Nové slovo: Layer bloat  
Hybridní cloud?

Tip:

Vyzkoušejte <https://stratus.fi.muni.cz/>.

# A co takhle cloud?

## There is no cloud



...just other people's computers



Privátní cloud:

- OpenNebula
- OpenStack
- oVirt

Nové slovo: Layer bloat  
Hybridní cloud?

Tip:

Vyzkoušejte <https://stratus.fi.muni.cz/>.

# A co takhle cloud?

## There is no cloud



...just other people's computers



Privátní cloud:

- OpenNebula
- OpenStack
- oVirt

Nové slovo: Layer bloat  
Hybridní cloud?



Tip:

Vyzkoušejte <https://stratus.fi.muni.cz/>.

# Kontejnery

- Virtualizace SW prostředí
  - knihovny, konfigurace, ...
- Bez virtualizace jádra (menší režie)
- Oddělené zdroje (sítě, PID, FS, ...)
- R-only (COW) obrazy FS (vrstvy ...)



# Kontejnery

- Virtualizace SW prostředí
  - knihovny, konfigurace, ...
- Bez virtualizace jádra (menší režie)
- Oddělené zdroje (sítě, PID, FS, ...)
- R-only (COW) obrazy FS (vrstvy ...)



# Kontejnery

- Virtualizace SW prostředí
  - knihovny, konfigurace, ...
- Bez virtualizace jádra (menší režie)
- Oddělené zdroje (sítě, PID, FS, ...)
- R-only (COW) obrazy FS (vrstvy ...)





# Kontejnery

- Virtualizace SW prostředí
  - knihovny, konfigurace, ...
- Bez virtualizace jádra (menší režie)
- Oddělené zdroje (sít, PID, FS, ...)
- R-only (COW) obrazy FS (vrstvy ...)



# Jmenné prostory v Linuxu



- Oddělení různých částí systému
- Stavebnice pro tvorbu kontejnerů
- Hierarchické
- Služby jádra: `clone(CLONE_NEW*)`, `setns(2)`, `unshare(2)`
- Příkazy: `unshare(1)`, `nsenter(1)`, `lsns(1)`

# Jmenné prostory v Linuxu



- Oddělení různých částí systému
- Stavebnice pro tvorbu kontejnerů
- Hierarchické
- Služby jádra: `clone(CLONE_NEW*)`, `setns(2)`, `unshare(2)`
- Příkazy: `unshare(1)`, `nsenter(1)`, `lsns(1)`

# Jmenné prostory v Linuxu



- Oddělení různých částí systému
- Stavebnice pro tvorbu kontejnerů
- Hierarchické
  - Služby jádra: `clone(CLONE_NEW*)`, `setns(2)`, `unshare(2)`
  - Příkazy: `unshare(1)`, `nsenter(1)`, `lsns(1)`

# Jmenné prostory v Linuxu



- Oddělení různých částí systému
- Stavebnice pro tvorbu kontejnerů
- Hierarchické
- Služby jádra: `clone(CLONE_NEW*)`, `setns(2)`, `unshare(2)`
- Příkazy: `unshare(1)`, `nsenter(1)`, `lsns(1)`

# Jmenné prostory v Linuxu



- Oddělení různých částí systému
- Stavebnice pro tvorbu kontejnerů
- Hierarchické
- Služby jádra: `clone(CLONE_NEW*)`, `setns(2)`, `unshare(2)`
- Příkazy: `unshare(1)`, `nsenter(1)`, `lsns(1)`

# Mount namespace

- Samostatný strom adresářů
- chroot(2)?
- ale: procfs, sysfs
- Sdílet změny i do podřízených NS?
- Mít část stromu r/only (nosuid, nodev, ...)

# User namespace

- Prostor UID/GID
- Mapování dovnitř
- UID=0 není venku root
- ale: mount(2), bind(2), ...
- subuid(5), subgid(5)

## Příklad: User namespace

```
unshare --user --map-auto --map-root-user \  
  --fork /bin/bash
```

```
nsenter -U -t 12345
```



# PID namespace

- nevidět cizí PID
- umožnit PID 1



## Příklad: PID namespace

```
unshare --pid --mount-proc --fork /bin/bash
```

# Network namespace

- `bind(2)` jen na jednu adresu
- `INADDR_ANY/IP6_ANY`
- komunikovat i s jinými adresami téhož počítače
- filtrovat packety(?)
- síťové zařízení: patří do jednoho netns
- virtual ethernet pair

# Další jmenné prostory

- CGroups
- IPC
- Time
- UTS (hostname a podobně)

# Aplikační kontejnery

- aplikace běží jako PID 1
- veškerá konfigurace *zvenku*
  - proměnné prostředí
  - adresář s konfigurací
  - přednastaveno zvenku (sít)
- Pod – spojení více kontejnerů (sidecars, ...)
- Docker, Podman, systemd-nspawn --as-pid2
- Kontejner jako balíček (Snap, FlatPak, ...)

# Systemové kontejnery

- celý user-space včetně initu
- LXC
- `systemd-nspawn -b`

## Příklad: systemd-nspawn (8)

```
# dnf -y --releasever=39 \  
    --installroot /var/lib/machines/f39 \  
    install fedora-release bash  
# systemd-nspawn -D /var/lib/machines/f39 bash  
# dnf -y --releasever=39 \  
    --installroot /var/lib/machines/f39 \  
    install procps systemd vim coreutils  
# systemd-nspawn -b -D /var/lib/machines/f39
```

# Kontejnery



## Čtení na dobrou noc

Not your grandmas' docker - Part 2

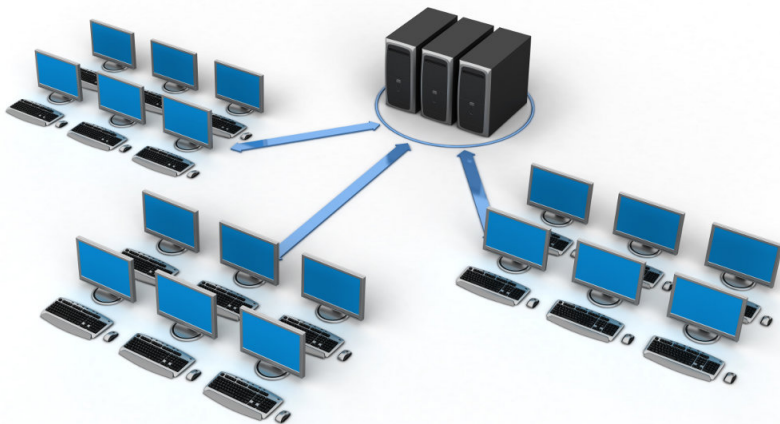
<https://osadalakmał.com/posts/container-part2/>

# Kapitola 10

## Sítě TCP/IP



# Sítě TCP/IP



# Sítě TCP/IP

- **internet** – několik sítí propojených dohromady.
- **Internet** – celosvětová síť s protokolem TCP/IP.
- **Transmission Control Protocol/Internet Protocol**
- **IETF** – Internet Engineering Task Force.
- **RFC** – Request For Comments – dokumenty, popisující jednotlivé protokoly.  
<https://ftp.fi.muni.cz/pub/rfc>.
- **Internet drafts** – návrhy protokolu.  
<https://ftp.fi.muni.cz/pub/internet-drafts>.  
Omezená časová platnost.

# Historie Internetu

- **1969** ARPANET – první síť s přepojováním packetů – 4 uzly.
- **1977** Začíná vývoj protokolů TCP/IP (Stanford, BBN, University College London).
- **1980** Provoz TCP/IP v ARPANETu. UCB implementuje TCP/IP pro BSD.
- **1983** TCP/IP se stává standardem v ARPANETu. Sun Microsystems – TCP/IP v komerční sféře.
- **1985** NSFnet – jádro současného Internetu.

# Charakteristiky TCP/IP

- Sít s přepojováním packetů (opak od přepojování okruhů; kombinace obou je např. ATM, MPLS).
- Stejné protokoly pro všechny sítě a druhy sítí.
- Směrování na základě cílové sítě, nikoli cílové adresy.

# Vrstvy protokolů TCP/IP

Nižší vrstvy přibližně podle ISO OSI:

- **1: fyzická** – například UTP kabel, optické vlákno, metalický okruh.
- 2: linková – síťové rozhraní (Ethernet, ATM LANE, PPP, HDLC, FDDI, RFC 1149 ^\_~).
- 3: síťová – IPv4, IPv6 (datagramy), ARP (získání HW adresy), RARP (získání vlastní IP adresy).
- 4: transportní – ICMP, ICMPv6 (řídící zprávy), IGMP (skupinová adresace), TCP (proudy dat), UDP (datagramy).
- 5+: aplikační – např. SMTP, FTP, telnet, NTP, SNMP, SSH, XMPP, ...

# Vrstvy protokolů TCP/IP

Nižší vrstvy přibližně podle ISO OSI:

- **1: fyzická** – například UTP kabel, optické vlákno, metalický okruh.
- **2: linková** – síťové rozhraní (Ethernet, ATM LANE, PPP, HDLC, FDDI, RFC 1149 ^\_~).
- **3: síťová** – IPv4, IPv6 (datagramy), ARP (získání HW adresy), RARP (získání vlastní IP adresy).
- **4: transportní** – ICMP, ICMPv6 (řídící zprávy), IGMP (skupinová adresace), TCP (proudy dat), UDP (datagramy).
- **5+: aplikační** – např. SMTP, FTP, telnet, NTP, SNMP, SSH, XMPP, ...

# Vrstvy protokolů TCP/IP

Nižší vrstvy přibližně podle ISO OSI:

- **1: fyzická** – například UTP kabel, optické vlákno, metalický okruh.
- **2: linková** – síťové rozhraní (Ethernet, ATM LANE, PPP, HDLC, FDDI, RFC 1149 ^\_~).
- **3: síťová** – IPv4, IPv6 (datagramy), ARP (získání HW adresy), RARP (získání vlastní IP adresy).
- **4: transportní** – ICMP, ICMPv6 (řídící zprávy), IGMP (skupinová adresace), TCP (proudy dat), UDP (datagramy).
- **5+: aplikační** – např. SMTP, FTP, telnet, NTP, SNMP, SSH, XMPP, ...

# Vrstvy protokolů TCP/IP

Nižší vrstvy přibližně podle ISO OSI:

- **1: fyzická** – například UTP kabel, optické vlákno, metalický okruh.
- **2: linková** – síťové rozhraní (Ethernet, ATM LANE, PPP, HDLC, FDDI, RFC 1149 ^\_~).
- **3: síťová** – IPv4, IPv6 (datagramy), ARP (získání HW adresy), RARP (získání vlastní IP adresy).
- **4: transportní** – ICMP, ICMPv6 (řídící zprávy), IGMP (skupinová adresace), TCP (proudy dat), UDP (datagramy).
- **5+: aplikační** – např. SMTP, FTP, telnet, NTP, SNMP, SSH, XMPP, ...



# Vrstvy protokolů TCP/IP

Nižší vrstvy přibližně podle ISO OSI:

- **1: fyzická** – například UTP kabel, optické vlákno, metalický okruh.
- **2: linková** – síťové rozhraní (Ethernet, ATM LANE, PPP, HDLC, FDDI, RFC 1149 ^\_~).
- **3: síťová** – IPv4, IPv6 (datagramy), ARP (získání HW adresy), RARP (získání vlastní IP adresy).
- **4: transportní** – ICMP, ICMPv6 (řídící zprávy), IGMP (skupinová adresace), TCP (proudy dat), UDP (datagramy).
- **5+: aplikační** – např. SMTP, FTP, telnet, NTP, SNMP, SSH, XMPP, ...

# Způsoby propojení sítí

- **Repeater** – v podstatě propojení na fyzické úrovni.
- **Bridge/Switch** – propojení na linkové úrovni.
- **Router** – propojení na síťové úrovni. Pro každou třídu protokolů může být zvláštní router. Nezávislost na 2. vrstvě.

# Adresování v IPv4

- Internet Protocol – verze 4, RFC 791.
- IP adresa – 4 bajty. Zapisuje se jako 4 dekadická čísla, oddělená tečkami.
- Adresa přidělována rozhraní, nikoli počítači.
- Adresa sítě vs. adresa v rámci sítě.
- Typy vysílání: unicast, broadcast, multicast.

# Třídni adresace

- Typ A - 0.x.x.x-127.x.x.x.
- Typ B - 128.0.x.x-191.255.x.x.
- Typ C - 192.0.0.x-223.255.255.x.
- Typ D - 224.0.0.0-239.255.255.255 - multicast.
- Typ E - 240.0.0.0-255.255.255.255 - rezerva.

# Beztržidní adresace

- Nedostatek adres, vyčerpání adresního prostoru.
- CIDR: prefix adresy (adresa, maska). RFC 1518.
- Zápis: podobně jako adresa nebo jen délka prefixu.

## Příklad: CIDR adresa

147.251.48.1/255.255.255.0

147.251.48.1/24

- Adresa sítě: *interface* and *netmask* - 147.251.48.0.
- Věšměrové vysílání - *interface* or *not netmask* - 147.251.48.255.

# Beztržidní adresace

- **Nedostatek adres**, vyčerpání adresního prostoru.
- **CIDR**: prefix adresy (adresa, maska). RFC 1518.
- **Zápis**: podobně jako adresa nebo jen délka prefixu.

## **Příklad: CIDR adresa**

147.251.48.1/255.255.255.0

147.251.48.1/24

- Adresa sítě: *interface* and *netmask* - 147.251.48.0.
- Věsměrové vysílání - *interface* or *not netmask* - 147.251.48.255.

# Beztrždní adresace

- Nedostatek adres, vyčerpání adresního prostoru.
- CIDR: prefix adresy (adresa, maska). RFC 1518.
- Zápis: podobně jako adresa nebo jen délka prefixu.

## Příklad: CIDR adresa

147.251.48.1/255.255.255.0

147.251.48.1/24

- Adresa sítě: *interface* and *netmask* - 147.251.48.0.
- Všesměrové vysílání - *interface* or not *netmask* - 147.251.48.255.

# Speciální IPv4 adresy

- **Moje adresa** - nezná-li odesílatel svoji adresu - 0.0.0.0.
- **Omezené všesměrové vysílání** (jen pro tuto síť, neprochází přes routery) - 255.255.255.255.
- **Loopback address** - pro adresování sebe sama - 127.0.0.1.
- **Privátní síť** - RFC 1918. Nesmí se objevit v Internetu:
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16

## Úkol:

Jaká je numericky nejvyšší adresa v bloku 172.16.0.0/12?



# Speciální IPv4 adresy

- **Moje adresa** - nezná-li odesílatel svoji adresu - 0.0.0.0.
- **Omezené všesměrové vysílání** (jen pro tuto síť, neprochází přes routery) - 255.255.255.255.
- **Loopback address** - pro adresování sebe sama - 127.0.0.1.
- **Privátní síť** - RFC 1918. Nesmí se objevit v Internetu:
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16



## Úkol:

Jaká je numericky nejvyšší adresa v bloku 172.16.0.0/12?

# Internet Protocol

- **Datagram** (packet) – balík dat omezené velikosti.
- **Směrování** – každý packet je směrován nezávisle na ostatních.
- **Nespolehlivost** – datagram může dojít vícekrát nebo se ztratit. datagramy mohou změnit pořadí.
- **Fragmentace packetů** – při průchodu do sítě s menší maximální velikostí datagramu (MTU).

# Formát IP packetu - I.

- Verze protokolu - 4 bity.
- Header length - 4 bity, značí počet 32-bitových jednotek.
- Type of service - 3 bity priorita, 1 bit latence, 1 bit propustnost, 1 bit bezeztrátovost. Zarovnáno na 8 bitů.
- Total length - 16 bitů.
- Identification - 16 bitů - k fragmentaci.
- Flags - 3 bity - Don't fragment, More fragments, 1 bit rezervovaný (ale: RFC 3514)
- Fragment offset - 13 bitů, značí násobek osmi bajtů.

## Úkol:

Nastudujte RFC 3514 ^\_~

# Formát IP packetu - I.

- Verze protokolu - 4 bity.
- Header length - 4 bity, značí počet 32-bitových jednotek.
- Type of service - 3 bity priorita, 1 bit latence, 1 bit propustnost, 1 bit bezeztrátovost. Zarovnáno na 8 bitů.
- Total length - 16 bitů.
- Identification - 16 bitů - k fragmentaci.
- Flags - 3 bity - Don't fragment, More fragments, 1 bit rezervovaný (ale: RFC 3514)
- Fragment offset - 13 bitů, značí násobek osmi bajtů.



## Úkol:

Nastudujte RFC 3514 ^\_~

# Formát IP packetu - II.

- **Time to live** - TTL, 8 bitů. Životnost datagramu. Každý směrovač sníží aspoň o 1.
- **Protocol** - 8 bitů. Označení vyšší vrstvy, které datagram patří.
- **Header checksum** - 16 bitů.
- **Source IP address** - 32 bitů.
- **Destination IP address** - 32 bitů.
- **Options** - délka je násobek 32 bitů.
- **Data** - délka je násobek 32 bitů, max. 65536 bajtů.

# Fragmentace paketů

- Prochází-li datagram do sítě s menší MTU.
- Minimální MTU je 576 bajtů.
- Všechny fragmenty mají stejné *identification*.
- Fragmenty mají *more fragments* flag nastaven na 1.
- Poslední fragment a nefragmentované packety mají tento flag nastavený na 0.
- Fragmenty (kromě prvního) mají nenulový *fragment offset*.

## Otázka:

Jak se pozná první fragment packetu?

## Úkol:

Promyslete, jestli fragmenty mohou být dále fragmentovány.

# Fragmentace packetů

- Prochází-li datagram do sítě s menší MTU.
- Minimální MTU je 576 bajtů.
- Všechny fragmenty mají stejné *identification*.
- Fragmenty mají *more fragments* flag nastaven na 1.
- Poslední fragment a nefragmentované packety mají tento flag nastavený na 0.
- Fragmenty (kromě prvního) mají nenulový *fragment offset*.



## Otázka:

Jak se pozná první fragment packetu?

## Úkol:

Promyslete, jestli fragmenty mohou být dále fragmentovány.

# Fragmentace paketů

- Prochází-li datagram do sítě s menší MTU.
- Minimální MTU je 576 bajtů.
- Všechny fragmenty mají stejné *identification*.
- Fragmenty mají *more fragments* flag nastaven na 1.
- Poslední fragment a nefragmentované pakety mají tento flag nastavený na 0.
- Fragmenty (kromě prvního) mají nenulový *fragment offset*.



## Otázka:

Jak se pozná první fragment packetu?



## Úkol:

Promyslete, jestli fragmenty mohou být dále fragmentovány.



# Fragmentace - poznámky

- **Znovusestavení datagramu** - smí provádět pouze cílový počítač. Někdy je též prováděno hraničním routerem nebo firewallem privátní sítě.
- **Path MTU discovery** - zjištění MTU cesty - posílá se nefragmentovatelný packet, sledují se odpovědi.

## Úkol:

V jakém pořadí je výhodné odesílat právě vygenerované fragmenty (a proč)?

## Čtení na dobrou noc

So you think you understand IP fragmentation?  
<https://lwn.net/Articles/960913/>

# Fragmentace - poznámky

- Znovusestavení datagramu - smí provádět pouze cílový počítač. Někdy je též prováděno hraničním routerem nebo firewallem privátní sítě.
- Path MTU discovery - zjištění MTU cesty - posílá se nefragmentovatelný packet, sledují se odpovědi.



## Úkol:

V jakém pořadí je výhodné odesílat právě vygenerované fragmenty (a proč)?

Čtení na dobrou noc

So you think you understand IP fragmentation?  
<https://lwn.net/Articles/960913/>

# Fragmentace - poznámky

- **Znovusestavení datagramu** - smí provádět pouze cílový počítač. Někdy je též prováděno hraničním routerem nebo firewallem privátní sítě.
- **Path MTU discovery** - zjištění MTU cesty - posílá se nefragmentovatelný packet, sledují se odpovědi.



## Úkol:

V jakém pořadí je výhodné odesílat právě vygenerované fragmenty (a proč)?



## Čtení na dobrou noc

So you think you understand IP fragmentation?  
<https://lwn.net/Articles/960913/>

# Volitelné položky v IP datagramu

- **Record route** – trasování cesty datagramu.
- **Timestamp** – trasování s časovým razítkem.
- **Loose source route** – minimální cesta datagramu.
- **Strict source route** – úplná cesta datagramu.
- **Security** – stupeň utajení datagramu.

# IPv4 nad Ethernetem

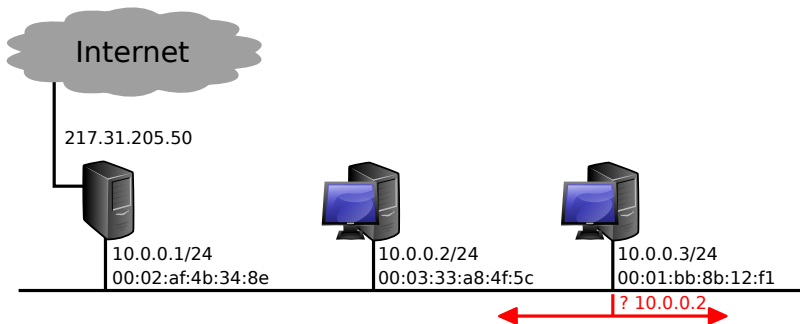
Formát rámce Ethernet v2:

- **Cílová adresa** - 6 bajtů.
- **Zdrojová adresa** - 6 bajtů.
- **Typ** (protokol vyšší vrstvy) - 2 bajty.
- **Data** - 46-1500 bajtů.
- **CRC** - 4 bajty.

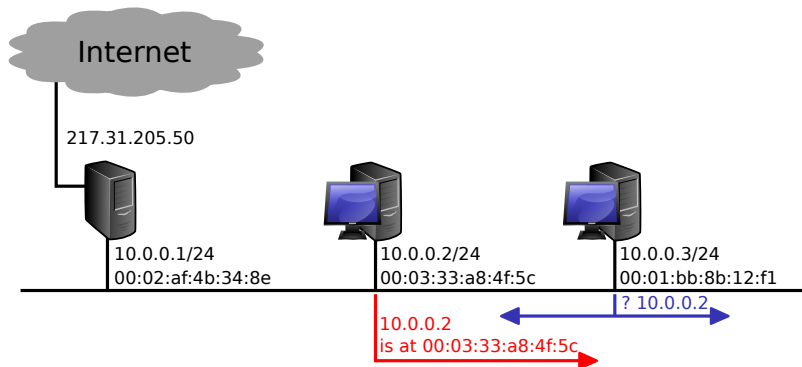
# ARP/RARP - formát packetů

- Převod L2 adres na L3 adresy a naopak
- HW type - 2 bajty (1 = Ethernet).
- Protocol type - 2 bajty (0x0800 = IP).
- HW length - 1 bajt (Ethernet = 6).
- Protocol length - 1 bajt (IP = 4).
- Operation - 2 bajty (1 = ARP request, 2 = ARP response, 3 = RARP request, 4 = RARP response).
- Sender layer 2 addr
- Sender layer 3 addr
- Target layer 2 addr
- Target layer 3 addr

# 📄 Příklad: Použití ARP

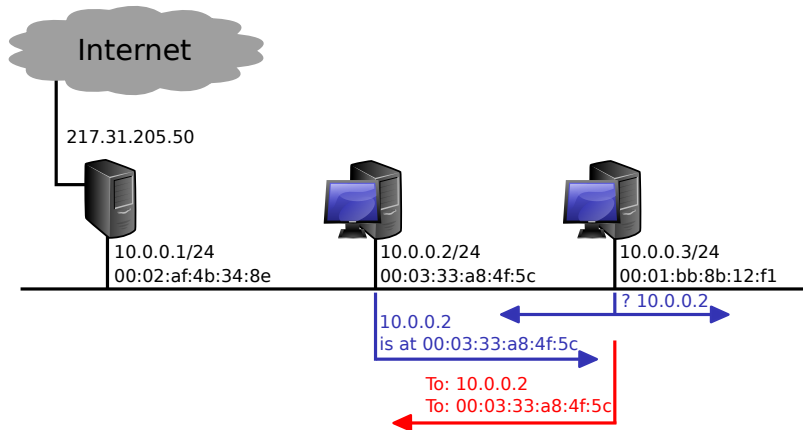


# ■ Příklad: Použití ARP

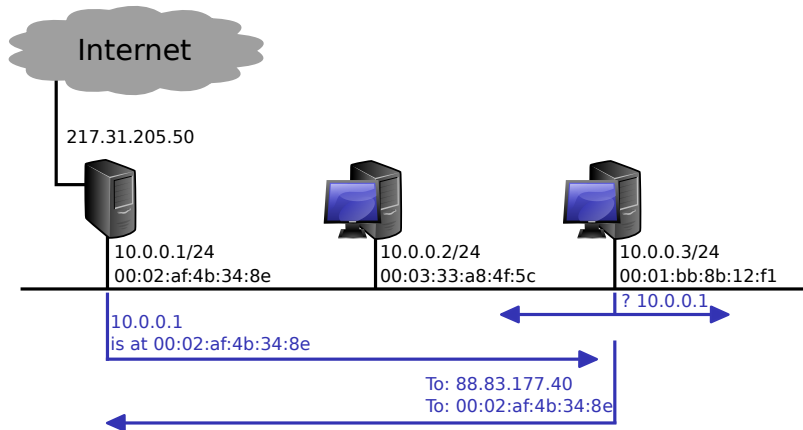




# Příklad: Použití ARP



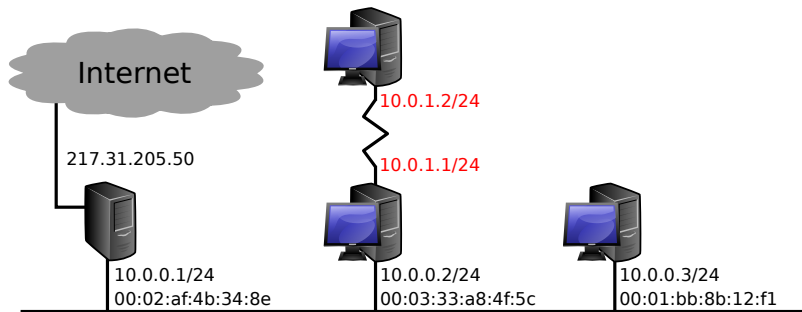
# ■ Příklad: ARP u nelokální adresy



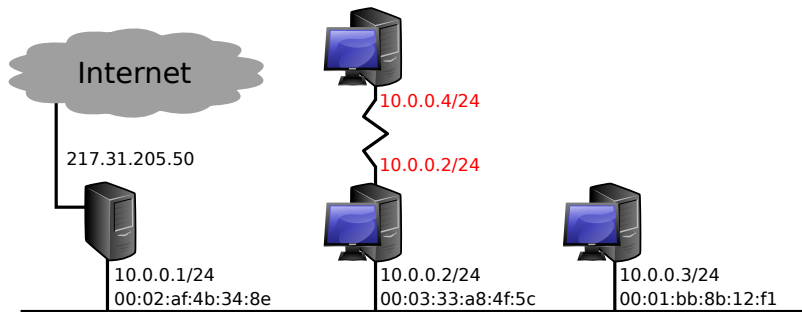
# ARP - poznámky

- **ARP cache** - není třeba ARP dotaz s každým packetem.
- **Proxy ARP** - náhrada/zjednodušení směrování.

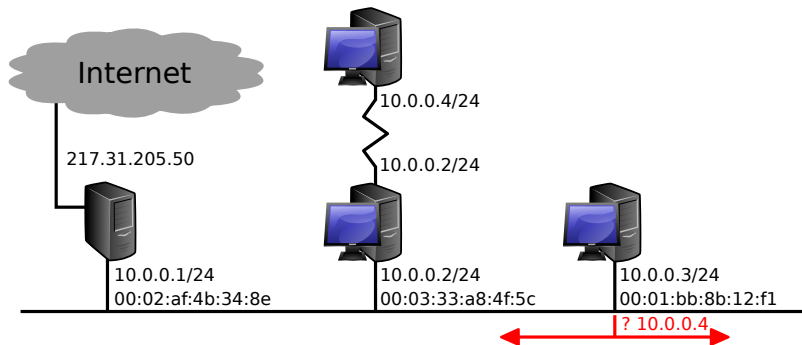
# 📄 Příklad: Proxy ARP



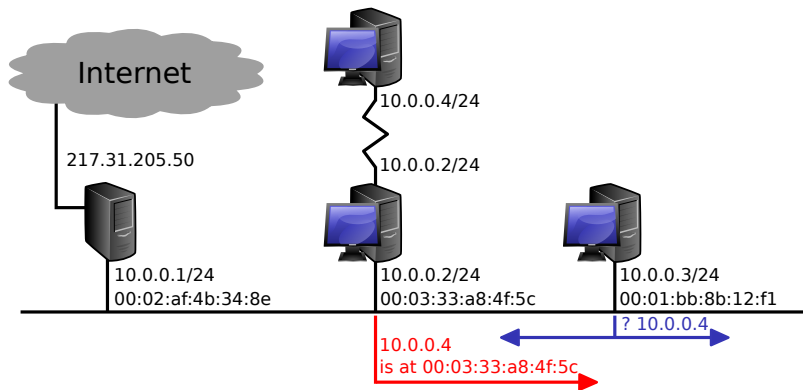
# 📄 Příklad: Proxy ARP



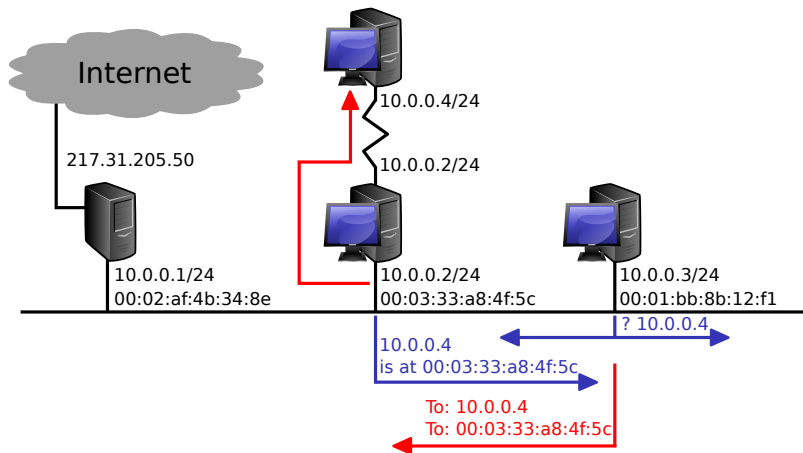
# 📄 Příklad: Proxy ARP



# ■ Příklad: Proxy ARP



# ■ Příklad: Proxy ARP





# IPv6 - Motivace

- Větší adresní prostor.
- Mobilita - práce ve více sítích, přechod mezi sítěmi za běhu aplikací, domovský agent.
- Zabezpečení - šifrované a podepisované packety - protokol IPSEC.
- Autokonfigurace - zjištění informací o síti přímo ze sítě.
- Způsoby přenosu - unicast, multicast, anycast.
- Více IPv6 adres na jedno rozhraní.

# Adresace v IPv6

- 128-bitová adresa
- Zápis - čtveřice šestnáctkových čísel.
- Příklad:  
3ffe:ffff:0000:f101:0210:a4ff:fee3:9562
- Vypuštění úvodních nul:  
3ffe:ffff:0:f101:210:a4ff:fee3:9562
- Vypuštění sekvence 0000:  
3ffe:ffff::f101:210:a4ff:fee3:9562.
- Prefix - podobně jako v IPv4 (např.:  
3ffe:ffff::12/64).

## Otázka:

Která IPv6 adresa je zapsatelná na nejméně znaků?

# Adresace v IPv6

- 128-bitová adresa
- Zápis - čtveřice šestnáctkových čísel.
- Příklad:  
3ffe:ffff:0000:f101:0210:a4ff:fee3:9562
- Vypuštění úvodních nul:  
3ffe:ffff:0:f101:210:a4ff:fee3:9562
- Vypuštění sekvence 0000:  
3ffe:ffff::f101:210:a4ff:fee3:9562.
- Prefix - podobně jako v IPv4 (např.:  
3ffe:ffff::12/64).



## Otázka:

Která IPv6 adresa je zapsatelná na nejméně znaků?

# Formát IPv6 packetů

- **Hlavička pevné délky**, řetězení hlaviček.
- **Verze protokolu** – 4 bity, hodnota vždy 6.
- **Priorita** – 8 bitů, třída provozu.
- **Identifikace toku** – 20 bitů.
- **Délka packetu** – 16 bitů.
- **Next header** – 8 bitů (identifikace další hlavičky, např. vyšší vrstvy).
- **Hop limit** – 8 bitů (ekvivalent TTL u IPv4).
- **Zdrojová adresa** – 128 bitů.
- **Cílová adresa** – 128 bitů.

# Speciální IPv6 adresy

- Loopback - `::1` (ekvivalent `127.0.0.1` v IPv4).
- Nespecifikovaná adresa - `::` (ekvivalent `0.0.0.0` v IPv4).
- Lokální adresa linky - `fe80::/10`
- Adresy pro příklady - `3ffe:ffff::/32`.
- IPv4 kompatibilní adresy - `::/96`.
- IPv4 mapované adresy - `::ffff:0:0/96`.

# Interpretace prefixu adresy



## Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

# Interpretace prefixu adresy



## Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

- `10` nejvyšších bitů stejných, zbytek jedničky

# Interpretace prefixu adresy



## Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

- 10 nejvyšších bitů stejných, zbytek jedničky
- `fe?f:ffff:ffff:ffff:ffff:ffff:ffff:ffff`



# Interpretace prefixu adresy

## ? Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

- 10 nejvyšších bitů stejných, zbytek jedničky
- `fe?f:ffff:ffff:ffff:ffff:ffff:ffff:ffff`
- ? = 8, z toho 2 nejvyšší bity zachovat, zbytek 1

# Interpretace prefixu adresy



## Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

- 10 nejvyšších bitů stejných, zbytek jedničky
- `fe?f:ffff:ffff:ffff:ffff:ffff:ffff:ffff`
- ? = 8, z toho 2 nejvyšší bity zachovat, zbytek 1
- 8 = `1000`, z toho 2 nejvyšší bity zachovat, zbytek 1

# Interpretace prefixu adresy



## Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

- 10 nejvyšších bitů stejných, zbytek jedničky
- `fe?f:ffff:ffff:ffff:ffff:ffff:ffff:ffff`
- ? = 8, z toho 2 nejvyšší bity zachovat, zbytek 1
- 8 = `1000`, z toho 2 nejvyšší bity zachovat, zbytek 1
- `1011` = b

# Interpretace prefixu adresy

## ? Úkol:

Link-local adresy jsou určeny prefixem `fe80::/10`. Jaká je numericky nejvyšší link-local adresa?

- 10 nejvyšších bitů stejných, zbytek jedničky
- `fe?f:ffff:ffff:ffff:ffff:ffff:ffff:ffff`
- ? = 8, z toho 2 nejvyšší bity zachovat, zbytek 1
- 8 = `1000`, z toho 2 nejvyšší bity zachovat, zbytek 1
- `1011` = b
- `febf:ffff:ffff:ffff:ffff:ffff:ffff:ffff`

# Privátní IPv6 adresy

- Site-local address – fec0::/10
  - ekvivalent privátních IPv4 adres dle RFC1918
  - nyní zastaralé (RFC 3879)
- Unique Local Address – RFC 4193.
  - prefix fc00::/7
  - 8. bit = 1 pro lokální přidělení, 0 pro globální.
  - 40 bitů: globální ID
  - 80 bitů: pro lokální přidělování (65 536 sítí /64)

# Privátní IPv6 adresy

- **Site-local address** - fec0::/10
  - ekvivalent privátních IPv4 adres dle RFC1918
  - nyní zastaralé (RFC 3879)
- **Unique Local Address** - RFC 4193.
  - prefix fc00::/7
  - 8. bit = 1 pro lokální přidělení, 0 pro globální.
  - 40 bitů: globální ID
  - 80 bitů: pro lokální přidělování (65 536 sítí /64)

# EUI-64 formát adresy

- EUI-64 formát adresy - lokální část se odvozuje z fyzické adresy.
- EUI-64 pro ethernet - MAC adresa, uprostřed vloženo fffe, 7. nejvyšší bit nastaven na 1 pro globálně přidělený identifikátor (MAC adresu) RFC 4291, sekce 2.5.1.

## Příklad: EUI-64

MAC adresa 00:D0:B7:6B:4A:B2

Prefix sítě fe80::/10

EUI-64 adresa je fe80::2d0:b7ff:fe6b:4ab2

- Autokonfigurace - směrovač vysílá *router advertisement*, kde je uveden /64 prefix lokální sítě. Viz též radvd(8).

# EUI-64 formát adresy

- **EUI-64 formát adresy** - lokální část se odvozuje z fyzické adresy.
- **EUI-64 pro ethernet** - MAC adresa, uprostřed vloženo fffe, 7. nejvyšší bit nastaven na 1 pro globálně přidělený identifikátor (MAC adresu) RFC 4291, sekce 2.5.1.

## **Příklad: EUI-64**

MAC adresa 00:D0:B7:6B:4A:B2

Prefix sítě fe80::/10

EUI-64 adresa je fe80::2d0:b7ff:fe6b:4ab2

- **Autokonfigurace** - směrovač vysílá *router advertisement*, kde je uveden /64 prefix lokální sítě. Viz též radvd(8).



# Výpočet EUI-64 adresy



## Úkol:

Jaká by byla EUI-64 adresa počítače `aisa.fi.muni.cz` (MAC adresa `00:25:B3:D7:D0:6A`) je-li adresní prefix `2001:718:801:230::/64`?

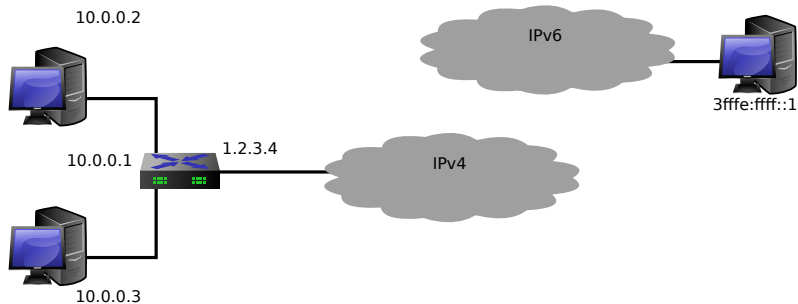
# Specifické vlastnosti IPv6

- **Fragmentace packetů** – na směrovačích není. Vysílající musí dělat *path MTU discovery*. Fragmentace popsána v samostatné *next header*.
- **Spolupráce s linkovou vrstvou** – NDP (neighbour discovery protocol). Náhrada ARP. Zjišťování adresního prefixu sítě, směrovacích informací, atd.

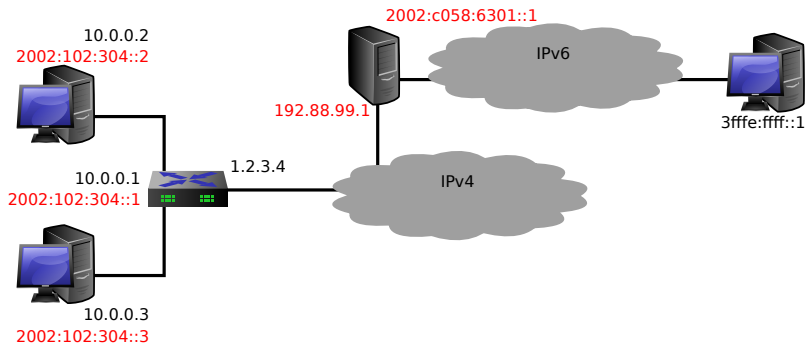
# Přechodové mechanismy

- **Dual stack** – podpora IPv4 a IPv6 v jednom počítači.
- **Tunelování** – zapouzdření IPv6 paketů do IPv4 (protokol SIT, 41).
- **Autotunelování** – (6to4):
  - adresy 2002:xxxx:yyyy:/48 vytvořené z IPv4 adresy
  - schování bloku /48 IPv6 adres za jednu IPv4 adresu
  - komunikace do nativního IPv6 internetu přes 6to4 relay
  - adresa relay: 192.88.99.1 (2002:c058:6301::).
- **6rd** – spolupráce ISP, nemá-li klient veřejnou IPv4 adresu.

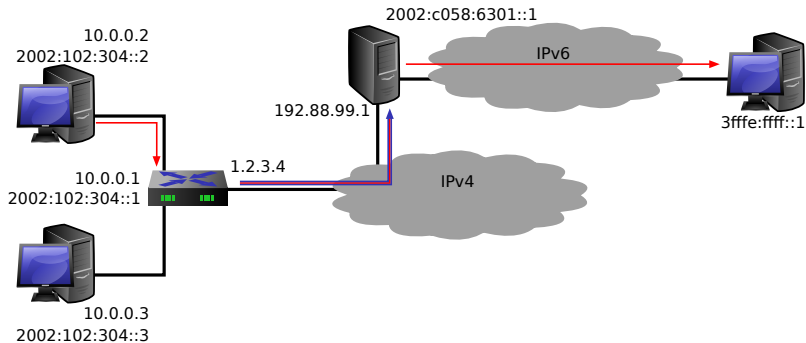
# Autotunelování 6to4



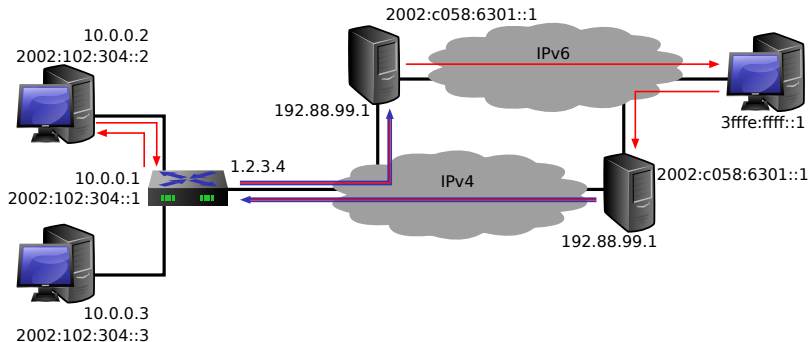
# Autotunelování 6to4



# Autotunelování 6to4



# Autotunelování 6to4



# Odkazy

- **Linux and IPv6 howto** - <http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/>
- **Pavel Satrapa: IPv6** - <https://knihy.nic.cz/files/edice/IPv6-2019.pdf>

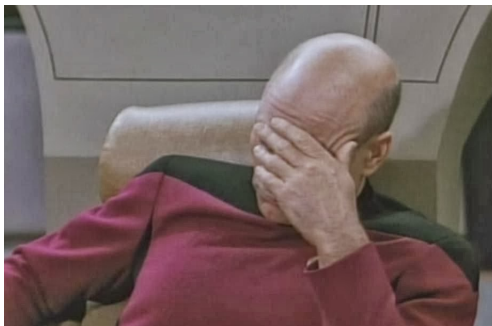


# Čtení na dobrou noc



## Zamyšlení s odstupem

The world in which IPv6 was a good design  
<https://apenwarr.ca/log/20170810>



# ICMP

- Internet Control Message Protocol.
- Chybové a řídicí zprávy IPv4.
- ICMP zpráva - uvnitř IPv4 datagramu.
- Podává se původnímu odesílateli příslušného datagramu.
- RFC 792.

# Formát ICMP zpráv

- **Type** – 8 bitů – typ ICMP zprávy.
- **Code** – 8 bitů – přídatný kód.
- **Checksum** – 16 bitů – kontrolní součet ICMP zprávy.

# ICMP Echo Request/Echo

- Pro testování dostupnosti počítače.
- Každý počítač v síti je povinen na ICMP echo request (*type* = 8) odpovědět ICMP echo (*type* = 0) se stejnou datovou částí (RFC 1122, sekce 3.2.2.6).
- Další položky: 16 bitů identifikace, 16 bitů číslo sekvence.

## Otázka:

K čemu je vlastně pole „identifikace“?

# ICMP Echo Request/Echo

- Pro testování dostupnosti počítače.
- Každý počítač v síti je povinen na ICMP echo request (*type* = 8) odpovědět ICMP echo (*type* = 0) se stejnou datovou částí (RFC 1122, sekce 3.2.2.6).
- Další položky: 16 bitů identifikace, 16 bitů číslo sekvence.



## Otázka:

K čemu je vlastně pole „identifikace“?

# Neodsažitelnost adresáta

- ICMP destination unreachable - type 3.
- Code obsahuje bližší informace:
  - Network unreachable
  - Host unreachable
  - Protocol unreachable
  - Port unreachable
  - Fragmentation needed
  - Source route failed
- Data - IP hlavička a prvních 64 bitů původního packetu.

## Otázka:

Kdy přesně se posílá host unreachable?

# Neodsažitelnost adresáta

- ICMP destination unreachable - type 3.
- Code obsahuje bližší informace:
  - Network unreachable
  - Host unreachable
  - Protocol unreachable
  - Port unreachable
  - Fragmentation needed
  - Source route failed
- Data - IP hlavička a prvních 64 bitů původního packetu.



## Otázka:

Kdy přesně se posílá **host unreachable**?

# Zahlčení routeru

- ICMP source quench
- Type = 4, code = 0, dále hlavička a 64 bitů ze začátku IP datagramu.
- Vysílající strana má reagovat snížením toku dat.
- Nemusí znamenat zahození datagramu.



Protokoly vyšších vrstev mají svoje řízení toku. Source quench téměř vždy znamená problém v konfiguraci sítě.



# Přesměrování datagramů

- ICMP redirect - type = 5.
- Žádost o přesměrování dalších packetů na jiný router.
- Code určuje typ přesměrování:
  - Redirect for the network
  - Redirect for the host
  - Redirect for the ToS and network
  - Redirect for the ToS and host
- Data - adresa nového routeru, IP hlavička a prvních 64 bitů dat IP datagramu.

## Úkol:

Jak pozná router, že má poslat ICMP redirect?

# Přesměrování datagramů

- ICMP redirect - type = 5.
- Žádost o přesměrování dalších packetů na jiný router.
- Code určuje typ přesměrování:
  - Redirect for the network
  - Redirect for the host
  - Redirect for the ToS and network
  - Redirect for the ToS and host
- Data - adresa nového routeru, IP hlavička a prvních 64 bitů dat IP datagramu.



## Úkol:

Jak pozná router, že má poslat ICMP redirect?

# Překročení času

- ICMP time limit exceeded - type = 11.
- Code:
  - TTL dosáhlo nuly
  - Překročen čas znovusestavení z fragmentů
- Data - IP hlavička a prvních 64 bitů dat IP datagramu.

# Chyba v datagramu

- ICMP parameter problem - type = 12.
- Code:
  - 0: položka „pointer“ není platná.
  - 1: položka „pointer“ je platná.
- Pointer - 8 bitů, zarovnáno na 32 bitů. Odkaz na místo, kde je problém.
- Data - IP hlavička a prvních 64 bitů dat IP datagramu.

# Další ICMP zprávy

- Timestamp request (type = 13).
- Timestamp reply (type = 14).
- Information request (type = 15) - žádost o adresu sítě.
- Information reply (type = 16).
- Address mask request (type = 17).
- Address mask reply (type = 18).

# ICMPv6

- Analogie ICMP pro IPv6
- Protokol (next header) číslo 58

Formát packetů:

- **Type** – 8 bitů, nejvyšší bit odlišuje chybové a informativní zprávy.
- **Code** – 8 bitů, bližší určení.
- **Checksum** – 16 bitů.

# Chybové zprávy ICMPv6

- 1 Destination unreachable.
- 2 Packet too big.
- 3 Time exceeded.
- 4 Parameter problem.

# Informativní zprávy - I.

- 128 Echo request.
- 129 Echo reply.
- 130 Group Membership Query.
- 131 Group Membership Report.
- 132 Group Membership Reduction.
- 133 Router Solicitation.
- 134 Router Advertisement.
- 135 Neighbor Solicitation.
- 136 Neighbor Advertisement.
- 137 Redirect.
- 138 Router Renumbering.



# Informativní zprávy - II.

- 139 ICMP Node Information Query.
- 140 ICMP Node Information Response.
- 141 Inverse Neighbor Discovery Solicitation Message.
- 142 Inverse Neighbor Discovery Advertisement Message.
- 143 MLDv2 Multicast Listener Report.
- 144 Home Agent Address Discovery Request Message.
- 145 Home Agent Address Discovery Reply Message
- 146 Mobile Prefix Solicitation.
- 147 Mobile Prefix Advertisement

# Informativní zprávy - III.

- 148 Certification Path Solicitation.
- 149 Certification Path Advertisement.
- 150 Experimental mobility protocols.
- 151 Multicast Router Advertisement.
- 152 Multicast Router Solicitation.
- 153 Multicast Router Termination.



## Neighbour dicoverly versus ARP

Multicast versus broadcast.

# Protokol UDP

- User Datagram Protocol
- Nespojovaná transportní služba
- **Porty** - rozlišení mezi více adresáty (a zdroji) v rámci jednoho počítače. 16-bitové celé číslo.
- **Well-known ports** - porty, na kterých lze očekávat obecně známé služby.

# Formát UDP rámce

- Source port – 16 bitů.
- Destination port – 16 bitů.
- Length – 16 bitů.
- Checksum – 16 bitů. Nepovinný. Součet UDP hlavičky a IP pseudohlavičky.

## IP pseudohlavička

- Zdrojová IP adresa, cílová IP adresa.
- Zarovnání – 8 nulových bitů.
- Protokol – 8 bitů.
- Délka packetu – 16 bitů.

# Formát UDP rámce

- Source port – 16 bitů.
- Destination port – 16 bitů.
- Length – 16 bitů.
- Checksum – 16 bitů. Nepovinný. Součet UDP hlavičky a IP pseudohlavičky.



## IP pseudohlavička

- Zdrojová IP adresa, cílová IP adresa.
- Zarovnání – 8 nulových bitů.
- Protokol – 8 bitů.
- Délka packetu – 16 bitů.

# Protokol TCP

- Transmission Control Protocol
- Spojovaná služba
- Spolehlivá služba
- Duplexní proud dat
- Buffering
- Porty – podobně jako v UDP

# Garance protokolu TCP

- Správné pořadí datagramů.
- Duplicitní datagramy jsou vyřazeny.
- Potvrzování přenosu dat.
- Opakování přenosu, nedojde-li potvrzení.

# TCP spojení

- Klient-server
- Server – *poslouchá* (listen) na portu a *přijme* (accept) spojení.
- Klient – *spojuje* se (connect) na port cílového stroje. Může navazovat i z konkrétního zdrojového portu.
- Navázání spojení – *three-way handshake*

## Identifikace TCP spojení

Celá čtveřice:

- Zdrojová IP adresa, TCP port
- Cílová IP adresa, TCP port



# TCP spojení

- Klient-server
- **Server** – *poslouchá* (listen) na portu a *přijme* (accept) spojení.
- Klient – *spojuje* se (connect) na port cílového stroje. Může navazovat i z konkrétního zdrojového portu.
- Navázání spojení – *three-way handshake*

## Identifikace TCP spojení

Celá čtveřice:

- Zdrojová IP adresa, TCP port
- Cílová IP adresa, TCP port

# TCP spojení

- Klient-server
- Server – *poslouchá* (listen) na portu a *přijme* (accept) spojení.
- **Klient** – *spojuje* se (connect) na port cílového stroje. Může navazovat i z *konkrétního zdrojového portu*.
- Navázání spojení – *three-way handshake*

## Identifikace TCP spojení

Celá čtveřice:

- Zdrojová IP adresa, TCP port
- Cílová IP adresa, TCP port

# TCP spojení

- Klient-server
- Server – *poslouchá* (listen) na portu a *přijme* (accept) spojení.
- Klient – *spojuje* se (connect) na port cílového stroje. Může navazovat i z *konkrétního zdrojového portu*.
- **Navázání spojení** – *three-way handshake*

## Identifikace TCP spojení

Celá čtveřice:

- Zdrojová IP adresa, TCP port
- Cílová IP adresa, TCP port

# TCP spojení

- Klient-server
- Server – *poslouchá* (listen) na portu a *přijme* (accept) spojení.
- Klient – *spojuje* se (connect) na port cílového stroje. Může navazovat i z *konkrétního zdrojového portu*.
- Navázání spojení – *three-way handshake*



## Identifikace TCP spojení

**Celá** čtveřice:

- Zdrojová IP adresa, TCP port
- Cílová IP adresa, TCP port

# Vytvoření TCP spojení



## Three-way handshake

- Klient → Server: **SYN (seq = x)**
  - Server → Klient: SYN-ACK (seq = y, ack = x+1)
  - Klient → Server: ACK (ack = y+1)
- 
- Číslo sekvence - 32 bitů
  - Pořadí dat rámcí v TCP spojení
  - **Bezpečnost!**

# Vytvoření TCP spojení



## Three-way handshake

- Klient → Server: SYN (seq = x)
  - Server → Klient: **SYN-ACK (seq = y, ack = x+1)**
  - Klient → Server: ACK (ack = y+1)
- 
- Číslo sekvence - 32 bitů
  - Pořadí dat rámcí v TCP spojení
  - **Bezpečnost!**

# Vytvoření TCP spojení



## Three-way handshake

- Klient → Server: SYN (seq = x)
  - Server → Klient: SYN-ACK (seq = y, ack = x+1)
  - Klient → Server: **ACK (ack = y+1)**
- 
- Číslo sekvence - 32 bitů
  - Pořadí dat rámcí v TCP spojení
  - **Bezpečnost!**

# Vytvoření TCP spojení



## Three-way handshake

- Klient → Server: SYN (seq = x)
  - Server → Klient: SYN-ACK (seq = y, ack = x+1)
  - Klient → Server: ACK (ack = y+1)
- 
- Číslo sekvence - 32 bitů
  - Pořadí dat rámcí v TCP spojení
  - Bezpečnost!



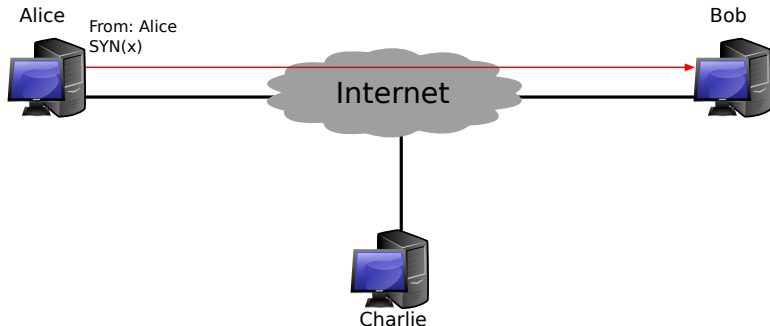
# Vytvoření TCP spojení



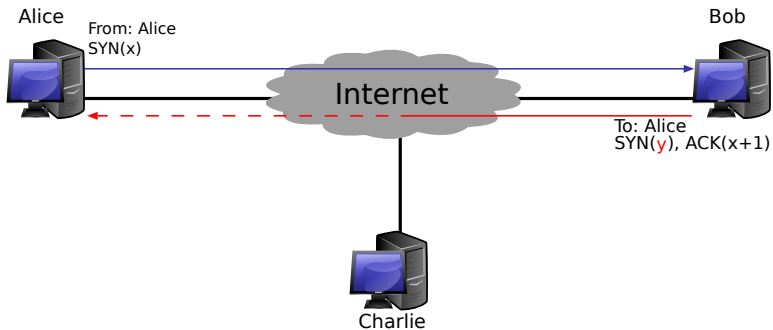
## Three-way handshake

- Klient → Server: SYN (seq = x)
  - Server → Klient: SYN-ACK (seq = y, ack = x+1)
  - Klient → Server: ACK (ack = y+1)
- 
- Číslo sekvence - 32 bitů
  - Pořadí dat rámcí v TCP spojení
  - **Bezpečnost!**

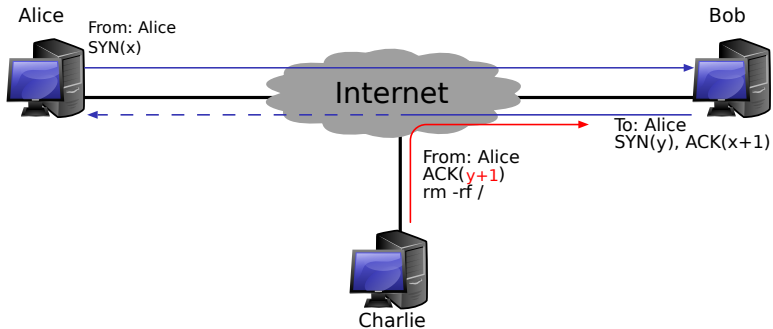
# Útok na sekvenční čísla



# Útok na sekvenční čísla



# Útok na sekvenční čísla

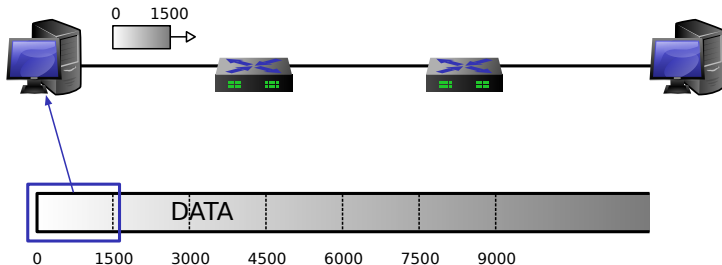


# Uzavření TCP spojení

- Uzavření spojení – každý směr zvlášť nebo simultánně.
- Ukončovací rámec – příznak FIN.
- Robustnost – i FIN se potvrzuje.

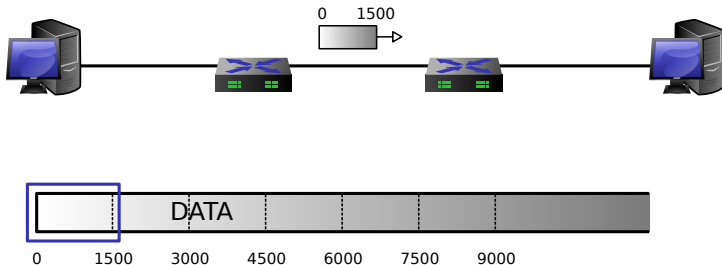
# Potvrzování packetů

- **Problém:** potvrzovat každý packet?



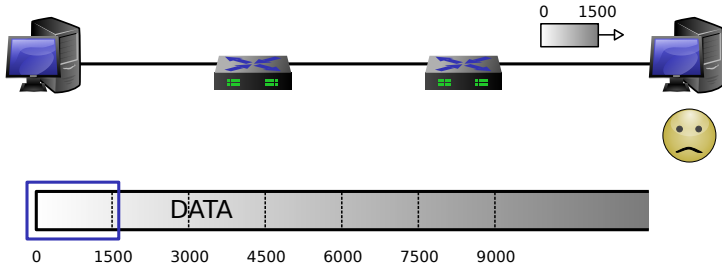
# Potvrzování packetů

- **Problém:** potvrzovat každý packet?



# Potvrzování packetů

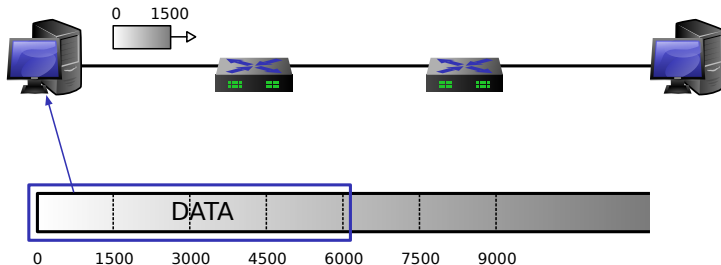
- **Problém:** potvrzovat každý packet?





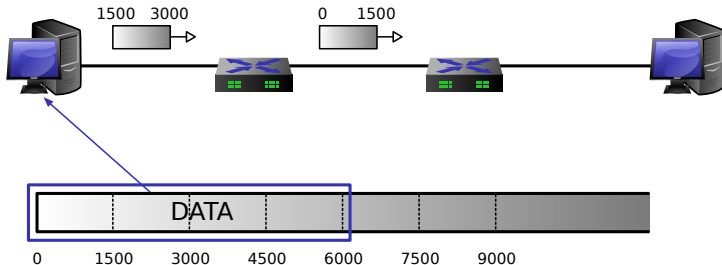
# Klouzající okno

- Velikost - dohodnuta při vytváření spojení.



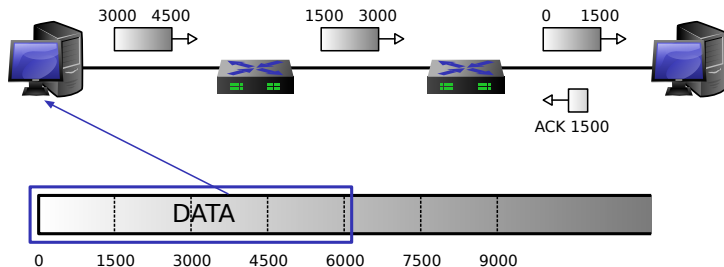
# Klouzající okno

- Velikost - dohodnuta při vytváření spojení.



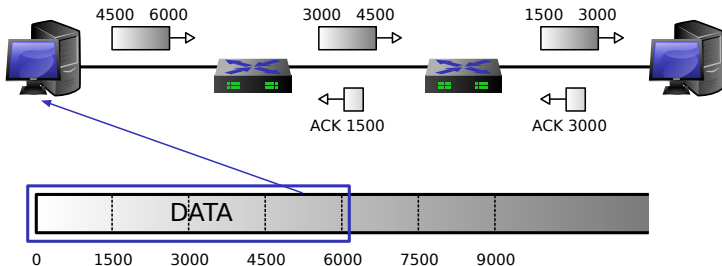
# Klouzající okno

- Velikost - dohodnuta při vytváření spojení.



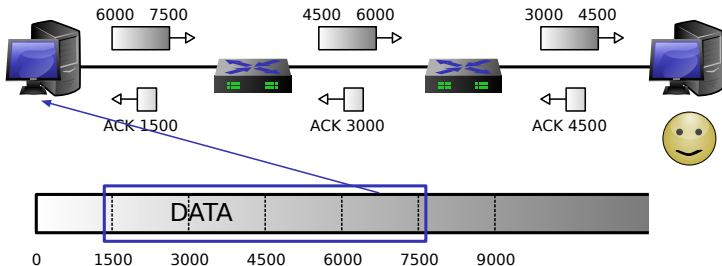
# Klouzající okno

- Velikost - dohodnuta při vytváření spojení.



# Klouzající okno

- Velikost - dohodnuta při vytváření spojení.



# Vlastnosti TCP

- **Potvrzování** – ACK packet, piggybacking při duplexním spojení.
- **Změna velikosti okna** – každý ACK packet obsahuje počet slov, který je druhá strana ochotna přijmout.
- **Exponential back-off** – v případě ztráty datagramu.
- **Out-of-band data** – urgentní data, zasílaná mimo pořadí.

# Formát TCP rámce

- Source port – 16 bitů.
- Destination port – 16 bitů.
- Sequence number – 32 bitů.
- Acknowledgement number – 32 bitů.
- Header length – 4 bity – počet 32-bitových slov, velikost hlavičky včetně volitelných položek.
- Window – 16 bitů. Počet slov, které je odesílatel schopen přijmout.
- Checksum – 16 bitů – kontrolní součet včetně pseudozáhlaví (viz UDP).
- Urgent pointer – poslední bajt urgentních dat.
- Options – zarovnáno na 32 bitů – volitelné položky (např. MSS, SACK, MD5, ...).

# Příznaky TCP rámce

- **URG** – 1 bit. Doručit tento segment co nejrychleji. Položka *Urgent pointer* je platná.
- **SYN** – 1 bit. Synchronizace sekvenčních čísel; žádost o zřízení spojení.
- **ACK** – 1 bit. Položka *Acknowledgement number* je platná.
- **RST** – 1 bit. Požadavek na reset spojení („Connection reset by peer“) – posílá se jako odpověď na packet bez SYN flagu, který nepřísluší žádnému existujícímu spojení.
- **PSH** – 1 bit (push). Požadavek na rychlé doručení tohoto segmentu vyšší vrstvě sítě.
- **FIN** – 1 bit. Ukončení spojení – odesílatel vyslal všechna data.



# Bufferbloat

- Jak TCP řídí tok dat?
  - Slow start, exponential back-off.
  - <https://vimeo.com/14439742>
- K čemu jsou buffery v routerech?
  - <http://guido.appenzeller.net/animations/>

# Bufferbloat

- Jak TCP řídí tok dat?
  - Slow start, exponential back-off.
  - <https://vimeo.com/14439742>
- K čemu jsou buffery v routerech?
  - <http://guido.appenzeller.net/animations/>

# SCTP

- Stream Control Transmission Protocol
- Spojovaný protokol
- Více datových proudů v jednom spojení
- Multihoming
- Multipath
- Zachování hranice zpráv
- Volitelné zachování pořadí
- Spolehlivý přenos uspořádaných i neuspořádaných zpráv

# DCCP

- Datagram Congestion Control Protocol
- Spojovaný protokol
- Nespolehlivý
- Zachování hranice zpráv
- Congestion control

# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
- Přesun z kernelu do user-space?
- QUIC: nad UDP + TLS, více streamů

Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>

# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
- Přesun z kernelu do user-space?
- QUIC: nad UDP + TLS, více streamů

Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>

# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
  - Přesun z kernelu do user-space?
  - QUIC: nad UDP + TLS, více streamů

Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>

# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
- Přesun z kernelu do user-space?
- QUIC: nad UDP + TLS, více streamů

Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>



# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
- Přesun z kernelu do user-space?
- **QUIC**: nad UDP + TLS, více streamů

Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>

# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
- Přesun z kernelu do user-space?
- QUIC: nad UDP + TLS, více streamů

Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>

# Protocol Ossification

- NATy, middleboxy, firewally, ...
- TCP Early Congestion Notification
- Problematické nasazování nových protokolů
- Přesun z kernelu do user-space?
- QUIC: nad UDP + TLS, více streamů



## Čtení na dobrou noc

A QUIC Look at HTTP/3

<https://lwn.net/Articles/814522/>

# Programování síťových aplikací

- Několik druhů API
- BSD Sockets – de facto standard, nejpoužívanější.
- Streams – pochází z UNIXu System V.

# BSD Sockets API

- API pro meziprocesovou komunikaci - někdy nazývané **BSD IPC** (oproti **SysV IPC** = semaforey, fronty zpráv a sdílená paměť).
- **Nezávislé na síťovém protokolu** - je možné provozovat nad různými *rodinami protokolů*.

## Příklad: Rodiny protokolů v Linuxu

PF_UNIX	PF_IPX	PF_BRIDGE	PF_AX25
PF_INET	PF_APPLETALK	PF_NETROM	PF_NETLINK
PF_INET6	PF_AAL5	PF_X25	...

# BSD Sockets API

- API pro meziprocesovou komunikaci - někdy nazývané **BSD IPC** (oproti **SysV IPC** = semaforey, fronty zpráv a sdílená paměť).
- **Nezávislé na síťovém protokolu** - je možné provozovat nad různými *rodinami protokolů*.



## Příklad: Rodiny protokolů v Linuxu

PF_UNIX	PF_IPX	PF_BRIDGE	PF_AX25
PF_INET	PF_APPLETALK	PF_NETROM	PF_NETLINK
PF_INET6	PF_AAL5	PF_X25	...

# Co je socket?

- **Socket** = schránka, zásuvka
- Jeden konec síťového spojení
- Deskriptor
- **Rozšíření abstrakce souboru**
  - **Běžné souborové operace** - `read(2)`, `write(2)`, `close(2)`
  - **Speciální socketové služby** - `sendmsg(2)`, `recvmsg(2)`, ...

# Vytvoření socketu

## socket(2)

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int proto);
```

Vytvoří socket a vrátí jeho deskriptor.

**domain** - rodina adres - způsob komunikace přes socket. Odpovídá rodině protokolů. AF\_UNIX, AF\_INET, atd.

**type** - sémantika komunikace. Viz dále.

**proto** - protokol. Obvykle existuje pro každou kombinaci (domain, type) nejvýše jeden. Pak zde může být 0. Viz /etc/protocols.



# Typy socketů

Položka `type` určuje chování socketu a jeho schopnosti.

- `SOCK_STREAM` – plně duplexní spolehlivý uspořádaný proud dat, případně podporuje i posílání dat mimo pořadí (out-of-band data).
- `SOCK_DGRAM` – datagramová služba.
- `SOCK_RAW` – přímý přístup na síťové zařízení. Povoleno jen superuživateli.
- `SOCK_SEQPACKET` – uspořádané spolehlivé duplexní spojení pro přenos packetů do jisté maximální délky. Může být požadováno načtení packetu jedním `read(2)` nebo podobnou službou.

# Sockety jako roura

## socketpair(2)

## Dvojice socketů

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int socketpair(int domain, int type, int proto,
               int sd[2]);
```

- Nepojmenovaná dvojice propojených socketů.
- sd[0] a sd[1] jsou ekvivaletní.

## Úkol:

Vysvětlete rozdíl mezi sd[0], sd[1] ze `socketpair(2)` a pd[0], pd[1] z `pipe(2)`.

# Sockety jako roura

## socketpair(2)

## Dvojice socketů

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int socketpair(int domain, int type, int proto,
               int sd[2]);
```

- Nepojmenovaná dvojice propojených socketů.
- sd[0] a sd[1] jsou ekvivaletní.

### ? Úkol:

Vysvětlete rozdíl mezi sd[0], sd[1] ze `socketpair(2)` a pd[0], pd[1] z `pipe(2)`.

## Příklad: /etc/protocols

```
ip    0    IP    # internet protocol
icmp  1    ICMP  # internet control message p.
igmp  2    IGMP  # internet group multicast p.
ggp   3    GGP   # gateway-gateway p.
tcp   6    TCP   # transmission control p.
pup   12   PUP   # PARC universal packet p.
udp   17   UDP   # user datagram p.
raw   255  RAW   # RAW IP interface
```

# Tabulka protokolů

## getprotoent(3), getprotoby\*(2)

```
#include <netdb.h>
```

```
struct protoent *getprotoent();  
struct protoent *getprotobyname(char *name);  
struct protoent *getprotobynumber(int proto);  
void setprotoent(int stayopen);  
void endprotoent();
```

```
struct protoent {  
    char *p_name;  
    char **p_aliases;  
    int p_proto;  
}
```

# Tabulka protokolů

- Funkce **nejsou reentrantní**
- Reentrantní verze: `getprotoent_r(3)`

## Úkol:

Napište programy `getprotobyname` a `getprotobynumber`, které na standardní vstup vypíší číslo protokolu na základě jména a naopak.

## Vzorová řešení programovacích úkolů

<http://www.fi.muni.cz/~kas/pv077/>

# Tabulka protokolů

- Funkce **nejsou reentrantní**
- Reentrantní verze: `getprotoent_r(3)`



## Úkol:

Napište programy `getprotobyname` a `getprotobynumber`, které na standardní vstup vypíší číslo protokolu na základě jména a naopak.

Vzorová řešení programovacích úkolů

<http://www.fi.muni.cz/~kas/pv077/>

# Tabulka protokolů

- Funkce **nejsou reentrantní**
- Reentrantní verze: `getprotoent_r(3)`

## ? Úkol:

Napište programy `getprotobyname` a `getprotobynumber`, které na standardní vstup vypíší číslo protokolu na základě jména a naopak.

## i Vzorová řešení programovacích úkolů

<http://www.fi.muni.cz/~kas/pv077/>



# Adresace socketů

- Adresa socketu - „Jak se se socketem spojit?“
- Adresace - závisí na rodině protokolů.

## struct sockaddr      Obecná adresa socketu

```
#include <sys/socket.h>

struct sockaddr {
    sa_family_t  sa_family;
    char         sa_data[14];
};
```

# Adresa socketu AF\_UNIX



- Adresace `cestou` v souborovém systému

## struct sockaddr\_un

```
#include <sys/socket.h>
#include <sys/un.h>

struct sockaddr_un {
    sa_family_t  sun_family;
    char         sun_path[UNIX_PATH_MAX];
};
```

# Speciality UNIXových socketů

- Zjištění práv protistrany
- Zasílání deskriptorů 
- Abstraktní sockety 

# Adresa socketu AF\_INET

- Adresace IPv4 adresou a portem

## struct sockaddr\_in

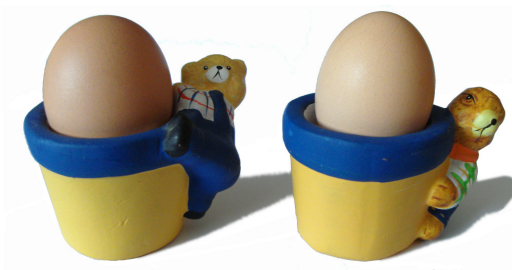
```
#include <sys/socket.h>
#include <netinet/in.h>

struct sockaddr_in {
    sa_family_t    sin_family;
    in_port_t      sin_port; /* net order */
    struct in_addr sin_addr;
};

struct in_addr {
    uint32_t s_addr; /* net order */
};
```

# Síťový formát dat

- Komunikace mezi různými stroji – nutnost stanovit pořadí bajtů v 16-bitovém a 32-bitovém čísle.
- Síťový formát dat – big endian.
- Nativní formát dat
  - little-endian: ia32, ia64, x86-64, AXP, ARM
  - big-endian: SPARC, HP-PA, Power
  - Některé platformy: podpora obojí endianity (Power/PPC, MIPS).



# Práce se síťovým formátem dat

## ntohl(3)

## Síťový formát dat

```
#include <arpa/inet.h>
```

```
uint32_t htonl(uint32_t hostl);
```

```
uint32_t ntohl(uint32_t netl);
```

```
uint16_t htons(uint16_t hosts);
```

```
uint16_t ntohs(uint16_t nets);
```

# Adresa socketu AF\_INET6

## struct sockaddr\_in6

```
#include <sys/socket.h>
#include <netinet/in.h>

struct sockaddr_in6 {
    sa_family_t      sin6_family;
    in_port_t        sin6_port; /* net order */
    uint32_t         sin6_flowinfo;
    struct in6_addr  sin6_addr;
    uint32_t         sin6_scope_id;
};

struct in6_addr {
    unsigned char s6_addr[16];
};
```

# Práce s IP adresami

## inet\_pton(3), inet\_aton(3)

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

```
int inet_pton(int af, char *cp, void *dst);
char *inet_ntop(int af, void *src, char *dst,
                socklen_t size);
```

```
int inet_aton(char *src, struct in_addr *dst);
char *inet_ntoa(struct in_addr in);
```

- inet\_pton(3) - třetí parametr musí mít dost místa pro příslušnou strukturu.
- Zastaralé: inet\_addr(3).



# Pojmenování socketu

## bind(2)

## Pojmenování socketu

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int bind(int fd, struct sockaddr *addr,
         int addrlen);
```

Přístupová práva k adresám:

- **AF\_UNIX**: práva do adresáře.
- **AF\_INET, AF\_INET6**: privilegované porty (0-1023).

## Upozornění:

Nezaměňovat privilegované a well-known porty!

# Pojmenování socketu

## bind(2)

## Pojmenování socketu

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int bind(int fd, struct sockaddr *addr,
         int addrlen);
```

Přístupová práva k adresám:

- **AF\_UNIX**: práva do adresáře.
- **AF\_INET, AF\_INET6**: privilegované porty (0-1023).

### **Upozornění:**

Nezaměňovat privilegované a well-known porty!

# Poznámky k bind(2)

- Všechna lokální rozhraní – INADDR\_ANY.
- ... v IPv6 – globální proměnná in6addr\_any, inicializátor IN6ADDR\_ANY\_INIT.
- Jen loopback v IPv6 – in6addr\_loopback, IN6ADDR\_LOOPBACK\_INIT.
- Po uzavření socketu – nelze jistou dobu další bind(2) (čekání na FIN packety atd.). Flag SO\_REUSEADDR.

# Well-known porty

## getservbyname(3)

## Získání čísla služby

```
#include <netdb.h>
```

```
struct servent *getservbyname(char *name,  
    char *proto);  
struct servent *getservbyport(int port,  
    char *proto);  
void setservent(int stayopen);  
struct servent *getservent();  
void endservent();
```

# Struktura servent

## struct servent

```
struct servent {  
    char *s_name;  
    char **s_aliases;  
    int s_port;  
    char *s_proto;  
};
```



## Příklad: Soubor /etc/services

```
discard      9/tcp      sink null
discard      9/udp      sink null
chargen      19/tcp     ttytst source
chargen      19/udp     ttytst source
ftp-data     20/tcp
ftp          21/tcp
fsp          21/udp     fspd
telnet       23/tcp
smtp         25/tcp     mail
time         37/tcp     timserver
```

# Jména a adresy

- Jednomu jménu může být přiřazeno více adres.
- Jeden počítač (rozhraní, adresa) může mít více jmen.
- **Resolver** – mechanismus převodu jmen na IP adresy a naopak.

# Převod jmen na IPv4 adresy

## gethostbyname(3)

```
#include <netdb.h>
extern int h_errno;

struct hostent *gethostbyname(char *name);
struct hostent *gethostbyaddr(char *addr,
    int len, int type);
void sethostent(int stayopen);
struct hostent *gethostent();
void endhostent();
void herror(char *s);
```



# Struktura hostent

## struct hostent

```
struct hostent {  
    char *h_name;  
    char **h_aliases;  
    int h_addrtype;  
    int h_length;  
    char **h_addr_list;  
};
```

## Příklad: Soubor /etc/hosts

```
127.0.0.1      localhost localhost.localdomain
147.251.50.60 pyrrha pyrrha.fi.muni.cz
```

### Úkol:

Napište program, který pomocí `gethostbyname(3)` vypíše IP adresy a všechny aliasy k zadanému jménu.

## Příklad: Soubor /etc/hosts

```
127.0.0.1      localhost localhost.localdomain
147.251.50.60 pyrrha  pyrrha.fi.muni.cz
```



### Úkol:

Napište program, který pomocí `gethostbyname(3)` vypíše IP adresy a všechny aliasy k zadanému jménu.

# Zjištění jména socketu

## getsockname(2)

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int getsockname(int s, struct sockaddr *name,
                int *namelen);
```

### Úkol:

Napište program, který zjistí, je-li na jeho standardním vstupu socket. Pokud ano, vypíše jeho adresu.

Napište program, který vytvoří pojmenovaný socket (AF\_UNIX) a výše uvedenému programu jej předá jako standardní vstup.

# Zjištění jména socketu

## getsockname(2)

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int getsockname(int s, struct sockaddr *name,
                int *namelen);
```



## Úkol:

Napište program, který zjistí, je-li na jeho standardním vstupu socket. Pokud ano, vypíše jeho adresu.

Napište program, který vytvoří pojmenovaný socket (AF\_UNIX) a výše uvedenému programu jej předá jako standardní vstup.

# Socket ve stavu LISTEN

## listen(2)

## Čekání na spojení

```
#include <sys/socket.h>
```

```
int listen(int sock, int backlog);
```

- **backlog** – max. počet příchozích spojení ve frontě. Další jsou odmítnuta s ECONNREFUSED.
- `listen(2)` lze volat pouze na sockety typu `SOCK_SEQPACKET` a `SOCK_STREAM`.

# Příjem příchozího spojení

## accept(2)

## Přijetí spojení na socketu

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int accept(int sock, struct sockaddr *addr,
           int *addrlen);
```

- Přijme spojení, čekající ve frontě.
- Vrátí **nový** deskriptor, odpovídající tomuto spojení.
- Pouze pro SOCK\_STREAM a SOCK\_SEQPACKET.
- Neblokující accept(2): na poslouchacím socketu vrací select(2) připravenost pro čtení.
- **addr**: adresa druhého konce socketu.

# TCP server



## Úkol:

Napište program `net read`, který dostane jako parametr jméno nebo číslo protokolu a jméno nebo číslo portu, otevře příslušný port a přijme na něm spojení.

Na standardní výstup vypíše adresu a port, ze které obdržel spojení, dále vše co přečte ze socketu a pak skončí.

Vyzkoušejte funkčnost příkazem `telnet stroj port`.



# Žádost o vytvoření spojení

## connect(2)

## Navázání spojení

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int connect(int sock, struct sockaddr *server,
            int addrlen);
```

## connect(2) a SOCK\_DGRAM

Pro SOCK\_DGRAM určuje, ze které adresy je ochoten socket přijímat data a na kterou adresu posílá data. Žádné další akce nejsou v nižších vrstvách protokolu provedeny.

# Žádost o vytvoření spojení

## connect(2)

## Navázání spojení

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int connect(int sock, struct sockaddr *server,
            int addrlen);
```



## connect(2) a SOCK\_DGRAM

Pro SOCK\_DGRAM určuje, ze které adresy je ochoten socket přijímat data a na kterou adresu posílá data. Žádné další akce nejsou v nižších vrstvách protokolu provedeny.

# TCP klient



## Úkol:

Napište program `netwrite`, který si otevře socket a spojí se protokolem TCP na zadanou adresu a port. Po ustavení spojení zapíše vše co přečte ze standardního vstupu do socketu a pak uzavře spojení.

# Zrušení spojení

## shutdown(2)

## Zrušení spojení

```
#include <sys/socket.h>
```

```
int shutdown(int sock, int how);
```

Parametr `how` může být jedno z následujících:

- 0 následující operace čtení jsou zakázány.
- 1 následující operace zápisu jsou zakázány.
- 2 všechny následující I/O operace jsou zakázány.

# Adresa protistrany spojení

## getpeername(2)

```
#include <sys/socket.h>
```

```
int getpeername(int sock, struct sockaddr *name,  
                socklen_t *namelen);
```

- Není-li socket spojený, vrátí -1 a errno = ENOTCONN.

# Nezávislost na síťovém protokolu

- `gethostbyname(3)` - vrací IPv4 adresy.
- Jak poznat z doménového jména IPv4 **nebo** IPv6 adresu?
- Jak mít aplikaci nezávislou na konkrétní `struct sockaddr_...`?
- Jak si ušetřit práci s `gethostbyname(3)`, `getprotobyname(3)` a `getservbyname(3)`?

# Resolvování nezávislé na L3 protokolu

## getaddrinfo(3)

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int getaddrinfo(const char *node,
               const char *service,
               const struct addrinfo *hints,
               struct addrinfo **result);
void freeaddrinfo(struct addrinfo *result);
const char *gai_strerror(int retval);
```

- **Reentrantní** – vrací dynamicky alokovanou strukturu.

# Struktura addrinfo

## struct addrinfo

```
struct addrinfo {  
    int             ai_flags;  
    int             ai_family;  
    int             ai_socktype;  
    int             ai_protocol;  
    size_t          ai_addrlen;  
    struct sockaddr *ai_addr;  
    char            **ai_canonname;  
    struct addrinfo *ai_next;  
};
```



# Poznámky ke getaddrinfo(3)

- Vrací zřetězený seznam struktur addrinfo.
- Pořadí prvků: RFC 3484.
- Místní preference: gai.conf(5).
- V parametru hints si lze upřesnit
  - rodinu protokolů (`ai_family` - může být i `AF_UNSPEC`)
  - typ socketu (`ai_socktype`)
  - protokol (`ai_protocol`)
  - další příznaky (`ai_flags`) - viz dále.

## Příklad: Soubor /etc/gai.conf

```
label      ::1/128      0
label      ::/0        1
label      2002::/16  2
label      ::/96      3
label      ::ffff:0:0/96 4

precedence ::1/128      50
precedence ::/0        40
precedence 2002::/16   30
precedence ::/96      20
precedence ::ffff:0:0/96 10
```

# Příznaky `ai_flags`

`AI_NUMERIC` - parametr `node` obsahuje adresu, nikoli DNS jméno.

`AI_NUMERICSERV` - položka `service` obsahuje číslo.

`AI_CANONNAME` - dohledat do `ai_canonname` oficiální jméno.

`AI_PASSIVE` - je-li `node` == NULL, dát do `ai_addr` nspecifikovanou adresu (`INADDR_ANY`), jinak zpětnovazebnou adresu.

`AI_ADDRCONFIG` - vrací rodinu protokolů podle toho, které protokoly jsou aktuálně na stroji konfigurované.

`AI_V4MAPPED` - je-li `ai_family` == `AF_INET6` a nenajde-li se žádná IPv6 adresa, vrací IPv4 adresy jako IPv6 mapované. Je-li navíc `AI_ALL`, vrací IPv4 jako v6-mapované vždy.

# Využití `getaddrinfo(3)`



## Úkol:

Napište program `getaddrinfo`, který zavolá funkci `getaddrinfo(3)` se zadaným jménem stroje a služby, a vypíše seznam vrácených struktur `addrinfo`.  
Vyzkoušejte jeho funkci na jménech jako je `www.kame.net`.

# I/O operace nad sockety

## recv\*(2)

## Čtení ze socketu

```
#include <sys/types.h>
#include <sys/socket.h>

int recv(int s, void *msg, int len,
         unsigned flags);
int recvfrom(int s, void *msg, int len,
             unsigned flags, struct sockaddr *from,
             int *fromlen);
int recvmsg(int s, struct msghdr *msg,
            unsigned flags);
```

- `recv(2)` se používá obvykle nad spojenými sockety.

# Příznaky pro recv\* (2)

- `MSG_DONTWAIT`: neblokující operace.
- `MSG_OOB`: zpracování out-of-band dat.
- `MSG_PEEK`: přečtení dat bez vymazání ze vstupní fronty.
- `MSG_WAITALL`: načtení přesně `len` bajtů dat.

# Struktura msghdr

## struct msghdr

```
struct msghdr {
    void          *msg_name;
    unsigned      msg_namelen;
    struct iovec  *msg_iov; /* viz readv(2) */
    unsigned      msg_iovlen;
    void          *msg_control;
    unsigned      msg_controllen;
    int           msg_flags;
};
```

# Řídící zprávy

## struct cmsghdr

```
struct cmsghdr {
    socklen_t cmsg_len;    /* včetně hlavičky */
    int       cmsg_level; /* protokol */
    int       cmsg_type;
    /* unsigned char cmsg_data[]; */
};
```



# Odesílání dat do socketu

## send\*(2) Zaslání zprávy do socketu

```
#include <sys/types.h>
#include <sys/socket.h>

int send(int s, void *msg, int len,
         unsigned flags);
int sendto(int s, void *msg, int len,
           unsigned flags, struct sockaddr *to,
           int tolen);
int sendmsg(int s, struct msghdr *msg,
            unsigned flags);
```

- `send(2)` funguje pouze na spojené sockety.
- Pokud je zpráva příliš dlouhá na atomický přenos, vrátí funkce chybu a `errno` nastaví na `EMSGSIZE`.

## Příznaky pro send\* (2)

- `MSG_OOB` - posílání out-of-band dat.
- `MSG_DONTROUTE` - pouze pro přímo připojené sítě.
- `MSG_DONTWAIT` - neblokující operace.
- `MSG_NOSIGNAL` - nesignalizuje SIGPIPE v případě chyby.

# UDP klient a server



## Úkol:

Napište program `udpread`, který bude přijímat zprávy na zadaném UDP portu. Pro každou zprávu vypíše, odkud ji obdržel (IP adresu a port) a obsah zprávy.

## Úkol:

Napište program `udpwrite`, který otevře UDP socket a bude posílat řádky standardního vstupu na daný UDP port a danou IP adresu.

# UDP klient a server

## ? Úkol:

Napište program `udpread`, který bude přijímat zprávy na zadaném UDP portu. Pro každou zprávu vypíše, odkud ji obdržel (IP adresu a port) a obsah zprávy.

## ? Úkol:

Napište program `udpwrite`, který otevře UDP socket a bude posílat řádky standardního vstupu na daný UDP port a danou IP adresu.

# Parametry socketu

## getsockopt(2), setsockopt(2)

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int getsockopt(int s, int level, int optname,
               void *optval, int *optlen);
int setsockopt(int s, int level, int optname,
               void *optval, int optlen);
```

- `level` je číslo protokolu (viz `getprotoent(3)`) nebo `SOL_SOCKET`.
- Viz `ip(7)`, `ipv6(7)`, `tcp(7)`, `udp(7)`, `socket(7)`.

# Parametry pro SOL\_SOCKET - I.

- **SO\_DEBUG** - nastaví zapisování ladící informace (superuser).
- **SO\_REUSEADDR** - povolí nové použití lokální adresy při `bind(2)`.
- **SO\_KEEPALIVE** - povolí posílání keep-alive packetů.
- **SO\_DONTROUTE** - obchází směrování pro odcházející zprávy.
- **SO\_LINGER** - nastavuje chování při uzavírání socketu.
- **SO\_BROADCAST** - získání práv na posílání broadcast packetů (může pouze superuživatel).
- **SO\_OOBINLINE** - OOB data jsou čtena v normální datové frontě.
- **SO\_SNDBUF, SO\_RCVBUF** - nastavení velikosti čtecího a zápisového bufferu.

# Parametry pro SOL\_SOCKET - II.

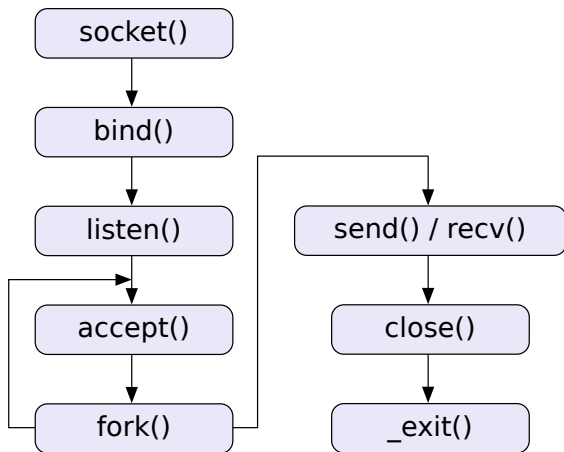
- **SO\_SNDLOWAT** - low-water mark pro posílání dat.
- **SO\_RCVLOWAT** - low-water mark pro čtení dat.
- **SO\_SNDTIMEO** - timeout pro výstupní operace.  
Maximální doba, po kterou je proces blokován ve službě jádra send(2).
- **SO\_RCVTIMEO** - timeout pro vstupní operace.
- **SO\_TYPE** - vrací typ socketu (například SOCK\_STREAM).
- **SO\_ERROR** - zkoumá, došlo-li na socketu k chybě.

# TCP/IP aplikace

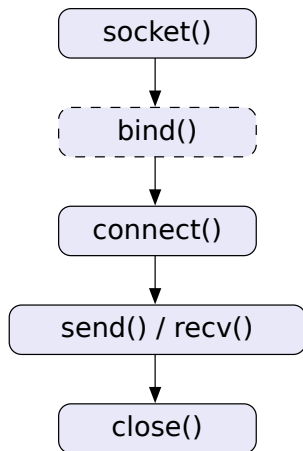
- Klient-server přístup.
- **Jednoprocesový server** - vše v jednom procesu; I/O operace multiplexovány pomocí `select(2)` nebo `poll(2)` nebo přes asynchronní I/O.
- **Víceprocesový server** - hlavní proces obvykle pouze přijme spojení přes `accept(2)`, předá potomkovi k vyřízení. Na každého klienta je jeden proces na serveru.
- **Stavový server** - výsledek předchozích operací ovlivňuje následující operace (FTP - změna adresáře).
- **Bezstavový server** - nezáleží na zopakování požadavku (NFS).



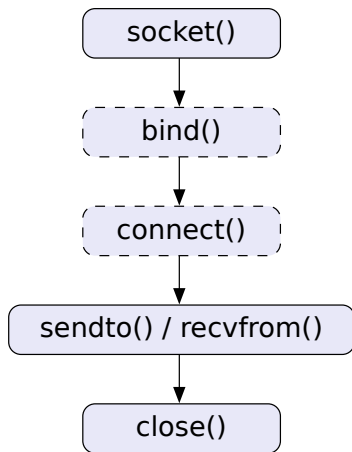
# Schéma TCP serveru



# Schéma TCP klienta



# Schéma UDP aplikace



# Kapitola 11

## Administrace sítě

# Konfigurace sítě

- Přidělení jména stroje.
- Přidělení IP adresy na interface.
- Směrovací tabulky.
  - Statické versus dynamické směrování.

# Nastavení jména stroje

## hostname(1)

## Jméno stroje

```
# hostname jméno  
$ hostname
```

- Jméno může být FQDN  nebo jen jméno bez domény (SysV).

## gethostname(2)

```
#include <unistd.h>
```

```
int gethostname(char *name, size_t len);  
int sethostname(const char *name, size_t len);
```

# Nastavení jména stroje

## hostname(1)

## Jméno stroje

```
# hostname jméno  
$ hostname
```

- Jméno může být FQDN  nebo jen jméno bez domény (SysV).

## gethostname(2)

```
#include <unistd.h>
```

```
int gethostname(char *name, size_t len);  
int sethostname(const char *name, size_t len);
```

# Informace o operačním systému

**uname(1)**

**Jméno systému**

```
$ uname [-snrvma]
```

- m - machine (hardware) type.
- n - node name (host name).
- r - operating system release.
- s - operating system name.
- v - operating system version.
- a - all of the above.



# Konfigurace síťového rozhraní

## ifconfig(8)

```
$ ifconfig [interface|-a]  
# ifconfig interface options ...
```

### Příklad: ifconfig eth0

```
eth0 Link encap:Ether HWaddr 00:24:1D:71:A7:7A  
inet addr:147.251.48.204 Bcast:147.251.48.255\  
Mask:255.255.255.0  
inet6 addr: 2001:718:801:230::cc/64 Scope:Glob  
inet6 addr: fe80::224:1dff:fe71:a77a/64 \  
Scope:Link  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:114326910 errors:0 dropped:0 ...  
...
```

# Konfigurace síťového rozhraní

## ifconfig(8)

```
$ ifconfig [interface|-a]
# ifconfig interface options ...
```

### Příklad: ifconfig eth0



```
eth0 Link encap:Ether HWaddr 00:24:1D:71:A7:7A
  inet addr:147.251.48.204 Bcast:147.251.48.255\
    Mask:255.255.255.0
  inet6 addr: 2001:718:801:230::cc/64 Scope:Glob
  inet6 addr: fe80::224:1dff:fe71:a77a/64 \
    Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:114326910 errors:0 dropped:0 ...
...
```

# Parametry pro `ifconfig(8)` - I.

`up` - aktivace rozhraní. Implicitně pokud se specifikuje nová adresa.

`down` - deaktivace rozhraní.

`[-]arp` - zapíná/vypíná použití ARP nad daným rozhraním.

`[-]promisc` - přijímá všechny pakety na dané síti.

`mtu N` - Maximum Transfer Unit.

`hw hw-adresa` - nastavení hardwarové adresy, pokud to daný interface podporuje.

# Parametry pro `ifconfig(8)` - II.

*adresa* - nastavuje adresu rozhraní.

`dstaddr` *adresa* (nebo `pointopoint` *adresa*) - cílová adresa pro point-to-point rozhraní.

`netmask` *netmask* - síťová maska rozhraní.

`[-]broadcast` [*adresa*] - nastavuje adresu pro všesměrové vysílání.

# Příklad: Konfigurace síťového rozhraní

```
# ifconfig lo 127.0.0.1 netmask 255.0.0.0

# ifconfig eth0 147.251.50.60 \  
    netmask 255.255.255.0 broadcast 147.251.50.255

# ifconfig ppp0 down
```

# Směrovací tabulka

- Adresování na základě sítě - adresa sítě, maska (adresní prefix).
- Položky: adresní prefix, adresa routeru nebo jméno rozhraní, metrika.

## route(8)

## Práce s IP směrovací tabulkou

```
$ route [-n]
# route add|del [-net|-host] target \
  [metric metric] [dev interface]
```

- Síť default je totéž co 0.0.0.0/0.
- Směřuje se podle nejdelšího prefixu a pak podle metriky.
- Program ifconfig(8) přidává automaticky cestu k lokálně připojené síti.



# Příklad: Směrovací tabulka

```
# route add 127.0.0.2 dev lo

# route add -net 147.251.48.0 \  
    netmask 255.255.255.0 gw 147.251.1.13

# route add default gw 147.251.48.14
```

# Další konfigurační nástroje

**ip(8)****Obecná konfigurace sítě**

```
# ip link ...  
# ip addr ...  
# ip route ...  
# ip tunnel ...
```

- NetworkManager (a nmcli(8))
- systemd-networkd(8)



# Směrovací protokoly

- Dynamická modifikace směrovacích tabulek.
- Tolerantnost k výpadku sítě.
- Zamezení vzniku směrovacích kruhů.
- Rozdělení zátěže.

# Druhy směrovacích protokolů



## Link-state protokol

- Routers si posílají informace o svých **sousedech**.
- Každý konstruuje **mapu sítě**.
- Každý počítá **nejkratší cesty** v mapě (Dijkstra).

## Distance-vector protocol

- Routers si posílají svoje směrovací tabulky s **metrikami**.
- Každý vybírá cestu s **nejnižší metrikou** (Bellman-Ford).

# Druhy směrovacích protokolů



## Link-state protokol

- Routers si posílají informace o svých **sousedech**.
- Každý konstruuje **mapu sítě**.
- Každý počítá **nejkratší cesty** v mapě (Dijkstra).

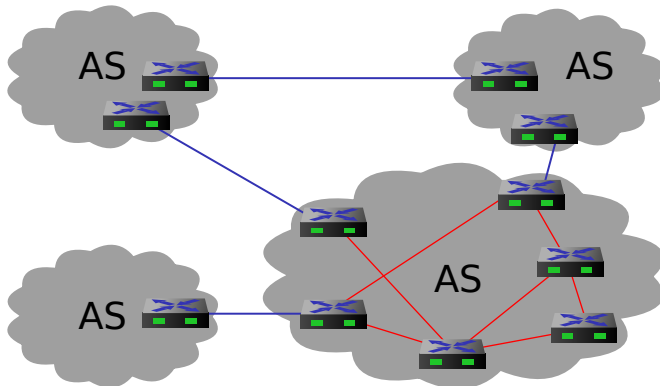


## Distance-vector protocol

- Routers si posílají svoje **směrovací tabulky** s **metrikami**.
- Každý vybírá cestu s **nejnižší metrikou** (Bellman-Ford).

# Směrování v Internetu

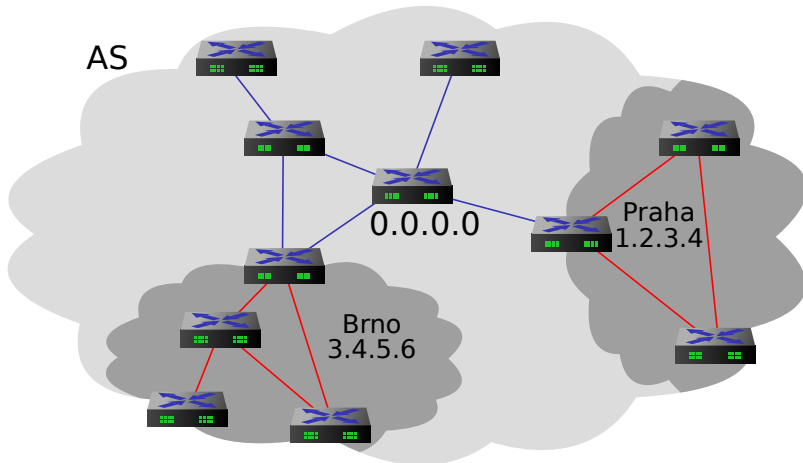
- **Autonomní systémy** - sítě jednotlivých poskytovatelů.
- **Vnitřní a vnější směrování** - v rámci nebo přes hranice AS.





# Směrovací protokoly

- **RIP** – DV protokol. Trigger update. Maximální průměr sítě 16 uzlů.
- **RIPv2** – odstraňuje některé nevýhody RIPu.
- **OSPF** – LS protokol. Open shortest paths first. Oblasti. Autentizace.
- **BGP** – DV protokol. Border gateway protocol. Externí směrovací protokol.
- **IS-IS** – LS protokol. Zvlášť topologie a zvlášť prefixy různých L3 protokolů.


# OSPF oblasti



# Směrovací protokoly v UNIXu

- **RouteD** – RIP.
- **GateD** – [www.gated.org](http://www.gated.org) .
- **Zebra, Quagga** – [www.quagga.net](http://www.quagga.net) – GPL, portabilní, včetně IPv6.
- **BIRD** – [bird.network.cz](http://bird.network.cz) .

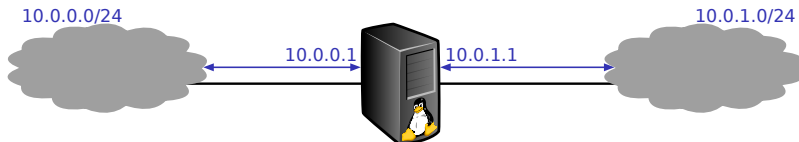
# Policy routing

- **Policy routing** - směrování podle jiného klíče, než je adresní prefix (například podle zdrojové adresy, TOS, atd.)
- **Linux** - ovládání programem `ip(8)` - nahrazuje `ifconfig(8)`, `route(8)` a několik dalších. 



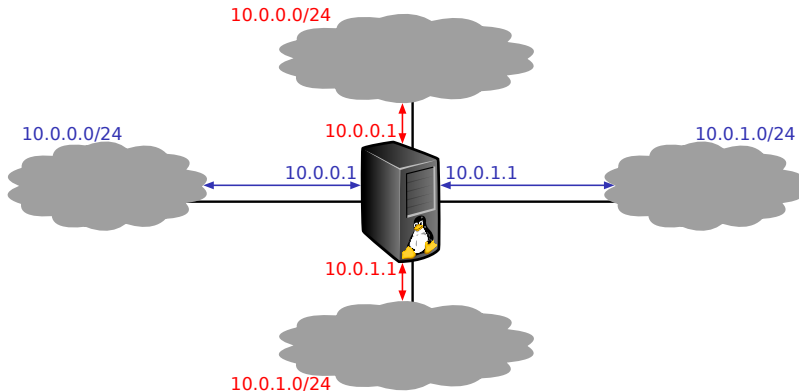


# Příklad: Použití policy routingu





# Příklad: Použití policy routingu



# Routing engine v Linuxu



- **Směrovací tabulky** – více než jedna směrovací tabulka v jádře; předdefinované tabulky `local` a `main`.
- **Pravidla** – priorita, popis packetu, akce.
- **Levá strana pravidla** – zdrojový prefix, cílový prefix, TOS, fwmark, zařízení.
- **Pravá strana** – číslo tabulky, nebo jedno z `prohibit`, `blackhole`, `unreachable`, dále zdrojový a cílový realm (doména).
- **Cílová doména (realm)** může být nastavena i směrovací tabulkou.

# Informace o nastavení sítě

## netstat(8)

```
$ netstat [-vncturi]
```

Bez parametrů vypíše seznam otevřených socketů.

Možné parametry:

- v podrobnější výpis.
- n číselný výpis bez převodu na doménová jména.
- c periodický výpis podobně jako u top(1).
- t TCP sockety.
- u UDP sockety.
- r vypíše směrovací tabulku jádra.
- i vypíše seznam všech síťových rozhraní.

 **Příklad: netstat (8)**

```
$ netstat -rn
```

Destination	Gateway	Mask	Flg	Iface
147.251.48.18	0.0.0.0	32	UH	eth0
147.251.48.0	0.0.0.0	24	U	eth0
127.0.0.0	0.0.0.0	8	U	lo
0.0.0.0	147.251.48.14	0	UG	eth0

# Výpis síťových packetů

## tcpdump (8)

```
# tcpdump [-exnp] [-c count]
           [-i interface] expression
```

- n nepřevádí adresy na doménová jména.
  - x podrobnější výpis.
  - e vypisuje také hlavičky linkové úrovně (například MAC adresy).
  - p nepřepne rozhraní do promiskuitního režimu.
- c *count* vypíše prvních *count* packetů.
- i *interface* jiné než implicitní rozhraní.

## Příklad: tcpdump (8)

```
# tcpdump -i eth0 -p port domain
tcpdump: verbose output suppressed, \
  use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet),\
  capture size 65535 bytes
15:15:33.277318 IP calypso.fi.muni.cz.36075 \
  > ares.fi.muni.cz.domain: 5129+ A? \
  aisa.fi.muni.cz. (33)
15:15:33.280797 IP ares.fi.muni.cz.domain \
  > calypso.fi.muni.cz.36075: 5129 1/3/2 A\
  147.251.48.1 (132)
```

# Program ping(8)

**ping(8)**

**ICMP Echo Request**

```
# ping [-nfqR] [-c count] [-s size] host
```

- n nepřevádí IP adresy na doménová jména.
- f flood ping.
- q bez výstupu o jednotlivých packetech.
- R record route.



## Příklad: ping (8)

```
$ ping -c 2 pyrrha.fi.muni.cz
PING pyrrha.fi.muni.cz (147.251.50.60):
  56 data bytes
64 bytes from 147.251.50.60: icmp_seq=0
  ttl=57 time=318.2 ms
64 bytes from 147.251.50.60: icmp_seq=1
  ttl=57 time=535.2 ms
- pyrrha.fi.muni.cz ping statistics -
2 pkts transmitted, 2 pkts received, 0% pkt loss
round-trip min/avg/max = 261.4/371.6/535.2 ms
```

# IP nad Ethernetem

## arp(8)

## Manipulace s ARP tabulkou

```
$ arp -a
# arp -d ipaddr
# arp -s ipaddr hwaddr [netmask netmask] \
    [pub|temp]
```

### Příklad: ARP tabulka

```
$ arp -a
pyrrha.fi.muni.cz (147.251.48.140) \
    at 00:e0:81:45:90:e2 [ether] on eth0
anxur.fi.muni.cz (147.251.48.3) \
    at 18:a9:05:e9:3b:10 [ether] on eth0
? (10.0.0.11) at 00:16:3e:01:12:34 [ether] \
    on brvm
```

# IP nad Ethernetem

## arp(8)

## Manipulace s ARP tabulkou

```
$ arp -a
# arp -d ipaddr
# arp -s ipaddr hwaddr [netmask netmask] \
    [pub|temp]
```

### Příklad: ARP tabulka

```
$ arp -a
pyrrha.fi.muni.cz (147.251.48.140) \
    at 00:e0:81:45:90:e2 [ether] on eth0
anxur.fi.muni.cz (147.251.48.3) \
    at 18:a9:05:e9:3b:10 [ether] on eth0
? (10.0.0.11) at 00:16:3e:01:12:34 [ether] \
    on brvm
```

# Point-to-point protocol

- Přenos **datagramů** po sériových linkách.
- Protokol je **symetrický** – není role master a slave (neexistuje PPP server a klient).
- IP, IPX, IPv6 a další L3 protokoly.
- Komprese paketů, šifrování a další transformace.
- Van Jacobsonova komprese IP hlaviček.
- Většina vlastností protokolu (escape znaky, atd.) je vyjednána při začátku spojení.
- Link Control Protocol (LCP) – vyjednávání o parametrech spojení.
- IP Control Protocol (IPCP) – vyjednávání o IP spojení.
- Autentizace – PAP, CHAP.

# Jména a adresy

- IP adresy – těžko zapamatovatelné.
- Jména počítačů.
- **Dříve:** soubor `hosts.txt` distribuovaný přes InterNIC (mechanismus stále používaný – `/etc/hosts`).
- Centrální registrace jmen počítačů.
- Těžkopádné – nutnost rozdělení pravomoci přidělování jmen.

# Vznik DNS

- Hierarchický systém jmen a adres.
- Systém domén, tečková notace *doménové adresy* (pyrrha.fi.muni.cz).
- **FQDN** - řetězec jmen oddělených tečkami.
- **Doména** - samostatný jmenný prostor.
- **Subdoména** - další jmenný prostor uvnitř domény.
- **Delegace autority** spravovat subdoménu.
- **Kořenová doména** - značí se tečkou (.).

# Co může obsahovat doménové jméno?

- Písmena (a - z bez rozlišení velikosti)
- Číslice (0-9)
- Pomlčka (-)
- Nic jiného (RFC 1034).

# Domain Name System

- **Definice** – RFC 1033–1035.
- RFC 1912 – Common DNS Operational and Configuration Errors.
- **BIND** – referenční implementace DNS (démon named(8))
- Name Server Operations Guide for BIND (v distribuci BINDu).



# Nameservery

- **Autoritativní nameserver** pro doménu – data o doméně ví „přímo“.
- **Cache-only nameserver** – není autoritativní pro žádnou doménu, slouží pouze k přeposílání dotazů.

# Autoritativní nameservery - delegace

- **Registrovaný nameserver** - na něj jsou delegována práva spravovat subdoménu.
- **Neregistrovaný nameserver** - není odkaz z nadřazené domény, přesto zná informace o své doméně.

# Autoritativní nameservery - data

- **Primární nameserver** pro určitou doménu (zónu) - informace má uloženy obvykle v souboru.
- **Sekundární nameserver** pro určitou doménu - data o této zóně si stahuje periodicky z primárního nebo nadřazeného sekundárního nameserveru této zóny.

# Kořenové nameservery

- nameservery pro doménu „.“
- v Internetu mají doménové jméno pod ROOT-SERVERS.NET.
- seznam je zveřejňován spolu s jejich IP adresami na ftp.internic.net.
- **Interní kořenový nameserver** - kořenový nameserver v síti s protokolem TCP/IP, která není připojena do Internetu.

# Zpracování DNS dotazů

- Nameserver ví, koho se ptát, když on sám neví
  - *forwarders* (například DNS servery poskytovatele připojení)
  - nebo zná kořenové nameservery.
- **Rekurzivní chování** – DNS server přeposílá dotazy sám.
- **Iterativní chování** – DNS server vrací „nevím, ale zeptejte se támhle“.

# Data v DNS

- **Dopředné mapování** – převod jména na IP adresu.
- **Reverzní mapování** – převod IP adres na jména.
- **Služební data** – adresy nameserverů, delegace subdomény, metadata domény, atd.
- **Další údaje** – textové popisy, adresy služeb, směrování pošty, zeměpisné souřadnice, aliasy, atd.

# Reverzní mapování

- Reverzní dotaz – dohledání FQDN na základě IP adresy.
- DNS – překlad FQDN na *nějaká data* (adresa, souřadnice, ...).
- ... možná i na jiné FQDN?
- Jak udělat z IP adresy FQDN?

## IPv4

- pseudodoména `in-addr.arpa`.
- subdomény po bajtech IPv4 adresy (od nejvýznamnějšího).
- Příklad: `147.251.48.14` → `14.48.251.147.in-addr.arpa`.

# Reverzní mapování

- Reverzní dotaz – dohledání FQDN na základě IP adresy.
- DNS – překlad FQDN na *nějaká data* (adresa, souřadnice, ...).
- ... možná i na jiné FQDN?
- Jak udělat z IP adresy FQDN?

## IPv4

- pseudodoména `in-addr.arpa`.
- subdomény po bajtech IPv4 adresy (od nejvýznamnějšího).
- Příklad: `147.251.48.14` →  
`14.48.251.147.in-addr.arpa`.



# Reverzní DNS v IPv6

- Analogické k IPv4.
- Pseudodoména `ip6.arpa`.
- Subdomény po půlbajtech (od nejvýznamnějšího).

## Příklad: Záznam v `ip6.arpa`

```
fec0::1 →  
1.0.0.0. ... 24× 0 ... .0.c.e.f.ip6.arpa.
```

# Reverzní DNS v IPv6

- Analogické k IPv4.
- Pseudodoména `ip6.arpa`.
- Subdomény po půlbajtech (od nejvýznamnějšího).

## Příklad: Záznam v `ip6.arpa`

```
fec0::1 →  
1.0.0.0. ... 24× 0 ... .0.c.e.f.ip6.arpa.
```

# Konfigurace BINDu

## named(8)

## Domain Name Server

```
# named [-d debuglevel] [-p port[/localport]] \  
        [-c configfile] [-u user] [-g group] \  
        [-t rootdir]
```

- Konfigurační soubor: /etc/named.conf
- Nespouštět pod rootem (použít -u a -g)
- Použít chroot (-t) nebo SELinux.

# Signály pro named (8)

**SIGHUP** - způsobí znovunačtení konfigurace

**SIGINT** - uloží aktuální databázi a cache do souboru  
`/var/tmp/named_dump.db`.

**SIGUSR1** - Zvýší *debuglevel* o jedničku.

**SIGUSR2** - Vypne ladění (*debuglevel* 0).

Viz též `rndc(8)`.



## Příklad: Soubor named.conf

```
options {
    directory "/var/named";
    forwarders {
        147.251.48.3; 147.251.4.33;
        3ffe:ffff::1;
    };
    forward only;
};
zone "." {
    type hint;
    file "named.root";
};
```



## 📄 Příklad: named.conf - pokračování

```
zone "fi.muni.cz" {
    type master;
    file "master/fi.muni.cz";
    also-notify {
        147.251.48.140;
        3ffe:ffff::1;
    };
    allow-transfer { any; };
};
zone "muni.cz" {
    type slave;
    file "slave/muni.cz";
    masters { 147.251.4.33; };
    allow-transfer { any; };
};
```

## 📄 Příklad: named.conf - pokračování

```
zone "48.251.147.in-addr.arpa" {  
    type master;  
    file "master/147.251.48";  
    also-notify { 147.251.48.140; };  
};  
zone "f.f.f.f.e.f.f.3.ip6.arpa" {  
    type master;  
    file "master/3.f.f.e.f.f.f.f";  
    also-notify { 3ffe:ffff::1; };  
};
```



## Příklad: Dopředný překlad

Příklad souboru `master/fi.muni.cz`:

```
$TTL 1D
@           IN SOA  anxur.fi.muni.cz. \
                postmaster.fi.muni.cz. (
                2010041701  ; Serial
                3H          ; Refresh
                15M         ; Retry
                2W          ; Expire
                1D         ) ; Min/neg TTL
```

## 📄 Příklad: Dopředný překlad

Příklad souboru `master/fi.muni.cz`:

```
$TTL 1D
@           IN SOA  anxur.fi.muni.cz. \
           ...
           IN NS   anxur
           IN NS   aisa
           IN MX   50 relay.muni.cz.
           IN MX   100 relay
           IN TXT  "Faculty of Informatics"
           IN TXT  "Masaryk University Brno"
```

# ■ Příklad: Zóna fi.muni.cz - pokračování

```
ns          IN CNAME anxur
ftp         IN CNAME odysseus
news       IN CNAME nimloth.ics.muni.cz.
time       IN CNAME pyrrha
relay      IN A      147.251.48.3
anxur      IN A      147.251.48.3
           IN HINFO  "Big Iron" "Linux"
           IN TXT    "Umisteni: Dolni sal"
calypso    IN A      147.251.50.61
           IN AAAA   3ffe:ffff::3d
           IN HINFO  "PC" "Linux"
           IN TXT    "Umisteni: Jan Kasprzak"
           IN MX    50 relay.muni.cz.
           IN MX    100 relay
```

## 📄 Příklad: Root cache - named.root

```
; formerly NS.INTERNIC.NET
.           3600000 IN NS  A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. 3600000 IN A   198.41.0.4
; formerly NS1.ISI.EDU
.           3600000 IN NS  B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. 3600000 IN A   128.9.0.107
; formerly C.PSI.NET
.           3600000 IN NS  C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. 3600000 IN A   192.33.4.12
; formerly TERP.UMD.EDU
.           3600000 IN NS  D.ROOT-SERVERS.NET.
...
```

## Příklad: Reverzní mapování IPv4

\$TTL 1D

```
48.251.147.in-addr.arpa. IN SOA anxur.fi.muni.cz.\
                                postmaster.fi.muni.cz. (
                                2001041700 ; Serial
                                3H          ; Refresh
                                15m         ; Retry
                                2W          ; Expire
                                1D         ) ; Minimum
                                IN NS      anxur.fi.muni.cz.
                                IN NS      aisa.fi.muni.cz.
1                                IN PTR    aisa.fi.muni.cz.
3                                IN PTR    anxur.fi.muni.cz.
                                IN PTR    relay.fi.muni.cz.
5                                IN PTR    thetis.fi.muni.cz.
```

...

## Příklad: Reverzní mapování IPv6

```
$TTL 1D
f.f.f.f.e.f.f.3.ip6.arpa. IN SOA \
                            anxur.fi.muni.cz.\
                            postmaster.fi.muni.cz. (
                            2001041700 ; Serial
                            3H          ; Refresh
                            15m         ; Retry
                            2W          ; Expire
                            1D         ) ; Minimum
      IN NS  anxur.fi.muni.cz.
      IN NS  aisa.fi.muni.cz.
d.3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 \
  IN PTR  calypso.fi.muni.cz.
...

```

# Další typy DNS záznamů

- **SRV** – server služby pro doménu
  - `_kerberos._udp IN SRV 0 0 88 thetis`
- **LOC** – zeměpisné souřadnice
  - `aisa IN LOC 49 12 35.943 N \  
16 35 56.099 E \  
215 1 1 1`
- **SSHFP** – otisk veřejného SSH klíče
  - `aisa IN SSHFP 2 1 123456789ab...890`

# Konfigurace DNS klienta

- Resolvovací mechanismy - např. přes NSswitch.
- DNS servery: /etc/resolv.conf
- Seznam nejvýše čtyř nameserverů (má vliv na velikost timeoutu).

## Příklad: Soubor resolv.conf

```
domain fi.muni.cz           nebo:
search fi.muni.cz ics.muni.cz muni.cz
nameserver 147.251.48.14
nameserver 147.251.4.33
```



# Konfigurace DNS klienta

- Resolvovací mechanismy - např. přes NSswitch.
- DNS servery: /etc/resolv.conf
- Seznam nejvýše čtyř nameserverů (má vliv na velikost timeoutu).

## Příklad: Soubor resolv.conf

```
domain fi.muni.cz           nebo:
search fi.muni.cz ics.muni.cz muni.cz
nameserver 147.251.48.14
nameserver 147.251.4.33
```

# File Transfer Protocol

- Protokol pro přenos souborů
- Možnost přenosu i mezi dvěma vzdálenými počítači.
- Řídící spojení – 21/tcp, iniciováno klientem.
- Datové spojení – zdrojový port (obvykle) 20/tcp, iniciováno serverem.
- Pasivní FTP – i datové spojení otevírá klient na server. Někdy lepší průchod přes proxy-servery a firewally.
- Anonymní FTP – obvykle přihlášení na dohodnutý účet ftp nebo anonymous. Povoluje se pouze čtení určitého podstromu (a FTP-démon pro tento strom zavolá chroot(2)).

## Otázka:

Proč má FTP problém s překladem adres?

# File Transfer Protocol


- Protokol pro přenos souborů
- Možnost přenosu i mezi dvěma vzdálenými počítači.
- Řídící spojení – 21/tcp, iniciováno klientem.
- Datové spojení – zdrojový port (obvykle) 20/tcp, iniciováno serverem.
- Pasivní FTP – i datové spojení otevírá klient na server. Někdy lepší průchod přes proxy-servery a firewally.
- Anonymní FTP – obvykle přihlášení na dohodnutý účet ftp nebo anonymous. Povoluje se pouze čtení určitého podstromu (a FTP-démon pro tento strom zavolá `chroot(2)`).




## Otázka:

Proč má FTP problém s překladem adres?

# Vzdálené volání procedury

- **RPC** – remote procedure call.
- Vykonání procedury asynchronně na vzdáleném stroji.
- **ONC RPC** – dříve Sun RPC .
- Jednoznačné číslo *balíku procedur* (= služby), verze balíku.
- Formát dat nezávislý na platformě.
- **XDR** – external data representation (serializace dat).
- Podobné služby: OSF DCE, CORBA, SOAP.

# Port mapper

- **Jmenná služba** pro RPC.
- Nejsou vyhrazené porty pro balíky procedur.
- **Port mapper** – RPC služba na pevném portu (111/tcp, 111/udp).
- Převod *čísla balíku a verze* na číslo portu.
- Forwardování požadavků.
- Vzdálená registrace služeb.
- **Implementace** – `rpc.portmap(8)` nebo `rpcbind(8)` .

# Výpis informací z portmapperu

```
rpcinfo(8)
```

```
Informace od portmapperu
```

```
$ rpcinfo -p
```

program	vers	proto	port	
100000	2	tcp	111	rpcbind
100000	2	udp	111	rpcbind
100005	1	udp	766	mountd
100005	1	tcp	768	mountd
100005	2	udp	771	mountd
100005	2	tcp	773	mountd
100003	2	udp	2049	nfs

# Network File System

- Sdílení souborů po síti.
- **Bezestavová služba** - tolerance k restartu serveru, ale jiné problémy.
- **Sdílený prostor** UID/GID nebo jmen uživatelů. Jiná autentizace až ve verzi 4.

# NFS klient


## Příklad: Připojení NFS svazku

```
# mount -t nfs aisa:/export/home /home
```

- Přístup je podobný jako k lokálním diskům.
- Nemožnost zamykání existencí souboru.
- Zamykání části souboru - démon `rpc.lockd(8)` a `rpc.statd(8)`.
- Asynchronní zápis/čtení - démoni `biod(8)` nebo `nfsiod(8)` uvnitř jádra.



# NFS server

- **Démoni** `rpc.mountd(8)`, `rpc.nfsd(8)`, případně `rpc.ugidd(8)` pro mapování UID a GID a `rpc.gssd(8)`.
- Někdy je `nfsd(8)` uvnitř jádra a běží v několika instancích pro urychlení paralelního přístupu.
- `nfsd(8)` má pevně vyhrazený port 2049.
- **Seznam sdílených adresářů**: `/etc/exports`, `/etc/dfs/dfstab` .
- `showmount(8)` - výpis informací z mount-démona (`export-list`, seznam připojených adresářů).
- **User-space NFS server** - alternativní implementace mimo jádro.

# Kerberos

- Centrální autentizační systém
- Třístranná autentizace:
  - uživatel
  - služba/server
  - Kerberos server (KDC) - důvěryhodná třetí strana
- Bez zasílání hesla po síti.

# Kerberos - základní pojmy

- **Key Distribution Center** – služba, která má databázi všech hesel.
- **Realm** – doména; oblast spravovaná jedním KDC.
- **Principal** – jméno uživatele nebo služby.
- **Heslo** (tajemství/secret) – pro uživatele i pro služby.
- **Lístky** (tickets) – vydává KDC, používají se pro prokazování totožnosti klienta vůči službě.

## Příklad: Principal

1885@IS.MUNI.CZ

krbtgt/FI.MUNI.CZ@FI.MUNI.CZ

# Kerberos - základní pojmy

- **Key Distribution Center** – služba, která má databázi všech hesel.
- **Realm** – doména; oblast spravovaná jedním KDC.
- **Principal** – jméno uživatele nebo služby.
- **Heslo** (tajemství/secret) – pro uživatele i pro služby.
- **Lístky** (tickets) – vydává KDC, používají se pro prokazování totožnosti klienta vůči službě.



## Příklad: Principal

1885@IS.MUNI.CZ

krbtgt/FI.MUNI.CZ@FI.MUNI.CZ

# Získání lístku

- **Žádost o lístek** – principal uživatele, adresa, principal služby, časové informace, další parametry.
- **Odpověď KDC**
  - (session key, jméno klienta, jméno serveru) zašifrováno heslem uživatele.
  - lístek (neprůhledný pro uživatele).
- **Uživatel** – zadáním hesla získá session key.

# Získání lístku

- **Žádost o lístek** – principal uživatele, adresa, principal služby, časové informace, další parametry.
- **Odpověď KDC**
  - (session key, jméno klienta, jméno serveru) zašifrováno heslem uživatele.
  - lístek (neprůhledný pro uživatele).
- **Uživatel** – zadáním hesla získá session key.

# Získání lístku

- **Žádost o lístek** – principal uživatele, adresa, principal služby, časové informace, další parametry.
- **Odpověď KDC**
  - (**session key**, jméno klienta, jméno serveru) zašifrováno heslem uživatele.
  - lístek (neprůhledný pro uživatele).
- **Uživatel** – zadáním hesla získá session key.

# Použití lístku

- **Obsah lístku** – (session key, jméno a adresa klienta, jméno serveru, čas, doba platnosti) zašifrováno heslem služby.
- **Autentizace** – klient vytvoří *autentizátor* a pošle i s lístkem službě.
  - **Autentizátor** – (jméno a adresa klienta) zašifrováno session key.
- **Ověření službou:**
  - rozšifruje lístek,
  - získá session key,
  - rozšifruje autentizátor
  - ověří, že jméno a adresa klienta odpovídá v autentizátoru i v lístku.



# Použití lístku

- **Obsah lístku** - (session key, jméno a adresa klienta, jméno serveru, čas, doba platnosti) zašifrováno heslem služby.
- **Autentizace** - klient vytvoří *autentizátor* a pošle i s lístkem službě.
  - **Autentizátor** - (jméno a adresa klienta) zašifrováno session key.
- **Ověření službou:**
  - rozšifruje lístek,
  - získá session key,
  - rozšifruje autentizátor
  - ověří, že jméno a adresa klienta odpovídá v autentizátoru i v lístku.

# Použití lístku

- **Obsah lístku** – (session key, jméno a adresa klienta, jméno serveru, čas, doba platnosti) zašifrováno heslem služby.
- **Autentizace** – klient vytvoří *autentizátor* a pošle i s lístkem službě.
  - **Autentizátor** – (jméno a adresa klienta) zašifrováno session key.
- **Ověření službou:**
  - rozšifruje lístek,
  - získá session key,
  - rozšifruje autentizátor
  - ověří, že jméno a adresa klienta odpovídá v autentizátoru i v lístku.

# Další vlastnosti

- Ticket granting service je také služba.
- Ticket granting ticket - TGT - lístek pro získávání dalších lístků.
- Získání TGT - program `kinit(1)` nebo `pam_krb5(8)`. Dále `klist(1)`, `kdestroy(1)`.
- Replay cache - proti odposlechnutí.
- Autentizace služby vůči klientovi.
- Forwardovatelné lístky - nejsou vázané na adresu.
- Proxy lístky - delegace části pravomocí na službu.
- Interakce mezi realmy - například `cross-realm trust`.
- Záložní KDC - replikace.

# Implementace

- MIT Kerberos V – referenční implementace.
- Heimdal – evropská implementace.
- Klientské aplikace – nutnost kerberizace (mj. rozšíření protokolu).
- Konfigurace klientů – /etc/krb5.conf.
- Klíče služeb – /etc/krb5.keytab, ktutil(8).

Přečtěte si `^_~`

Designing an Authentication System  
(rozhovor ve čtyřech scénách).

<http://web.mit.edu/kerberos/www/dialogue.html>.

# Implementace

- MIT Kerberos V – referenční implementace.
- Heimdal – evropská implementace.
- Klientské aplikace – nutnost kerberizace (mj. rozšíření protokolu).
- Konfigurace klientů – /etc/krb5.conf.
- Klíče služeb – /etc/krb5.keytab, ktutil(8).



## Přečtěte si <sup>^</sup>\_<sub>~</sub>

Designing an Authentication System  
(rozhovor ve čtyřech scénách).

<http://web.mit.edu/kerberos/www/dialogue.html>.

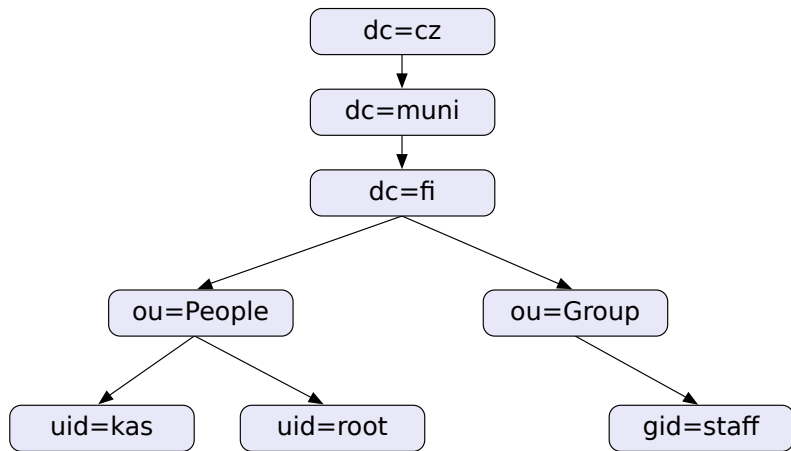
# LDAP

- Light-weight Directory Access Protocol
- Původ – odlehčená varianta DAP pro přístup k X.500 adresářům.
- Adresář
  - ne jako ve filesystému,
  - analogie telefonního seznamu (kartičky).
- Adresářová služba – databáze pro rychlé vyhledávání a občasné modifikace dat.

# LDAP - vlastnosti

- Celosvětový distribuovaný adresář - i odkazy mezi LDAP servery.
- Vyhledávání - v daném podstromu, objekt určitých atributů.
- Odkazy - referrals - něco jako symbolické linky.
- Read-mostly databáze - nejsou transakce, rychlé vyhledávání.

# LDAP strom





# LDAP - organizace dat

- Uzly stromu - objekty.
- Distinguished name (DN) poloha ve stromu (cesta ke kořeni) - např.:  
uid=kas,ou=People,dc=fi,dc=muni,dc=cz
- Relativní DN - v rámci jedné úrovně (uid=kas).
- Objekty - patří do tříd (i více), mají různé atributy.
- Schéma - definice tříd (názvy atributů, typy hodnot, povinné/nepovinné atributy). Definice ve formátu ASN.1.

# Autentizace v LDAPu

- **Autentizace proti LDAP serveru** – externí (SASL), simple (heslo je součástí objektu), anonymní.
- **Subjekt práv** – obecné DN. Není zvlášť databáze uživatelů.
- **Objekt práv** – podstrom, jednotlivá DN, *self*, jednotlivé atributy.
- **Přístupová práva** – *authenticate, compare, read, search, write*.

## Příklad: Formát LDIF

```
$ ldapsearch -H ldap://ldap.fi.muni.cz/ \
  -b ou=People,dc=fi,dc=muni,dc=cz uid=kas -x
# kas, People, fi.muni.cz
dn: uid=kas,ou=People,dc=fi,dc=muni,dc=cz
uid: kas
objectClass: account
objectClass: posixAccount
userPassword:: e2NyeXB0fXg=
loginShell: /bin/bash
uidNumber: 11561
gidNumber: 10100
homeDirectory: /home/kas
gecos: Jan Kasprzak
host: aisa
host: anxur
```

# LDAP v UNIXu

- **Napojení na nsswitch** – možnost uložení tabulek uživatelů, skupin, služeb, protokolů ...
- **Modul nss\_ldap** – konfigurace v `/etc/ldap.conf`.
- Mapování podstromu LDAP objektů a jejich atributů na záznamy v tabulce uživatelů.
- **Automatický převod** – MigrationTools – [www.padl.com](http://www.padl.com).
- **Uživatelé** – třídy `account`, `posixAccount` a `shadowAccount`.
- **Zabezpečení** – možnost použít LDAP nad SSL (`nss_ldap` umí kontrolovat i certifikát serveru).

# Ukládání hesel v LDAPu

- Několik algoritmů ukládání.
- Formát hesla - např.: {SMD5}F92mezjPoWxSE.
- Hashovací metody - CRYPT, SMD5, MD5, SSHA, SHA.
- Generování hashované podoby hesla - s`lappasswd(8)`.
- Heslo v LDIF formátu - často kódované base64. Oddělovač `::` v LDIF. Podobně se kódují binární data.

Úkol: Jaké mám heslo? ^\_~

```
dn: uid=kas,ou=People,dc=fi,dc=muni,dc=cz
...
userPassword:: e2NyeXB0fXg=
```



# Ukládání hesel v LDAPu

- Několik algoritmů ukládání.
- Formát hesla - např.: {SMD5}F92mezjPoWxSE.
- Hashovací metody - CRYPT, SMD5, MD5, SSHA, SHA.
- Generování hashované podoby hesla -  
s`lappasswd(8)`.
- Heslo v LDIF formátu - často kódované base64.  
Oddělovač `::` v LDIF. Podobně se kódují binární data.

## ? Úkol: Jaké mám heslo? ^\_~

```
dn: uid=kas,ou=People,dc=fi,dc=muni,dc=cz
...
userPassword:: e2NyeXB0fXg=
```

# Implementace LDAPu

- **Řádkoví klienti** – `ldapsearch(1)`, `ldapadd(1)`, `ldapdelete(1)`, `man ldapmodify(1)`.
- **Ostatní klienti** – `GQ(1)`, `nss_ldap`, Mozilla.
- **Servery** – OpenLDAP, iPlanet/SunONE , Oracle Internet Directory, Fedora Directory Server , ...

# Elektronická pošta

- Off-line komunikace.
- Textová komunikace (později i další druhy dat).



## Komponenty poštovního systému

- **MTA** – program pro přenos zpráv po síti (sendmail, qmail, exim).
- **MUA** – uživatelský program pro čtení a posílání zpráv (elm, exmh, mutt).
- **MDA** – program pro lokální doručení do schránky (mail(1), procmail(1), deliver(1)).



# Obálka zprávy

- **Obálkový odesílatel** – kam poslat chybovou zprávu.
- **Obálkový adresát** – skutečný příjemce zprávy.

## Otázka:

Kdo je obálkovým odesílatelem chybové zprávy?

## Upozornění:

Obálka  $\neq$  hlavičky!

# Obálka zprávy

- Obálkový odesílatel – kam poslat chybovou zprávu.
- Obálkový adresát – skutečný příjemce zprávy.



## Otázka:

Kdo je obálkovým odesílatelem chybové zprávy?

## Upozornění:

Obálka  $\neq$  hlavičky!

# Obálka zprávy

- Obálkový odesílatel – kam poslat chybovou zprávu.
- Obálkový adresát – skutečný příjemce zprávy.



## Otázka:

Kdo je obálkovým odesílatelem chybové zprávy?



## Upozornění:

Obálka  $\neq$  hlavičky!

# Formát zpráv

- RFC 2822 „Internet Message Format“ (dříve RFC 822).
- **Hlavička** – obsahuje řídicí informace zprávy a má pevnou strukturu.
- **Tělo zprávy** oddělené prázdným řádkem.

## Hlavičky zprávy

- řádky tvaru *klíč: hodnota*
- řádky tvaru *bílý znak hodnota* – pokračování hodnoty z předchozího řádku.

# Formát zpráv

- RFC 2822 „Internet Message Format“ (dříve RFC 822).
- **Hlavička** – obsahuje řídicí informace zprávy a má pevnou strukturu.
- **Tělo zprávy** oddělené prázdným řádkem.



## Hlavičky zprávy

- řádky tvaru *klíč: hodnota*
- řádky tvaru *bílý znak hodnota* – pokračování hodnoty z předchozího řádku.

# Původce zprávy

**From:** mailbox autora/autorů zprávy.

**Sender:** skutečný odesílatel zprávy, je-li ve From: více mailboxů.

**Reply-To:** na jakou adresu se má poslat odpověď.

# Adresát zprávy

**To:** adresa hlavního příjemce.

**Cc:** carbon copy - další příjemci.

**Bcc:** blind carbon copy - totéž, ale tato hlavička se při odesílání ze zprávy odstraní.

# Identifikace

**Message-Id:** jednoznačná identifikace zprávy. Používá se například k detekci zacyklení pošty.

**In-Reply-To:** „v odpovědi na“ – identifikátor předchozí zprávy (na kterou tato zpráva odpovídá).

**References:** identifikátory předchozích zpráv.



# Trasování zprávy

**Received:** každý MTA po cestě přidá jeden takovýto řádek se služebními informacemi.

**Return-Path:** u doručené zprávy cesta k odesílateli.

# Ostatní položky

**Date:** datum vzniku zprávy.

**Subject:** předmět, věc.

**Keywords:** klíčová slova.

**X-cokoli:** nestandardní hlavičky (X-Face:).

# MIME

- Multipurpose Internet Mail Extensions - RFC 1521, 2045-2049 a 2231.
- Původně - jen ASCII znaky.
- Je-li MIME - ASCII v hlavičkách, v těle libovolné (je-li uvedeno).

## Povinné hlavičky MIME

```
Mime-Version: 1.0  
Content-Type: typ[podtyp[; parametry ...]]  
Content-Transfer-Encoding: přenosové kódování
```

# MIME

- Multipurpose Internet Mail Extensions - RFC 1521, 2045-2049 a 2231.
- Původně - jen ASCII znaky.
- Je-li MIME - ASCII v hlavičkách, v těle libovolné (je-li uvedeno).



## Povinné hlavičky MIME

```
Mime-Version: 1.0  
Content-Type: typ[podtyp[; parametry ...]]  
Content-Transfer-Encoding: přenosové kódování
```

# Jednoduché typy/podtypy:

`text` / plain, html, richtext, ...

`image` / gif, jpeg, g3fax, ...

`audio` / basic, ...

`video` / mpeg, quicktime, ...

`application` / octet-stream, postscript, pdf, ...

*x-nestandardní*

# Složené typy/podtypy

`multipart/mixed` - více objektů různých typů.

`multipart/parallel` - paralelně prezentovatelné části (například text a zvuk).

`multipart/alternative` - MUA má zobrazit jednu z částí (například `text/plain` a `text/html`).

# Přenosová kódování

- Přenosové kódování (např. hlavička Content-Transfer-Encoding:).
- Nespecifikují znakovou sadu.

7bit - jen ASCII.

base64 - úsporné kódování (3 bajty na 4 znaky).

quoted-printable - ne-ASCII bajty a rovnítko  
kódovány jako *=hexa kód*.

8bit - přímé použití libovolných bajtů.

## Příklad: Zápis slova „ježek“ v UTF-8

QP: je=C5=B5ek

Base64: amXFvmVr

# Přenosová kódování

- Přenosové kódování (např. hlavička Content-Transfer-Encoding:).
- Nespecifikují znakovou sadu.

**7bit** – jen ASCII.

**base64** – úsporné kódování (3 bajty na 4 znaky).

**quoted-printable** – ne-ASCII bajty a rovnítko kódovány jako *=hexa kód*.

**8bit** – přímé použití libovolných bajtů.

## Příklad: Zápis slova „ježek“ v UTF-8

QP: je=C5=B5ek  
Base64: amXFvmVr



# Přenosová kódování

- Přenosové kódování (např. hlavička Content-Transfer-Encoding:).
- Nespecifikují znakovou sadu.

`7bit` – jen ASCII.

`base64` – úsporné kódování (3 bajty na 4 znaky).

`quoted-printable` – ne-ASCII bajty a rovnítko kódovány jako *=hexa kód*.

`8bit` – přímé použití libovolných bajtů.

## Příklad: Zápis slova „ježek“ v UTF-8

QP: je=C5=B5ek

Base64: amXFvmVr

# Kódování MIME Words

- **Hlavičky** – jen ASCII znaky.
- **RFC 2047** – kódování ne-ASCII znaků v hlavičkách.
- **Formát** `=?přenosové kódování?charset?text?=  
=?`

**Příklad: „Křemílek a Vochomůrka“**

```
=?Q?UTF-8?K=C5Tem=C3=ADlek=20a=20?=  
=?Q?UTF-8?Vochom=C5=AFrka?=  
=?
```

# Kódování MIME Words

- **Hlavičky** – jen ASCII znaky.
- **RFC 2047** – kódování ne-ASCII znaků v hlavičkách.
- **Formát** `=?přenosové kódování?charset?text?=  
=?`



## **Příklad: „Křemílek a Vochomůrka“**

```
=?Q?UTF-8?K=C5Tem=C3=ADlek=20a=20?=  
=?Q?UTF-8?Vochom=C5=AFrka?=  
=?
```

# SMTP

- Simple Mail Transfer Protocol – protokol pro přenos pošty nad TCP/IP.
- RFC 2821 – RFC 821.
- Rozšiřitelnost – ESMTP (8-bitový přenos, ETRN, atd).
- Inicializace – HELO, EHLO.
- Předání obálky:  
MAIL FROM:<odesílatel>  
RCPT TO:<adresát>
- Předání zprávy – DATA; zakončeno tečkou na samostatném řádku.
- Ukončení relace – QUIT.

# SMTP - další možnosti

- Vynucený přenos - ETRN.
- Kontrola odesilatele - VRFY.
- Expanze aliasů - EXPN.
- Dotaz na velikost - SIZE.
- Zvýšení propustnosti - PIPELINING.
- Přenos 8-bitových dat - 8BITMIME.

# Formát mailboxů v UNIXu

- **mbox** – jeden soubor.
  - Zprávy začínají „From *mezera*“ na začátku řádku.
  - Následuje adresa odesílatele a datum přijetí.
  - Zbytek je vlastní zpráva (hlavičky, tělo).
  - Odsazení „From *mezera*“ většítkem.
- **Maildir** – Podadresáře: tmp, new, cur. Odstraňuje problémy se zamykáním.
- **MMDf folder** – podobné jako mailbox, ale zprávy jsou odděleny čtyřmi znaky Ctrl-A.
- **MH-folder** – adresář; jednotlivé zprávy jsou uloženy v souborech s číselnými názvy. Soubory označené ke smazání mají název začínající čárkou.

# Lokální klienti

- Přímý přístup k mailboxu (zamykání). Někdy set-gid pro skupinu mail.
- Odesílání – na vstup `/usr/sbin/sendmail -t`.

# Síťoví klienti



- Čtení pošty – POP-3 nebo IMAP.
- Odesílání – SMTP přes relay.



# POP-3

- Post Office Protocol
- Výlučný přístup k mailboxu.
- Atomická operace během celé session.
- Nelze více schránek v rámci jednoho účtu.

# IMAP

- Internet Mail Access Protocol.
- Sdílený přístup k mailboxu.
- Práce s více schránkami v rámci jednoho účtu.
- Možnost čtení pošty z více počítačů.

# Kapitola 12

## Zdroje

# Zdroje obrázků

- Strany 49, 50, 57, 93, 146, 155, 157, 416: <http://www.sxc.hu>
- Strana 21: <https://static.zerochan.net/Andou.Mahoro.full.114965.jpg>
- Strana 3: <http://cm.bell-labs.com/cm/cs/who/dmr/otherunix.html>
- Strana 203: <http://all-free-download.com/>
- Strana 115: [https://www.oldcomputr.com/wp-content/uploads/2015/11/digital\\_vt100-7342.jpg](https://www.oldcomputr.com/wp-content/uploads/2015/11/digital_vt100-7342.jpg)
- Strana 191: NOAA's National Ocean Service
- Strana 190: Free Software Foundation
- Strana 201: <https://banksy.newtfire.org>
- Ostatní obrázky: Servisní středisko pro e-learning na MU <http://is.muni.cz/stech/>