Obsługa terminalu dla opornych

Xerun#0773

3 maja 2021

O czym jest ten dokument?

W tym dokumencie postaram się w sposób przystępny przekazać tajniki działania terminala w systemie Linux. Pokażę, że nie jest to nic strasznego i nie trzeba się tego bać. Terminal jest bardzo przydatnym narzędziem, które tylko wygląda nieprzystępnie. Dodatkowo ten dokument może funkcjonować jako baza wiedzy o pewnych komendach i ich prostych przykładach, do której można zajrzeć w dowolnym momencie. Dla ułatwienia takiego użytkowania można skorzystać ze spisu treści, którego elementy po kliknięciu powinny zaprowadzić Cię do odpowiedniego fragmentu. Warto też wspomnieć, że Twój system może się różnić lekko od mojego, więc nie wszystko musi działać lub wyglądać tak samo. Postaram się podawać alternatywy (jeśli takie znam) w przypadku, gdy z jakiegoś powodu dla Ciebie dana komenda nie działa.

Spis treści

1 Wprowadzenie

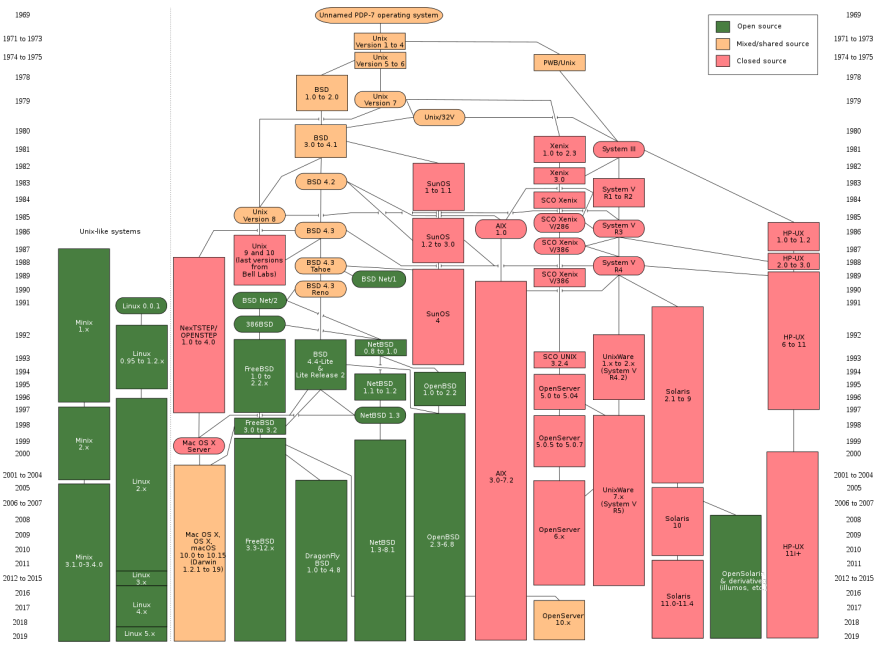
1.1 O Linuxie, czyli historia dla zainteresowanych

Ta sekcja jest przeznaczona dla zainteresowanych. Jeśli jesteś jedną z tych osób co zasypiały na historii, to nie krępuj się i przejdź do następnej. Myślę, że jednak warto jest wiedzieć "na czym się siedzi".

Linux jest potocznym określeniem rodziny systemów operacyjnych, czyli w skrócie rozbudowanych zbiorów programów, zarządzających komputerem i umożliwiających wygodną pracę i rozrywkę. Poprawna nazwa brzmi **GNU/Linux** a samo określenie Linux dotyczy jedynie jądra, na którym system jest zbudowany, ponieważ jednak Linux jest określeniem lepiej rozpowszechnionym, także i ja pozwolę sobie je stosować w odniesieniu do całego systemu.

Linux nazywamy również systemem **unixopodobnym**. Sama nazwa pochodzi ze zbitki słów Linus (będącego imieniem twórcy) i Unix. Ale cóż to Unix? **UNIX** to system operacyjny napisany w 1969 w Bell Labs przez Dennisa Ritchie i Kena Thompsona. Był on rewolucyjny z paru względów, ważnym dla nas jest jednak to, że wprowadził standard **POSIX**, których między innymi właśnie Linux się (mniej więcej) "trzyma". Systemy zbliżone budową do systemu Unix, albo/i opartych na standardach POSIX, można nazwać unixopodobnymi.

Poniższa grafika prezentuje uproszczoną wersję rodziny systemów unixopodobnych. Może nie widać zbyt wiele bez przybliżania, lecz wyraźnie pokazuje to jak rozrosły się systemy tego typu.



Grafika od Wikimedia Commons

Linux nie stanowi produktu jednej firmy, jak np. Windows, jest natomiast wynikiem pracy wielu firm i ogromnej liczby różnych grup ludzi, często działającej na tym polu bezinteresownie (non-profit). GNU/Linux jest bazowym komponentem, na którego podstawie rodzą się inne produkty, nazywane **dystrybucjami**.

Poszczególne dystrybucje różnią się od siebie mniej lub bardziej wyglądem oraz przeznaczeniem – możemy więc tutaj wybierać pomiędzy dużymi, kompletnymi systemami na domowe biurka, na serwery, do firm a między małymi, lekkimi, a zarazem szybkimi dystrybucjami nadającymi się np. na starsze komputery, czy urządzenia pendrive. Podstawy terminala są jednak niemalże takie same niezależnie od danej dystrybucji. Osobiście piszę ten dokument na dystrybucji o nazwie "Fedora", która "utrzymywana" jest przez firmę Red Hat, ale większość komend, (a w szczególności tych z pierwszych rozdziałów) działać będzie również na np. Ubuntu.

Jednymi z obecnie najpopularniejszych dystrybucji (w losowej kolejności) są:

- Ubuntu
- Debian
- Linux Mint
- Manjaro
- Fedora
- Pop!_OS
- Elementary
- Gentoo

Już teraz wiem, że ta lista się dobrze nie zestarzeje. Ponadto proszę o nieinteresowanie się tym ostatnim wymienionym.

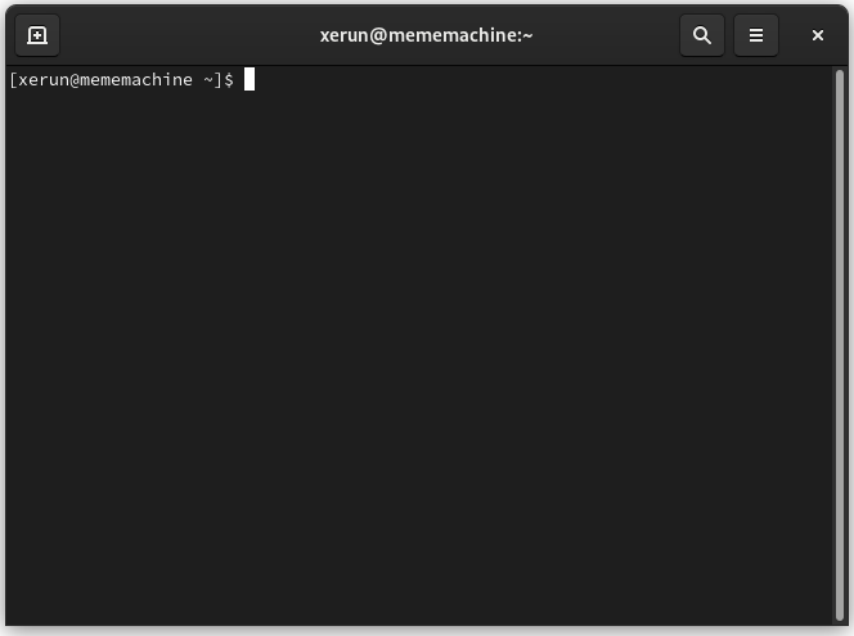
Dużą zaletą Linuksów jest to, że jako oparte głównie o **wolne oprogramowanie**, są one dostępne w większości zupełnie za darmo i to całkiem legalnie. Istnieją oczywiście komercyjne wersje zawierające dodatkowe oprogramowanie, jednakże nawet i one mają konkurencyjne ceny w stosunku do np. oprogramowania Microsoftu.

1.2 Co to terminal?

Bardzo ważną cechą systemów uniksowych jest **rozdzielenie systemu**, czyli procesów wykonujących zadania, **od interfejsu**. Dotyczy to nie tylko interfejsu graficznego, ale także i tekstowego.

Jądro Linux ma zaimplementowaną **emulację terminali**. Historycznie terminale były to komputery o minimalnych możliwościach obliczeniowych i bez własnej pamięci masowej, ale z monitorem i klawiaturą. Za pomocą tych terminali można było sterować główną jednostką obliczeniową. Taki podział był spowodowany m.in. przez bardzo wysoką cenę komputerów w tamtym czasie, i umożliwiał pracę na wielu stanowiskach, z użyciem tylko jednego komputera.

Obecnie wbudowana obsługa terminali w GNU/Linuxie umożliwia zarówno zdalną pracę (np. zdalną administrację serwerem), jak i wykorzystanie wielozadaniowości systemu także w środowisku tekstowym (uruchomienie wielu terminali odpowiada funkcjonalnie otwarciu wielu okien, gdzie w czasie gdy jeden program jest zajęty i nie odpowiada, możemy pracować w innym).



Przykład aplikacji emulującej terminal

Taka budowa ma również skutki dla twórców oprogramowania – umożliwia to napisanie aplikacji która nie ma praktycznie interfejsu, tylko komunikuje się z użytkownikiem za pomocą wypisywanych linijek tekstu, a użytkownik wpisuje polecenia.

1.3 Co to komendy i polecenia?

Komendy i **polecenia** są synonimami. Oznaczają one dowolną instrukcję daną przez użytkownika do systemu, np. uruchomienie programu. Komendy są egzekwowane przez wpisanie ich do linii komend (w terminalu) i naciśnięcie klawisza ENTER.

Warto wiedzieć, że do niemalże każdej komendy na systemie Linux istnieje wbudowana instrukcja obsługi, do której dostać się można za pomocą polecenia:

```
man nazwa_komendy
```

Może się to okazać przydatne, jeśli wyjaśnienie interesującej Cię komendy nie istnieje w tym dokumencie, lub uznasz, że obecne nie jest dla Ciebie wystarczające. Nazwa `man` *prawdopodobnie* pochodzi od słowa “manual“, ale kto wie tbh, równie dobrze może oznaczać człowieka. Do dziś nie wiadomo do końca co poeta miał na myśli. W moim wypadku instrukcje serwowane są w języku angielskim, na niektórych dystrybucjach jednak istnieje również wersja polska tych tekstów.

1.4 Uruchamianie terminala

Zależnie od dystrybucji możecie się spotkać z różnymi nazwami aplikacji emulujących terminal. Aby wam ułatwić znalezienie odpowiedniej aplikacji wypiszę nazwy, pod którymi może się ona kryć:

- Terminal
- Konsole
- Terminator
- Yakuake
- Kitty
- xterm

(a) Ikony aplikacji 'Terminal' (b) 'Konsole'

Po uruchomieniu terminala możecie być słusznie lekko zagubieni. Spotykacie się z jakimiś nazwami w nawiasach, jakimś dolarem i migającym kursorem. Wyjaśnię więc wszystko od początku ...

Oto przykładowy widok po uruchomieniu aplikacji emulującej terminal na swoim systemie (w moim przypadku to "Terminal"):

```
[user@linux ~]$
```

Oczywiście jest to tylko wersja tekstowa, ale mniej więcej reprezentuje to, co powinniście widzieć. Kolory mogą być inne, a nazwy `user` i `linux` są definitywnie u was inne.

Zacznę więc od wyjaśnienia od lewej do prawej. To enigmatyczne `user` to jesteście *wy*. To wasza nazwa użytkownika i pod tą nazwą znani jesteście dla systemu. Nazwa ta nie zawiera nigdy żadnych dużych liter i zwykle jest też nazwą waszego katalogu domowego (spokojnie, to też wyjaśnię potem).

Następnie widzimy znak `@` oraz `linux` . To pierwsze po prostu w języku angielskim "at" oznacza "na", tymczasem `linux` oznacza po prostu nazwę systemu albo komputera, na którym się znajdujecie. W luźnym tłumaczeniu cała fraza `user@linux` oznacza "użytkownik na tym systemie".

Kolejny symbol to ~ . Jest to aktualny katalog, w którym się znajdujesz, ta część będzie się zmieniać zależnie od tego, jak zmienisz lokalizację – przykładowo:

`[user@linux Wideo]$` albo `[user@linux Pobrane]$`

O zmianie katalogu dowiesz się więcej w następnej sekcji, a konkretnie w "Zmiana katalogu – `cd` "

~ w systemie Linux oznacza **katalog domowy** (o czym potem w "Domek – czyli `/home` sweet `/home` ")

Fraza zostaje zamknięta w nawiasach i następuje po niej symbol `$` , który można interpretować jako "wykonuje".

Cała fraza oznacza zatem mniej więcej "użytkownik na tym komputerze i w tym katalogu wy-

konuje...".

1.5 Ważna uwaga odnośnie klawisza 'Tab' i strzałek

Klawisz 'Tab' jest od teraz Twoim największym przyjacielem. W przyszłych rozdziałach będzie Ci bardzo pomocny. Jego funkcją w terminalu jest **automatyczne uzupełnianie**, co w praktyce oznacza że po wciśnięciu parę razy tego klawisza dostaniesz prawdopodobnie to czego oczekujesz. Czy to ścieżka do jakiegoś pliku, albo niedokończona komenda, dobrą praktyką jest wciśnięcie 'Tab' parę razy, a on albo dokończy za Ciebie, albo pokaże ci jakie ma alternatywy, a ty zareagujesz odpowiednio. Jeśli to co teraz czytasz brzmi przerażająco, to spokojnie, wszystko będzie dobrze – 'Tab' tu jest i Ci pomoże.

Innymi przydatnymi klawiszami są strzałki. Dzięki nim możesz przemieszczać kursor w tekście (strzałkami lewo – prawo), jak w większości znanych Ci już pewnie programów, oraz uzyskać dostęp do poprzednio wykonanych komend (strzałkami góra – dół). Może to być przydatne gdy chcesz wykonać jakieś długie polecenie po kilka razy. Nikomu się nie chce pisać czegoś ponownie, więc najlepiej jest użyć strzałek i wyedytować komendę jeśli jest taka potrzeba.

2 Katalogi – czyli łażenie po świecie

Należy się przyzwyczaić do myślenia, że obsługując terminal zawsze będziemy w jakimś katalogu. Czy to nasz katalog domowy czy katalog z filmami, nasze działania nie są odizolowane od świata. W tej sekcji przedstawię, jak oddziaływać na ten świat i jak się w nim swobodnie poruszać.

2.1 Domek – czyli `/home` sweet `/home`

Katalog domowy to katalog oddzielny dla każdego użytkownika, którego celem jest przechowywanie jego plików i dokumentów (czegokolwiek sobie zapragnie). Katalogi te znajdują się w katalogu `/home` (czyli "w domku") i opatrzone są twoją nazwą użytkownika. Przykładowo, będąc zalogowanym jako `user` , ścieżka naszego katalogu domowego to `/home/user`

Domyślnie, po otwarciu terminala, to w tym katalogu (znanym też jako $\sim$) będziemy się znajdować i w nim rozpoczniemy naszą podróż.

2.2 Gdzie jestem? – `pwd`

Najprostszą komendą, dzięki której nigdy się nie zgubisz, jest `pwd` (ang. **p**rint **w**orking **d**irectory)

Jeśli czujecie się zagubieni, nie wiecie gdzie się znajdujecie lub na jakim katalogu operujecie, to ta komenda jest dla was przyjacielem.

2.2.1 Przykłady

1. **`[user@linux ~]$ pwd`**
   `/home/user`
2. **`[user@linux Wideo]$ pwd`**
   `/home/user/Wideo`

...i to wszystko, co ta komenda oferuje tbh.

2.3 Co jest w moim katalogu? – `ls` i inne fafarstki

Polecenie służące do wypisywania plików i katalogów w danym katalogu. Bez podania żadnych argumentów wypisuje pliki w obecnym katalogu roboczym, czyli tym, w którym się aktualnie znajdujesz. Jedna z najważniejszych komend, jakie można znać na Linuxie.

Komenda nie jest trudna, jej nazwa może też się kojarzyć z angielskim słowem "list" czyli wypisywać. Oto składnia:

`ls -opcje katalog`

I jak już wcześniej wspominałem, nie wpisując argumentu **`katalog`** otrzymamy wynik dla obecnego katalogu roboczego.

Dzięki tej komendzie dowiemy się również, jak wygląda podstawowa struktura katalogu, co w nim jest, jakie są tutaj typy danych itd.

2.3.1 Niewidoczne rzeczy w katalogach

Katalogi w systemie Linux mogą zawierać wiele rzeczy, również takich domyślnie niewidocznych dla nas, zwykłych śmiertelników. Zwykle są to pliki konfiguracyjne, lub foldery zawierające takowe, których normalnie nie chcielibyście widzieć, bo tylko wchodzą w drogę, mają jednak wszystkie cechę wspólną – ich nazwa zaczyna się od " `.` ". Przykładowe pliki i katalogi to: **`.bashrc`**, **`.cache`**, **`.config`**.

Poza wspomnianymi katalogami i plikami konfiguracyjnymi (lub własnymi ukrytymi rzeczami) istnieją stałe 2 nazwy katalogów które należy zapamiętać:

1. `.` – oznacza aktualny katalog roboczy.
2. `..` – oznacza katalog nadrzędny, "wyższy" w hierarchii.

Są to skróty, których celem jest umilenie Ci czasu. Dzięki nim, zamiast wpisywać całą ścieżkę pliku lub katalogu do jakiegoś polecenia, po prostu ich używasz.

Na przykład, jeśli znajdujesz się w katalogu **`Muzyka`**, ale chcesz mieć prostą do wpisania ścieżkę do katalogu **`Wideo`**, który znajduje się w katalogu nadrzędnym, to w tym wypadku ścieżka do niego wygląda tak: **`../Wideo`**

Łatwiej jest to sobie zwizualizować mając drzewko hierarchii folderów:

```
home
└── user
    ├── CLionProjects
    ├── Dokumenty
    ├── GOG Games
    ├── MultiMC
    ├── Muzyka
    │   └── ..... Tutaj jesteś
    ├── Obrazy
    ├── Pobrane
    ├── Publiczny
    ├── Pulpit
    ├── PycharmProjects
    ├── Szablony
    └── Wideo
        └── ..... Tu chcesz coś zrobić
```

Komenda “ **ls .** “ będzie robić dokładnie to samo co zwykła komenda “ **ls** “. Po prostu w tym drugim wypadku program się domyśla, że chodzi o kropkę.

2.3.2 Opcje

- **a** (all) Wypisuje *wszystkie* elementy w katalogu, nawet te ukryte.
- **l** (long) Wypisuje dodatkowe dane w kolumnach (uprawnienia, ilość dowiązań, właściciela, grupę, do której należy plik/katalog, rozmiar, datę modyfikacji, nazwę pliku).
- **h** (human-readable) Pokazuje jednostki bardziej czytelne dla ludzi (np. 2M(egabajty), 1G(igabajt)).
- **S** (sort by Size) Sortuje wynik po rozmiarze (największy pierwszy).
- **t** (sort by time) Sortuje wynik po czasie (najnowszy pierwszy).

Komendy można łączyć ze sobą jednocześnie, np. **-al** wypisuje wszystkie, nawet ukryte, elementy w kolumnach razem z dodatkowymi danymi. Kolejność nie ma znaczenia, **-al** robi dokładnie to samo co **-la**

2.3.3 Przykłady

1. Proste wykonanie:

   ```
   [user@linux ~]$ ls

   CLionProjects  'GOG Games'   Muzyka   Pobrane     Pulpit           Szablony
   Dokumenty       MultiMC      Obrazy   Publiczny   PycharmProjects  Wideo
   ```

Jak wydać powyżej, polecenie **ls** wypisuje każdy nieukryty element w katalogu roboczym. Sortowane są one alfabetycznie po nazwie, niezależnie od typu (plik/katalog/coś innego). Elementy, których nazwy są wieloczłonowe są opatrzone w cudzysłowie, żeby ich przypadkowo nie wziąć za oddzielne elementy.

2. Wyświetlanie jako lista:

```
[user@linux ~]$ ls -l
```

```
razem 48
drwxr-xr-x. 7 user user 4096 01-14 00:16  CLionProjects
drwxr-xr-x. 6 user user 4096 01-30 13:33  Dokumenty
drwx------. 3 user user 4096 08-19 23:49 'GOG Games'
drwxr-xr-x. 5 user user 4096 01-11 23:24  MultiMC
drwxr-xr-x. 2 user user 4096 08-15 15:59  Muzyka
drwxr-xr-x. 4 user user 4096 01-30 18:21  Obrazy
drwxr-xr-x. 2 user user 4096 02-14 00:15  Pobrane
drwxr-xr-x. 2 user user 4096 08-15 15:59  Publiczny
drwxr-xr-x. 3 user user 4096 02-13 14:19  Pulpit
drwxr-xr-x. 3 user user 4096 11-18 14:14  PycharmProjects
drwxr-xr-x. 2 user user 4096 08-15 15:59  Szablony
drwxr-xr-x. 6 user user 4096 02-13 18:38  Wideo
```

3. Wyświetlenie posortowanej po rozmiarze listy elementów z katalogu Wideo:

```
[user@linux ~]$ ls -lhS Wideo
```

```
razem 238M
-rw-r--r--. 1 user user  71M 12-09 05:34  projekt2hq.mp4
-rw-r--r--. 1 user user  30M 02-08 18:17 '2021-02-08 18-05-21.mkv'
-rw-r--r--. 1 user user  30M 12-09 05:23  projekt2.mp4
-rw-r--r--. 1 user user  19M 02-03 00:52 '2021-02-03 00-45-41.mkv'
-rw-r--r--. 1 user user 180K 02-03 00:45 '2021-02-03 00-45-01.mkv'
-rw-r--r--. 1 user user  65K 01-15 22:09  projekt3.kdenlive
drwxrwxr-x. 2 user user 4,0K 02-13 18:38  Film
drwxr-xr-x. 2 user user 4,0K 02-08 19:11 'Projekt 3'
drwxr-xr-x. 2 user user 4,0K 01-15 22:04  Projekt2
drwxr-xr-x. 2 user user 4,0K 01-15 21:51  titles
```

2.4 Zmiana katalogu – cd

To najważniejsza i zarazem najprostsza komenda w tej całej sekcji. Dzięki niej zrobisz wszystko i nic.

cd , czyli **c**hange **d**irectory, to polecenie służące do przemieszczania się między folderami. Poniżej składnia komendy:

```
cd katalog
```

2.4.1 Przykłady

Zakładając, że wasz katalog domowy wygląda tak po wpisaniu komendy **ls** :

```
CLionProjects  'GOG Games'   Muzyka   Pobrane     Pulpit            Szablony
Dokumenty       MultiMC      Obrazy   Publiczny   PycharmProjects   Wideo
```

1. Jeśli chcemy wejść do katalogu Wideo, możemy wpisać:
   [user@linux ~]$ cd Wideo
2. albo pełną ścieżkę:
   [user@linux ~]$ cd /home/user/Wideo
3. A po wpisaniu:
   [user@linux Wideo]$ cd ..
   wrócimy do katalogu *wyżej* (czyli w tym wypadku ~).

2.5 Tworzenie katalogów – `mkdir`

mkdir służy do tworzenia katalogów. To wszystko. Banalnie prosta komenda robiąca jedną rzecz – coś do czego się musicie przyzwyczaić, bo takich jest wiele.

Składnia:

mkdir nazwaKatalogu1 [...]

Te tajemnicze kwadratowe nawiasy **[...]** oznaczają ewentualne dodatkowe nazwy katalogów, bo komenda, rzecz jasna, obsługuje tworzenie wielu na raz. Zamiast wpisania samej nazwy katalogu, można też wpisać ścieżkę zakończoną nazwą (całkowitą bądź relatywną do naszej pozycji). Jeśli chcemy stworzyć katalog o wieloczłonowej nazwie to musimy go opatrzyć w cudzysłowie.

2.5.1 Opcje

Oczywiście są jakieś dodatkowe (być może przydatne opcje):

- **-p** (parent) Tworzy brakujące nieistniejące katalogi nadrzędne jeśli polecenie tego wymaga.
- **-v** (verbose) Pokazuje wiadomość potwierdzającą o każdym utworzonym katalogu.
- **-m** (mode) Zmienia tryb tworzenia katalogu tak jak w poleceniu **chmod** .

2.5.2 Przykłady

1. Stworzenie katalogu 'Nowy Folder'.
   `[user@linux ~]$ mkdir ``Nowy Folder``
2. W stworzonym katalogu 'Nowy Folder' tworzymy katalogi 'Praca domowa' oraz 'Research':
   `[user@linux ~]$ mkdir ``Nowy Folder/Praca domowa`` ``Nowy Folder/Research``
3. Tworzenie katalogu 'aaaaa' w nieistniejącym katalogu 'bbbbb':
   `[user@linux ~]$ mkdir -p bbbbb/aaaaa`

2.6 Podsumowanie i ogólna struktura katalogów na Linuxie

Po przyswojeniu tej całej wiedzy o przemieszczaniu się, warto byłoby mieć jakąś "mapę", dzięki której zorientujesz się gdzie co *mniej więcej* się znajduje.

Przedstawię strukturę katalogów w znanej z poprzednich sekcji formie drzewka i dodam jakieś komentarze odnośnie tego jakich elementów można się spodziewać wewnątrz poszczególnych katalogów.

2.6.1 Katalog główny – `/`

Katalogiem głównym nazywamy "najwyższy" katalog w hierarchii. Jego ścieżka to `/`.

Chciałbym nadmienić też dwie ważne rzeczy:

1. Do części katalogów poza `/home/user` możesz nie mieć dostępu. Pewne jest natomiast to, że pliki poza własnym katalogiem domowym będą raczej dostępne nam **wyłącznie tylko do odczytu**, co oznacza że nie będziesz w stanie zmienić ich treści bez odpowiednich *uprawnień*. Jest to normalne, a powód i jego rozwiązanie odnajdziesz w rozdziale Co mogę, a czego nie mogę – czyli uprawnienia.
2. Ta struktura może się nieco różnić zależnie od dystrybucji. Postaram się jak najogólniej przedstawić temat, ale jeśli jakiegoś katalogu u Ciebie nie ma, lub masz więcej, to się nie bój.

```
/
├── bin ..... Zawiera zbiór programów udostępnionym wszystkim użytkownikom systemu (np. ls lub cd ).
│
└── boot ..... Katalog ten zawiera większość programów wymaganych podczas rozruchu systemu Linux.
```

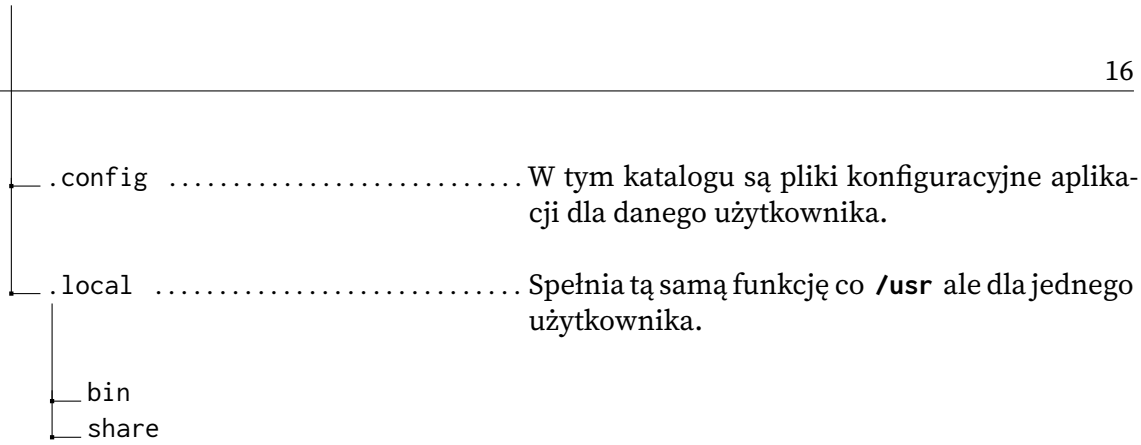
- `dev` — Katalog ten zawiera pliki urządzeń i inne pliki specjalne. Tutaj znajdziemy na przykład pliki reprezentujące nasz procesor, ram, dyski i inne elementy komputera...powiem o tym więcej w Pliki – czyli wszystko co powiedziałem wcześniej było kłamstwem
- `etc` — Tutaj znajdują się pliki konfiguracyjne systemu i usług.
  - `passwd` — Przykładowym plikiem może być `passwd`, który przechowuje dane dotyczące różnych użytkowników albo innych istot.
- `home` — To jest domek o którym już wspomniałem tutaj: Domek – czyli **`/home`** sweet **`/home`** . W skrócie znajdują się tutaj katalogi należące dla użytkowników systemu, w których mogą przechowywać swoje pliki.
  - `user`
- `lib` — Katalog, w którym znajdują się *biblioteki dzielone*, zawierające funkcje dzielone przez wiele programów.
- `lib64` — To samo co wyżej ale w 64 bitach.
- `lost+found` — System w tym miejscu przechowuje pliki odnalezione podczas wykonywania testów dysku.
- `media` — Katalog zawierający pliki z urządzeń (np. pendrive, dodatkowe dyski twarde) montowanych automatycznie.
- `mnt` — To samo co wyżej, ale nie montowanych automatycznie.
- `opt` — Zawiera programy, których kod, dane i konfiguracja znajdują się w jednym katalogu.
- `proc` — Wirtualny katalog, który pozwala na komunikację z jądrem systemu (proszę nie tykać).
- `root` — Katalog domowy administratora systemu.
- `run` — Zawiera dane wykonawcze programów uruchamianych podczas wczesnego startu systemu.

- `sbin` ..... Przechowuje on najważniejsze dla systemu operacyjnego pliki wykonywalne, przeznaczone do użycia przez niego samego lub administratora systemu.
- `srv` ..... Zawiera dane dla usług (serwerów) udostępnianych przez system.
- `sys` ..... Sam nie wiem tbh. nikt nie pamięta o istnieniu tego katalogu.
- `tmp` ..... Zawiera pliki tymczasowe.
- `usr` ..... Zawiera dodatkowe programy, które umożliwiają pracę zwykłemu użytkownikowi systemu.
  - `bin` ..... Kod tych programów.
  - `share` ..... Dane tych programów
- `var` ..... Służy do przechowywania wszystkich zmieniających się danych, jak artykuły grup dyskusyjnych, poczta elektroniczna, strony WWW itp.

2.6.2 Katalog domowy

Tutaj dowiesz się o części rzeczy zwykle domyślnie się znajdujących w katalogu domowym. Jakie pliki i jakie katalogi za co odpowiadają. O plikach i jak nimi manipulować powiem więcej w sekcji Pliki – czyli wszystko co powiedziałem wcześniej było kłamstwem, a tutaj pozostawię mapkę w celu lepszego rozeznania w terenie.

`home`
- `user`
  - `.bash_history` ..... To jest plik w którym zapisywane są twoje wszystkie komendy. Czasem możesz tam zajrzeć jeśli chcesz, chociaż nie wiem do końca po co.
  - `.bashrc` ..... Plik zawierający skrypty, których zadaniem jest ustawianie wartości domyślnych podczas uruchamiania.
  - `.cache` ..... W tym katalogu programy przechowują pliki do których potrzebują czasem dostępu. Może też zawierać starsze wersje aplikacji i inne fafarstki.



3 Pliki – czyli wszystko co powiedziałem wcześniej było kłamstwem

We wcześniejszych sekcjach pisałem dużo o katalogach, nie byłem jednak do końca szczery. Tak na prawdę wszystko w Linuxie jest plikiem, nawet katalogi. Twój procesor? – plik. Twój RAM? – plik. Twoja myszka lub gładzik? – plik. Wiedza ta jest dość kluczowa do zrozumienia niektórych operacji (niekoniecznie w tym rozdziale ale ogólnie).

A teraz Cię podwójnie okłamałem, bo w rzeczywistości jest dokładnie jedna rzecz która nie jest plikiem w Linuxie i jest nią proces. Ale o tym potem...

3.1 Rodzaje plików

Rozróżniamy z grubsza 3 rodzaje plików. Możemy zobaczyć jaki rodzaj ma dany plik przykładowo używając komendy `ls -l` . Pierwszy znak pierwszej kolumny sygnalizuje nam typ.

1. Zwykłe pliki (**-**) – Przykładowo są to pliki tekstowe, pliki binarne programu, pliki pdf itd. Ogólnie rzecz ujmując zbiór danych zapisanych na dysku.
2. Pliki katalogowe (**d**) – Są to pliki zawierające listę innych plików, to co dotychczas nazywaliśmy katalogami albo folderami.
3. Pliki specjalne
   - Plik urządzenia blokowego (**b**) – plik specjalny reprezentujący urządzenie, do którego dostęp realizowany jest poprzez większe porcje danych zwane blokami.
   - Plik urządzenia znakowego (**c**) – plik specjalny reprezentujący urządzenie, do którego dostęp realizowany jest znak po znaku (bajt po bajcie).
   - Plik nazwanego potoku (**p**) – plik wymiany informacji między procesami, działający jako kolejka FIFO (first in first out).
   - Plik dowiązania symbolicznego (**l**) – plik wskazujący na inny plik
   - Plik gniazda (**s**) – plik wymiany między procesami.

Skrypty skrypty skrypty...

Bla bla bla wszystko tu jest WIP

7.1 Twój pierwszy skrypt – `Hello World!`

W pliku **`helloworld.sh`**:

```
1 #!/bin/bash
2 echo "Hello World!"
```

```
[user@linux ~]$ chmod +x helloworld.sh
```

Nadajemy uprawnienia egzekucji naszego skryptu.

```
[user@linux ~]$ ./helloworld.sh

Hello World!
```

7.2 Dodawanie czyli – $((...))

Tworzę zmienną **suma**,

```
1 #!/bin/bash
2 suma=$((2+3))
3 echo $suma
```

8 Istoty drzemiące w głębinach systemu – czyli demony