Linux i sieci: Podstawowe polecenia Unix'a

Projekt „Matematyka dla Ciekawych Świata",
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2019-05-05

1 Praca w terminalu

Podstawowym sposobem wydawania poleceń w systemach typu Unix jest wpisywanie ich w terminalu. Terminal może pracować w trybie tekstowym lub może być uruchomiony (jako tzw. emulator terminala) w trybie graficznym.

1.1 Komendy

Komendy składają się z nazwy polecenia oraz opcji i argumentów. Nazwą polecenia może być nazwa funkcji wbudowanej, nazwa programu (znajdującego się w ścieżce wyszukiwania programów) lub pełna ścieżka do programu. Po nazwie polecenia mogą występować opcje i/lub argumenty. Są one oddzielane od nazwy polecenia i od siebie przy pomocy spacji[1]. Nie ma silnego rozróżnienia opcji od argumentów, typowo stosowaną konwencją jest rozpoczynanie opcji od pojedynczego myślnika (opcje krótkie - jednoliterowe) lub dwóch myślników (opcje długie). W przypadku stosowania tej konwencji po pojedynczym myślniku może występować kilka bezargumentowych opcji jednoliterowych. Typowo argumenty opcji oddzielane są od nich spacją (w przypadku opcji krótkich) lub znakiem równości (w przypadku opcji długich). Jeżeli któryś z składników komendy (np. argument) zawiera spacje należy je zabezpieczyć przy pomocy odwrotnego ukośnika lub ujęcia zawierającego je napisu w apostrofy lub cudzysłowia.

1.2 Powłoka

1.2.1 bash

Bash jest chyba najpopularniejszą powłoką (interpreterem poleceń). Jest zgodny ze składnią z sh, zapewnia m.in. obsługę zmiennych (zasadniczo napisowych) oraz omówionych w dalszej części skryptu znaków uogólniających. Umożliwia edycję linii poleceń, dopełnianie komend, ścieżek, itp przy pomocy tabulatora oraz zapewnia dostęp do historii linii poleceń (poruszanie się po historii strzałkami, wyszukiwanie przy pomocy Ctrl+R, polecenie `history`).

1.2.2 screen i tmux

`screen` i `tmux` są tzw. multiplexerami terminala - pozwala na uzyskanie wielu okien konsoli (także np. wyświetlanych jedno obok drugiego) na pojedynczym terminalu. Ponadto pozwalają na odłączanie i podłączanie sesji, co pozwala na łatwe pozostawienie działającego programu po wylogowaniu i powrót do niego później.

1.3 Uzyskiwanie pomocy

Informację na temat działania danej komendy oraz jej opcji można uzyskać w wbudowanym systemie pomocy przy pomocy poleceń `man` lub `info` / `pinfo`. Większość poleceń obsługuje także opcje `--help` lub `-h`, które wyświetlają informację na temat ich użycia.

1. zasadniczo dowolnego ciągu białych znaków: spacji, tabulatorów, „escapowanych" nowych linii.

Do zapamiętania

Zarówno w tekstach pomocy jak i w tym dokumencie stosowana jest konwencja polegająca na oznaczaniu opcjonalnych argumentów poprzez umieszczanie ich w nawiasach kwadartowych (jeżeli podajemy ten argument do komendy nie obejmujemy go już tymi nawiasami) oraz rozdzielaniu alternatywnych opcji przy pomocy `|`. Np. `a [b] c|d` oznacza iż polecenie `a` wymaga argumentu postaci `c` albo `d`, który może być poprzedzony argumentem `b`.

1.4 more i less

Jeżeli wynik jakiejś komendy nie mieści się na ekranie do jego obejrzenia możemy użyć poleceń `more` lub `less`. Są to programy umożliwiające przeglądanie tekstu ekran po ekranie. `less` posiada większe możliwości od more (w szczególności posiada możliwość przeglądanie dokumentu w tył)[2]. Programy te kończą się po wciśnięciu klawisza `q`. `less` umożliwia także wyszukiwanie – klawisz `/` pozwala na wprowadzenie szukanej frazy, a `n` na wyszukanie kolejnego wystąpienia. Programy te umożliwiają też wyświetlanie wskazanych jako argumenty plików.

1.5 Przekierowania

Typowo program posiada trzy strumienie danych: jednen wejściowy (stdin) i dwa wyjściowe (stdout i stderr). Standardowe wyjście możemy przekierować na standardowe wejście innego programu przy pomocy `|`, np:

```
ls --help | less
```

Konstrukcja ta przekieruje wynik komendy `ls` uruchomionej z opcją `--help` do komendy `less`.

Możemy także przekierować standardowe wyjście do pliku (przy pomocy `>` lub `>>`, gdy chcemy dopisywać do pliku) lub pobrać standardowe wejście z pliku (przy pomocy `<`). `2>` pozwala na przekierowanie standardowego wyjścia błędu do pliku, a `>&` i `|&` pozwala na przekierowanie obu strumieni odpowiednio do pliku lub standardowego wejścia innego polecenia.

1.6 vi i vim

`vi` jest chyba najbardziej zaawansowanym edytorem, którego obecność gwarantuje standard POSIX. `vim` jest mocno rozbudowanym jego klonem, oferującym bardzo zaawansowane funkcjonalności, powszechnie stosowanym jako zamiennik oryginalnego `vi`. `vim` obsługuje 3 podstawowe tryby pracy: komend (służący do wydawania opisanych niżej poleceń), wizualny (służący do zaznaczania i wydawania niektórych komend), edycji (wstawiania/nadpisywania - służący do wprowadzania tekstu). Podstawowa klawiszologia:

- `Esc` powrót do trybu komend
- `i` tryb wstawiania
- `R` tryb zastępowania
- `Insert` zmiana trybu wstawiania i zastępowania
- `v` tryb wizualny (umożliwia zaznaczenie przy pomocy strzałek)
- `y` skopiuj; `d` - wytnij (skopiuj i usuń)
  po `y`, `d` można podać np. `20l` lub `20[strzałka w prawo]` co oznacza 20 kolejnych znaków, `2w` oznacza dwa słowa
- `x` wytnij (skopiuj i usuń) znak (może być poprzedzone ilością znaków do wycięcia)
- `yy` skopiuj linię; `dd` - wytnij (skopiuj i usuń)
  w obu wypadkach może być poprzedzone ilością linii do skopiowania/wycięcia

2. wybrane przydatne opcje: `-X` nie czyści ekranu przy wychodzeniu z less'a (całość historii wyświetlania pliku pozostaje w historii terminala) `-F` automatycznie kończy gdy wyświetlany tekst mieści się na jednym ekranie

- p wkleja po; P - wkleja przed
- u cofa ostatnią operację
- / szukanie
- n wyszukanie następnego wystąpienie; N wyszukanie poprzedniego wystąpienie
- G przejście do wskazanej linii, numer podajemy przed G, 0 oznacza ostatnią linię w pliku, więc 0G spowoduje przejście do niej
- :[zakres]s@regexp@napis@[g] wyszukaj i zastąp wyrażenie regularne regexp przez napis; zakres może być:
  - numerem linii,
  - przedziałem z numerami linii postaci pierwsza,ostatnia, gdzie:
    . oznacza bieżącą linię, $ oznacza ostatnią linię w pliku, wartość numeryczna poprzedzona + oznacza tyle kolejnych linii od bieżącej, a poprzedzona - przed bieżącą,
  - znakiem % (co oznacza cały plik),
  - zakresem zaznaczonym w trybie wizualnym;

  podanie opcji g powoduje zastępowanie wszystkich wystąpień a nie tylko pierwszego;
  znak @ pełni rolę separatora i może zostać zamiast niego użyty inny znak
- :e ścieżka otwarcie wskazanego pliku
- :w zapis (można także podać ścieżkę pod jaka ma zostać zapisany plik)
- :q wyjście ; :q! wyjście bez zapisywania ; :wq zapis i wyjście
- :n - następny plik ; :N - poprzedni plik
- :split - podział okna ; :vs - pionowy podział okna ; Ctrl+W a następnie strzałka - przełączanie między oknami
- :%!xxd pokazanie wartości numerycznych i umożliwienie edycji pliku jako binarnego;
  :%!xxd -r powrót do normalnej edycji

2 Operacje na systemie plików

2.1 Podstawowe komendy

2.1.1 katalog roboczy

- cd [ścieżka] zmiana bieżącego katalogu,
  warto zauważyć, iż katalogi w ścieżce oddzielamy ukośnikami /, bieżący katalog oznaczamy kropką ., nadrzędny oznaczamy dwiema kropkami ..,
  ścieżki zaczynające się od ukośnika / oznaczają *ścieżki bezwzględne* (od korzenia systemu plików), pozostałe oznaczają *ścieżkę względną* (wyrażoną względem bierzącego katalogu),
  katalog domowy oznacza się tyldą ~
- pwd wyświetla ścieżkę do bieżącego katalogu

2.1.2 wyświetlanie i wyszukiwanie

- ls [opcje] [ścieżka] listowanie zawartości katalogu, do ważniejszych opcji należy zaliczyć:
  -a wyświetlaj pliki ukryte (zaczynających się od kropki)
  -l wyświetlaj pliki w formie listy z szczegółowymi informacjami (uprawnienia, rozmiar, data modyfikacji, właściciel, grupa, rozmiar)
  -1 wyświetlaj pliki w formie 1 plik w jednej linii (bez dodatkowych informacji; stosowane domyślne gdy wynik komendy przekazywany jest strumieniem do innej komendy lub pliku)

`-h` stosuj jednostki typu k, M, G zamiast podawać rozmiar w bajtach
`-t` sortuj wg daty modyfikacji
`-S` sortuj wg rozmiaru
`-r` odwróć kolejność sortowania
`-c` użyj daty utworzenia zamiast daty modyfikacji (stosowane w połączeniu z `-l` i/lub `-t`)
`-d` wyświetlaj informacje o katalogu zamiast jego zawartości

- `find [opcje] [katalog startowy] [wyrażenie]` wyszukiwanie w systemie plików w oparciu o nazwę/ścieżkę lub właściwości pliku, do ważniejszych opcji należy zaliczyć:
  `-P` wypisuj informacje o linkach symbolicznych a nie plikach przez nie wskazywanych (domyślne)
  `-L` wypisuj informacje o wskazywanych przez linki symboliczne plikach
  do ważniejszych elementów wyrażenia należy zaliczyć:
  `-name "wyrażenie"` pliki których nazwa pasuje do wyrażenia korzystającego z shellowych znaków uogólniających
  komenda find (w odróżnieniu np. od ls) samodzielnie interpretują wyrażenia zawierające shellowe znaki uogólniające, w związku z czym konieczne może się okazać zabezpieczenie ich przed interpretacją przez powłokę np. przy pomocy umieszczenia wewnątrz pojedynczych cudzysłowów
  `-iname "wyrażenie"` jak -name, tyle że nie rozróżnia wielkości liter
  `-path "wyrażenie"` pliki których ścieżka pasuje do wyrażenia korzystającego z shellowych znaków uogólniających
  `-ipath "wyrażenie"` jak -path, tyle że nie rozróżnia wielkości liter
  `-regex "wyrażenie"` pliki których ścieżka pasuje do wyrażenia regularnego
  `-iregex "wyrażenie"` jak -regexp, tyle że nie rozróżnia wielkości liter

  `warunek -o warunek` łączy warunki sumą logiczną OR zamiast domyślnego iloczynu logicznego AND
  `! warunek` negacja warunku

  `-mtime [+|-]n` pliki których modyfikacja odbyła się n*24 godziny temu
  `-mmin [+|-]n` pliki których modyfikacja odbyła się n minut temu
  `-ctime [+|-]n` pliki które zostały utworzone n*24 godziny temu
  `-cmin [+|-]n` pliki które zostały utworzone n minut temu
  `-size [+|-]n[c|k|M|G]` pliki których rozmiar wynosi n (c - bajtów, k - kilobajtów, M - Megabajtów, G - gigabajtów)
  w powyższych testach + oznacza więcej niż, - oznacza mniej niż, uwaga: porównywaniu podlegają liczby całkowite, np. +1 oznaczacza > 1 ***w liczbach całkowitych tzn.*** $\geqslant 2$

  `-exec polecenie \{\} \;` dla każdego znalezionego pliku wykonaj `polecenie` podstawiając ścieżkę do tego pliku pod `\{\}` (zastosowane odwrotne ukośniki służą zabezpieczeniu nawiasów klamrowych i średnika przed zinterpretowaniem ich przez powłokę)
  `-execdir polecenie \{\} \;`, podobnie jak `-exec` tyle że polecenie zostanie uruchomione w katalogu w którym znajduje się wyszukany plik
- `du [opcje] ścieżka1 [ścieżka2 [...]]` wyświetlanie informacji o zajętej przestrzeni dyskowej przez wskazane pliki / katalogi, do ważniejszych opcji należy zaliczyć:
  `-s` podaje łączną ilość zajętego miejsca dla każdego argumentów (zamiast wypisywać rozmiar każdego pliku)
  `-c` podaje łączną ilość zajętego miejsca dla wszystkich argumentów
  `-h` stosuje jednostki typu k, M, G
  podawany rozmiar może się różnić (w obie strony) od wyniku ls: ls podaje rozmiar pliku (ile zawiera informacji lub ile zostało zadekarowane że może jej zawierać), a du to ile zajmuje na dysku
- `df [opcje]` wyświetlanie informacji o zajętości miejsca na poszczególnych systemach plików

W ścieżkach i nazwach plików można korzystać z znaków uogólniających powłoki takich jak:

- `?` oznaczający dowolny znak
- `*` oznaczający dowolny (także pusty) ciąg znaków

- `[a-z AD]` oznaczający dowolny znak z wymienionych w zbiorze ujętym w nawiasach kwadratowych, zbiór może być definiowany z użyciem zakresów, np. a-z AD oznacza dowolną małą literę od a do z włącznie, spację, dużą literą A lub D
- `[!a-z]` oznaczający dowolny znak z wyjątkiem znaków wymienionych w podanym zbiorze, zbiór może być definiowany z użyciem zakresów, np. a-z oznacza dowolną małą literę od a do z włącznie

Ścieżki mogą być podawane także jako napisy ujęte w cudzysłowie pojedynczym (', np. `'aaa bbb'`) lub podwójnym (", np. `"aaa bbb"`) celem np. ochrony spacji w nich występujących. Oba typy cudzysłowów zabezpieczają przed rozwijaniem znaków uogólniających (zastępowaniem napisu ze znakami listą pasujących nazw / ścieżek). Cudzysłów pojedynczy (w odróżnieniu od podwójnego) zabezpiecza także przed interpretacją umieszczonych wewnątrz innych znaków specjalnych takich jak odwołania do zmiennych.

Warto zauważyć, iż w przypadku komendy `ls` znaki uogólniające muszą być rozwinięte przez powłokę. Natomiast w przypadku komendy `find` na ogół chcemy aby nie były rozwijane przez powłokę, ale interpretowane przez samą komendę `find`. W tym celu powinniśmy je zabezpieczyć przed rozwinięciem przy pomocy cudzysłowów.

Zauważ również, że jeżeli komanda `ls` w wyniku rozwinięcia znaków uogólniających dostanie jako argument ścieżkę do katalogu to wylistuje jego zawartość (zachowanie to zmienia opcja `-d`).

2.1.3 kopiowanie, przenoszenie, usuwanie, ...

- `cp [opcje] źródło1 [źródło2 [...]] cel` kopiuje wskazany plik (lub pliki) do wskazanej lokalizacji, w przypadku kopiowania wielu plików cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na (rekursywne) kopiowanie katalogów
  `-a` podobnie jak `-r`, dodatkowo zachowując atrybuty plików
  `-l` zamiast kopiować tworzy twarde dowiązania (hard links)
  `-s` zamiast kopiować tworzy linki symboliczne do plików
  `-f` nadpisywanie bez pytania
  `-i` zawsze pytaj przed nadpisaniem
- `ln źródło1 [źródło2 [...]] cel` tworzy link twardy do wskazanego pliku (lub plików) w wskazanej lokalizacji, w przypadku wskazania wielu plików źródłowych cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:
  `-s` tworzy dowiązania symboliczne zamiast twardych
  `-r` używa ścieżki względnej zamiast bezwzględnej przy tworzeniu dowiązań symbolicznych
- `mv [opcje] źródło1 [źródło2 [...]] cel` przenosi wskazane pliki / katalogi do wskazanej lokalizacji, w przypadku przenoszenia wielu plików cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:
  `-f` nadpisywanie bez pytania
  `-i` zawsze pytaj przed nadpisaniem
- `rm [opcje] ścieżka1 [ścieżka2 [...]]` usuwa wskazane pliki, do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na na (rekursywne) kasowanie katalogów wraz z zawartością
  `-f` usuwanie bez pytania
  `-i` zawsze pytaj przed usunięciem
- `mkdir [opcje] ścieżka1 [ścieżka2 [...]]` tworzy wskazane katalogi, do ważniejszych opcji należy zaliczyć:
  `-p` pozwala na tworzenie całej ścieżki a nie tylko ostatniego elementu, nie zgłasza błędu gdy wskazany katalog istnieje

2.2 Struktura katalogów

Systemy unix'owe posiadają drzewiasty system plików zaczynający się w katalogu głównym oznaczanym przez ukośnik (`/`), w którym zamontowany jest główny system plików (rootfs), inne systemy plików mogą

być montowane w kolejnych katalogach. Do najistotniejszych katalogów należy zaliczyć:

- `/bin` zawierający pliki wykonywalne podstawowych programów
- `/sbin` zawierający pliki wykonywalne podstawowych programów administracyjnych
- `/lib` zawierający pliki podstawowych bibliotek
- `/usr` zawierający oprogramowanie dodatkowe (wewnętrznie ma podobną strukturę do głównego - tzn. katalogi `/usr/bin`, `/usr/sbin`, `/usr/lib`, itd)
- `/etc` zawierający konfiguracje ogólnosystemowe
- `/var` zawierający dane programów i usług (takie jak kolejka poczty, harmonogramy zadań, bazy danych)
- `/home` zawierający katalogi domowe użytkowników (często montowany z innego systemu plików, dlatego też root ma swój katalog domowy w `/root`, aby był dostępny nawet gdy takie montowanie nie doszło do skutku)
- `/tmp` zawierający pliki tymczasowe (typowo czyszczony przy starcie systemu); w Linuxie występuje też `/run` przeznaczony do trzymania danych tymczasowych działających usług takich jak numery pid, blokady, itp
- `/dev` zawierający pliki reprezentujące urządzenia; w Linuxie występuje też `/sys` zawierający informacje i ustawienia dotyczące m.in. urządzeń
- `/proc` zawierający informacje o działających procesach (w Linuxie także interfejs konfiguracyjny dla wielu parametrów jądra)

Pliki i katalogi których nazwa rozpoczyna się od kropki traktowane są jako pliki ukryte.

Z punktu widzenia programisty czy też użytkownika (prawie) wszystko jest plikiem, których istnieją różne rodzaje (zwykły plik, katalog, urządzenie znakowe, urządzenie blokowe, link symboliczny, kolejka FIFO, ...); pewnym wyjątkiem są urządzenia sieciowe (które nie mają reprezentacji w systemie plików (ale gniazda związane z nawiązanymi połączeniami obsługuje się zasadniczo tak jak pliki).

Zadanie 2.0.1

Wyświetlić nazwy (mogą być wraz z pełną ścieżką) wszystkich plików i katalogów znajdujących się bezpośrednio w `/etc/` których druga litera to `a` natomiast trzecia to `p` lub `s`.
Wskazówka: to zadanie nie wymaga stosowania find.

Zadanie 2.0.2

Wyszukaj (rekurencyjnie) wszystkie pliki w katalogu `/etc/` zmodyfikowane w przeciągu ostatnich 48 godzin.

Zadanie 2.0.3

Utworzyć w katalogu `/tmp` linki symboliczne `ll1` i `ll2` wskazujące na `/etc/passwd` odpowiednio poprzez ścieżkę bezwzględną i względną.

Zadanie 2.0.4

Zmodyfikuj rozwiązanie zadania 2.0.2 tak aby wyświetlać szczegóły (w tym datę modyfikacji) dla wyszukanych plików.

3 Praca zdalna

3.1 powłoka zdalna

Komenda `ssh [user@]host` umożliwia uzyskanie powłoki zdalnego systemu poprzez szyfrowane połączenie, przydatne opcje:
`-L portLokalny:hostZdalny:portZdalny` tworzy tunel przekierowujący dane kierowane na portLokalny komputera na którym działa klient ssh do portu portZdalny na serwerze hostZdalny poprzez serwer SSH (przydatne gdy hostZdalny jest osiągalny z hostSSH ale nie z komputera lokalnego)
`-D port` tworzy tunel dynamiczny na wskazanym porcie (może on być użyty jako proxy typu SOCKS np. w Firefoxie w celu zapewnienia dostępu do zasobów WWW dostępnych z serwera SSH a niedostępny z komputera lokalnego)
`-p port` określa inny niż domyślny port serwera SSH
`-X` aktywuje przekazywanie komend X serwera ze strony zdalnej do klienta (pozwala na uruchomienie po stronie zdalnej aplikacji z GUI, które zostanie wyświetlone na lokalnym X serwerze)

3.2 zdalne kopiowanie

Najprostszą metą kopiowania plików pomiędzy różnymi systemami jest wykorzystanie do tego ssh, typowo robi się to na jeden z 3 sposobów:

- poleceniem `scp [opcje] źródło1 [źródło2 [...]] cel` , które kopiuje wskazany plik (lub pliki) do wskazanej lokalizacji, w przypadku kopiowania wielu plików cel powinien być katalogiem, do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na (rekursywne) kopiowanie katalogów
  `-P port` określa port SSH
  W odróżnieniu od `cp` źródło lub cel w postaci `[user@]host:[ścieżka]` wskazują na zdalny system dostępny poprzez SSH.
- poleceniem `rsync [opcje] źródło cel` , które kopiuje (synchronizacjiuje) pliki i drzewa katalogów (zarówno lokalnie jak i zdalnie), do ważniejszych opcji należy zaliczyć:
  `-r` pozwala na (rekursywne) kopiowanie katalogów
  `-l` kopiuje linki symboliczne jako linki symboliczne (zamiast kopiowania zawartości pliku na który wskazują)
  `-t` zachowuje czas modyfikacji plików
  `-u` kopiuje tylko gdy plik źródłowy nowszy niż docelowy
  `-c` kopiuje tylko gdy plik źródłowy i docelowy mają inne sumy kontrolne
  `--delete` usuwa z docelowego drzewa katalogów elementy nie występujące w drzewie źródłowym

  `-e 'ssh'` pozwala na kopiowanie na/z zdalnych systemów za pośrednictwem ssh, źródło lub cel w postaci `[user@]host:[ścieżka]` wskazują na zdalny system `--partial --partial-dir=."-tmp-"` zachowuje skopiowane częściowo pliki w katalogu .-tmp- (pozwala na przerwanie i wznowienie transferu pliku)
  `--progress` pokazuje postęp kopiowania
  `--exclude="wzorzec"` pomija (w kopiowaniu i kasowaniu) pliki pasujące do wzorzec (wzorzec może zawierać znaki uogólniające powłoki) `-n` symuluje pracę (pokazuje co zostałoby skopiowane, ale nie kopiuje)
- złożonego polecenia opartego na przekierowaniu wyjścia jakiejś komendy do ssh, które uruchamia po zdalnej stronie proces odbierający te dane na swoim standardowym wejściu, np.:
  - `tar -czf - ścieżka1 [ścieżka2 [...]] | ssh [user@]host 'cat > plik.tgz'` archiwizuje wskazane pliki/katalogi bezpośrednio na zdalny system z użyciem tar i kompresji gzip do pliku `plik.tgz`
  - `tar -cf - ścieżka1 [ścieżka2 [...]] | ssh [user@]host 'tar -xf - -C cel'` kopiuje wskazane pliki/katalogi na zdalny system z użyciem tar do katalogu `cel`

Zadanie 3.0.1

Wylistuj zawartość /etc/ z wskazanej maszyny zdalnej.

Zadanie 3.0.2

Skopiuj plik /etc/hostname z wskazanej maszyny zdalnej do /tmp/hostname.

4 Operacje na plikach

4.1 grep i wyrażenia regularne

Polecenie `grep [opcje] wyrażenie [plik1 [plik2 [...]]]` wyszukuje pasujące do wyrażenia regularnego wyrażenie linie w plikach, przydatne opcje:
`-v` zamiast pasujących wypisz nie pasujące
`-i` ignoruj wielkość liter
`-a` przetwarzaj pliki binarne jak tekstowe
`-E` korzystaj z „*Extended Regular Expressions*" (ERE) zamiast „*Basic Regular Expressions*" (BRE)

`-P` korzystaj z „*Perl-compatible Regular Expressions*" (PCRE) zamiast „*Basic Regular Expressions*" (BRE)

`-r` rekursywnie przetwarzaj podane katalogi wyszukując w wszystkich znalezionych plikach
`-R` jak -r, ale zawsze podąża za linkami symbolicznymi
`--exclude="wyrażenie"` pomiń pliki pasujące do wyrażenie (może zawierać znaki uogólniające powłoki)

`-l` wypisuje pliki z pasującymi liniami
`-L` wypisuje pliki z bez pasujących linii

Wyrażenia regularne[3] konstruuje się w oparciu o następujące znaki specjalne:

```
.      - dowolny znak
[a-z]  - znak z zakresu
[^a-z] - znak z poza zakresu (aby mieć zakres z ^ należy dać go nie na początku)
^      - początek napisu/linii
$      - koniec napisu/linii

*      - dowolna ilość powtórzeń
?      - 0 lub jedno powtórzenie
+      - jedno lub więcej powtórzeń
{n,m}  - od n do m powtórzeń

()     - pod-wyrażenie (może być używane dla powtórzeń, a także referencji
 ↪ wstecznych)
```

4.2 sed i inne narzędzia przetwarzania tekstów

- `sed [opcje] [pliki]` edytuje plik zgodnie z podanymi poleceniami, przydatne opcje:
  `-e "polecenie"` - wykonuj na pliku polecenie (może wystąpić wielokrotnie celem podania wielu poleceń)
  `-f "plik"` - wczytaj polecenia z pliku plik
  `-E` - używaj rozszerzonych wyrażeń regularnych
  `-i` - modyfikuj podany plik zamiast wypisywać zmieniony na stdout
  przydatne polecenia:
  `s@regexp@napis@[g]` - wyszukaj dopasowania do wyrażenia regularnego regexp i zastąp je przez

3. Podana składnia dotyczy „*Extended Regular Expressions*", przy BRE niektóre z znaków sterujących wymagają zabezpieczenia odwrotnym ukośnikiem.

napis, podanie opcji g powoduje zastępowanie wszystkich wystąpień a nie tylko pierwszego, znak `@` pełni rolę separatora i może zostać zamiast niego użyty inny znak
`y@zbiór1@zbiór2@` - zastąp znaki z zbiór1 znakami odpowiadającymi im pod względem kolejności znakami z zbiór2, znak `@` pełni rolę separatora i może zostać zamiast niego użyty inny znak
możliwe jest też m.in. adresowanie linii na których ma być wykonywana operacja, np: `0,/regexp/ s@regexp@napis@` wykona polecenie s na liniach od początku pliku do linii pasującej do wyrażenia regularnego regexp, czyli zastąpi tylko pierwsze wystąpienie w pliku

- `tail [opcje] [plik]` wyświetla ostatnie linie pliku, przydatne opcje:
  `-n x` określa że ma zostać wyświetlone x ostatnich linii
  `-f` uruchamia dopisywania (gdy do pliku zostaną dopisane nowe linie tail je wyświetli)
- `head [opcje] [plik]` wyświetla początkowe linie pliku, przydatne opcje:
  `-n x` określa że ma zostać wyświetlone x pierwszych linii
- `diff ścieżka1 ścieżka2` porównuje pliki lub katalogi (w przypadku tych drugich porównuje ze sobą pliki o takich samych nazwach oraz zgłasza fakt występowania pliku tylko w jednym z katalogów), przydatne opcje:
  `-r` rekursywnie przetwarzaj podane katalogi
  `-u` wypisuje różnice w formacie "unified"
  `-c` wypisuje różnice w formacie "context"
- `patch` stosuje plik łaty (wynik diff'a) w celu zmodyfikowania plików, typowo:
  `patch -pn < plik.diff` co powoduje zastosowanie zmian opisanych w plik.diff na plikach w bieżącym katalogu, n określa ilość poziomów ścieżek podanych w pliku łaty które mają zostać zignorowane
- `sort [plik]` sortuje linie w wskazanym pliku, przydatne opcje:
  `-n` traktuj liczby jako wartości numeryczne a nie napisy
  `-i` ignoruj wielkość liter
  `-r` odwróć kolejność sortowania
  `-k n` sortuj wg kolumny n
  `-t sep` kolumny rozdzielane są przy pomocy separatora sep
- `cut [opcje] [pliki]` wybiera z pliku zadany zestaw kolumn, przydatne opcje:
  `-f nnn` wypierz kolumny określone przez nnn (np. 1,3-4,6- oznacza kolumnę 1, kolumny od 3 do 4 i od 6, a -3 oznacza 3 pierwsze kolumny)
  `-d sep` kolumny rozdzielane są przy pomocy separatora sep (musi być pojedynczym jedno bajtowym znakiem, aby ominąć to ograniczenie należy skorzystać z awk)
- `paste` łączy (odpowiadające sobie pod względem numerów) linie z dwóch plików
- `join` łączy linie z dwóch plików w oparciu o porównanie wskazanego pola
- `comm` porównuje dwa posortowane pliki pod względem unikalności linii (może wypisać wspólne lub występujące tylko w jednym z plików)
- `uniq` usuwa powtarzające się linie z posortowanego pliku, przydatne opcje:
  `-c` wypisz liczbę powtórzeń
  `-d` wypisz tylko linie z 2 lub więcej wystąpieniami
  `-u` wypisz tylko linie z 1 wystąpieniem

Zadanie 4.0.1

Wyświetl plik `/etc/passwd` z zastąpionym `false` przez `FALSE`.

Zadanie 4.0.2

Zmodyfikuj rozwiązanie zadania 2.0.1 w taki sposób aby wyświetlone były tylko nazwy plików bez ścieżek. Przedstaw rozwiązanie z użyciem i bez użycia komendy `cd`.

Zadanie 4.0.3

Napisz polecenie które wyszuka wszystkie wystąpienia napisu `nameserver` w plikach znajdujących się w katalogu `/etc` (wraz z jego podkatalogami).

Zadanie 4.0.4

Zmodyfikuj rozwiązanie zadania 4.0.3 tak aby wyłącznie wypisywało nazwy plików (mogą być ze ścieżką) zawierających wyszukiwany napis.

5 Użytkownicy, uprawnienia i procesy

5.1 Uprawnienia do plików

Podstawowe unixowe uprawnienia do plików składają się z trzech członów: uprawnienia dla właściciela (u), grupy (g) i pozostałych użytkowników (o). W każdym z członów mogą być przyznane uprawnienia do czytania (r), pisania (w) i wykonywania (x); w odniesieniu do plików jest to intuicyjne (uprawnienie do wykonywania jest potrzebne do uruchomienia programów), natomiast w stosunku do katalogów wygląda to następująco: uprawnienia do czytania pozwalają na listowanie zawartości, do wykonania pozwalają na dostęp, do zawartości katalogu (wejścia do niego) do pisania na tworzenie nowych obiektów wewnątrz niego i zmienianie nazw istniejących.

Rozszerzeniem podstawowych uprawnień opisanych powyżej jest mechanizm Filesystem Access Control List (ACL, fACL).

Wszystkie poniższe komendy przyjmują opcję `-R` powodującą rekursywne wykonywanie zmian na drzewku katalogów/plików rozpoczynającym się w podanej ścieżce.

- `chown [opcje] właściciel ścieżka` zmiana właściciela pliku
- `chgrp [opcje] grupa ścieżka` zmiana grupy do której należy plik pliku
- `chmod [opcje] uprawnienia ścieżka` zmiana prawa dostępu do pliku(ów)
- `getfacl [opcje] [ścieżka]` dczyt uprawnień związanych z listami kontroli dostępu fACL
- `setfacl [opcje] [ścieżka]` ustawianie uprawnień związanych z listami kontroli dostępu fACL
- `chattr` zmienia atrybuty plików związanych z systemem plików (np. zabrania jakiejkolwiek modyfikacji pliku)

5.2 Użytkownicy

- `id [użytkownik]` informacja o użytkowniku (m.in. grupy do których należy)
- `whoami` informacja o aktualnym użytkowniku
- `w lub who` informacja o zalogowanych użytkownikach
- `passwd [użytkownik]` zmiana hasła
- `su [użytkownik]` przełącza użytkownika (aby przełączony użytkownik miał dostęp do "naszego" x serwera wcześniej wydajemy `xhost LOCAL:użytkownik`)
- `sudo` program pozwalający na wykonywanie uprzywilejowanych komend przez wyznaczonych użytkowników

5.3 Procesy i zasoby

- `ps [opcje]` wyświetla aktualnie działające procesy i informacje o nich
  np. kombinacja opcji `-Alf` powoduje wyświetlenie wszystkich procesów, w długim, pełnym formacie wypisywania
- `top` monitorowanie procesów obciążających CPU, pamięć, itd

- `iotop` monitorowanie procesów obciążających I/O
- `kill [opcje] pid` przesyła sygnał do procesów o podanych PID
- `killall [opcje] nazwa` przesyła sygnał do procesów o pasującej nazwie

5.4 Planowanie zadań

Typowo system zapewnia usługę uruchamiania zadań o zadanym czasie. Z usługi tej można skorzystać przy pomocy poleceń:

- `crontab` pozwala oglądać i edytować tablice zaplanowanych zadań cyklicznych (dla cron'a)
- `at` pozwala jednorazowo zaplanować zadanie

Pliki konfiguracyjne crona / obsługiwane crontab-em mają postać: `minuty godzina dzienMiesiaca miesiac dzienTygodnia polecenie`. Wpis oznacza że polecenie ma zostać wykonane jeżeli wszystkie warunki będą spełnione, jeżeli jakiś warunek nie jest nam potrzebny można użyć gwiazdki `*`, z kolei `*/n` oznacza wykonywanie jeżeli dana wartość jest podzielna przez n. Np.: `*/20 3 * * 1 ls` oznacza wykonanie komendy ls w każdy poniedziałek o godzinie 3:00 3:20 i 3:40

Standardowe wyjście, wyjście błędu oraz powiadomienie o niezerowym kodzie powrotu domyślnie są wysyłane na lokalny adres mailowy użytkownika będącego właścicielem danego contaba. Niekiedy dostępny jest także `anacron` pozwalający na mniej precyzyjne planowanie zadań.

6 Literatura dodatkowa

Więcej informacji o podstawach działania w systematach „unixowych" oraz pełniejsza lista poleceń zamieszczona została pod adresem http://vip.opcode.eu.org/#Systemy_unix-owate.