1. Terminal i polecenia podstawowe

1.1. Konwencje używane w tekście

Zasadniczy tekst niniejszego dokumentu opisany jest czcionką o kroju proporcjonalnym. Czcionka o kroju stałym pojawia się w celu wskazania, że sformatowany nią tekst jest *poleceniem*, wpisywanym z konsoli bądź *komunikatem* systemu operacyjnego wysyłanym do użytkownika. Elementy poleceń zapisane kursywą są *parametrami*, tzn. ciągami znaków o ściśle przypisanym znaczeniu, które muszą zostać określone przed wydaniem polecenia. Wielokropek (**`...`**) stojący za pewnym parametrem oznacza, że parametr ten może zostać *powtórzony* dowolną liczbę razy. Ujęcie pewnego fragmentu polecenia w *nawiasy kwadratowe* (**`[]`**) oznacza, że jest to element opcjonalny i może zostać pominięty.

Na przykład:

`cat` ***`[nazwa_pliku...]`***

- **`cat`** jest nazwą pewnego polecenia systemu Linux
- polecenie to można wydać bez podawania parametrów (nawiasy kwadratowe) bądź też z dowolną liczbą parametrów (wielokropek), będących nazwami plików

Parametry poleceń mogą być *przełącznikami* bądź *argumentami*. Argument to najczęściej **nazwa pliku**, na którym zostanie wykonana wskazana operacja, zaś

przełącznik (*opcja*) jest wskazówką, w jaki szczególny sposób owa operacja ma zostać wykonana. Każde polecenie ma własny unikalny zestaw przełączników, chociaż niektóre z nich (na przykład **-h** (od ang. *help*), najczęściej oznaczające żądanie wypisania krótkiego wyjaśnienia działania polecenia) mogą być wspólne dla wielu różnych poleceń.

Uwaga! Polecenie, argumenty i przełączniki zawsze rozdziela się co najmniej jednym białym znakiem (w tej roli najczęściej występuje tu spacja).

Przykłady kompletnej konwersacji z systemem będą prezentowane jako tekst w ramce. W przykładach takich linie rozpoczynające się od znaku dolara (**$**) bądź kratki (**#**) odzwierciedlają tekst wprowadzony przez użytkownika, pozostałe zaś są odpowiedzią systemu.

Na przykład:

```
$ pwd
/home/user
```

Przykład powyższy obrazuje odpowiedź systemu na wydanie przez użytkownika polecenia **pwd**.

Niektóre z przełączników mogą mieć wersję długą (najczęściej pod postacią słowa lub słów języka angielskiego) i krótką (najczęściej pod postacią jednej litery alfabetu łacińskiego lub cyfry). Przełączniki krótkie poprzedza się jednym znakiem minus (**-**), długie dwoma (**--**). Na przykład polecenie **watch** ma przełącznik **v** (od ang. *version*), który może zostać użyty jako krótki albo długi:

```
watch -v
watch --version
```

W obu przypadkach wynikiem wykonania polecenia będzie ujawnienie wersji używanego programu.

```
$ watch -v
watch from procps-ng 3.3.17
$ watch --version
watch from procps-ng 3.3.17
```

Niektóre z poleceń uruchomione bez parametrów prezentują skrótowy opis przeznaczenia i sposobu uruchomienia, ale nie jest to regułą. Wiele z poleceń reaguje w takiej sytuacji uruchomieniem zachowania domyślnego, które może zdezorientować nieświadomego użytkownika. Na przykład polecenie **cat** w takiej sytuacji zaczyna czekać na wprowadzanie danych z klawiatury nie sygnalizując tego w żaden jawny sposób, co może wywoływać mylne wrażenie, że konsola po prostu się zawiesiła. W takich sytuacjach pomocny może być długi przełącznik **--help**, który w większości przypadków powoduje ujawnienia skrótowego opisu przeznaczenia i działania danego polecenia. Warto zauważyć, że nie wszystkie polecenia dopuszczają postać krótką tego przełącznika, czyli **-h** -– dzieje się tak na przykład w przypadku wspomnianego już polecenia **cat**).

```
$ cat -h
cat: invalid option -- 'h'
Try 'cat --help' for more information.
$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.
With no FILE, or when FILE is -, read standard input.
  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
  -e                       equivalent to -vE
  -E, --show-ends          display \$ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      suppress repeated empty output lines
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as \^I
  -u                       (ignored)
  -v, --show-nonprinting   use \^{} and M-notation, except for LFD and TAB
      --help     display this help and exit
      --version  output version information and exit
```

Przełączniki w postaci krótkiej w większości przypadków mogą być podawane w dowolnej kolejności. Wyjątkiem będą takie polecenia, w których kolejność przełączników narzuca kolejność wykonywanych operacji, ale jest to sytuacja dość rzadka i najczęściej wynika z samej natury działania polecenia.

Przełączniki mogą być również łączone ze sobą, co skraca zapis i czas wydawania polecenia. Na przykład wszystkie poniższe przypadki użycia przełączników **-l** i **-a** są równoważne:

```
ls -l -a
ls -a -l
```

```
ls -al
ls -la
```

W przypadkach, w których plik będący argumentem pewnego polecenia ma nazwę tożsamą z zapisem jakiegokolwiek przełącznika tego polecenia, pomocny może okazać się argument specjalny zapisywany jako **--** powodujący, że wszystkie następujące po nim argumenty traktowane są wyłącznie jako nazwy plików, nawet jeśli ich nazwy zaczynają się od **-** bądź **--** . Konwencja ta warta jest zapamiętania, chociaż takie przypadki zdarzają się raczej rzadko.

Jeśli, na przykład w pewnym katalogu istnieje plik o nazwie **--help** (co jest absolutnie dopuszczalne), próba wypisania jego zawartości poleceniem:

```
cat --help
```

spali na panewce, ponieważ zamiast spodziewanych danych zobaczymy na terminalu tekst pomocy. Aby wybrnąć z tej opresji, powinniśmy napisać:

```
cat -- --help
```

1.2. Praca z terminalem i wydawanie poleceń

Po poprawnym zalogowaniu użytkownika do sesji terminalowej i wyświetleniu systemowej notki powitalnej, system przechodzi w tryb pracy interaktywnej. Oznacza to, że każdy ciąg znaków wprowadzony przez użytkownika i zakończony klawiszem *Enter* jest traktowany przez system (dokładniej: przez proces interpretera poleceń nazywany *powłoką*, ang. *shell*) jak polecenie z ewentualnymi parametrami, które powłoka ma rozpoznać i spróbować wykonać. Należy pamiętać, że rejestr liter użytych przy pisaniu polecenia oraz jego parametrów, w tym także przełączników, ma krytyczne znaczenie: **pwd** jest czymś innym niż **PWD**, a przełącznik **-h** zapewne wymusza zupełnie inne zachowanie niż **-H**. Zasada ta jest jedną z cech silnie odróżniających systemy wywodzące się z Uniksa od systemów rodziny Windows, gdzie rejestr liter nie ma znaczenia ani w nazwach poleceń, ani w ich parametrach, ani nawet w nazwach plików.

Gotowość do wykonywania poleceń jest sygnalizowana przez powłokę wyświetleniem ciągu znaków o zwyczajowej nazwie *prompt*. Słowo to nie doczekało się dobrego polskiego odpowiednika, chociaż można spotkać się z niezbyt fortunnym określeniem *poganiacz*.

Postać prompta jest w pełni konfigurowalna i może różnić się w różnych dystrybucjach Linuksa, najczęściej jednak przybiera postać podobną do poniższej:

```
user@host: dir $
```

gdzie:

user nazwa zalogowanego użytkownika
host nazwa komputera, na którym pracuje powłoka
dir nazwa katalogu bieżącego (pojęcie to zostanie omówione w dalszej części tekstu
$ oznacza, że powłoka pracuje na rzecz użytkownika, który nie jest administratorem systemu (tzn. nie jest użytkownikiem *root* – nie myl z katalogiem **/root**!); prompt administratora zwyczajowo kończy się znakiem **#** (hash), co w założeniu ma wywoływać u użytkownika wzmożenie jego uwagi, jako że w tej sytuacji ewentualnie popełniony błąd może mieć daleko idące konsekwencje dla stabilności całego systemu

Długość polecenia nie jest ograniczona i można je pisać zdając się na domyślne zachowanie terminala, który linię wykraczającą poza prawy margines ekranu samoczynnie przełamie, ale można też przerwać pisanie polecenia w dowolnym momencie, kończąc wpisany fragment znakiem odwrotnego ukośnika (\) i naciskając klawisz *Enter*. System uzna, że polecenie będzie kontynuowane w kolejnej linii i wstrzyma się z jego wykonaniem do chwili skompletowania całości, a zamiast prompta na początku nowej linii zostanie wyświetlony znak **>**.

Pewne kombinacje klawiszy są przez system traktowane specjalnie – najważniejsze z nich ujęte są w poniższym zestawieniu:

Klawisz	Działanie
Alt-F*n* ewentualnie **Ctlr-Alt-F*n***	przełączenie się na wirtualną konsolę o numerze *n* (na przykład **Alt-F2** przełącza na drugą konsolę wirtualną); uwaga – działa tylko i wyłącznie, gdy terminal pracuje w faktycznej konsoli na fizycznym terminalu, a nie w konsoli emulatora terminala
↑ ↓	przywołanie poprzedniego/następnego polecenia z historii poleceń
Tab	aktywowanie „podpowiadacza" nazw plików
Ctrl-Z	przerzucenie bieżącego procesu na drugi plan (polecenie **fg** przywróci proces z powrotem na pierwszy plan)
Ctrl-L	wyczyszczenie okna terminala
Ctrl-C	przerwanie i zakończenie bieżącego procesu
Ctrl-D	wysłanie znaku końca pliku (tak zwany znak *EOF*) do bieżącego procesu; ma sens tylko wtedy, gdy proces faktycznie oczekuje na wprowadzenie danych z konsoli; wysłanie tej kombinacji do powłoki systemowej wywołuje efekt natychmiastowego wylogowania się z systemu

1.3. Uzyskiwanie pomocy

W skład systemu Linux zwyczajowo wchodzi złożony zbiór dokumentów tekstowych opisujących różne aspekty systemu i jego narzędzia. Są to tak zwane *manual pages*, w polskim żargonie znane jako *manuale*. Dostęp do dokumentacji możliwy jest dzięki wbudowanej, interaktywnej przeglądarce, którą uruchamia się poleceniem:

```
man nazwa_polecenia
```

gdzie *nazwa_polecenia* to nazwa programu, usługi systemowej, pliku konfiguracyjnego bądź innych obiektów systemowych, dla których chce się uzyskać pomoc, na przykład:

```
man cd
```

pozwala na uzyskanie pomocy dla polecenia **cd**.

Istnieje także strona pomocy dla samej pomocy systemowej -– można uzyskać do niej dostęp poleceniem:

```
man man
```

Wykonanie polecenia nie uda się, gdy wskazany dokument pomocy nie istnieje.

Strony dokumentacji podzielone są na sekcje, które grupują informacje zawarte w pomocy. Nazwy oraz kolejność sekcji podlegają standaryzacji, dzięki czemu ich studiowanie jest znacznie uproszczone.

W przypadku dokumentów opisujących narzędzia systemowe najczęściej spotykane sekcje to:

- NAME: nazwa oraz krótki komentarz lub wyjaśnienie
- SYNOPSIS: sposoby uruchamiania programu lub polecenia wraz z listą możliwych przełączników
- DESCRIPTION: pełen opis programu oraz szczegółowy opis możliwych do zastosowania przełączników
- CONFIGURATION: opis konfiguracji usługi lub programu
- FILES: opis plików, które wpływają na sposób działania programu lub usługi, na przykład plików konfiguracyjnych
- SEE ALSO: wskazówki dotyczące podobnych lub powiązanych poleceń
- BUGS: opisanie słabych punktów programu oraz okoliczności, w których program może dawać niepoprawne wyniki albo też zachować się w nieprzewidywalny sposób.

Całość dokumentacji podzielona jest na rozłączne rozdziały, które zawierają opisy poleceń i programów określonego typu. I tak:

- rozdział nr 1: „Programy wykonywalne lub polecenia powłoki” (ang. *User commands*) – programy konsolowe, z reguły dostępne dla każdego użytkownika systemu, choć niekoniecznie z takimi samymi funkcjonalnościami, na przykład polecenie **`date`** wykonywane przez zwykłego użytkownika potrafi jedynie podać bieżący czas na wiele różnych sposobów, a użyte przez administratora umożliwia natychmiastową zmianę ustawień zegara systemowego
- rozdział nr 2: „Wywołania systemowe” (ang. *System calls*) – opis funkcji języka C, stanowiących interfejs do usług jądra systemu, na przykład **`man fork`** udostępni opis funkcji jądra służącej do tworzenia nowych procesów
- rozdział nr 3: „Wywołania biblioteczne” (ang. *Library calls*) – standardowe funkcje biblioteczne realizujące funkcjonalności implementowane poza jądrem, na przykład **`man asin`** udostępni opis grupy funkcji obliczających wartość funkcji trygonometrycznej *sin(x)*
- rozdział nr 4: „Pliki specjalne” (ang. *Special files*) – opis przeznaczenia i zachowania plików przechowywanych w katalogu **`/dev`**, które są abstrakcjami urządzeń fizycznych, na przykład polecenie **`man 4 hd`** przedstawi opis plików **`/dev/hd*`** funkcjonujących jako obrazy zawartości dysków IDE wykrytych przez jądro w systemie

- rozdział nr 5: „Formaty plików i konwencje” (ang. *File formats and configuration files*) – opisy przeznaczenia i wewnętrznej struktury różnorodnych plików tekstowych używanych do konfigurowania właściwości systemu operacyjnego, na przykład **`man 5 passwd`** przedstawi opis struktury pliku **`/etc/passwd`**, zawierający dane wszystkich kont użytkowników znanych w danej instalacji
- rozdział nr 6: „Gry” (ang. *Games*) – opis gier i programów służących rozrywce, na przykład polecenie **`man fortune`** zaprezentuje opis programu, który po uruchomieniu wypisuje na konsoli losowa wybraną sentencję, cytat bądź aforyzm (teksty te czerpane są z bazy rozbudowywanej od wielu dekad przez kolejne pokolenia użytkowników systemów uniksowych)
- rozdział nr 7: „Różnorodne” (ang. *Overview, conventions, and miscellaneous*) – opisy różnorodnych konwencji, standardów, protokołów i innych trudno klasyfikowalnych zagadnień, na przykład polecenie **`man 7 man`** udostępni opis formatu plików wykorzystywanych przez polecenie **`man`**
- rozdział nr 8: „Polecenia do administracji systemem” (ang. *System management commands*) – dokumentacja narzędzi z reguły dostępnych tylko dla administratora systemu, na przykład polecenie **`man fdisk`** przedstawi instrukcję użycia podstawowego programu używanego do zarządzania partycjami twardych dysków

Strony pomocy są jednoznacznie identyfikowane za pomocą hasła tematycznego i numeru rozdziału, na przykład **`passwd(1)`** oznacza, że dla polecenia **`passwd`** pomoc systemowa znajduje się w rozdziale nr 1. Jeżeli pewne hasło opisane jest w więcej niż jednym rozdziale, polecenie **`man`** domyślnie wyświetli to, które znajduje się w rozdziale o niższym numerze. Oznacza to, że polecenie postaci:

```
man unlink
```

spowoduje wyświetlenie pomocy dla polecenia **`unlink`** (ten sam efekt dałoby polecenie **`man 1 unlink`**), natomiast polecenie postaci:

```
man 2 unlink
```

wyświetli pomoc dla możliwej do użycia w kodzie napisanym w języku C/C++ funkcji systemowej o nagłówku:

```
int unlink(const char *pathname);
```

dostępnej poprzez plik nagłówkowy **<unistd.h>**.

Pomoc systemowa wyświetlana jest za pomocą systemowego narzędzia **less**, które obsługuje się z klawiatury za pomocą następujących poleceń klawiszowych:

spacja lub *PgDn* – przejście do następnej strony
Ctrl-B lub *PgUp* – przejście do poprzedniej strony ki
q – zamknięcie i opuszczenie przeglądarki
/ – wyszukiwanie w tekście w przód; po znaku **/** należy wpisać tekst do wyszukania i nacisnąć *Enter*
? – wyszukiwanie wstecz (dalsze postępowanie jak wyżej)
n, *N* – przejście do następnego (*n*)/poprzedniego (*N*) wystąpienia poszukiwanego wyrażenia

W zestawie powyższym szczególnie warte zapamiętania jest polecenie *q*, jako że intuicyjnie nasuwające się użycie klawisza *Esc* nie przynosi żadnego rezultatu.

Przeszukiwanie dokumentów pomocy systemowej w celu odnalezienia tych, które zawierają określone słowa kluczowe jest możliwe dzięki programom **apropos** oraz **whatis**, które przeglądają wszystkie dokumenty pomocy dostępne w systemie i wyświetlają listę znalezisk, na przykład wydanie któregoś z poniższych poleceń:

```
apropos passwd
whatis passwd
```

doprowadzi do pojawienia się na konsoli listy dokumentów, które nawiązują bądź mogą nawiązywać do słowa *passwd*. Różnica między obydwoma poleceniami polega na głębokości przeprowadzonego przeszukania:

- **apropos** szuka podanych słów w tytułach dokumentów oraz w rozdziałach DESCRIPTION tych dokumentów
- **whatis** szuka podanego słowa wewnątrz jednowierszowych opisów haseł pomocy (a nie w treści samych dokumentów)

Z tego też powodu wyniki dostarczane przez **apropos** są z reguły znacznie obszerniejsze.

Warto dodać, że oba polecenie akceptują też wieloznaczne definicje szukanych haseł, zapisane przy użycia znaków specjalnych bądź pod postacią wyrażeń regularnych. Oba te tematy pojawią się niebawem w naszych rozważaniach.

Dokładniejsze informacje na temat obu poleceń znajdziesz oczywiście w manualu.

1.4. Drzewo katalogów systemu GNU/Linux

Struktura katalogów systemu Linux tworzy drzewo (z pewnymi wyjątkami, o których opowiemy w dalszej kolejności). Drzewo to ma korzeń, nazywany katalogiem głównym (ang. *root directory*) i oznaczanym znakiem ukośnika:

```
/
```

Każdy katalog może zawierać dowolną (w tym również zerową) liczbę plików, w tym katalogów (w systemach wywodzących się z Uniksa katalog jest szczególną formą pliku). Katalog zawarty w innym katalogu nazywamy *podkatalogiem* (ang. *subdirectory*).

W każdym momencie jeden z katalogów wchodzących w skład drzewa katalogów jest *katalogiem bieżącym* (*roboczym*). Katalog roboczy jest zawsze rozpoznawany pod nazwą skróconą zapisywaną jako pojedynczy znak kropki:

```
.
```

W większości przypadków użycie nazwy pliku pozbawionej informacji o jego położeniu w konkretnym katalogu spowoduje, że powłoka uzna, że chodzi o plik w katalogu bieżącym. Pozwala to na znacznie ułatwienie prac związanych z manipulowaniem plikami.

Reguła ta nie dotyczy jednak uruchamiania programów. Na przykład próba uruchomienia w konsoli własnoręcznie napisanego i skompilowanego programu o nazwie **`hello`** spowoduje, że powłoka przeszuka pewien zestaw katalogów domyślnych i jeśli nie znajdzie w nich pliku o podanej nazwie, zasygnalizuje błąd. Stanie się to także wtedy, gdy plik taki będzie znajdował się w katalogu bieżącym. W takim przypadku polecenie uruchomienie programu **`hello`** musi zawierać wskazania katalogu, w którym należy go szukać i tu właśnie użycie kropki pozwala nam uniknąć żmudnego wprowadzania pełnej ścieżki.

```
$ hello
bash: hello: command not found
$ ./hello
Hello world!
```

Każdy katalog – z wyjątkiem głównego – ma jeden katalog nadrzędny (jest to tak zwany *katalog-rodzic* lub *nadkatalog*, ang. *parent directory*), oznaczany dwiema

kropkami:

```
..
```

Choć brzmi to zdecydowanie paradoksalnie, z punktu widzenia logiki organizacji drzewa katalogów systemów uniksowych, nadkatalogiem katalogu głównego jest sam katalog główny. Przyjęcie takiej zaskakującej fikcji pozwala ujednolicić wiele mechanizmów systemu plików.

Położenie pewnego katalogu względem katalogu głównego zapisuje się jednoznacznie ciągiem nazw katalogów prowadzących od katalogu głównego aż do danego katalogu. Nazwy katalogów rozdziela się znakiem ukośnika (zauważ, że tym samym żaden katalog nie może w swojej nazwie zawierać ukośnika), a całość tworzy tak zwaną *nazwę kanoniczną*. Na przykład:

```
/home/user/subdir
```

opisuje katalog o nazwie *subdir* będący podkatalogiem katalogu *user* będącego podkatalogiem katalogu *home* będącego z kolei podkatalogiem katalogu głównego. Zauważ, że brak pierwszego znaku ukośnika może w sposób radykalny zmienić znaczenie takiego zapisu, a nawet uczynić go niepoprawnym, ponieważ w takim przypadku podgałąź drzewa katalogów będzie interpretowana tak, jakby „wyrastała" z katalogu bieżącego, a nie z katalogu *root*!

Po poprawnym zalogowaniu się katalogiem bieżącym użytkownika (o ile nie zdefiniowano tego inaczej) staje się tak zwany *katalog domowy* (ang. *home directory*), przydzielony każdemu z użytkowników systemu w chwili tworzenia jego konta. Katalog domowy danego użytkownika jest dla niego rozpoznawalny pod skróconym oznaczeniem zapisywanym przy pomocy symbolu tyldy:

```
~
```

Poruszanie się po drzewie katalogów (a tym samym zmianę katalogu bieżącego) umożliwia polecenie **cd** (ang. *change directory)*:

```
cd [nazwakatalogu]
```

Poprawne wykonanie polecenia spowoduje zmianę katalogu bieżącego na wskazany w parametrze. Wykonanie polecenia nie uda się, gdy zajdzie któraś z poniższych okoliczności:

– wskazany katalog nie istnieje

– użytkownik wykonujący polecenie **cd** nie ma uprawnień pozwalających na przemieszczenie się do wskazanego katalogu

W przypadku pomyślnego wykonania katalog bieżący zostanie zmieniony na katalog wskazany w parametrze. Zwróć uwagę na fakt, że postać w jakiej zapisano nazwę katalogu oraz to, w jakim katalogu polecenie wydano, ma dramatyczny wpływ na uzyskany w efekcie wynik. Na przykład:

– **cd home**
przejdź do tego podkatalogu *home*, który znajduje się w katalogu bieżącym
– **cd /home**
przejdź do podkatalogu *home* katalogu głównego

Polecenie cd ma kilka interesujących oboczności:

cd .
określa żądanie przejścia do katalogu bieżącego, a więc – mimo formalnej poprawności – skutek wykonania takiego polecenia będzie żaden (katalog bieżący nie ulegnie zmianie)

cd ..
określa żądanie przejścia do katalogu nadrzędnego i – co może wydać się paradoksalne – zostanie pomyślnie wykonane również na poziomie katalogu głównego, chociaż w tym przypadku nie odniesie żadnych skutków

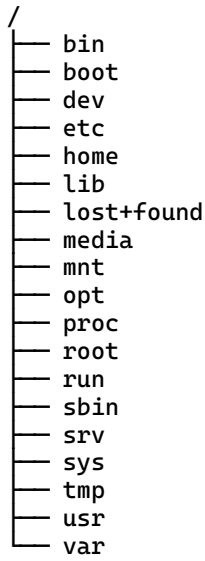
cd
jest najprostszym i najszybszym sposobem przejścia do własnego katalogu domowego, będąc w tej postaci skrótowym zapisem polecenia **cd** ~

cd -
zmienia katalog bieżący na ten, w którym przebywaliśmy poprzednio; regułę tę należy brać dosłownie: kolejno wykonywane polecenia tej postaci będą nas przemieszczać tylko i wyłącznie pomiędzy dwoma ostatnio odwiedzonymi katalogami

1.5. Standardowa hierarchia katalogów systemu GNU/Linux

Struktura katalogów systemu GNU/Linux jest w znaczącej części zestandaryzowana, zarówno jeśli chodzi o konwencje nazewnicze, jak i przeznaczenie poszczególnych katalogów. Zbiór takich konwencji nosi nazwę FHS (od ang. *Filesystem Hierarchy*

Standard – standard hierarchii systemu plików). Różne dystrybucje Linuksa mogą tę konwencję rozszerzać bądź częściowo ignorować, jednak jej zręby wydają się być uniwersalne i najczęściej pierwszy poziom podkatalogów prezentuje się jak poniżej:



Rys. 1..1: Zawartość katalogu głównego według standardu FHS

Przeznaczenie wybranych katalogów prezentuje poniższa lista:

- **`/bin`** – katalog zawierający podstawowe programy, dostępne dla wszystkich użytkowników i niezbędne do rutynowego użytkowania systemu
- **`/boot`** – katalog zawierający pliki niezbędne do poprawnego wystartowania systemu operacyjnego
- **`/dev`** – katalog zawierający tak zwane *pliki specjalne*, reprezentujące dostępne w systemie urządzenia; można uznać, że każdy plik w tym katalogu jest abstrakcją pewnego urządzenia fizycznego rozpoznanego przez system; pewna operacja wykonana na takim pliku będzie faktycznie wykonana ma stowarzyszonym z plikiem urządzeniu; na przykład odczyt 512 bajtów z pliku **`/dev/sda`** może *de facto* oznaczać odczytanie pierwszego sektora urządzenia dyskowego dostępnego w bieżącej konfiguracji komputera
- **`/etc`** – katalog z lokalnymi plikami konfiguracyjnymi systemu; to tu przechowywany jest na przykład plik **`passwd`** przechowujący informace o znanych systemowi kontach użytkowników
- **`/home`** – katalog zawierający podkatalogi domowe użytkowników systemu;

zwyczajowo nazwą katalogu domowego użytkownika *xyz* będzie **/home/xyz**; w instalacjach o dużej liczbie kont użytkowników może zostać wykorzystana konwencja z dodatkowym pośrednim poziomem katalogów i wtedy nazwa przybierze postać **/home/x/xyz**

- **/lib** – katalog przechowujący biblioteki i moduły, wykorzystywane przez pracujące procesy i jądro systemu
- **/lost+found** – katalog zakładany automatycznie w chwili tworzenia systemu plików i przeznaczony do umieszczania w nim plików zagubionych w wyniki awarii, a odnalezionych przez narzędzia naprawcze bez możliwości ustalenia w jakim katalogu pliki te były przechowywane oryginalnie; jest to swego rodzaju „*biuro rzeczy znalezionych*”
- **/media** – katalog, którego podkatalogi reprezentują urządzenia automatycznie i dynamicznie podłączone do systemu, na przykład pendrajwy; operację dołączenia nowego urządzenia do drzewa katalogów nazywa się *montowaniem* (ang. *mount*), a odłączenie urządzenia – *odmontowaniem* (ang. *unmount*)
- **/mnt** – katalog, którego podkatalogi reprezentują urządzenia statycznie dołączone do drzewa katalogów (najczęściej w chwili startu systemu operacyjnego)
- **/opt** – katalog zawierający oprogramowanie opcjonalne, najczęściej instalowane przzez użytkownika i nie stanowiące standardowego wyposażenia systemu operacyjnego
- **/proc** – wirtualny system plików, który dostarcza między innymi informacji o procesach obecnych w systemie oraz o rozpoznanej przez jądro konfiguracji sprzętowej; „wirtualność” plików w tym katalogu oznacza, że pliki te nie są przechowywane fizycznie na nośniku, a są generowane „w locie” na konkretne żądanie
- **/root** – zwyczajowo katalog domowy użytkownika *root*, czyli administratora systemu;
- **/run** – katalog, którego zadaniem jest przechowywanie plików tymczasowych tworzonych przez pracujące procesy, a które to pliki muszą być chronione przed nieuprawnionym usunięciem lub zmodyfikowaniem, czego nie zapewnia starszy koncepcyjnie mechanizm udostępniany przez katalog **/tmp**
- **/sbin** – pełni rolę podobną do katalogu **/bin**, z tym, że przechowuje programy i narzędzia przeznaczone do użytku przez administratora systemu
- **/srv** – katalog przechowujący dane potrzebne do poprawnej pracy serwerów różnych usług systemu operacyjnego
- **/sys** – katalog zawierający wirtualny system plików o działaniu podobnym do katalogu **/proc**, z tym, że przeznaczony do bezpośredniej komunikacji z

usługami jądra systemu, badania ich aktualnego stanu bądź wpływania na ich stan
- **`/tmp`** – katalog przeznaczony do przechowywania plików tymczasowych
- **`/usr`** – katalog zawierający zestaw oprogramowania użytkowego dostępnego dla użytkowników oraz pliki umożliwiające pisanie własnych programów, w tym pliki nagłówkowe i i biblioteki
- **`/var`** – katalog ten zawiera pliki, które często zmieniają swoją zawartość i rozmiar, na przykład kroniki (tak zwane *logi*), pliki tymczasowe różnych procesów i temu podobne

1.6. Manipulowanie katalogami

Jeżeli uprawnienia użytkownika na to pozwalają, może on tworzyć nowe katalogi i usuwać katalogi już istniejące. Używa się do tego celu poleceń **`mkdir`** i **`rmdir`**.

- **`mkdir [przełącznik...] nazwakatalogu...`**
  (od ang. *make directory* – powoduje utworzenie katalogu o wskazanej nazwie; wykonanie polecenie nie uda się, gdy zajdzie jedna z poniższych okoliczności:
  - taki katalog już istnieje
  - posiadane uprawnienia nie pozwalają na tworzenie plików (katalogów) we wskazanym miejscu drzewa katalogów

  Oto kilka przykładów.
  - utworzenie katalogu *xyz* w katalogu bieżącym:
    `mkdir xyz`
  - utworzenie katalogu *xyz* w bezpośrednim nadkatalogu:
    `mkdir ../xyz`
  - utworzenie katalogu *xyz* w katalogu głównym (z reguły możliwe do wykonania tylko w przypadku posiadania uprawnień administracyjnych):
    `mkdir /xyz`
- **`rmdir [przełącznik...] nazwakatalogu...`**
  (od ang. *remove directory*) – usuwa katalog o wskazanej nazwie, przy czym usuwany katalog musi być pusty; wykonanie polecenia nie uda się, gdy ma miejsce jedna z poniższych sytuacji:
  - katalog o takiej nazwie nie istnieje
  - katalog nie jest pusty (zawiera niezerową liczbę plików)
  - użytkownik nie dysponuje odpowiednimi uprawnieniami

  Oto kilka przykładów:
  - usunięcie katalogu *xyz* z własnego katalogu domowego:
    `rmdir ~xyz`

  - usunięcie podkatalogu *xyz* z katalogu bieżącego:
    `rmdir xyz`

  Uwaga: dla poleceń **`rmdir`** i **`mkdir`** dostępny jest przełącznik **`-p`** (od ang. *path*), który pozwala odpowiednio usuwać i tworzyć więcej niż jeden katalog jednocześnie, o ile tylko tworzą one spójną hierarchię, na przykład jednoczesne usunięcie katalogów *ghi*, *def* oraz *abc*, które wspólnie tworzyły hierarchię, można przeprowadzić jednym poleceniem zamiast trzema:
  `rmdir -p abc/def/ghi`
- w dowolnym momencie można zapytać powłokę o nazwę kanoniczną katalogu bieżącego – dokonuje się tego poleceniem:

  `pwd`

  (od ang. *print working directory*), które wypisuje na konsolę pełną (tzn. zaczepioną w katalogu głównym) nazwę aktualnie bieżącego katalogu
- **`pushd`** ***`katalog...`***
  Polecenie **`pushd`** powoduje odłożenie nazw wskazanych katalogów (należy podać co najmniej jedną) na szczyt tak zwanego *stosu katalogów* (ang. *directory stack*; stos katalogów przechowywany jest do końca sesji, po czym jest niszczony; dodatkowo polecenie to wypisuje aktualny stan stosu; odłożenie na stos katalogu bieżącego można wykonać na przykład tak:

  `pushd .`

- **`popd`**
  Polecenie **`popd`** powoduje zdjęcie ze szczytu stosu katalogów znajdującej się tam ewentualnie nazwy (jeśli stos jest pusty, wykonanie polecenia sprowadza się do wyświetlenia komunikatu o błędzie), a następnie wykonuje polecenie **`cd`** dla tej właśnie nazwy; oba powyższe polecenia pozwalają na wygodne zapamiętywanie dowolnego ciągu położeń w drzewie katalogów i szybki powrót do wcześniej zajmowanych miejsc; uwaga: **`pushd`** i **`popd`** są realizowane wewnętrznie przez powłokę, a nie przez zewnętrzny program i co za tym idzie, nie mają swojej strony pomocy; dokładny opis można uzyskać wykonując polecenie

  `man bash`

  i wyszukując stosowny fragment dotyczący współdziałania obu poleceń.
- **`ls`** ***`[przełącznik...] [nazwa_katalogu...]`***

Polecenie **ls** (od ang. *list*) wyświetla informację o zawartości katalogu bądź katalogów, na przykład:

- **ls**
  wyświetla zwięzłe zestawienie zawartość katalogu bieżącego (tylko nazwy i tylko pliki/katalogi widoczne)
- **ls -a**
  wyświetla zwięzłą informację o zawartości katalogu bieżącego, uwzględniając wszystkie pliki (od ang. *all*), to znaczy także te, których nazwa zaczyna się od znaku „.” (umownie są to pliki *ukryte*)
- **ls -al**
  wyświetla informację o wszystkich plikach w katalogu bieżącym z wykorzystaniem tak zwanego *długiego formatu* (od ang. *long*), czyli podając typ każdego obiektu w katalogu (pierwszy znak linii: **d** – katalog, znak **-** – plik zwykły, **l** – dowiązanie), prawa dostępu, liczbę dowiązań, nazwę właściciela, nazwę grupy właścicielskiej, rozmiar (w bajtach), data ostatniej modyfikacji oraz samą nazwę pliku/katalogu
- **ls -al ~**
  jak wyżej, przy czym wyświetlana jest zawartość katalogu domowego
- **ls -al /etc**
  jak wyżej, ale wyświetlana jest zawartość katalogu **/etc**

1.7. Manipulowanie plikami

Plik to kontener do składowania danych. Plik jest przechowywany w systemie plików i opatrzony jest szeregiem atrybutów takich jak nazwa, rozmiar, prawa dostępu, etc. W systemach GNU/Linux większość obiektów systemowych (na przykład urządzeń) jest na poziomie usług systemu prezentowana tak, jakby były plikami, nawet wtedy, gdy ich fizyczna reprezentacja nie ma z plikiem nic wspólnego. Pozwala to na zachowanie spójnego i przenośnego sposobu dostępu i obsługi do wielu heterogenicznych zasobów, np komunikacja procesów poprzez sieć odbywa się tak, jakby transmisja danych była realizowana poprzez zapisy do plików i odczyty z nich.

W systemach wywodzących się z Uniksa nazwy plików nie mają formalnego podziału na nazwę i rozszerzenie (całość nazwy pliku traktowana jest jak jeden spójny ciąg znaków), jednakże mimo tego tradycyjne rozumienie pojęcia *rozszerzenie* daje się pomyślnie stosować.

Możliwe jest stosowanie w nazwach plików znaków specjalnych (na przykład: $, %, :, #), ale nie jest to zalecane.

W szczególności (co stanowi przedmiot bardzo złośliwego, klasycznego uniksowego żartu) można wręcz nadać plikowi nazwę następującą:

```
*
```

Bezrefleksyjna próba usunięcia takiego pliku poleceniem **rm** o poniższej postaci:

```
rm *
```

spowoduje mały kataklizm w katalogu bieżącym, jako że w takim przypadku gwiazdka nie oznacza samej siebie, a staje się znakiem wieloznacznym, pasującym do nazwy dowolnego pliku.

Poprawna i mniej niszczycielska postać takiego polecenia prezentuje się następująco:

```
rm "*"
```

Nazwy plików mogą również zawierać spacje. Powoduje to pewne komplikacje przy manipulowaniu takimi plikami. Na przykład plik nazywający się „*tajny plik*" nie da się wyświetlić poleceniem **cat** wydanym w sposób następujący:

```
cat tajny plik
```

Dzieje się tak dlatego, że w takim przypadku spacja nie jest brana za część nazwy pliku, a za separator rozdzielający parametry polecenia, co w tym przypadku oznacza, że polecenie **cat** uruchomiono z dwiema nazwami plików, a nie z jedną.

W przypadkach, gdy nazwa pliku zawiera znaki nie-alfanumeryczne, może być konieczne albo ujęcie nazwy w cudzysłowy, albo poprzedzenie znaków specjalnych znakiem odwróconego ukośnika (****). Dotyczy to w szczególności sytuacji, gdy znaków **** i **"** używa się wewnątrz nazw plików. Oba poniższe polecenia są poprawne i równoważne:

```
cat "tajny plik"
```

```
cat tajny\ plik
```

Podstawowe operacje manipulacji plikami można realizować z wykorzystaniem następujących poleceń:

– **cp *[przełącznik...] nazwa_pliku... nowa_nazwa_lub_katalog***
  kopiowanie pliku określonego przez pierwszy i ewentualne kolejne argumenty pod nazwę lub do katalogu określonego ostatnim argumentem, na przykład:

  - **`cp abc.txt xyz.txt`**
    kopiuje plik *abc.txt* pod nową nazwę *xyz.txt* w katalogu bieżącym

  - **`cp /tmp/abc.txt ~`**
    kopiuje plik *abc.txt* z katalogu */tmp* do katalogu domowego użytkownika

  - **`cp abc.txt ~/xyz.txt`**
    kopiuje plik abc.txt z katalogu bieżącego pod nową nazwę xyz.txt w katalogu domowym użytkownika

  - **`cp abc.txt xyz.txt ~/documents`**
    kopiuje pliki *abc.txt* i *xyz.txt* z katalogu bieżącego do podkatalogu *documents* w katalogu domowym użytkownika; przypadek ten można uogólnić następującą regułą: jeśli do polecenia **`cp`** podano więcej niż dwie nazwy plików, to wszystkie nazwy oprócz ostatniej są nazwami obiektów do skopiowania, a nazwa ostatnia musi być nazwą istniejącego katalogu, do którego mają trafić kopiowane obiekty.

Przydatnym przełącznikiem polecenia **`cp`** jest **`-r`** (od ang. *recursive*), który służy do kopiowania całych struktur katalogów. Uwaga! Domyślnie polecenie **`cp`** w żaden sposób nie ostrzega przed utratą (nadpisaniem) istniejących plików.

- **`rm [przełącznik...] nazwa_pliku...`**
  usuwanie plików podanych jako argumenty wywołania, na przykład:
  - **`rm abc.txt xyz.txt`**
    usuwa pliki *abc.txt* i *xyz.txt* w katalogu bieżącym
  - **`rm /tmp/abc.txt`**
    usuwa plik *abc.txt* z katalogu */tmp*

  Uwaga! Domyślnie polecenie **`rm`** w żaden sposób nie ostrzega przed utratą istniejących plików. Ponadto, w czasie pracy w konsoli nie działa żaden mechanizm ochronny, znany ze środowisk graficznych pod nazwą *kosza*. Usunięcie pliku/katalogu przy pomocy polecenia **`rm`** jest w większości przypadków nieodwracalne, dlatego wskazana jest daleko idąca ostrożność.
  Przydatnym przełącznikiem polecenia **`rm`** jest przełącznik **`-r`**, który służy do usuwania całych struktur katalogów wraz z zawartością, co pozwala unikać niewygód związanych z użyciem polecenie **`rmdir`**. Opcji używa się często w połączeniu z przełącznikiem **`-f`** (od ang. *force*), który powoduje, że

operacja usuwania przeprowadzana jest bez interakcji z użytkownikiem i bez sygnalizowania błędów.

– **mv *[przełącznik...] nazwa_pliku... nowa_nazwa***
zmienia nazwę pliku określonego pierwszym argumentem wywołania na nazwę określoną ostatnim argumentem wywołania, jednak jeśli ostatni argument jest katalogiem, to wówczas plik zostanie przeniesiony do tego katalogu, na przykład:

  – **mv abc.txt xyz.txt**
  zmiana nazwy pliku *abc.txt* na *xyz.txt* w katalogu bieżącym, chyba że *xyz.txt* jest nazwą katalogu – w takim przypadku plik *abc.txt* zostanie przeniesiony do tego katalogu bez zmiany nazwy
  – **mv /tmp/abc.txt ~**
  przeniesienie pliku **abc.txt** z katalogu **/tmp** do katalogu domowego użytkownika

Uwaga! Domyślnie polecenie **mv** w żaden sposób nie ostrzega przed utratą (nadpisaniem) istniejących plików. Ponadto, wartym zapamiętania jest fakt, że w odróżnieniu od poleceń **cp** i **rm**, polecenie **mv** nie używa opcji **-r**.

– **touch *[przełącznik...] nazwa_pliku...***
modyfikuje informacje na temat czasów modyfikacji i dostępu pliku, ale pozwala także na utworzenie nowego pliku, na przykład:

```
touch abc.txt
```

utworzy nowy (pusty) plik **abc.txt** w katalogu bieżącym o ile taki plik nie istniał już wcześniej; w przeciwnym przypadku **touch** ograniczy się do aktualizacji jego metadanych odnoszących się do czasów modyfikacji i dostępu.

– **cat *[nazwa_pliku...]***
polecenie **cat** (od ang. *concatenate*) zostało wymyślone jako uniwersalny „sklejacz”, pozwalający połączyć ze sobą (skonkatenować) zawartość dowolnej liczby plików, jednak na razie użyjemy go do zupełnie innego celu, a mianowicie do wypisywania zawartości pliku na terminal; wypisywanie plików tekstowych zwykle przebiega bez zakłóceń, jednak próba potraktowania w ten sam sposób pliku binarnego (na przykład wykonywalnego albo zawierającego mapę bitową) może doprowadzić terminal do zachowań niecodziennych (na przykład zaśmiecenie ekranu dziwnymi znaczkami, kakofonia pisków, a na koniec niemożność napisania czegokolwiek, co byłoby widoczne na ekranie). Efekt taki ma źródło w fakcie, że pewne sekwencje znaków są przez terminal traktowane jako znaki sterujące i w niesprzyjają-

cych okolicznościach może to spowodować kompletną dezorganizację jego ustawień. W takich sytuacjach należy zachować zimną krew i nawet na ślepo (nie widząc efektów na ekranie) wpisać polecenie:

```
reset
```

i nacisnąć klawisz *Enter*. Terminal zostanie przywrócony do stanu stabilnego.

– **`less [nazwa_pliku...]`**
prosty *page*r (czyli narzędzie do dzielenia pliku tekstowego na strony i wyświetlania go w sposób interaktywny); klawisze sterujące wyświetlaniem zostały opisane w punkcie poświęconym poleceniu **`man`**

Polecenia dotyczące plików i katalogów można także wydawać z wykorzystaniem tak zwanych *wzorców uogólniających*, znanych także jako *znaki wieloznaczne* (ang. *wildcards*), które tworzy się z zastosowaniem następujących operatorów:

`*` – zastępuje dowolny ciąg znaków (także pusty)
`?` – zastępuje dokładnie jeden dowolny znak
`[<znaki>]` – zastępuje dokładnie jeden znak z podanego zbioru, na przykład **`[xyz]`** zastępuje dowolny z podanych trzech znaków **`x`**, **`y`** lub **`z`**
`[^<znaki>]` – znak *caret* (**`^`**) użyty jako pierwszy znak zbioru oznacza dopełnienie tego zbioru, czyli na przykład **`[^xyz]`** oznacza jeden dowolny znak nie będący ani literą **`x`**, ani **`y`**, ani **`z`**

Oto przykładowe polecenia z wykorzystaniem wzorców uogólniających:

– **`cp ./*.txt`** $\sim$
kopiowanie wszystkich plików z rozszerzeniem **`.txt`** z katalogu bieżącego do katalogu domowego użytkownika

– **`rm ./[0-9]*`**
usunięcie tych plików z katalogu domowego, których nazwa rozpoczyna się od cyfry dziesiętnej

1.8. Inne przydatne polecenia

– **`id`**
Polecenie **`id`** wydane bez parametrów spowoduje wyświetlenie zestawu identyfikatorów przypisanych użytkownikowi wykonującemu to polecenie; są to kolejno:
  – identyfikator użytkownika (*uid*)
  – identyfikator grupy macierzystej (podstawowej) użytkownika (*gid*)
  – lista identyfikatorów wszystkich grup, do których należy użytkownik (*groups*)

```
$ id
uid=1000(user) gid=1000(user) groups=1000(user),6(disk),18(audio)
```

– **who**
Polecenie **who** spowoduje wyświetlenie listy wszystkich zalogowanych użytkowników (dokładniej: wszystkich sesji terminalowych wszystkich zalogowanych użytkowników); podobną funkcję realizują polecenia **w** i **finger**

```
$ who
user     tty1         2022-01-01 12:00
user     pts/0        2022-01-01 13:00 (192.168.0.100)
$ w
14:00:00 up 100 days, 10:20,  2 users,  load average: 0,10, 0,12, 0,12
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
user     tty1     --                11sie22  6days  0.83s  0.21s -zsh
user     pts/0    192.168.0.100    13:27    0.00s  0.17s  0.04s w
$ finger
Login     Name       Tty      Idle  Login Time   Office     Office Phone
user      user      *tty1       6d  Jan 01 11:08
user      user       pts/0          Jan 01 13:27 (192.168.0.100)
```

– **exit**
Polecenie **exit** spowoduje zakończenie bieżącej sesji i wylogowanie użytkownika.

– **watch polecenie** ***[parametr...]***
Polecenie **watch** pozwala wykonywać dowolne inne polecenie w sposób ciągły, nadając mu tym samym pozory interaktywności; domyślne działanie polecenia **watch** można opisać następująco:

1. wyczyść ekran terminala
2. wykonaj wskazane polecenie
3. odczekaj 2 sekundy
4. idź do punktu 1

Kombinacja klawiszy *Ctrl-C* przerywa powyższą pętlę i kończy wykonanie **watch**. Wydanie polecenia postaci:

```
watch ls -al
```

umożliwi bieżący (z dokładnością do dwóch sekund) podgląd zmian w zawartości bieżącego katalogu

1.9. Źródła uzupełniające

1. Cytowanie znaków i łańcuchów w powłoce Bash: `https://www.gnu.org/software/bash/manual/bash.html#Quoting`
2. Manual polecenia **`apropos`**: `https://man7.org/linux/man-pages/man1/apropos.1.html`
3. Manual powłoki Bash: `https://www.man7.org/linux/man-pages/man1/bash.1.html`
4. Manual polecenia **`cat`**: `https://man7.org/linux/man-pages/man1/cat.1.html`
5. Manual polecenia **`cd`**: `https://www.man7.org/linux/man-pages/man1/cd.1p.html`
6. Manual polecenia **`cp`**: `https://man7.org/linux/man-pages/man1/cp.1.html`
7. Manual polecenia **`finger`**: `https://man.openbsd.org/finger`
8. Manual polecenia **`id`**: `https://man7.org/linux/man-pages/man1/id.1.html`
9. Manual polecenia **`ls`**: `https://man7.org/linux/man-pages/man1/ls.1.html`
10. Manual polecenia **`man`**: `https://man7.org/linux/man-pages/man1/man.1.html`
11. Manual polecenia **`mkdir`**: `https://man7.org/linux/man-pages/man1/mkdir.1.html`
12. Manual polecenia **`mv`**: `https://www.man7.org/linux/man-pages/man1/mv.1.html`
13. Manual polecenia **`pwd`**: `https://www.man7.org/linux/man-pages/man1/pwd.1.html`
14. Manual polecenia **`reset`**: `https://www.man7.org/linux/man-pages/man1/reset.1.html`
15. Manual polecenia **`rm`**: `https://www.man7.org/linux/man-pages/man1/rm.1.html`
16. Manual polecenia **`rmdir`**: `https://www.man7.org/linux/man-pages/man1/rmdir.1.html`
17. Manual polecenia **`touch`**: `https://man7.org/linux/man-pages/man1/touch.1.html`
18. Manual funkcji systemowej **`unlink()`**: `https://man7.org/linux/man-pages/man2/unlink.2.html`
19. Manual polecenia **`unlink`**: `https://man7.org/linux/man-pages/man1/unlink.1.html`
20. Manual polecenia **`w`**: `https://man7.org/linux/man-pages/man1/w.1.`

html

21. Manual polecenia **watch**: https://man7.org/linux/man-pages/man1/watch.1.html
22. Manual polecenia **whatis**: https://man7.org/linux/man-pages/man1/whatis.1.html
23. Manual polecenia **who**: https://man7.org/linux/man-pages/man1/who.1.htm
24. Standard FHS 3.0: https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html
25. Znaki wieloznaczne w interpretacji powłoki Bash: https://www.gnu.org/software/bash/manual/bash.html#Pattern-Matching

1.10. Zadania do samodzielnego wykonania

1. Zbadaj szczegółowo działanie „podpowiadacza" nazw plików, który uaktywnia się, gdy podczas pisania polecenia użyjesz klawisza *Tab*; w tym celu przy użyciu polecenia **touch** załóż w dowolnym pustym katalogu pliki o poniższych nazwach:
   - **aaa**
   - **aab**
   - **aac**
   - **aba**
   - **abb**
   - **abc**

   a następnie będąc w tymże katalogu napisz w linii poleceń

   ```
   cat a
   ```

   i naciśnij dwukrotnie klawisz *Tab*. Teraz dopisz literę **a** lub **b** i ponownie naciśnij klawisz *Tab*. Uważnie obserwuj zachowanie konsoli.
2. Przetestuj zachowanie opisane w poniższym fragmencie strony pomocy powłoki *Bash*:
   Uzupełnianie complete (TAB):
   Usiłuje przeprowadzić uzupełnianie tekstu przed punktem. Bash próbuje uzupełniania traktując tekst kolejno: jako zmienną (jeżeli tekst zaczyna się od $), nazwę użytkownika (jeśli tekst zaczyna się od ~), nazwę hosta (jeśli tekst zaczyna się od @) lub polecenie (łącznie z aliasami i funkcjami). Jeżeli żadne z powyższych nie daje dopasowania, to próbowane jest uzupełnianie nazw plików.
3. Sprawdź, jaki efekt wywołują następujące postaci polecenia **cd**:

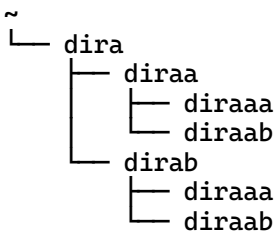
   - **`cd .`**
   - **`cd ..`**
   - **`cd`**
   - **`cd -`**
4. Zbadaj zachowanie poleceń **`pushd`** i **`popd`**.
5. Polecenie **`history`** powoduje wypisanie historii dotychczas wydanych poleceń. Każde z poleceń w historii opatrzone jest numerem. Wydanie polecenia

   ```
   !n
   ```

   (gdzie *n* jest numerem jednego z poleceń w historii) powoduje ponowienie wykonania tego polecenia. Sprawdź, czy po wydaniu takiego polecenia numeracja w historii zmienia się.
6. Użyj dokumentu pomocy polecenia **`ls`**, aby odpowiedzieć na poniższe pytania:
   - jak wypisać nazwy plików/katalogów w jednej kolumnie?
   - jak posortować listę plików według ich rozmiaru?
7. Polecenie postaci

   ```
   ls katalog
   ```

   spowoduje wypisanie zawartości katalogu *katalog*. Jak nakłonić polecenie **`ls`**, aby zamiast tego podało tylko informacje o katalogu *katalog* bez wypisywania jego zawartości?
8. Czy możliwe jest istnienie w pewnym katalogu pliku i katalogu o takiej samej nazwie?
9. Załóż w swoim katalogu domowym następujące poddrzewo katalogów:

   W każdym podkatalogu stwórz poleceniem **`touch`** jeden plik nazwany tak, jak ciąg znaków umieszczony po słowie *dir* w nazwie katalogu (na przykład w katalogu *dira* plik *a*). Użyj tej struktury do przetrenowania następujących działań:

   - skopiowania pliku z *dira/diraa/diraaa* do *dira/dirab/diraab*
   - przemianowania katalogu *dira/dirab/diraab* na *fake*
   - przeniesienia całej gałęzi zaczynającej się na katalogu *dira/dirab* to katalogu *dira/diraa*
10. Uwaga! Bądź ostrożny – to niebezpieczne!
    Jaki efekt wywołuje użycie polecenia **`rm -rf *`** ?
11. Usuń całe drzewo katalogów z zadania 9. wraz z zawartością.
12. Sprawdź zachowanie poleceń **`cp`** i **`mv`** w sytuacji, gdy wywołano je z dwoma argumentami i drugi argument określa:
    - istniejący plik
    - istniejący katalog
    - nieistniejący plik/katalog
13. Sprawdź zachowanie poleceń **`cp`** i **`mv`** w sytuacji, gdy wywołano je z trzema argumentam, z których dwa pierwsze są nazwami istniejących plików, a trzeci:
    - jest nazwą istniejącego pliku
    - jest nazwą istniejącego katalogu
    - jest nazwą nieistniejącego pliku/katalogu
14. Jaki efekt wywołuje użycie opcji **`-h`** polecenia **`ls`**?
15. Polecenie **`df`** powoduje wypisanie informacji o wolnym miejscu na dyskach dostępnych w systemie. Użyj polecenia **`watch`**, aby co jedną sekundę wyświetlać bieżące dane o dostępnej pamięci dyskowej.
16. Co odróżnia od siebie dane prezentowane przez polecenia **`w`**, **`who`** i **`finger`**?
17. Zbadaj co zawiera i do jakich celów używany jest plik ∼**`/.bash_history`**?
    Jaki efekt wywoła usunięcie tego pliku?
18. Co zawiera plik **`/proc/cpuinfo`**?
19. Jaki efekt wywołuje opcja **`-v`** dodana do poleceń **`cp`**, **`mv`** i **`rm`**?
20. Spróbuj użyć polecenia **`cat`** do wypisania na ekranie zawartości pliku **`/proc/kcore`**.
    Wyjaśnij uzyskany efekt wiedząc, że plik ten (a dokładniej rzecz biorąc pseudoplik) odwzorowuje zawartość całej pamięci operacyjnej komputera i tym samym, daje dostęp do przestrzeni adresowej wszystkich obecnych w systemie procesów.