# Kurs administrowania systemem Linux

## Zajęcia nr 6: Podstawowe czynności administracyjne w Linuksie

Instytut Informatyki Uniwersytetu Wrocławskiego

3 kwietnia 2025

# System plików (*rootfs*)

**Składniki systemu plików Uniksa**

- *Namespace*: mechanizm nazywania i hierarchicznego grupowania obiektów (katalogi to pliki, drzewiasta struktura katalogów, montowanie podsystemów plików).
- *Access control*: mechanizmy ograniczania dostępu do obiektów.
- *API*: sposoby komunikacji użytkownika z jądrem w celu uzyskania informacji o i dostępu do obiektów.
- Implementacja.

# Montowanie systemów plików

## Montowanie

`mount -t` *`typ`* `-o` *`opcje urządzenie katalog`*

W Linuksie typ zwykle wykrywany automatycznie, we FreeBSD — nie.

## Konfiguracja punktu montowania w `/etc/fstab`

*`urządzenie katalog typ opcje backup fsck`*

- Wpis w `/etc/fstab` uwalnia od konieczności podawania wszystkich opcji w wierszu poleceń lub skrypcie (np. inicjalizacja systemu) — wystarczy podać tylko katalog lub urządzenie.
- Dodatkowy katalog `/etc/fstab.d/` — po jednym pliku na urządzenie.
- *`backup`* = 1: `dump(8)` robi backup, 0: nie robi.
- *`fsck`* — czy wołać `fsck(8)` = 2: tak, 0: nie, 1: *rootfs*.
- W SystemD zastąpiony jednostkami `*.mount` (po jednej dla każdego systemu plików).

# Identyfikatory i nazwy urządzeń

**Numer urządzenia nadawany przez jądro**

- Para *major*:*minor* typu `dev_t` (def.: `<sys/types.h>`).
- *major* — klasa (rodzaj) urządzenia (def.: `<linux/major.h>`), np. `TTY_MAJOR=4`, `SCSI_DISK0_MAJOR=8`) — określa m.in. sterownik jądra.
- *minor* — identyfikator konkretnego urządzenia danej klasy.
- Np. pierwszy dysk SATA: `8:0`, drugi dysk SATA: `8:16`, druga partycja 1. dysku SATA: `8:2`.
- Uwaga: *minor* zależy od kolejności wykrywania urządzeń!

**Plik urządzenia w katalogu /dev**

- Tworzony przez syscall `mknod(2)` — uniwersalny syscall do tworzenia i-*nodów*. Funkcje pomocnicze: `makedev(3)`, `major(3)` i `minor(3)`.
- Daje możliwość odwoływania się do urządzenia poprzez nazwę pliku (m. in. zwykła kontrola dostępu), deskryptor pliku (m. in. zwykłe syscalle typu `read`) itd.
- Konfiguracja statyczna: polecenie `mknod(1)` i skrypt `MAKEDEV(8)`.
- Obecnie pseudosystem plików `devfs` zarządzany dynamicznie przez `udev(7)`.

# Nazwy urządzeń znakowych w katalogu /dev

**Nazwy urządzeń znakowych**

- `tty`*n* — terminale (wirtualne),
- `tty` — dla każdego procesu jego terminal sterujący,
- `console` — konsola systemowa („najważnieszy” terminal),
- `ttyUSB`*n*, `ttyACM`*n* — terminale USB (pełne i uproszczone),
- `ttyS`*n* — terminale szeregowe (RS232),
- `vcs`*n*, `vcsa`*n* — niskopoziomowy dostęp do terminali wirtualnych,
- `pts/`*n* — pseudoterminale (`pts/ptsmx` — tworzenie, zob. `pts(4)`),
- pseudourządzenia `full`, `null`, `zero`, `random`, `urandom`, `kmsg` i in.

# Nazwy urządzeń blokowych w katalogu /dev

**Nazwy urządzeń blokowych**

- sd*x* — SATA/SCSI (driver: libATA), *x* — litera a, b,..., z, aa, ab,..., np. sda, sdd,
- nvme*n* — dyski NVMe, *n* — liczba 0, 1,..., np. nvme0,
- nvme*n*n*m* — *namespace'y* NVMe, *m* — liczba 1, 2,..., np. nvme0n1,
- sr*n* — ATAPI ROM (CD-ROM, DVD-ROM), np. sr0,
- dm-*n* — pseudourządzenia tworzone przez *device mappera*, np. dm-0 (zwykle rootfs),
- loop*n* — pseudourządzenia tworzone przez *losetup*, np. loop0,
- hd*x* — dyski IDE, np. hda,
- fd*n* — napęd dyskietek, np. fd0 (nie mylić z katalogiem /dev/fd/).

**Całe urządzenia i ich partycje**

- *urządzenie* — całe urządzenie, np. sda,
- *urządzenie*p*m* — partycja *m* (numeracja od 1), np. sda1,
- *urządzenie*p*m* (jeśli nazwa kończy się cyfrą), np. nvme0n1p1, sdc1p2.

# Linki symboliczne w katalogu /dev

**Dodatkowe aliasy**

- `fd` → `/proc/self/fd`,
- `stdin`, `stdout`, `stderr` → `/proc/self/fd/{0,1,2}`,
- `rtc` → `rtc0`, `cdrom` → `sr0` itp.
- `block/`, `char/` — mapowanie *major:minor* na nazwy urządzeń (odpowiednio: blokowych i znakowych).

**Nazwy dysków niezależne od (przypadkowej) numeracji jądra**

- `mapper/` — nazwy symboliczne urządzeń *device mappera*,
- `disk/` — unikatowe nazwy, ID i UUID partycji i systemów plików.

# Unikatowa identyfikacja urządzeń

**Systemy plików i kontenery**

- ext2/3/4: 16-bajtowy losowy UUID i 16-bajtowa etykieta umieszczone w superbloku,
- FAT32: 4-bajtowa sygnatura (generowana na podstawie czasu) i 11-znakowa etykieta umieszczone w BPB.
- Kontener LUKS/LUKS2 — 16-bajtowy losowy UUID.
- Linki dostępne w katalogu `/dev/disk/by-uuid/`.

**Systemy partycji**

- MSDOS: 4-bajtowa sygnatura umieszczona w MBR (generowana na podstawie czasu),
- GPT: 16-bajtowy losowy UUID umieszczony w nagłówku GPT (także 4-bajtowa sygnatura umieszczona w *protective/hybrid MBR*).

**Partycje**

- MSDOS: brak. Symulowane przez Linux w postaci *SygnaturaDysku-NumerPartycji*.
- GPT: 16-bajtowy losowy UUID każdej partycji (nie mylić z UUID typu partycji), 36-znakowa (UTF-16LE) etykieta partycji.
- Linki dostępne w katalogach `/dev/disk/{by-partuuid,by-partname}/`.

# Katalog /dev/disk/

**Katalog /dev/disk/**

- Tworzony przez `udev`.
- Reguły tworzenia w `/usr/lib/udev/rules.d/60-persistent-storage.rules` (można modyfikować).
- Zarządzanie: `udevadm(8)`.
- W podkatalogach — linki symboliczne do plików urządzeń blokowych.

**Podkatalogi /dev/disk/**

- `by-uuid` — uuid systemów plików
- `by-partuuid` — uuid partycji
- `by-label` — etykiety systemów plików
- `by-partlabel` — etykiety partycji
- `by-path` — ścieżka obsługi w hardware (rodzaj i numer magistrali, sterownik itp.). Zob.: `udevadm test-builtin path_id /sys/block/sda`
- `by-id` — unikatowa nazwa tworzona na podstawie danych z hardware'u (model number, serial number, WWN — 8/16 bajtów, odpowiednik MAC adresu dla dysków)

# Identyfikacja urządzenia blokowego podczas montowania

**Można podać ścieżkę dostępu do:**

- faktycznego pliku urządzenia, np. `/dev/sda`;
- linku wskazującego (być może rekurencyjnie) na faktyczny plik urządzenia, np. `/dev/disk/by-partuuid/834939da-ae1d-f638-80af-09866613b8ed`;
- kostrukcji `LABEL=`, `PARTLABEL=`, `UUID=`, `PARTUUID=`, np. `PARTUUID=834939da-ae1d-f638-80af-09866613b8ed`.

# Ważne opcje montowania

- `ro` — read only (uwaga na księgowanie!)
- `discard` — wysyłaj polecenia TRIM do dysku (SSD). Dawniej polecano `fstrim(8)` jako alternatywę.
- `noatime`, `nodiratime` — nie zapisywać czasu ostatniego odczytu (uwaga na programy `mutt` i podobne).
- `user` — wolno montować zwykłemu użytkownikowi (zwykle CDROM, pendrive itp.). Por. UDisks2 i PolicyKit.
- `mode`, `umask`, `fmask`, `dmask` — nadaj odpowiednie prawa dostępu (plikom, katalogom): bardzo przydatne w przypadku FAT (np. `fmask=0177,dmask=0077`).
- `noexec` — zabraniaj uruchamiania programów z tego systemu.
- `nosuid` — zabraniaj uruchamiania programów z SUID.
- `noauto` — nie montować automatycznie w czasie rozruchu (`fstab`)
- `defaults` — w razie braku opcji (`fstab`)

# Moje rozwiązanie

- Każdy dysk zewnętrzny, pendrive, karta SD itp. *dysk*, ma własny podkatalog `/media/`*dysk*`/`.
- Domyślne prawa dostępu do `/media/`*dysk*`/` to `000` (zabezpiecza przed omyłkowym pisaniem do katalogu bez zamontowanego urządzenia).
- Dysk *dysk* ma własny wpis w pliku `fstab` bazujący na `UUID` lub `LABEL`, jeśli trzeba, to z opcją `user`.
- Zawsze wiadomo, gdzie dany dysk się znajduje i czy jest zamontowany.
- Jak sprawdzić automatycznie, czy dysk *dysk* jest zamontowany?

# Przykład: montowanie /tmp

## ręcznie

```
# mount -t tmpfs none /tmp -o size=2000m,mode=1777,strictatime
```

## /etc/fstab

```
none /tmp tmpfs size=2000m,mode=1777,strictatime 0 0
```

## systemd: jednostka tmp.mount

```
[Unit]
Description=Temporary Directory
ConditionPathIsSymbolicLink=!/tmp
DefaultDependencies=no
Conflicts=umount.target
Before=local-fs.target umount.target

[Mount]
What=tmpfs
Where=/tmp
Type=tmpfs
Options=mode=1777,size=2000m,strictatime

[Install]
WantedBy=local-fs.target
```

# Praca z jądrem obcym

- Zamontowany system plików może być dowiązany w innych punktach montowania.
- Ważne dla `chroot` — proces *chrootowany* potrzebuje dostępu do `/sys`, `/proc` i `/dev`.
- Często przydatne podczas ratowania systemu: jądro systemu ratunkowego użyczone dla *userlandu* systemu ratowanego.

## *W-chroot-owanie* w system ratowany

```
# mkdir /target
# mount /dev/rootfs-ratowanego /target
# for FS in proc sys dev; do \
      mount -o rbind /$FS /target/$FS; done
# chroot /target
# jesteśmy w systemie ratowanym
```

- Psudosystemy `proc` i `sys` można tworzyć osobno.
- Czasem trzeba montować także `run`.

# Specjalne sposoby montowania

**Tworzenie albo dowiązywanie pseudosystemów plików**

- `mount -t proc` *`cokolwiek`* `/target/proc` — tworzy *nowy* pseudosystem plików.
- `mount -o bind /proc /target/proc` albo
  `mount --bind /proc /target/proc` — używa istniejącego.

**Dowiązanie rekurencyjne**

- Pseudosystemy plików mają wewnątrz zamontowane kolejne pseudosystemy, np.
  `/dev/pts`
  `/dev/shm`
  `/dev/mqueue`
  `/dev/hugepages`
  `/sys/kernel/security`
  `/sys/kernel/debug`
  `/sys/firmware/efi/efivars`
  także liczne związane z cgrupami.
- Opcja `-o rbind` lub `--rbind`

# Loopback

`losetup(8)`

- Tworzy urządzenie blokowe z obrazu w pliku.
- Opcja `-f`: znajdź wolny numer *n* urządzenia `/dev/loop`*n*.
- Opcja `--show`: wypisz nazwę utworzonego urządzenia.

## Montowanie obrazu płyty ISO

```
# mount $(losetup -f --show plyta.iso) /mnt
```

## Skrót: opcja `loop` programu `mount`

```
# mount -o loop plyta.iso /mnt
```

# Partycje zaszyfrowane

- W Linuksie przeważnie `dm-crypt` z nagłówkiem LUKS.
- Narzędzie: `cryptsetup`.
- Idea: partycja zaszyfrowana jest mapowana na partycję wirtualną w `/dev/mapper/`.
- Konfiguracja: `/etc/crypttab`.

## Montowanie systemu zaszyfrowanego

```
# cryptsetup open /dev/dysk nazwa
Pytanie o hasło
# mount /dev/mapper/nazwa /punkt_montażowy
```

## Odmontowywanie systemu zaszyfrowanego

```
# umount /punkt_montażowy
# cryptsetup close nazwa
```

**Rodzaje kontroli dostępu**

- *Mandatory*: scentralizowana, prawa przydziela aministrator.
- *Discretionary*: rozproszona. Obiekty mają właścicieli. Prawa przydziela właściciel obiektu.
- W Uniksie stosuje się oba modele.

**Kontrola dostępu do plików**

- Każdy plik ma właściciela i grupę.
- Każdy plik ma *access mode*: 12 flag określających możliwe rodzaje dostępu.
- Niektóre systemy plików wspierają też atrybuty, *extended attributes* i ACL (*Access Control Lists*).

# Tryb dostępu i typ pliku

4 najstarsze bity

| | | |
|---|---|---|
| 1 | p | pipe |
| 2 | c | character |
| 4 | d | directory |
| 6 | b | block |
| 8 | - | regular |
| a | l | link |
| c | s | socket |

12 bitów zapisywanych tradycyjnie ósemkowo

| | |
|---|---|
| 4000 | set user ID |
| 2000 | set group ID |
| 1000 | sticky bit |
| 0400 | read by owner |
| 0200 | write by owner |
| 0100 | execute program / search directory by owner |
| 0040 | read by group |
| 0020 | write by group |
| 0010 | execute/search by group |
| 0004 | read by others |
| 0002 | write by others |
| 0001 | execute/search by others |

# Szczególne tryby dostępu

**Działający proces ma prawa pewnego użytkownika i pewnej grupy**

- user i group: *real*, *effective*, *saved*, *filesystem*
- `execve(2)` ustala podczas uruchamiania procesu, uwzględniając flagi SETUID i SETGID.
- Możliwa dynamiczna zmiana za pomocą `set(|e|re)uid(2)` i `set(|e|re)gid(2)`
- Uwaga: SETUID i SETGID są niebezpieczne! Nie działają dla skryptów.

**Flagi modyfikujące prawa dostępu**

- SETUID — dla plików wykonywalnych.
- SETGID — dla plików wykonywalnych i katalogów (zob. np. grupa `staff` i prawa `drwxrwsr-x` dla `/usr/local/`).
- *Sticky bit* (dla katalogów, dawniej też zwykłych plików).
- Prawa dostępu do dowiązań symbolicznych są ignorowane i nie można ich zmienić za pomocą polecenia `chmod`.

# Alternatywy dla niebezpiecznej flagi `setuid`

**Capabilities**

- Atrybuty plików wykonywalnych, zob. `capabilities(7)`.
- Pozwalają na selektywne nadawanie uprawnień.
- API jądra: `capset(2)`, `capget(2)`, biblioteka `libcap`, narzędzia: `setcap(8)`, `getcap(8)`.
- Przykłady: `CAP_NET_RAW`, `CAP_SYS_NICE`, `CAP_SYS_TIME` itp.

**Inne możliwości**

- `su(1)`, `sudo(1)`, `doas(1)` — impersonacja (pełna, selektywna).
- PolicyKit — "the sudo of systemd" (zło!)

# Ustawianie praw dostępu

**Domyślna maska dostępu podczas kreacji plików**

- Syscall `umask(2)` — maska dostępu dla procesu. Używana podczas `open(2)` i `mkdir(2)`.
- Uwaga: maska odwrotna (*wildcard mask*) — gasi wybrane bity w podanym trybie dostępu.
- Np. jeśli umask = `00077`, to `open("file", O_WRONLY|O_CREAT, 00644)` utworzy plik z prawami dostępu `00644 & ~00077 = 00600`.
- Uwaga: literał całkowitoliczbowy w C jest ósemkowy jeśli zaczyna się zerem — stąd dodatkowe zero z przodu.
- Proces potomny (`fork(2)`) dziedziczy maskę.
- Polecenie powłoki `umask` — ustala maskę dla powłoki (i pośrednio dla wszystkich procesów uruchamianych przez powłokę).
- Domyślna wartość: `0022`. Można zmienić np. na `0077`.
- Dla procesów uruchamianych przez SystemD: podawać `UMask=`.

**Zmiana dla istniejących plików**

- `chmod(1)`, `chgrp(1)` — zmiana praw dostępu i grupy istniejącego pliku.

# Tryb dostępu — zapis literowy `rwx`

**Podczas ujawniania informacji**

- Ciąg 10 symboli ze zbioru `-dlbcprwxsStT`.
- Stosowany przez programy `ls`, `stat` itp.
- Dodatkowo informacja o typie pliku (`-dlbcps`): pierwszy symbol.
- SUID: `s` zamiast `x` w pierwszej trójce (`S` jeśli brak `x`).
- SGID: `s` lub `S` zamiast `x` w drugiej trójce.
- *sticky bit*: `t` lub `T` zamiast `x` w trzeciej trójce.
- Przykłady: `-rws--S---`, `drwxrwxrwt`.

**W programie chmod**

- Składnia: `[ugoa]*([-+=]([rwxXst]*|[ugo]))+|[-+=][0-7]+`
- Bardziej elastyczne, niż zapis ósemkowy, szczególnie w połączeniu z opcją `-R`.
- `X` oznacza `x` tylko dla katalogu lub jeśli już był.
- Przykład: `chmod -R go-wx,go+rX *`

# Listy kontroli dostępu

**Access Control Lists (ACL)**

- Pozwalają na przydzielanie uprawnień *per* użytkownik, zob. `acl(5)`.
- Uwaga: nie zawsze są dostępne. Trzeba włączyć podczas montowania.
- Narzędzia: `setfacl(1)`, `getfacl(1)`.
- Działają dla plików zwykłych. Dla katalogu: domyślna lista dla plików tworzonych w tym katalogu.
- Przykład: `user:tomasz:rw-`
- Skomplikowane, rzadko używane.

# Atrybuty plików

| | |
|---|---|
| a | append only |
| A | no atime updates |
| c | compressed |
| C | no copy on write |
| d | no dump |
| D | synchronous directory updates |
| e | extent format |
| i | immutable |
| j | data journalling |
| s | secure deletion |
| S | synchronous updates |

| | |
|---|---|
| t | no tail-merging |
| T | top of directory hierarchy |
| u | undeletable |

Read-only attributes

| | |
|---|---|
| E | compression error |
| h | huge file |
| I | indexed directory |
| N | inline data |
| X | compression raw access |
| Z | compressed dirty file |

Narzędzia: `lsattr`, `chattr`.

# Atrybuty rozszerzone

**Extended attributes**

- Pary *nazwa=wartość* przyporządkowane do pliku.
- Narzędzia: `setfattr(1)`, `getfattr(1)`, zob. `xattr(7)`.
- Cztery klasy: `security`, `system`, `trusted`, `user`.
- Np. `wget(1)` i `curl(1)` z opcją `--xattr` zapisują atrybuty

  `user.xdg.origin.url=`*URL-ściągniętej-strony*
  `user.xdg.referer.url=`*URL-strony-z-linkiem-do-ściągniętej-strony*
- XDG definiuje wiele atrybutów: kodowanie, język, *creator*, *producer* itp.
- To są metadane przechowywane *poza* zawartością pliku.

**Alternatywa: metadane przechowywane wewnątrz pliku — exif**

- Wiele formatów: PDF, muzyka (MP3), obrazy (JPEG) filmy (MP4, AVI).

# Dostęp użytkownika nieuprzywilejowanego do katalogów

**Zapis możliwy tylko do:**

- `/home/`*`user`*`/`
- `/tmp/`, `/var/tmp/`
- `/var/mail/`*`user`*
- I niewiele więcej, ale uwaga na drobiazgi, np.:
  `drwx-wx--T root crontab /var/spool/cron/crontab`

**Odczyt**

- Katalogi systemowe `/usr/`, `/var/` itp.
- Uwaga: domyślenie `drwxr-xr-x` dla katalogu `/home/`*`user`*`/`
- Domyślnie każdy użytkownik ma swoją grupę. Możliwość tworzenia dodatkowych *working groups*.
- Dobra izolacja danych różnych użytkowników, ale uwaga na dane spoza `/home/`*`user`*`/`.
- Warto zakładać dla siebie wiele kont w celu separacji danych.