6 System plików

Wiemy już jak poruszać się po systemie plików. Warto również wiedzieć w jaki sposób dane są reprezentowane na dysku.

Podział na partycje

Każdy dysk, na samym początku podzielony jest na *partycje*. Każda partycja to fragment wydzielonej przestrzeni dysku, który system operacyjny obsługuje niezależnie od pozostałych partycji. W ten sposób można na jednym fizycznym dysku zmieścić różne systemy plików i różne systemy operacyjne.

Istnieją dwa podstawowe reprezentacje partycji:

* MBR (Master Boot Record) — wprowadzony w 1983. Informacja MBR zapisana jest zawsze w pierwszym dostępnym sektorze dysku. Obsługuje dyski do pojemności 2 TB operując 512-bajtowymi blokami, i standardowo pozwala na stworzenie maksymalnie czterech partycji. Nowsze wersje pozwalają na przekroczenie tych limitów, ale takie zabiegi nie są rozpoznawane przez wszystkie systemy operacyjne.
* GPT (GUID Partition Table) — wprowadzony z końcem XX wieku, obecnie obsługiwany przez wszystkie nowsze systemy operacyjne. Informacja GPT też zapisywana jest w pierwszych dostępnych sektorach dysku. Teoretyczna maksymalna pojemność dysku obsługiwana przez GPT to 9.4ZB ($2^{64} \times 512$) ale może być zwiększona poprzez zwiększenie wielkości bloku. GPT pozwala na stworzenie 128 partycji.

System operacyjny Windows przypisuje kolejne litery do kolejnych partycji, poczynając od litery `C`. Każda pełna ścieżka zaczyna się od litery reprezentującą którąś z partycji.

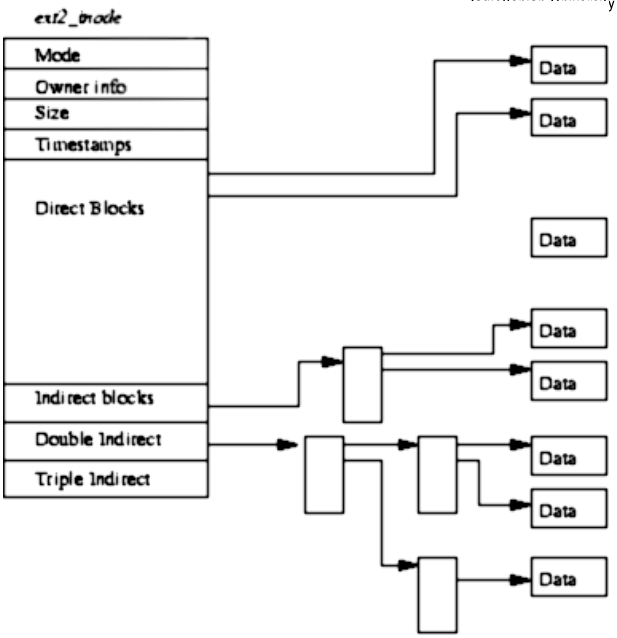
Pod linuxem nie ma takiego twardego związku. Partycja systemowa przypisana jest do katalogu głównego `/` ale pozostałe mogą być podpięte jako dowolny podkatalog, za pomocą komendy `mount`. Linux pozwala też na montowanie innych partycji, także wyjmowalnych lub połączonych przez sieć. Wpisując `mount` bez argumentów możemy zobaczyć wszystkie aktualnie podpięte partycje. Pomocna jest również komenda `df`.

```
$ mount
$ df -h
```

I-węzły i bloki danych

Każda partycja ma swój niezależny system plików. Współczesne systemy linuxowe operują w systemie `ext2`, `ext3` lub `ext4`. Systemy te są do siebie podobne. Systemy przechowują dwa rodzaje informacji:

* informacje o plikach, trzymane w *tablicy i-węzłów* (ang. *i-node table*).
* dane (zawartość plików), przechowywane w *blokach*, gdzieś w przestrzeni dyskowej.

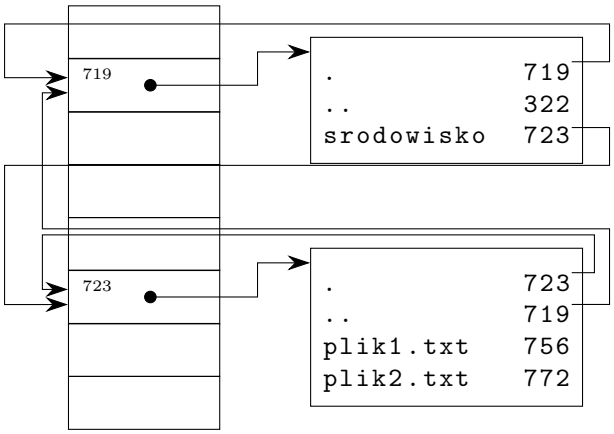


Każdy I-węzeł (ang. *inode*) reprezentuje dokładnie jeden plik. Przechowywane są tam wszystkie informacje o danych, ale nie sama zawartość pliku:

* Rodzaj pliku (zwykły plik, katalog, ...)
* Właściciel pliku
* Prawa dostępu
* Wielkość
* Czas modyfikacji
* Adresy bloków które przechowują dane

I-węzeł nie przechowuje nazwy pliku!

We wszystkich blokach do których odwołuje się dany i-węzeł znajduje się fragment zawartości pliku. Każdy blok ma z góry ustaloną wartość. Na systemach TCS jest to 4KB. Do każdego bloku odwołuje się zawsze dokładnie jeden i-węzeł. Nie jest możliwe podzielenie bloku na mniejsze kawałki. W konsekewncji, jeśli wielkość pliku nie jest dokładną wielokrotnością wielkości bloku, trochę przestrzeni dyskowej zostaje zmarnowane.



Katalog jest specjalnym rodzajem pliku. I-węzeł odpowiadający za katalog ma taką samą strukturę jak każdy inny plik. Zawartość katalogu zdefiniowana jest nie w węźle, a w bloku danych. To tam wypisane są nazwy zawieranych plików, i adres i-węzłów im odpowiadających.

Każdy katalog, nawet gdy jest on pusty, ma zawsze zdefiniowane dwie pozycje:

* `.` – wskazuje na swój własny i-węzeł.
* `..` – wskazuje na i-węzeł odpowiadający za katalog nadrzędny.

Podając parametr `-i` do komendy `ls` możemy podejrzeć informację o i-węzłach:

```
$ ls -ali
723 drwxrwxr-x   2 pdan pdan  4096 Oct  9 06:42 .
719 drwx-----x  10 pdan users 4096 Oct  9 06:42 ..
756 -rw-rw-r--+  1 pdan pdan    52 Oct  6 07:56 plik1.txt
772 -rw-rw-r--   1 pdan pdan    68 Oct  6 06:39 plik2.txt
```

Każda pozycja poprzedzona jest numerem i-węzła. Zwróćmy uwagę na liczbę występującą bezpośrednio po prawach dostępu - jest to "liczba połączeń", czyli liczba krawędzi przy-

chodzących z któregokolwiek katalogu do danego i-węzła. W przypadku zwykłych plików, wartość ta zwykle wynosi 1, bo plik znajduje się w dokładnie jednym katalogu. Katalogi natomiast mają wiele przychodzących krawędzi z powodu `.` i `..`

Dowiązania („skróty")

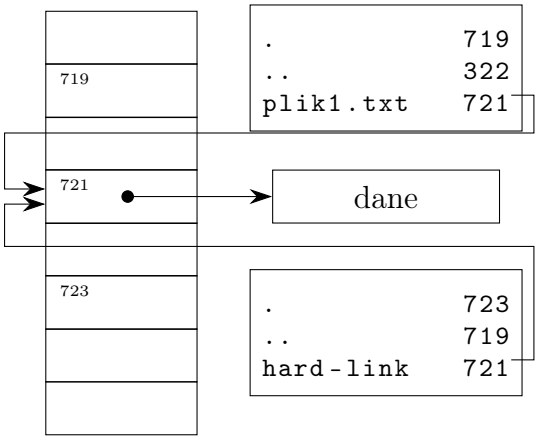
Dowiązania (ang. *link*) to cecha niektórych systemów plików pozwalająca na tworzenie dodatkowych, niestandardowych połączeń w systemie plików. Takie połączenia pozwalają na szybszą nawigację w strukturze plików: dany plik można łatwo dowiązać do katalogu nas interesującego, mimo że tak na prawdę znajduje się on zupełnie gdzie indziej. Dowiązania można stosować zarówno do plików jak i do całych katalogów.

Istnieją dwa rodzaje dowiązań:
* *dowiązanie symboliczne* (*symbolic link*, *symlink*) operuje na nazwach plików,
* *dowiązanie twarde* (*hard link*) operuje na numerach i-więzów.

Plik **dowiązania symbolicznego** przechowuje nazwę pliku docelowego. W przypadku gdy dowolna aplikacja próbuje otworzyć plik poprzez dowiązanie, system operacyjny automatycznie przeszuka drzewo katalogów by znaleźć właściwy plik docelowy.

Plik docelowy nie jest "świadom" istnienia dowiązań symbolicznych nań wskazujących. Jeśli plik docelowy zostanie usunięty bądź jego nazwa ulegnie zmianie, to nie będzie on osiągalny poprzez dowiązanie. Próba użycia takiego dowiązania zakończy się błędem.

Dowiązania symboliczne tworzymy: `ln -s <plik docelowy> <nazwa dowiązania>`



Plik **dowiązania twardego**, to po prostu wpis w katalogu używający numer i-węzła, który już jest wykorzystywany przez inny plik. Twarde linki są nierozróżnialne od oryginału. Wszystkie wpisy prowadzące do danego i-węzła są równoważne.

Twarde dowiązanie nie jest kopią pliku! Wszystkie pliki podpięte pod ten sam i-węzeł używają tą samą, jedną przestrzeń dyskową. Jest to ten sam plik, o wielu nazwach.

Naturalnie, zmiana nazwy jednego z połączeń, łącznie z przeniesieniem pliku w inne miejsce, nie wpływa na pozostałe powiązania. Co więcej, próba usunięcia pliku poprzez jedno z połączeń nie wpłynie negatywnie na działanie pozostałych twardych dowiązań.

Dzieje się tak ponieważ komenda `rm` wcale nie usuwa pliku, tylko usuwa wpis w katalogu. Jeśli po takiej operacji docelowy i-węzeł nie jest wskazywany przez żaden inny wpis, plik zostanie naprawdę usunięty.

Warto dodać, że gdy jakakolwiek aplikacja otwiera plik, tworzy ona podobne połączenie z plikiem. Gdy w trakcie działania aplikacji wszystkie połączenia do pliku zostaną usunięte, plik nadal pozostaje na dysku. Dopiero gdy aplikacja zamknię plik, zostaje on usunięty.

Takie podejście jest szczególnie przydane na przykład podczas aktualizacji plików systemowych bez potrzeby restartowania komputera.

Twarde dowiązania tworzy się poleceniem `ln <plik docelowy> <nazwa dowiązania>`

Uwaga:
Twarde dowiązania nie mogą być używane dla katalogów. Twarde dowiązania nie mogą być też stosowane przy łączeniu plików z różnych partycji czy dysków sieciowych (każda partycja dysponuje własną numeracją i-węzłów, która jest unikalna tylko w obrębie tej partycji).

W obu tych przypadkach można bez problemów używać dowiązań symbolicznych.
Jeżeli chcemy sprawdzić „fizyczną" ścieżkę: `pwd -P`.

Prawa dostępu

Każdy i-węzeł przechowuje informację o właścicielu pliku i prawach dostępu. Prawa dostępu zdefiniowane są dla trzech kategorii użytkowników:

* właściciel pliku (`u`)
* grupa przypisana do pliku (`g`)
* pozostali użytkownicy (`o`)

Dla każdej kategorii można zdecydować, czy użytkownik może: odczytać (r), zapisać (w) lub uruchomić (x) dany plik.

Właściciel pliku może zmienić uprawnienia za pomocą komendy `chmod`. Składnia komendy to: `chmod` KOU <nazwa> , gdzie

* K — znak lub kombinacja definiująca kategorię użytkowników co do której dokonujemy zmianę (`u`, `g`, `o`); można użyć `a` dla wszystkich (równoważne z `ugo`),
* O — znak operacji: prawa dostępu można nadać `+` albo udebrać `-`.
* U — znak lub kombinacja znaków definiująca rodzaju uprawnień — `r`, `w` lub `x`.

Przykłady nadawania praw:

`chmod u+x nazwa`	daje użytkownikowi prawo do uruchamiania
`chmod ug+rx nazwa`	nadaje użytkownikowi i grupie prawo do czytania i uruchamiania
`chmod a-rx nazwa`	zabiera wszystkim prawo do czytania i uruchamiania

Można również użyć systemu ósemkowego. Czytaniu (r) odpowiada 4, pisaniu (w) 2, wykonaniu (x) 1. Każda cyfra odpowiada jednemu z kategorii użytkowników. Na przykład:

```
chmod 755 helloshell.sh
```

nadaje użytkownikowi prawa do pisania, czytania i uruchamiania (4+2+1), prawo czytania i uruchamiania (4+0+1) grupie i innym.

Ponieważ prawa dostępu są zapisane w i-węźle, są one stałe, niezależnie przez które dowiązania plik jest otwierany. Dowiązanie symboliczne jest osobnym i-węzłem ze swoimi prawami, ale są one ignorowane – liczą się tylko i wyłącznie prawa dostępu zapisane w i-węźle docelowego pliku który otwieramy.

Uprawnienia do katalogów nadaje się w dokładnie taki sam sposób co do zwykłych plików. Jak już wyjaśniliśmy wcześniej, z punktu widzenia systemu plików, katalog to po prostu specjalny rodzaj pliku. Jednakże, w konteście katalogów, warto wyjaśnić co dane uprawnienia oznaczają w praktyce:

* `r` – odczyt pozwala użytkownikowi odczytać nazwy w katalogu.
* `w` – zapis pozwala użytkownikowi stworzyć, zmienić lub usunąć pliki z danego katalogu.
* `x` – pozwala uzyskac dostęp do plików powiązanych z nazwami w danym katalogu.

W szczególności, katalog z uprawnieniami `r--` pozwoli nam zaledwie na wypisanie wszystkich nazw plików, bez podania jakiejkolwiek informacji czym właściwie są te pliki. Innymi słowy, mamy nazwy ale nie mamy dostępu do powiązanych z nimi i-węzłów.

Natomiast katalog z uprawnieniami `--x` pozwoli nam otworzyć dowolny plik, ale nie powie nam jakie właściwie pliki tam są.

SUID i SGID

Istnieje jeszcze specjalna flaga SUID i SGID, włączana odpowienio przez `u+s` i `g+s`. Plik wykonywalny z taką flagą dziedziczy uprawnienia właściciela pliku gdy zostanie on uruchomiony. Programy z taką flagą pozwalają innym użytkownikom na wykonanie operacji której normalnie wykonać by nie mogli.

Dla przykładu, program `/bin/mount` pozwala zamontowac nowy dysk czy zasób w określonym miejscu w systemie plików. Operacja taka wymaga uprawnien głównego administratora (tzw. roota). W normalnej sytuacji `mount` byłby dla mnie bezużyteczny, bardzo szybko bym się przekonał, że nie mam wystarczających uprawnień.

Na szczęście `mount` ma ustawioną flagę SUID i `root` jest właścicielem pliku. Niezależnie kto uruchomi ten program, jego działanie będzie miało uprawnienia roota, i niezbędne zmiany będą mogły być dokonane.

Naturalnie, trzeba być bardzo ostrożnym nadając flagi SUID lub SGID szczególnie gdy jest się administratorem. Jeśli program z taką flagą zawiera błąd, złośliwy użytkownik może próbować uzyskać szerszy dostęp do komputera niż oryginalnie założono.

Zużycie pamięci dyskowej - `quota`

Pracując na dowolnym komputerze, czy to prywatnym czy wspólnym serwerze, warto zwrócić uwagę na ilość zajmowanego miejsca na dysku. Gdy go zabraknie, wiele komend może okazać się niemożliwymi do wykonania. Na wspólnych serwerach administratorzy często wprowadzają limity danych dla użytkowników (by uniknąć wyczerpania miejsca).

Komenda `quota` podaje informację o ilości zużytej pamięci oraz dostępnych limitach.

`-g`	`--group`	Wyświetla limity dla grupy, do której należy użytkownik
`-v`	`--verbose`	Uwzględnia wszystkie dostępne systemy plików, nawet tam gdzie nie ma limitów
`-s`	`--human-readable`	Wyświetla ilość danych w wygodnych jednostkach
`-q`	`--quiet`	Wyświetla informacje tylko gdy limity są przekroczone

Na serwerach `tcs` limity są określone dla grupy użytkownika a niekoniecznie dla loginu. Dlatego rekomendujemy używać parametr `-g`:

```
$ quota -sg
Disk quotas for group staff (gid 50): none
Disk quotas for group users (gid 100): none
Disk quotas for group fuse (gid 109): none
Disk quotas for group pdan (gid 627):
Filesystem   space   quota   limit   grace   files   quota   limit   grace
random.direct:/exports/home/student
             9687M   8192M  16384M   6days     103       0       0
```

aktualne zużycie — miękki limit — twardy limit — czas na poprawę — ilość danych

aktualne zużycie — miękki limit — twardy limit — czas na poprawę — ilość plików

Użytkownik ma prawo przekroczyć miękki limit na krótki okres czasu (tzw. *grace period*). Twardy limit nie może zostać przekroczony. Przy każdej operacji która mogłaby naruszyć ten limit zostanie wywołany błąd tak jakby nie było miejsca na dysku.
Limit 0 oznacza, że nie ma ograniczenia.

Kompresja

Pracując z danymi często potrzebujemy spakować lub rozpakować dane. Kompresji można dokonać różnymi programami, najbardziej popularne są `gzip` i `tar`.

`gzip <nazwa>`	kompresuje `<nazwa>`, tworząc `<nazwa>.gz`
`gzip -d <nazwa>.gz`	dekompresuje `<nazwa>.gz`
`tar czvf <katalog>.tar.gz <katalog>`	kompresuje `<katalog>`
`tar tzvf <katalog>.tar.gz`	listuje skompresowany `<katalog>`
`tar xzvf <katalog>.tar.gz`	dekompresuje `<katalog>`

Skompresowane pliki nie są plikami tekstowymi i typowo mają końcówkę `.gz`.

Drzewo katalogów w systemie Linux

Niezależne od wersji systemu Linux, katalog główny i niektóre podkatalogi wyglądają bardzo podobnie.

`/`	katalog główny. Tu wszystko się znajduje.
`/bin`	podstawowe programy systemu i powłoki. Część z komend które przedstawiliśmy na poprzednich zajęciach to są tak na prawdę osobne programy, które znajdują się właśnie tutaj.
`/boot`	pliki startowe i jądro linuxa
`/dev`	urządzenia peryferyjne komputera, reprezentowane jako specjalne pliki, oznaczane literą `c`. Pliki te nie rezydują na żadnym z dysków, ale wykorzystują te same funkcje zapisu i odczytu, pozwalając różnym aplikacjom z łatwością komunikować się z tymi urządzeniami.
`/etc`	pliki konfiguracyjne
`/home`	prywatne pliki użytkowników
`/initrd`	zawiera informacje potrzebne do startu systemu. Na TCSie widnieje jako symlink do jednego z plików w `/boot`.
`/lib`	zawiera biblioteki dynamiczne potrzebne do uruchamiania różnych programów. Wasze programy w C też z tych plików korzystają.
`/media`	tu domyślnie mountowane są dyski zewnętrzne (CD-ROMy, dyskietki, pendrive-y)
`/opt`	dodatkowe programy niezależnych dystrybutorów, nie związanych z daną wersją Linuxa. Na TCS-ie znajdziecie tutaj na przykład `satori`
`/proc`	wirtualny system plików reprezentujące aktualne zasoby systemu. Tu można odczytać np. parametry procesora (`/proc/cpuinfo`) czy pamięci (`/proc/meminfo`).
`/root`	katalog domowy głównego administratora systemu.
`/tmp`	pliki czasowe. Mogą ulec usunięciu np. przy restarcie systemu.
`/usr`	programy i biblioteki dla programów użytkownika. Dla programisty to bardzo ważny katalog!
`/usr/bin`	narzędzia programistyczne. Tu znajdziecie m. in. kompilatory C i C++
`/usr/games`	bo każdy programista musi się czasem rozerwać :)
`/usr/include`	pliki nagłówków do C i C++, zarówno te standardowe jak i dodatkowych pakietów zainstalowanych na Linuxie
`/usr/lib`	pliki bibliotek
`/usr/local`	podobny do `/usr`. Ideowo tu znajdują się pliki bardziej lokalne dla danej maszyny, ale rozróżnienie to jest często niejasne w praktyce.
`/usr/share`	domyślne pliki konfiguracyjne, rysunki, dokumentacje zainstalowanych bibliotek, itp.
`/usr/src`	pliki źródłowe
`/var`	Tu pojawiają się m.in. niektóre logi, cache plików internetowych, zadania dla drukarek, itp. Wszystkie te pliki mają cechę, że ich zawartość i liczba często zmienia się w czasie w wyniku normalnego działania systemu.