Politechnika Wrocławska

Systemy operacyjne 2

Laboratorium nr 3

Operacje na dowiązaniach

Szymon Datko

szymon.datko@pwr.edu.pl

Wydział Informatyki i Telekomunikacji,
Politechnika Wrocławska

semestr letni 2022/2023

HR EXCELLENCE IN RESEARCH

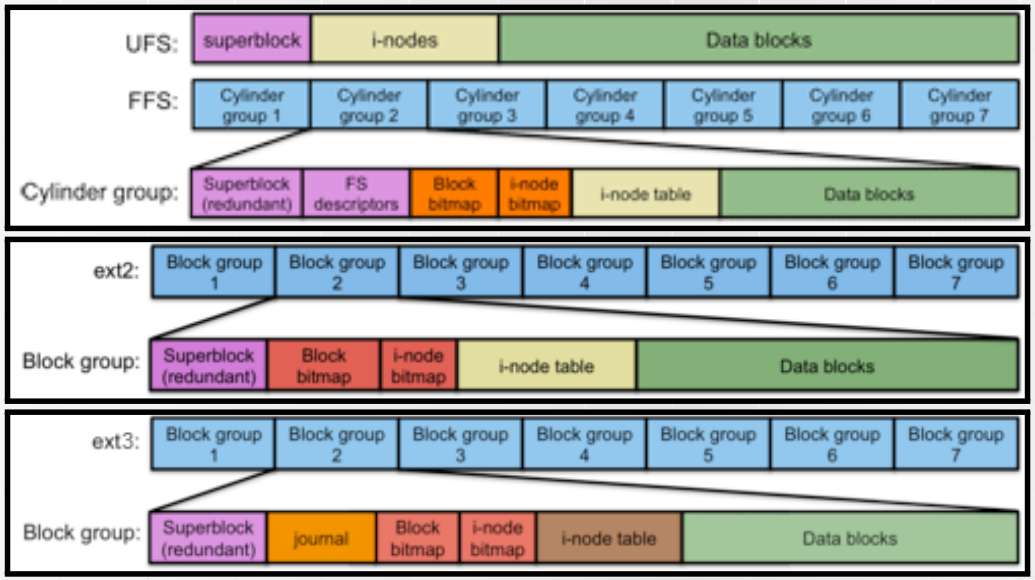
Politechnika Wrocławska

Część I

Omówienie zagadnień

Budowa systemu plików

- System plików opisuje sposób organizacji plików w ramach partycji.
- W uproszczeniu składa się z nagłówka, tablicy i-węzłów i bloków danych.
- Współczesne systemy dzielą dodatkowo całość na cylindry/grupy bloków.
  - Oryginalny pomysł zakładał ograniczenie ruchów głowicy dysku twardego.
  - Pojęcie cylindra zastąpiono lepiej pasującym do współczesnych nośników.

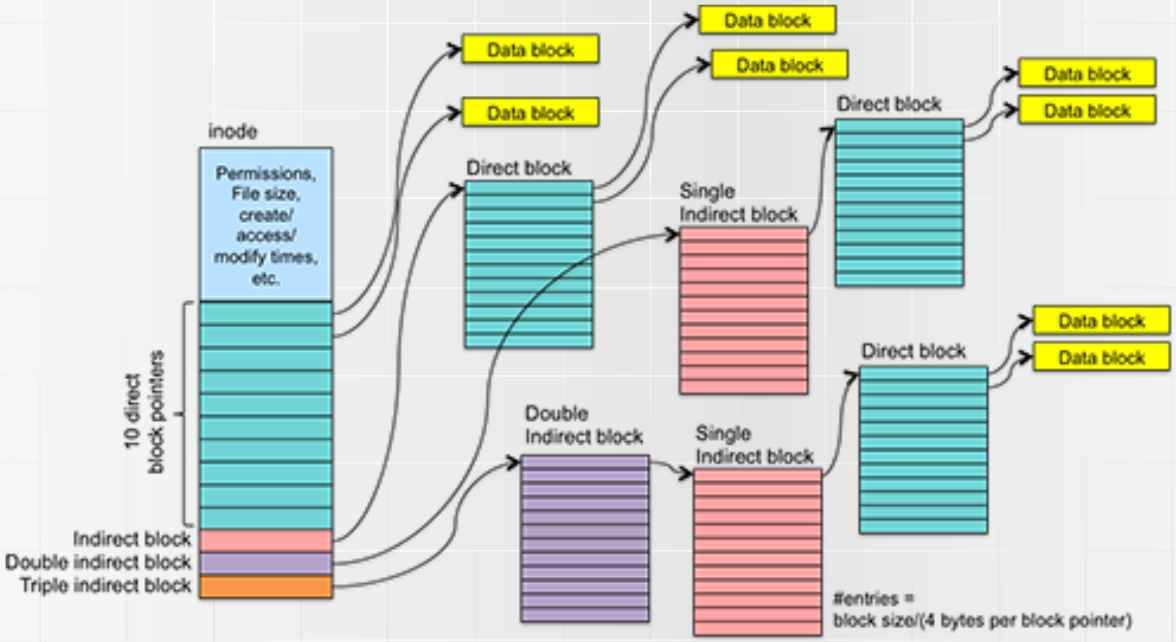


Źródło: https://www.cs.rutgers.edu/~pxk/416/notes/13-fs-studies.html

Inny ciekawy artykuł: https://prog.world/how-to-write-your-own-filesystem-in-rust/

Budowa systemu plików – i-węzły

- Metadane zawierają najważniejsze informacje o samym pliku.
  - Rodzaj pliku, uprawnienia, czasy dostępów, itd. (Ale nie nazwa!)
- Następnie zapisane są adresy fragmentów zwartości w blokach danych.
  - W UFS było to 10 bezpośrednich adresów, zaś w FFS i kolejnych już 12.
- Na końcu znajdują się adresy do bloków pośrednich (1-, 2- i 3-krotnych).
  - Wskazują adresy do bloków danych albo do kolejnych bloków pośrednich.



Źródło: https://www.cs.rutgers.edu/~pxk/416/notes/13-fs-studies.html

Rodzaje i typy/formaty plików

- W systemie Linux (prawie) wszystko jest plikiem.
- Jeśli coś nie jest plikiem, to jest procesem (który jest częściowo zmapowany w `/proc`).

Podstawowe **rodzaje** plików obejmują:

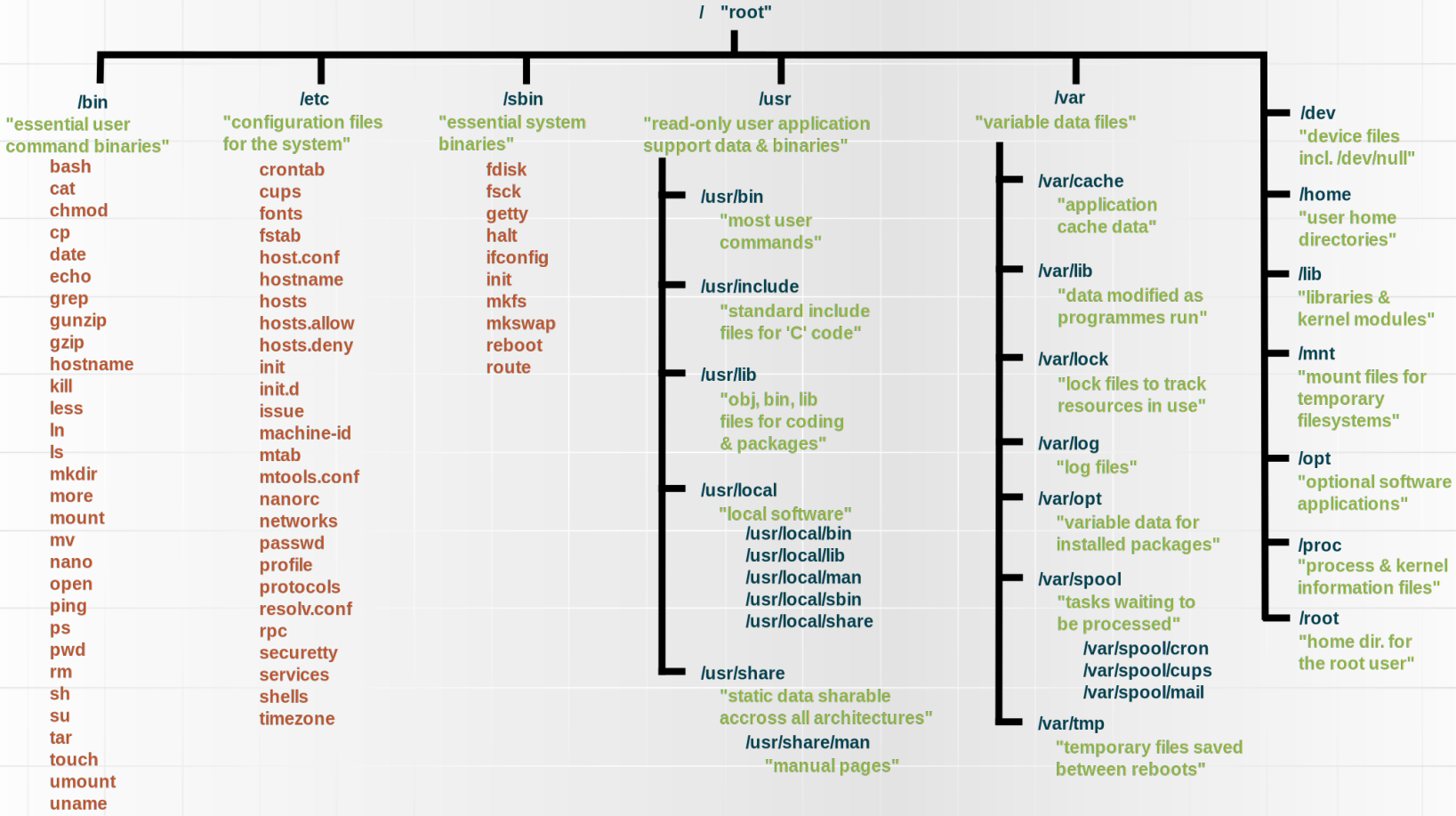
- **pliki zwykłe** – `-` – *regular file*,
  - binarne lub tekstowe,
- **katalogi** – `d` – *directory*,
  - ogólnie: pliki zawierające spis zawartych w nich plików (nazwy i i-węzły),
- dowiązania miękkie – `l` – *symbolic link*,
- gniazdka sieciowe – `s` – *socket*,
- urządzenie znakowe – `c` – *character device*,
- urządzenie blokowe – `b` – *block device*,
- łącza nazwane – `p` – *named pipe*.

Tak zwane rozszerzenie jest tylko i wyłącznie fragmentem nazwy pliku.

- To zawartość określa właściwy **typ**/**format** pliku (archiwum, obraz, itd.).
  - Słowo więcej na ten temat: kurs Inżynieria Obrazów, slajdy do Lab4.
- Do określenia typu/formatu pliku można wykorzystać program `file`.

Standardowa struktura katalogów

- Powszechnie stosowany jest *Filesystem Hierarchy Standard* lub wariacje.



+ Katalog `/sys` – zapewnia dostęp do niektórych funkcji/modułów jądra systemu.
- Komenda dla dociekliwych: `man hier`.

Źródło: https://en.wikipedia.org/wiki/Unix_filesystem

Tworzenie plików i dowiązania twarde

- W skrócie: tworzenie polega na zajęciu i-węzła i zapisaniu zawartości.
- Dowiązanie twarde polega na utworzeniu kolejnego odnośnika do i-węzła.
- Innymi słowy: jest to alternatywna nazwa i położenie dla jakiegoś pliku.
- Ciekawostka – usunięcie pliku polega na usunięciu samego i-węzła.
  - Domyślnie zawartość nie zostaje usunięta (bloki danych niezmienione).
  - Dlatego funkcja systemowa w języku C do kasowania plików to `unlink()`.
- Każdy nowo utworzony katalog posiada od razu dwa dowiązania twarde.
  - Jest to nazwa katalogu oraz wskazanie do samego siebie `.` (kropka).
  - Każdy podkatalog ma dodatkowo dowiązanie `..` do folderu nadrzędnego.
- Nie można tworzyć własnych, dodatkowych dowiązań do katalogów!
  - Powodowało by to ryzyko komplikacji w strukturze systemu plików.
  - Np. nieskończonej pętli zagnieżdżeń i niejasności katalogu nadrzędnego.

Więcej: https://askubuntu.com/questions/210741/why-are-hard-links-not-allowed-for-directories

Tworzenie plików i dowiązania twarde – przykład

```
[sdatko@polluks test]$ mkdir katalog  # utworzenie nowego katalogu

[sdatko@polluks test]$ ls -li .  # zwrócić uwagę na istniejące od razu dwa dowiązania do katalogu
14262663 drwxr-xr-x 2 sdatko sdatko 4096 03-24 12:01 katalog

[sdatko@polluks test]$ ls -lia katalog/  # zwrócić uwagę, że katalog nadrzędny ma już trzy dowiązania
14262663 drwxr-xr-x 2 sdatko sdatko 4096 03-24 12:01 .
14262662 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:01 ..

[sdatko@polluks test]$ mkdir katalog/podkatalog  # stworzenie nowego podkatalogu

[sdatko@polluks test]$ ls -li .  # w katalogu pojawiło się dodatkowe dowiązanie twarde
14262663 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:02 katalog

[sdatko@polluks test]$ ls -lia katalog/  # przez to poprzedni katalog ma teraz trzy dowiązania
14262663 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:02 .
14262662 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:01 ..
14270854 drwxr-xr-x 2 sdatko sdatko 4096 03-24 12:02 podkatalog

[sdatko@polluks test]$ touch katalog/plik1 katalog/plik2  # utworzenie dwóch nowych plików zwykłych

[sdatko@polluks test]$ ln katalog/plik2 katalog/dowiązanie-twarde  # stworzenie dowiązania twardego

[sdatko@polluks test]$ ls -lia katalog/  # zwrócić uwagę na numery i-węzłów i liczbę dowiązań
14262663 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:08 .
14262662 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:01 ..
14177897 -rw-r--r-- 2 sdatko sdatko    0 03-24 12:07 dowiązanie-twarde
14177655 -rw-r--r-- 1 sdatko sdatko    0 03-24 12:07 plik1
14177897 -rw-r--r-- 2 sdatko sdatko    0 03-24 12:07 plik2
14270854 drwxr-xr-x 2 sdatko sdatko 4096 03-24 12:02 podkatalog
```

Dowiązania miękkie

- Inny sposób na nadanie alterantywnej nazwy/położenia jakiemuś plikowi.
- Fizycznie jest nowy plik specjalny, którego zawartość stanowi ścieżka.
  - Może ona wskazywać nie tylko na pliki zwykłe, ale także na katalogi, itd.
  - Wskazywany element może także nie istnieć (dowiązanie wiszące).
- Ścieżka ta może być:
  - bezwzględna – rozpoczynająca się zawsze od katalogu `/`,
  - względna – określona zawsze w odniesieniu do położenia dowiązania.
- Ścieżka kanoniczna – najkrótsza możliwa ścieżka bezwzględna do pliku.
  - Tak: `/etc/systemd/journald.conf`.
  - Nie: `/./dev/../etc/./././systemd/../systemd/journald.conf`.

```
[sdatko@polluks test]$ ln -s plik2 katalog/miękiszon  # stworzenie dowiązania miękkiego

[sdatko@polluks test]$ ls -lia katalog/  # zwrócić uwagę na numery i-węzłów i liczbę dowiązań
14262663 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:08 .
14262662 drwxr-xr-x 3 sdatko sdatko 4096 03-24 12:01 ..
14177897 -rw-r--r-- 2 sdatko sdatko    0 03-24 12:07 dowiązanie-twarde
14170391 lrwxrwxrwx 1 sdatko sdatko    5 03-24 12:24 miękiszon -> plik2
14177655 -rw-r--r-- 1 sdatko sdatko    0 03-24 12:07 plik1
14177897 -rw-r--r-- 2 sdatko sdatko    0 03-24 12:07 plik2
14270854 drwxr-xr-x 2 sdatko sdatko 4096 03-24 12:02 podkatalog
```

Politechnika Wrocławska

Wyszukiwanie dowiązań na dysku

- Najprościej wygląda sytuacja z dowiązaniami miękkimi.
  - Wystarczy wyszukać po prostu pliki specjalnego typu.
  - Ewentualnie sprawdzić też ścieżki wskazywane przez te dowiązania.

- Dowiązania twarde stanowią znacznie większe wyzwanie.
  - Wyglądają jak różne pliki o identycznej zawartości na dysku twardym.
  - W praktyce mówimy o jednej zawartości (wskazywanej przez i-węzeł).
  - Fizycznie jeden i-węzeł jest wskazany pod różnymi nazwami w folderach.
  - Zasadniczo jedyną możliwością jest przegląd zupełny numerów i-węzłów.
  - Nieco prostszym zadaniem jest znalezienie dowiązań konkretnego pliku.
    - Efektywnie chodzi o znalezienie dokładnie tego samego i-węzła.
    - Program `test` ma odpowiednią opcję (`-ef`) do sprawdzenia tego.

Politechnika Wrocławska

Część II

Podstawowe komendy

Program `ln`

- Narzędzie do tworzenia i zmieniania dowiązań twardych i miękkich.
- Przykłady użycia:
  - `ln plik-oryginalny ścieżka-nowego-dowiązania-twardego`
  - `ln -s ścieżka-dowiązania plik-dowiązania-miękkiego`

```
NAME
       ln - make links between files

SYNOPSIS
       ln [OPTION]... [-T] TARGET LINK_NAME
       ln [OPTION]... TARGET
       ln [OPTION]... TARGET... DIRECTORY
       ln [OPTION]... -t DIRECTORY TARGET...

DESCRIPTION
       In  the  1st  form,  create  a link to TARGET with the name LINK_NAME.  In the 2nd
       form, create a link to TARGET in the current directory.  In the 3rd and 4th forms,
       create  links to each TARGET in DIRECTORY.  Create hard links by default, symbolic
       links with --symbolic.  By default, each destination (name of new link) should not
       already  exist.  When creating hard links, each TARGET must exist.  Symbolic links
       can hold arbitrary text; if later resolved, a relative link is interpreted in  re-
       lation to its parent directory.

       Mandatory arguments to long options are mandatory for short options too.
```

Program `readlink`

- Narzędzie do odczytywania ścieżek z dowiązań miękkich.
- Przykłady użycia:
  - `readlink ścieżka-do-dowiązania-miękkiego`
  - `readlink -f ścieżka-do-dowolnego-pliku`

```
NAME
       readlink - print resolved symbolic links or canonical file names

SYNOPSIS
       readlink [OPTION]... FILE...

DESCRIPTION
       Note  realpath(1) is the preferred command to use for canonicalization functional-
       ity.

       Print value of a symbolic link or canonical file name

       -f, --canonicalize
              canonicalize by following every symlink in every  component  of  the  given
              name recursively; all but the last component must exist

       -e, --canonicalize-existing
              canonicalize  by  following  every  symlink in every component of the given
              name recursively, all components must exist
```

Program realpath

- Narzędzie do rozwiązywania i generowania ścieżek.
- Przykłady użycia:
  - `realpath ścieżka-do-dowiązania`
  - `realpath --relative-to=/etc ścieżka-do-pliku`

```
NAME
       realpath - print the resolved path

SYNOPSIS
       realpath [OPTION]... FILE...

DESCRIPTION
       Print the resolved absolute file name; all but the last component must exist

       -e, --canonicalize-existing
              all components of the path must exist

       -m, --canonicalize-missing
              no path components need exist or be a directory

       -L, --logical
              resolve '..' components before symlinks

       -P, --physical
              resolve symlinks as encountered (default)
```

Program dirname

- Narzędzie do odnajdywania katalogu nadrzędnego w ścieżce.
- Przykłady użycia:
  - `dirname jakaś/taka/ścieżka  # zwróci: jakaś/taka`

```
NAME
       dirname - strip last component from file name

SYNOPSIS
       dirname [OPTION] NAME...

DESCRIPTION
       Output  each  NAME with its last non-slash component and trailing slashes removed;
       if NAME contains no /'s, output '.' (meaning the current directory).

       -z, --zero
              end each output line with NUL, not newline

       --help display this help and exit

       --version
              output version information and exit

EXAMPLES
       dirname /usr/bin/
              -> "/usr"
```

Program basename

- Narzędzie do wyodrębniania ostatniej nazwy pliku ze ścieżki.
- Przykłady użycia:
  - `basename jakaś/taka/ścieżka  # zwróci: ścieżka`

```
NAME
       basename - strip directory and suffix from filenames

SYNOPSIS
       basename NAME [SUFFIX]
       basename OPTION... NAME...

DESCRIPTION
       Print  NAME with any leading directory components removed.  If specified, also re-
       move a trailing SUFFIX.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --multiple
              support multiple arguments and treat each as a NAME

       -s, --suffix=SUFFIX
              remove a trailing SUFFIX; implies -a

       -z, --zero
              end each output line with NUL, not newline
```

Politechnika Wrocławska

Część dla dociekliwych

Zadanie dodatkowe

Politechnika Wrocławska

Zadanie dodatkowe – dla zainteresowanych

Obsługa i przekierowywanie strumieni wejścia/wyjścia oraz mechanizmy heredoc i herestring - co, jak i po co stosujemy?

Proszę pokazać, w jaki sposób można przekierować standardowe wyjście programu do pliku, co można zrobić ze standardowym wyjściem błędów oraz jak otworzyć i wykorzystać dodatkowe strumienie? Jak można przekierować wyjście wszystkich komend do pliku, bez jawnego przekierowywania wyjścia każdej z tych komend i jak odwrócić później to przekierowanie?

Przy omawianiu mechanizmu heredoc, proszę w szczególności wskazać, kiedy zmienne powłoki mogą zostać zinterpolowane, a kiedy nie – jak to wymusić?