Temat zajęć	Podstawy pracy z terminalem systemu Linux. Polecenia podstawowe.
Zakres materiału	Polecenia: `apropos, cat, cd, cp, exit, history, id, less, ls, man, mkdir, more, mv, popd, pushd, pwd, reset, rm, rmdir, touch, watch, whatis,who`

Konwencje używane w tekście

Zasadniczy tekst niniejszego dokumentu opisany jest czcionką szeryfową. Czcionka bezszeryfowa pojawia się w celu wskazania, że sformatowany nią tekst jest *poleceniem* wpisywanym z konsoli bądź *komunikatem* systemu operacyjnego wysyłanego do użytkownika. Elementy poleceń zapisane kursywą są *parametrami*, tzn. ciągami znaków o ściśle przypisanym znaczeniu, które muszą zostać określone przez wydaniem polecenia. Trzykropek (…) 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` *`[nazwapliku...]`*

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

Parametry poleceń mogą być *przełącznikami* bądź *argumentami.* Argument to najczęściej nazwa pliku, na którym zostanie wykonana wskazana operacja, przełącznik (*opcja*) jest wskazówką, w jaki sposób owa operacja ma zostać wykonana. Każde polecenie ma własny unikalny zestaw przełączników. Polecenie, argumenty i przełączniki rozdziela się co najmniej jedną spacją.

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 (`--`). Np. 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.

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 i w takich przypadkach pomocny może być długi przełącznik `--help`, który w większości przypadków powoduje ujawnienia skrótowego opisu (nie wszystkie polecenia dopuszczają postać krótką, czyli `-h`).

Przełączniki w postaci krótkiej w większości przypadków mogą być podawane w dowolnej kolejności oraz łączone ze sobą. Na przykład wszystkie poniższe przykładu użycia przełączników `-l` i `-a` są równoważne:

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

```
ls -la
```

Praca z terminalem systemu Linux/Unix i wydawanie poleceń

Po poprawnym zalogowaniu się użytkownika do systemu i wyświetleniu systemowej notki powitalnej, system przechodzi w tryb pracy interaktywnej, co oznacza, że każdy ciąg znaków wprowadzony przez użytkownika i zakończony klawiszem *Enter* jest traktowany jak polecenie z ewentualnymi parametrami, które system ma rozpoznać i wykonać. Należy pamiętać, że rejestr liter użytych przy pisaniu polecenia ma znaczenie: `pwd` jest czymś innym niż `PWD`.

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 użytkownikach
- *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 (tzw. użytkownikiem *root* – nie myl z katalogiem `/root`!); prompt administratora będzie się kończył znakiem `#` (*hash*)

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:

Kombinacja	Znaczenie
`Alt-F`*n* ewentualnie `Ctlr-Alt-Fn`	przełączenie się na wirtualną konsolę o numerze n (np. Alt-F2 przełącza na drugą konsolę wirtualną); uwaga – działa tylko i wyłącznie, gdy terminal pracuje w konsoli wirtualnej, a tym samym nie działa 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 bieżącego procesu

`Ctrl-D`	wysłanie znaku końca pliku (EOF) do bieżącego procesu; wysłanie tej kombinacji do powłoki systemowej wywołuje efekt natychmiastowego wylogowania się z systemu

Uzyskiwanie pomocy

W skład systemu Linux zwyczajowo wchodzi złożony zbiór dokumentów tekstowych opisujących różne aspekty systemu i jego narzędzia. Są to tzw. „*manual pages*”, zwane w żargonie „*manualami*”. Dostęp do dokumentacji możliwy jest dzięki interaktywnej przeglądarce, którą uruchamia się poleceniem:

```
man nazwa_polecenia
```

gdzie *nazwa_polecenia* to nazwa programu lub usługi, dla której chce się uzyskać pomoc, np.:

```
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.

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 konfiguracyjnych;
- SEE ALSO – wskazówki dotyczące podobnych lub powiązanych poleceń.
- BUGS – opisanie słabych punktów programu oraz okoliczności, w których może dawać niepoprawne wyniki.

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

- rozdział nr 1 – polecenia użytkownika;
- rozdział nr 2 – wywołania systemowe;
- rozdział nr 3 – funkcje biblioteczne;
- rozdział nr 4 – pliki specjalne;
- rozdział nr 5 – formaty plików;
- rozdział nr 6 – gry;
- rozdział nr 7 – konwersje i rozmaitości;
- rozdział nr 8 – administracja i polecenia administratora;

Strony pomocy są oznaczane za pomocą hasła i numeru rozdziału, np.:

`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. Np. polecenie:

```
man unlink
```

wyświetli pomoc dla polecenia `unlink` (ten sam efekt dałoby `man 1 unlink`).
Polecenie postaci:

```
man 2 unlink
```

wyświetli pomoc dla wywołania systemowej funkcji *unlink()*.
Pomoc systemowa wyświetlana jest za pomocą przeglądarki `more`, którą obsługuje się za pomocą następujących poleceń klawiszowych:

- *spacja* lub *PgDn* – przejście do następnej strony;
- *Ctrl+B* lub *PgUp* – przejście do poprzedniej strony;
- *q* – zamknięcie i opuszczenie przeglądarki;
- / – wyszukiwanie tekstu w przód, po znaku / należy wpisać tekst do wyszukania;
- *?* – wyszukiwanie w tył;
- *n, N* – przejście do następnego (n)/poprzedniego (N) wystąpienia poszukiwanego wyrażenia.

Wyszukiwanie stron pomocy systemowej jest możliwe dzięki programom `apropos` oraz `whatis`, które wyszukują podanych słów w pomocy systemowej, np.:

```
apropos passwd
whatis passwd
```

Manipulowanie katalogami

Struktura katalogów systemu Linux tworzy drzewo (z pewnymi wyjątkami, na które przyjdzie pora później) z jednym korzeniem, nazywanym katalogiem głównym (*root*) i oznaczanym znakiem ukośnika:

```
/
```

Każdy katalog może zawierać dowolną (również zerową) liczbę plików, w tym katalogów (w systemie Linux katalog jest szczególną formą pliku). Katalog zawarty w innym katalogu nazywamy podkatalogiem.

W każdym momencie jeden z katalogów wchodzących w skład drzewa katalogów jest katalogiem bieżącym (roboczym), oznaczanym znakiem kropki:

```
.
```

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

Każdy katalog - z wyjątkiem głównego - ma jeden katalog nadrzędny („*nadkatalog*”), oznaczany dwiema kropkami:

```
..
```

Położenie pewnego katalogu względem katalogu głównego zapisuje się jednoznacznie ciągiem nazw katalogów prowadzących od katalogu głównego do danego katalogu; nazwy katalogów rozdziela się znakiem ukośnika (zauważ, że oznacza to, że żaden katalog nie może w swojej nazwie zawierać ukośnika) , a całość tworzy tzw. *nazwę kanoniczną*. Np.

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

opisuje katalog o nazwie *subdir* będący podkatalogiem katalogu *user* będącego podkatalogiem katalogu *home* będącego podkatalogiem katalogu głównego. Zauważ, że brak pierwszego znaku ukośnika może w sposób radykalny zmienić znaczenie takiego zapisu, a nawet uczynić go niepoprawnym.

Po poprawnym zalogowaniu się katalogiem bieżącym (o ile nie zdefiniowano tego inaczej) staje się tzw. *katalog domowy* (ang. *home directory*), przydzielony każdemu z użytkowników systemu. Katalog domowy oznaczany jest symbolem tyldy:

```
~
```

Poruszanie się po drzewie katalogów 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:

- wskazany katalog nie istnieje
- użytkownik wykonujący polecenie `cd` nie ma uprawnień pozwalających na przemieszczenie się do wskazanego katalogu

Struktura katalogów systemu Linux jest w znaczącej części zestandaryzowana, zarówno jeśli chodzi o konwencje nazewnicze, jak i przeznaczenie poszczególnych katalogów. Zbiór takich konwencji nosi nazwę FHS (od *Filesystem Hierarchy Standard* – standard hierarchii systemu plików). Różne dystrybucje Linuksa mogą tę konwencję rozszerzać bądź ignorować, jednak jej zręby wydają się być uniwersalne i najczęściej pierwszy poziom podkatalogów prezentuje się jak poniżej:

```
/
|-- bin
|-- boot
|-- dev
|-- etc
|-- home
|-- lib
|-- lost+found
|-- media
|-- mnt
|-- opt
|-- proc
|-- root
|-- sbin
|-- srv
```

```
|-- sys
|-- tmp
|-- usr
`-- var
```

Przeznaczenie wybranych katalogów prezentuje poniższa lista:

- `/bin` – katalog zawierający programy niezbędne do uruchomienia systemu;
- `/dev` – katalog zawierający pliki specjalne, reprezentujące dostępne w systemie urządzenia;
- `/etc` – katalog z lokalnymi plikami konfiguracyjnymi systemu;
- `/home` – w tym katalog znajdują się podkatalogi domowe użytkowników systemu;
- `/proc` – wirtualny system plików, który dostarcza m.in. informacji o bieżących procesach w systemie i jego jądrze;
- `/root` – zwyczajowo katalog domowy użytkownika *root*, czyli administratora systemu;
- `/usr` – katalog zawierający zestaw oprogramowania użytkowego dostępnego dla użytkowników;
- `/var` – katalog ten zawiera pliki, które często zmieniają swoją zawartość i/lub rozmiar, np. kroniki (tzw. *logi*)

Jeżeli uprawnienia użytkownika na to pozwalają, może on tworzyć i usuwać katalogi. Używa się do tego celu poleceń:

- `mkdir` *`[przełącznik...] nazwakatalogu...`*

  (od ang. *make directory*) – powoduje utworzenie katalogu o wskazanej nazwie; nie uda się, gdy taki katalog już istnieje bądź gdy posiadane uprawnienia nie pozwalają na tworzenie katalogów we wskazanym katalogu; oto kilka przykładów:

  `mkdir xyz` – utworzenie katalogu *xyz* w katalogu bieżącym;
  `mkdir ../xyz` – utworzenie katalogu *xyz* w katalogu bezpośrednio nadrzędnym

- `rmdir` *`[przełącznik...] nazwakatalogu...`*

  (od ang. *remove directory*) – usuwa katalog o wskazanej nazwie; katalog musi być pusty; nie uda się, gdy katalog nie istnieje bądź nie jest pusty bądź z powodu braku odpowiednich uprawnień; oto kilka przykładów:

  `rmdir ~/xyz` – usunięcie katalogu *xyz* z katalogu domowego;
  `rmdir xyz` – usunięcie katalogu *xyz* z katalogu bieżącego.

  Uwaga: dla poleceń *rmdir* i *mkdir* dostępny jest m.in. przełącznik `-p`, który pozwala odpowiednio usuwać i tworzyć więcej niż jeden katalog jednocześnie, o ile tylko tworzą one spójną hierarchę, np.:

  `rmdir -p abc/def/ghi` – usunie katalogi *ghi, def* oraz *abc*, które tworzyły hierarchię.

- W dowolnym momencie można zapytać system o nazwę kanoniczną katalogu bieżącego. Dokonuje się tego poleceniem

  `pwd` (od ang. *print working directory*).

- `cd` *`[przełącznik...] [nazwa_katalogu]`*

zmiana katalogu na 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 to, jak polecenie zostanie wykonane, np.:

```
cd home
cd /home
```

Polecenie `cd` ma kilka oboczności, które poznasz wykonując ćwiczenia.

- `pushd` *`katalog...`*
  Polecenie `pushd` powoduje odłożenie nazw katalogów (należy podać co najmniej jedną) określonych w argumentach na szczyt tzw. *stosu katalogów*; 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 np. 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 pewnego położenia w drzewie katalogów i szybki powrót w to samo miejsce; 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 obu poleceń.

- `ls` *`[przełącznik...] [nazwa_katalogu...]`*– wyświetlenie zawartości katalogu bądź katalogów, np.:

  ```
  ls
  ```
  wyświetla zawartość katalogu bieżącego;

  ```
  ls -a
  ```
  wyświetla zawartość katalogu bieżącego, uwzględniając wszystkie pliki, tzn. także te, których nazwa zaczyna się od znaku "." (umownie są to pliki ukryte)

  ```
  ls -al
  ```
  wyświetla wszystkie pliki z katalogu bieżącego z uwzględnieniem tzw. "długiego formatu", czyli podając typ każdego obiektu w katalogu (pierwszy znak linii: *d* – katalog, znak "-" – plik zwykły, *l* – dowiązanie), prawa dostępu, liczbę dowiązań, właściciela, nazwę grupy, rozmiar (w bajtach), data ostatniej modyfikacji oraz nazwę

  ```
  ls -al ~
  ```
  jak wyżej, przy czym wyświetlana jest zawartość katalogu domowego;

  ```
  ls -al /etc
  ```

jak wyżej, ale wyświetlana jest zawartość katalogu **`/etc`**.

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 Linux większość obiektów systemowych (np. urządzeń) jest prezentowana tak, jakby były plikami, nawet wtedy, gdy fizyczna reprezentacja nie ma z plikiem nic wspólnego. Pozwala to na zachowanie spójnego sposobu dostępu i obsługi do wielu heterogenicznych zasobów w jeden transparentny sposób. Nazwy plików nie mają podziału na nazwę i rozszerzenie, jednakże można takie podejście stosować. Możliwe jest stosowanie w nazwach plików znaków specjalnych (np.: $, % lub #), ale nie jest to zalecane.

W szczególności (co stanowi przedmiot 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 (czemu?).
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. Np. plik nazywający się „*fajny plik*” nie da się wyświetlić (dlaczego?) poleceniem *cat* wydanym w sposób następujący:

```
cat  fajny plik
```

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 "fajny plik"
cat fajny\ plik
```

Podstawowe operacje manipulacji plikami można realizować z wykorzystaniem następujących poleceń:

- `cp [przełącznik...] nazwa_pliku... nowa_nazwa_lub_katalog`
  kopiowanie pliku określonego przez pierwszy argument pod nazwę lub do katalogu określonego drugim argumentem, np.:

  ```
  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

Przydatnym przełącznikiem polecenia `cp` jest przełącznik `-r`, który służy do kopiowania całych struktur katalogów. Uwaga! Domyślnie polecenie `cp` w żaden sposób nie ostrzega przed utratą istniejących plików.

- `rm [`*`przełącznik...`*`] `*`nazwa_pliku...`*
  usuwanie plików podanych jako argumenty wywołania, np.:

- `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. Przydatnym przełącznikiem polecenia `rm` jest przełącznik `-r`, który służy do usuwania całych struktur katalogów.

- `mv [`*`przełącznik...`*`] `*`nazwa_pliku... nowa_nazwa`*
  zmiana nazwy pliku określonego pierwszym argumentem wywołania na nazwę określoną ostatnim argumentem wywołania. Jeśli ostatni argument wywołania jest katalogiem, to wówczas plik zostanie przeniesiony do tego katalogu, np.:

  `mv abc.txt xyz.txt`
  zmiana nazwy pliku **`abc.txt`** na nazwę **`xyz.txt`** w katalogu bieżącym

  `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 istniejących plików.

- `touch [`*`przełącznik...`*`] `*`nazwa_pliku...`*
  modyfikuje informacje na temat czasów modyfikacji i odczytu pliku, ale pozwala także na utworzenie pliku, np.:

  `touch abc.txt`
  utworzenie (pustego) pliku *abc.txt* w katalogu bieżącym.

Polecenia dotyczące plików (i katalogów) można także wydawać z wykorzystaniem tzw. *wzorców uogólniających (*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 zakresu, np.: `[xyz]`;
- `[^<znaki>]` – znak *caret* (`^`) na początku oznacza dopełnienie zbioru, czyli dla przykładu `[^xyz]`oznacza dowolny znak nie będący literą *x*, *y* i *z*.

Oto przykładowe polecenia z wykorzystaniem wzorców uogólniających:

- `cp ./*.txt ~`
  kopiowanie wszystkich plików z rozszerzeniem *.txt* z katalogu bieżącego do katalogu domowego użytkownika;

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

- `cat` *`nazwa_pliku`*
  Polecenie *cat* (od and. *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 terminalu; wypisywanie plików tekstowych zwykle przebiega bez zakłóceń, jednak próba potraktowania w ten sam sposób pliku binarnego (np. wykonywalnego) może doprowadzić terminal do zachowań bardzo niecodziennych (zaśmiecenie ekranu dziwnym znaczkami, kakofonia pisków, a na koniec niemożność napisania czegokolwiek). Efekt taki ma źródło w tym, że pewne sekwencje znaków są przez terminal traktowane jako znaki sterujące i w niesprzyjających okolicznościach może to spowodować kompletną dezorganizację 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 początkowego

- `more` *`nazwa_pliku`*
  Prymitywny *pager* (czyli narzędzie do dzielenia pliku tekstowego na strony i wyświetlania go w sposób interaktywny); klawisze sterujące wyświetlaniem zostały opisane w punkcie poświęconym poleceniu man

- `less` *`nazwa_pliku`*
  Zaawansowany i wygodny pager, pozwalający na efektywne przeglądanie i przeszukiwanie zawartości plików

Inne przydatne polecenia

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

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

- `exit`
  Polecenie `exit` spowoduje zakończenie sesji i wylogowanie użytkownika

- `watch [`*`polecenie [parametr...]`*`]`

Polecenie `watch` pozwala wykonywać dowolne inne polecenie w sposób ciągły, nadając im tym samym pozory interaktywności; domyślne działanie polecenia *watch* można opisać następująco:

1. wyczyść ekran terminala
2. wykonaj wskazane polecenie
3. odczekaj 5 sekund
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 5 sekund) podgląd zmian w zawartości bieżącego katalogu

Zadania do wykonania w czasie zajęć

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 w tymże katalogu napisz w linii poleceń

   ```
   cat a
   ```

   i naciśnij klawisz *Tab*. Teraz dopisz literę `a` lub `b` i ponownie naciśnij klawisz *Tab*. W drugim kroku w pustej linii poleceń napisz literę

   ```
   y
   ```

   i naciśnij klawisz *Tab* dwukrotnie. Czy wiesz już, jak możesz wykorzystać ten mechanizm w codziennej pracy?
2. Przetestuj zachowanie opisane w poniższym fragmencie strony pomocy powłoki bash:
   Uzupełnianie (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 strony pomocy polecenia `ls`, aby odpowiedzieć na poniższe pytania:
   - jak wypisać wyłącznie nazwy i tylko 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 posiadanie w pewnym katalogu pliku i katalogu o takiej samej nazwie?
9. Załóż w swoim katalogu domowym następujące poddrzewo katalogów:

   ```
   dira
   |
   |--diraa
   |     |--diraaa
   |     \--diraab
   \--dirab
         |--diraa
         \--dirabb
   ```

   W każdym podkatalogu stwórz poleceniem `touch` jeden plik nazwany tak, jak ciąg znaków umieszczony po słowie *dir* w nazwie katalogu (np. w katalogi `dira` plik `a`). Użyj tej struktury do przetrenowania następujących działań:
   - skopiowania plików z `dira/diraa/diraaa` do `dira/diraa/diraab`
   - przemianowania katalogu `dira/dirab/dirabb` na `fake`
   - przeniesienia całej gałęzi zaczynającej się na katalogu `dirab` to katalogu `diraab`
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 drugi argument określa:
    - istniejący plik
    - istniejący katalog
    - nieistniejący plik/katalog
13. Polecenia `cp` i `mv` mogą mieć więcej niż dwa argumenty określające nazwy plików/katalogów; sprawdź, co robią oba polecenia, kiedy użyje się je z trzema nazwami, z których dwie pierwsze są nazwami istniejących plików, a trzecia:
    - 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 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` podana dla poleceń *cp*, *mv* i *rm*?
20. Poleceniem `cat` wypisz na ekran zawartość pliku `/proc/kcore`. Wyjaśnij uzyskany efekt.