Pliki i systemy plików.

Paweł Józiak

Wydział Matematyki i Informatyki Politechniki Warszawskiej

22 XI 2021

Plan zajęć

Plan zajęć

Struktura informacji o pliku - *i-węzeł*

- `i-węzeł` (*i-node*) to struktura przechowujaca informacje o pliku z wyjątkiem jego nazwy oraz danych, tj.
  - typ,
  - wielkość,
  - właściciela i grupę,
  - uprawnienia,
  - znaki czasowe: ostatnia zmiana i-węzła, ostatnia modyfikacja pliku (*ctime* oraz *mtime*), a także czas ostatniego dostępu (*atime*),
  - ID urządzenia dyskowego oraz wskaźniki na bloki dysku,
  - liczba linków (tj. twardych linków, zwykle 1).

Uzyskiwanie informacji o pliku – *stat*

- `stat` wyświetla wszystkie informacje o pliku zapisane w jego strukturze i-węzła.

Przykładowy rezultat *stat*

```
$ stat plik
  File: plik
  Size: 7               Blocks: 8          IO Block: 4096   regular file
Device: 805h/2053d      Inode: 10765343    Links: 1
Access: (0666/-rw-rw-rw-)  Uid: ( 1000/   pawel)   Gid: ( 1000/   pawel)
Access: 2021-11-17 13:59:31.986141442 +0100
Modify: 2021-11-21 22:22:11.262012815 +0100
Change: 2021-11-21 22:22:14.146006665 +0100
 Birth: -
```

- Nazwa pliku wyświetlana w pierwszej linii to argument komendy stat, nie zaś część struktury i-węzła!

Struktura informacji o pliku - *i-węzeł* II

- Jednolita struktura używana do zwyczajnych plików, linków symbolicznych, katalogów i innych obiektów systemu plików (por. następne slajdy).
- Tworzona i trzymana w pamięci w momencie udostępniania systemu plików (*montowanie*).
- Niezależna od typu systemu plików oraz implementacji.
- Stanowi logiczny interfejs do zasobów dyskowych.
- Przechowuje jedynie informacje numeryczne: właściciel pliku, grupa, uprawnienia są przechowywane jako liczby.

Nazwy plików i katalogi

- I-węzły nie zawierają nazwy pliku.
- Obiekty systemu plików są identyfikowane przez tzw. i-numery (numery i-węzła).
- Katalogi to obiekty systemu plików, które przechowują listę par *(nazwa, i-numer)*.

Katalogi a i-numery

```
$ ls -i1 /etc/cron.d/
4981511 anacron
4987301 e2scrub_all
4981161 php
4981512 popularity-contest
```

- Wiele nazw może być powiązanych z pojedynczym i-numerem. Wówczas mówimy o twardych linkach do pliku.
- Liczba twardych linków jest przechowywana w i-węźle. Jeśli ta wartość osiągnie 0, zakres pamięci jest uwalniany.

Znaki czasu

- `Access` : czas ostatniego odczytu treści pliku. W wielu systemach plików wyłączony ze względu na aspekty wydajnościowe.
- `Modify` : czas ostatniej modyfikacji pliku, tj. zmiany treści pliku.
- `Change` : czas ostatniej zmiany pliku, tj. zmiany metadanych pliku – nazwy, uprawnień, położenia etc.
- `Birth/Create` : w systemach Linux pole niezaimplementowane.

Typy plików

- Najczęściej spotykane typy plików to:
  - `-` – zwykły plik,
  - `d` – katalog,
  - `l` – link symboliczny,
  - `b` – urządzenie blokowe,
  - `c` – urządzenie znakowe
  - `p` – tzw. *pipe*, poza zakresem kursu
  - `s` – gniazdo (*socket*), poza zakresem kursu.
- Wiele komend operuje na plikach. Co do zasady, powinny działać jednakowo na każdym typie pliku.

Urządzenia blokowe

- Pozwalają na dowolny, buforowany dostęp do zawartości urządzenia, która reprezentowana jest przez wielo-bajtowe jednostki – bloki.
- Zwykle reprezentują urządzenia pamięci trwałej.
- Linux (lecz niekoniecznie Unix) używa następującej konwencji nazewniczej
  - `/dev/sda` , `/dev/sdb` , ... — dyski (SATA, PATA, USB, SCSI, sprzętowe wolumeny RAID),
  - `/dev/sda1` , `/dev/sda2` , `/dev/sda3` , `/dev/sda4` — fizyczne partycje na pierwszym dysku,
  - `/dev/sda5` , `/dev/sda6` , ... – logiczne dyski w rozszerzonym schemacie partycjonowania na pierwszy dysku,
  - `/dev/md0` , `/dev/md1` , ... -– programowe wolumeny RAID (zob. `mdadm`),
  - `/dev/sr0` — dysk optyczny (zwykle również pod symlinkiem `/dev/cdrom`).

Znajdowanie urządzeń blokowych

```
$ ls -l /dev | grep -P '^b' # za pomocą wyrażeń regularnych
$ lsblk # Dedykowaną komendą
```

Urządzenia znakowe

- Pozwala na strumieniowy (sekwencyjny) dostęp do zawartości, która reprezentowana jest przed jednoznakowe – tj. jednobajtowe – jednostki.
- Reprezentuje fizyczne urządzenia, TTY, a także niektóre systemowe generatory:
  - `/dev/null` — "czarna dziura", przyjmuje i odrzuca każde wejście, nie produkuje żadnego wyjścia;
  - `/dev/random` oraz `/dev/urandom` – generatory pseudolosowego strumienia znaków (`random` — lepsza losowść, `urandom` — szybszy),
  - `/dev/zero` — produkuje ciągły strumień bajtów 0 na wyjściu, przyjmuje i odrzuca każde wejście;
  - `/dev/full` – produkuje ciągły strumień bajtów 0 na wyjściu, produkuje *ENOSPC* (*No space left on device*) przy próbie zapisu.

✍ *Pisanie na TTY to pisanie do pliku (wymaga uprawnień roota)*

Wysyłanie komunikatu na TTY

```
# echo "Hello" > /dev/tty8
```

Plan zajęć

Pliki ukryte

- W systemie UNIX nie ma atrybutów specjalnych bądź ukrytych.
- Pliki, których nazwa zaczyna się od `'.'` (kropki) są uważane za ukryte i większość komend domyślnie ich nie używa. W szczególności, pliki te nie łapią się do jokera `'*'` – specjalny joker `'.*'` musi być dla nich używany.
- Czasem puste pliki nazwane np. `.keep` bądź podobnie, są umiejscawiane w katalogach, które chcemy chronić przed przypadkowym usunięciem.

✍ *Domyślnie, `rm` nie usuwa plików ukrytych.*

Usuwanie plików ukrytych w katalogu

```
$ rm * # nie usuwa plików ukrytych
$ rm * .* # usuwa wszystkie pliki, także ukryte
```

Obliczanie zajętości dysku katalogów – *du*

- `ls -l` oraz `stat` obliczają wielkość katalogu jako obiektu systemu plików, nie jako suma wielkości jego zawartości.
- Komenda `du` wyświetla zliczoną rekursywnie zawartość katalogu. Z flagą `-s` wyświetla tylko sumę. Z flagą `-h` (*human readable*) wyświetla tę wielkość zaokrągloną do adekwatnego rzędu wielkości (kilobajty, megabajty, gigabajty etc),
  ✍ *Zwróć uwagę, że `du` uwzględnia również ukryte pliki i katalogi*

Wyświetlanie zajętości dysku za pomocą *du*

```
$ du -sh ./katalog
2.0M    ./katalog
$ du -h ./katalog
1.5M    ./katalog/podkatalog1
470K    ./katalog/.podkatalog2
2.0M    ./katalog
```

Zaawansowane zarządzanie plikami – *lsof* oraz *dd*

- `lsof` listuje aktualnie otwarte pliki. Dla każdego pliku wypisuje, który proces (jego ID, właściciela i grupę) go używa.
- `lsof` *`plik`* wyświetla informacje jedynie o wskazanym pliku.
- Pomaga rozwiązać problemy *File already in use* mechanizmu zamka.

Syntaks *dd*

```
# dd if=zrodlo of=destynacja [count=ile_blokow] [bs=wielkosc_bloku]
```

- `dd` kopiuje bloki w liczbie `count` , o wielkości `bs` , z pliku wejściowego `if` do pliku wyjściowego `of` .
- Komenda współpracuje równie dobrze ze zwykłymi plikami, jak i urządzeniami blokowymi i urządzeniami znakowymi.
- Jeśli `bs` nie jest określony, domyślna jednostka jest używana (zwykle: 512B, wielkość jednego sektora dysku). Większe wartości przyspieszają operację.
- Jeśli parametr `count` nie jest określony, bajty są kopiowane aż odczytany będzie znak EOF.

dd – przykłady

Zachowanie MBR do pliku

```
# dd if=/dev/sda of=/root/mbr.bak count=1 bs=512
```

Nadpisanie dysku ciągiem zer/losowych znaków

```
# dd if=/dev/zero of=/dev/sdb
# dd if=/dev/urandom of=/dev/sdb
```

Utworzenie pliku tymczasowej pamięci dyskowej SWAP

```
# dd if=/dev/zero of=/swapfile bs=1024 count=524288
```

Utworzenie boot-owalnego USB z obrazu ISO LiveDVD

```
# dd if=liveDVD.iso of=/dev/sdb
```

Plan zajęć

Systemy Plików Unix

- Choć zewnętrzne systemy plików (na czele z FAT oraz NTFS) są wspierane przez współczesne Linuxy i systemy *BSD, najczęstsze w użyciu są jednak dedykowane systemy plików:
- `ext` , `ext2` , `ext3` oraz `ext4` (*extended file system*) — domyślny w większości współczesnych dystrybucji Linuxa,
- `UFS` (*Unix File System*) lub `FFS` (*Berkeley Fast File System*) — domyślne w systemach *BSD oraz Solaris (choć istnieją różnice implementacyjne), wielu dostawców adaptuje je do własnych potrzeb,
- `AdvFS` (*Tru64 UNIX Advanced File System*) — dla systemu Tru64,
- `JFS` (*Journaled File System*) – stworzony dla IBM AIX,
- `ReiserFS` (niegdyś wielka nadzieja, dziś nieutrzymywany – dlaczego? można znaleźć w portalach plotkarskich),
- `VxFS` (*Veritas File System*) — stworzony dla HP-UX,
- `XFS` – pierwotnie dla IRIX,
- `btrfs` -– nowy, obiecujący System Plików dla Linuxa (niedostatecznie stabilny jednak, by oprzeć na nim rozwiązania produkcyjne),
- `HAMMER2` — jeden z bazowych wyróżników systemu DragonFly BSD.

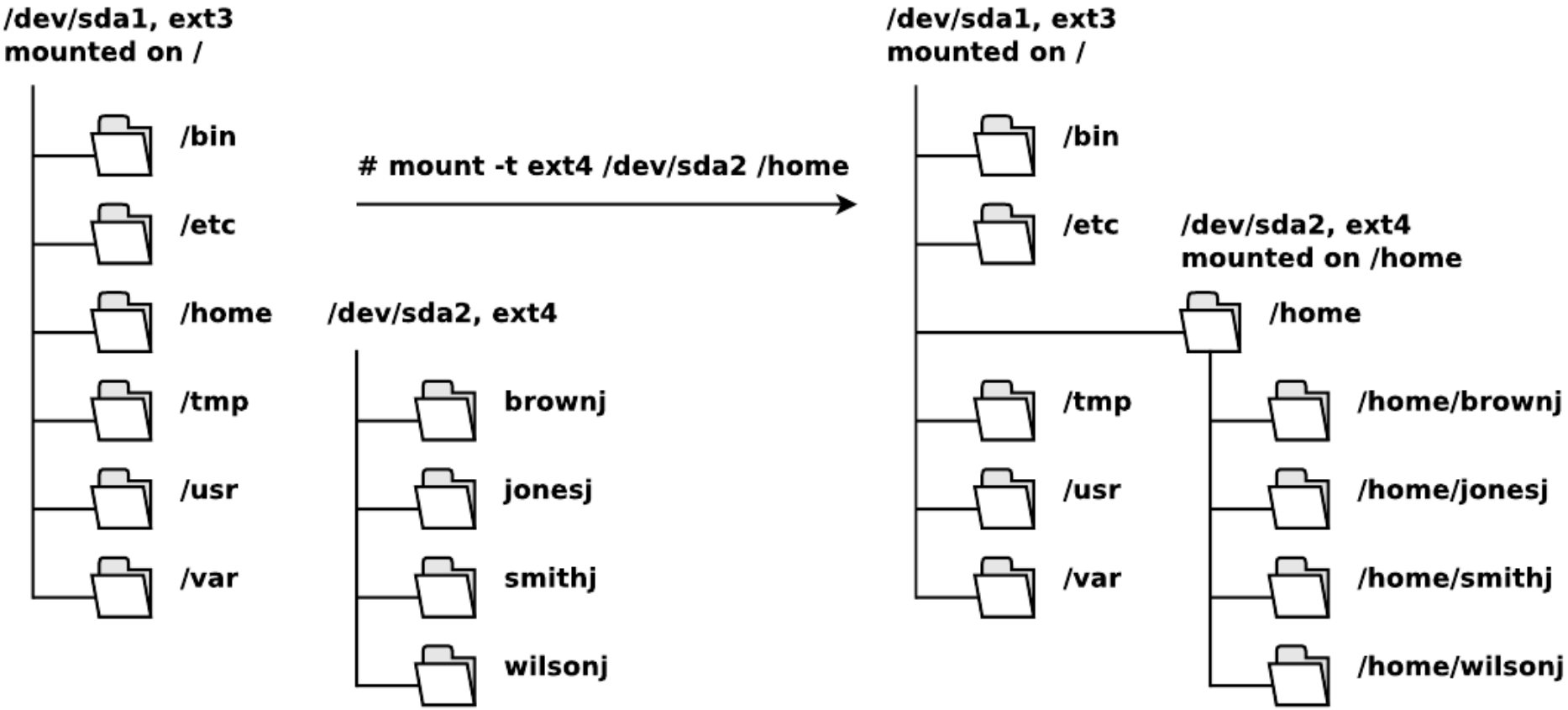
Po co systemy plików?

- Wybór konkretnego systemu plików to zagadnienie związane z wydajnością, stabilnością, bezpieczeństwem, tolerancją błędów, indeksowaniem i innymi cechami systemu plików.
- Systemy plików są nierozróżnialne dla użytkowników oraz programistów.
- Ta sama struktura i-węzłów będzie używana, niezależnie od typu systemu plików.
- Warto przemyśleć wybór systemu plików dla dedykowanych, niestandardowych, zastosowań:
  - wiele małych plików,
  - niewiele wielkich plików,
  - historia systemu plików, audyty bądź wymagania specjalnej obsługi błędów,
  - częste operacje I/O,
  - i inne.
- Podstawowym systemem plików w gros dystrybucji Linuxa jest `ext4` ,
- Jego poprzednicy, `ext2` / `ext3` , nie dorównują mu wydajnościowo, lecz pewne narzędzia *ratowania i odzyskiwania* będą dostępne.

Montowanie systemu plików

- Aby udostępnić zasoby systemu plików, musi on być zamontowany.
- Jedynie jeden system plików może być zamontowany w korzeniu drzewa katalogów (/) – tzw. podstawowy system plików.
- Zamontowanie pewnego systemu plików oznacza, że jego lokalny folder-korzeń jest zidentyfikowany z katalogiem istniejącym w dotychczasowym systemie plików. W ten sposób widzimy systemy plików jako gałęzie struktury hierarchicznej.
- Jeśli katalog jest używany jako plik montowania, jego dotychczasowa zawartość staje się niedostępna.
- System plików rodzica i dziecka mogą się różnić.

Montowanie systemu plików – przykład



Montowanie pseudo-systemu plików

- Poza montowaniem zwykłych systemów plików, niektóre przestrzenie adresowe bądź zestawy danych mogą być montowane i udostępnione w systemie, jakby były systemami plików.
- Dzięki temu Unix pozwala na dostęp do wszelkich zasobów w tym jądra i procesów (`procfs`), urządzeń (`devfs`) etc za pomocą standardowego interfejsu systemu plików i narzędzi z tym związanych.
- Zwykle katalogi: `/dev` , `/proc` oraz `/sys` zawierają zamontowane takie pseudo systemy plików (przestrzenie adresowe).

Montowanie systemu plików – *mount*

Syntaks *mount*

```
# mount [-t type] [-o options] device mountpoint
```

- Typ systemu plików
  - Zwykle flagę `-t` można swobodnie ominąć i zdać się na autodetekcję.
  - Są dwa sterowniki Linuksa dla NFS: `-t ntfs` (domyślny) oraz `-t ntfs-3g` (rekomendowany),
  - `-t iso9660` pozwala montować obrazy ISO (w trybie do odczytu).
- Wśród popularnych opcji można znaleźć:
  - `-o acl` bądź `-o acls` – uruchamia wsparcie dla ACL,
  - `-o noatime` – bez aktualizacji *atime* plików (zwiększa wydajność),
  - `-o nodev` – montowanie bez plików urządzeń ,
  - `-o noexec` – blokuje wykonywanie programów z tego systemu plików,
  - `-o nosuid` – ignoruje efekt *setuid*/*setgid*,
  - `-o ro` – montuje w trybie do odczytu.
- Opcje można grupować, np. `-o acl,noatime,nodev,nosuid` .
- Opcje można zmienić dla zamontowanego systemu: `-o remount` .

Odmontowywanie systemem plików – *umount*

Syntaks *umount*

```
# umount device|mountpoint
```

- Pewne systemy plików (`ext3` , `ext4` , `tmpfs` etc) można montować kilkukrotnie w różnych punktach montowania w tym samym czasie.
- Wskazanie punktu montowania zamiast nazwy urządzenia może uchronić przez niejednoznacznością.

Tabele montowania – */etc/fstab*

- `/etc/fstab` zawiera listę systemów plików do zamontowania przez `mount -a` (wywoływanym przy starcie systemu).

Syntaks */etc/fstab*

```
device  mountpoint  type  options  dump/pass
```

- Znaczenie pól jest następujące:
  - `device` – urządzenie blokowe bądź adres zdalnego systemu plików do zamontowania,
  - `mountpoint` – punkt montowania (albo `none` dla SWAP),
  - `type` – typ systemu plików, albo `auto` ,
  - `options` – rozdzielone przecinkiem opcje montowania systemu plików; `noauto` wymusza montowanie ręczne (nie za pomocą `mount -a`), `user` zezwala na montowanie przez użytkownika,
  - `dump/pass` – flagi dla narzędzi pomocniczych.
- Uwaga! Współczesne Linuksy do zastosowań desktopowych automatycznie montują przez `udev` wyciągalne nośniki (CD, pendrive etc). Wpisy `/etc/fstab` dla nich mogą doprowadzić do nieprzewidywalnych wyników!

Tabele montowania – */etc/mtab*

- `/etc/mtab` zawiera informacje o aktualnie zamontowanych systemach plików,
- Zwykle jest to symlink do pliku w pseudo-systemie plików `/proc` (np. `/proc/self/mounts`) albo `/sys` .
  ✍ *Poniższe dwie komendy powinny wyświetlić tę samą informację*

Listowanie systemów plików i ich punktów montowania

```
$ mount
$ cat /etc/mtab
```

Plan zajęć

Podstawowe zagadnienia Hierarchi Systemu Plików

- Separacja przestrzeni Systemu oraz przestrzeni Użytkownika – osobne katalogi dla plików krytycznych systemu, narzędzi systemowych oraz programów użytkownika (aplikacji, serwisów itd)
- Katalogi zawierające pliki użytkownika oraz dane użytkownika mogą być rozłożone na różne systemy plików i odmontowane jeśli będzie taka potrzeba.
- Pliki z prawem `read-write` oraz `read-only` leżące w osobnych katalogach pozwalają na optymalizację przez użycie innych typów systemów plików dla nich.
- Uproszczone zarządzanie uprawnieniami dla osobnych katalogów dla: binarek (programów), plików konfiguracyjnych, bibliotek, danych oraz logów.

Hierarchia Systemu Plików

- Poniższe katalogi występują w większości systemów Uniksowych:
  - `/bin` — binarki (Uniksowa nazwa na programy),
  - `/boot` – pliki inicjalizacyjne,
  - `/dev` – *devices*, urządzenia,
  - `/etc` – *edit-to-configure*, konfiguracja systemowa,
  - `/home` – katalog katalogów domowych (zwykłych użytkowników),
  - `/lib` – uwspólnione bilioteki,
  - `/mnt` – tymczasowe punkty montowania,
  - `/opt` – opcjonalne (dodatkowe) oprogramowanie,
  - `/proc` – przestrzeń adresowa procesów i informacji o jądrze,
  - `/root` – katalog domowy użytkownika `root`,
  - `/sbin` – binarki *superużytkownika*, *specjalne* bądź *systemowe*,
  - `/sys` – system podobny do `/proc` ,
  - `/tmp` – pliki tymczasowe,
  - `/usr` – zasoby systemowe o charakterze niekrytycznym,
  - `/var` – zmienne dane.

Idee przyświecające Partycjonowaniu

- Powody dla używania wielu partycji w systemie to m.in:
- zabezpieczenie przed przepełnieniem dysku,
- różne opcje montowania dla zwiększonego bezpieczeństwa (`ro` , `nosuid` , `noexec` , ...) and oraz używalności (`acl` etc),
- *quota* na jedynie wybranych systemach plików,
- uproszczone zarządzanie problemami (selektywne i/lub równoległe diagnostyki/naprawy systemów plików),
- optymalizacja wydajności (różne typy dysków, systemów plików etc),
- zróżnicowanie strategii tolerancji błędów i backupów,
- Używanie wielu systemów plików w maszynach desktopowych bywa często kwestionowane.

Rekomendowany schemat partycjonowania

- Schemat partycjonowania powszechnie praktykowany składa się z:
  - `/boot` – nie montowany po uruchomieniu systemu,
  - `/` – główna partycja ze wszystkimi krytycznymi zasobami,
  - `/home` – montowana z `nosuid,acl` ; czasem używa się dla `/home` zewnętrznej macierzy dyskowej,
  - `/usr` – montowane jako `ro` za wyjątkiem aktualizacji systemu,
  - `/var` – system plików zoptymalizowany pod częsty odczyt/zapis; warto rozważyć również opcję `noexec` .