Najważniejsze polecenia

cat

wyświetla zawartość plików

Postać: `cat [opcje] [plik...]`

Przykład:

```
> cat /etc/passwd
```

Polecenie `cat` może też posłużyć do tworzenia plików tekstowych

```
> cat > pliktekstowy
to jest tekst
który zostanie umieszczony
w pliku o nazwie pliktekstowy
Aby zakończyć wciśnij Ctrl+ D
```

lub do łączenia kilku plików w jedną całość - rezultat można przekierować do pliku:

```
> cat pliktekstowy dane.txt > nowy.txt
```

more

wyświetla zawartość pliku strona po stronie

Postać: `more [opcje] plik`

Przykład:

```
> more /etc/passwd
```

wyświetli zawartość pliku `passwd`

```
> ls /bin | more
```

pozwala przejrzeć listę plików w katalogu `/bin`

less

wyświetl zawartość pliku strona po stronie

Postać: `less [opcje] plik`

Jest to ulepszona wersja polecenia `more` pozwalająca poruszać się po pliku zarówno w przód jak i w tył.

Przykład:

```
> less /etc/passwd
```

Programy `more` i `less` posiadają wiele funkcji dostępnych za pomocą skrótów klawiszowych o których możemy się dowiedzieć wciskając `h`. Inne przydatne funkcje uzyskamy wciskając: `q` - wyjście z programu, `/` - poszukuje *wyrażenia* w pliku.

head

wyświetla początek pliku

Postać: `head [opcje] plik...`

Przykład:

```
> head /etc/passwd
```

wyświetli 10 pierwszych linii w pliku `passwd`

Najważniejsze opcje:

`-n liczba` wyświetli określoną liczbę początkowych linii

`-c liczba` wyświetli określona liczbę początkowych znaków

Przykład:

```
> head -c 10 /etc/passwd
```

wyświetli 10 pierwszych znaków pliku `passwd`

```
> ls | head -n 3
```

wyświetli nazwy trzech pierwszych plików z bieżącego katalogu

tail

wyświetla koniec pliku

Postać: `tail [opcje] plik...`

Działanie i opcje takie same jak w poleceniu `head` z tą różnicą, że wyświetlane jest zakończenie pliku

Przykład:

```
> tail -n 4 /etc/passwd
```

wyświetli cztery ostatnie linie pliku `passwd`

cmp

porównuje pliki znak po znaku

Postać: `cmp [opcje] plik1 plik2`

Polecenie wyświetla pozycje pierwszej napotkanej różnicy w zawartości plików.

Przykład:

```
> cmp plik1.txt plik2.txt
plik1.txt plik2.txt różnią się: bajt 30 linia 2
```

Najważniejsze opcje: `-c` wypisuje różniące się znaki

diff

znajduje różnice pomiędzy plikami

Postać: `diff [opcje] plik1 plik2`

Przykład:

```
> diff plik1.txt plik2.txt
```

Wynikiem działania jest wyświetlenie fragmentów które są różne w obu plikach wraz z informacją jak należy zmienić pierwszy z plików aby otrzymać drugi (`c` zamień, `d` usuń, `a` dodaj fragment tekstu).

Np.:

`1,10c2,5` oznacza, że należy zamienić linie od 1 do 10 w pierwszym pliku na tekst który występuje w liniach od 2 do 5 w drugim pliku.

`3a5` oznacza, że w linii trzeciej pierwszego pliku należy dodać 5 linię z drugiego pliku

wc

liczy ilość znaków, słów i linii w pliku

Postać: `wc [opcje] plik...`

Najważniejsze opcje:
`-c` drukuje liczbę znaków/bajtów w pliku
`-w` drukuje liczbę wyrazów w pliku
`-l` drukuje ilość linii w pliku
Przykład:

```
> wc -c dane.txt
```

wyświetli ilość bajtów zajętych przez plik
Przykład:

```
> wc -l *.txt
```

wyświetli liczbę linii we wszystkich plikach o rozszerzeniu .txt znajdujących się w bieżącym katalogu.

sort

sortuje zawartość pliku tekstowego

Postać: `sort [opcje] plik...`

Przykład:

```
> sort dane.txt > posortowane.txt
```

spowoduje posortowanie linii zawartych w pliku `dane.txt` i przesłanie wyniku do pliku `posortowane.txt`
Niektóre opcje:
`-r` odwraca kolejność sortowania
`-u` usuwa duplikaty
`-f` wyłącza rozróżnianie małych i dużych liter
`-n` sortowanie liczb (standardowo dane sortowane traktowane są jako ciągi znaków)
Przykład:

```
> du . | sort -n
```

wyświetli listę plików w bieżącym katalogu posortowaną według rozmiaru
`+liczba` pozwala pominąć przy sortowaniu określoną liczbę pól (pola standardowo są rozdzielone białymi znakami (przestarzała wersja))
`-k poz1[,poz2]` pozwala specyfikować względem którego pola (kolumny) chcemy sortować
`-t separator` używa podanego znaku jako separatora pól (kolumn)
Przykład:

```
> ls -l | sort +4 -n
```

wyświetli posortowaną listę plików według piątej kolumny otrzymanej za pomocą polecenia `ls -l`

```
> sort -k 5 -t : /etc/passwd
```

Wyświetli posortowana listę użytkowników (piąta kolumna pliku `passwd`, gdzie kolumny sa oddzielone dwukropkami).

grep

wyświetla linie pasujące do wzorca

Postać: `grep [opcje] wzorzec [plik...]`

Przykład:

```
> grep student /etc/passwd
```

wyświetli linie z pliku `/etc/passwd` zawierającą słowo tudent Często stosuje się to polecenie jako filtr w strumieniu, np:

```
> ls /bin | grep z | wc -l
```

wyświetli liczbę plików z katalogu `bin` zawierających w nazwie literę Najważniejsze opcje:

`-v` wyświetlane są wiersze w których wzorzec nie pojawia się

`-l` wyświetli tylko nazwę pliku w którym znaleziono wzorzec

`-i` nie rozróżnia dużych i małych liter we wzorcu

`-A n` wyświetla także n kolejnych linii

`-B n` wyświetla także n poprzedzających linii

cut

Wypisuje wybrane fragmenty linii

Postać: `cut [opcja]... [plik]...`

Niektóre opcje:

`-b N` wypisuje tylko podane bajty

`-f N` wypisuje tylko podane kolumny (standardowo separatorami kolumn są białe znaki)

`-d znak` użyj podanego znaku jako separatora kolumn

Przykład:

```
> cut -c 1 /etc/passwd
```

wyświetli tylko pierwszy znak z każdej linii.

```
> cut -c 4-7 plik
```

wyświetli znaki od 4-go do 7-go.

```
> cut -f 2- plik
```

Wyświetli linie bez pierwszej kolumny

```
> cut -d : -f 5 /etc/passwd
```

wyświetli imiona i nazwiska użytkowników (5 kolumna pliku gdzie kolumny oddzielone są dwukropkiem).

tr

Zamienia znaki wczytane ze standardowego wejścia.

Postać: `tr łańcuch1 łańcuch2`
`tr -d łańcuch`
`tr -s łańcuch`

Najważniejsze opcje:
`-d` usuń podane w łańcuchu znaki
`-s` usuń wielokrotne wystąpienia tych samych znaków
Przykład:
`> echo $PATH | tr : ' '`
wyświetla wartość zmiennej $PATH zastępując dwukropki spacjami.
`> echo Witaj swiecie | tr ai ia`
w podanym haśle zamienia literę 'i' na 'a' oraz literę 'a' na 'i'
`> echo Witaj swiecie | tr [a-z] [A-Z]`
zamienia małe litery na duże
`> cat plik | tr -d ' '`
usuwa spacje z pliku
`> cat plik | tr -s ' '`
usuwa powtórzenia spacji w pliku

Pętla while

Najpierw sprawdza warunek czy jest prawdziwy, jeśli tak to wykonane zostanie polecenie lub lista poleceń zawartych wewnątrz pętli, gdy warunek stanie się fałszywy pętla zostanie zakończona.
Składnia:

while warunek
do
polecenie
done
Przykład:

#!/bin/bash
x=1;
while [$x -le 10]; do
echo "Napis pojawił się po raz: $x"
x=$[x + 1]
done
Sprawdzany jest warunek czy zmienna x o wartości początkowej 1 jest mniejsza lub równa 10, warunek jest prawdziwy w związku z czym wykonywane są polecenia zawarte wewnątrz pętli: echo "Napis pojawił się po raz: $x" oraz x=$[x + 1], które zwiększa wartość zmiennej x o 1. Gdy wartość x przekroczy 10, wykonanie pętli zostanie przerwane.

sed

Edytor strumieniowy

Postać: `sed [-n] [-e skrypt] [opcja]... [plik]...`

Odczytuje kolejne linie ze strumienia wejściowego (lub pliku), dokonuje edycji zgodnie z podanym skryptem i wynik wyświetla na standardowym wyjściu.
Najważniejsze opcje:
`-n` hamuje normalne wyjście (wyświetlanie tylko linii wskazanych w skrypcie komendą `p`)
`-e` wykonają podany skrypt (pojedyncze polecenie). Jeśli podajemy tylko jedną komendę ta opcja nie jest wymagana.
Składnia skryptu:
`[adres[,adres]] funkcja [argumenty]`
`adres` to numer linii pliku (`$` oznacza numer ostatniej linii) lub wyrażenie regularne umieszczone pomiędzy znakami `/`
. Określa on zakres linii strumienia na których będę dokonywane operacje. Na przykład `1,3` pasuje do pierwszych trzech linii, `/bash/` pasuje do wszystkich linii zawierających wyrażenie `bash`, zaś `/begin/,/end/` dotyczy wszystkich kolejnych linii z których pierwsza zawiera słowo `begin` a ostatnia słowo `end`. `funkcja` do wyboru mamy wiele możliwości edycji strumienia. Najważniejsze to:
`a tekst` dodaj podany tekst przed następną linią
`c tekst` zamień linię podanym tekstem
`d` usuń linię
`i tekst` wstaw podany tekst
`p` wyświetl bufor (aktualnie edytowaną linię)
`s/wyrażenie/łańcuch/` zastępuje podanym `łancuchem` pierwsze znalezione w buforze `wyrażenie`
`s/wyrażenie/łańcuch/g` zastępuje podanym `łancuchem` wszystkie znalezione w buforze `wyrażenia`
`=` wyświetla numer linii
Przykłady:
`> sed -n '1p' plik`
wyświetli pierwsza linię pliku
`> sed -n '3,$p' plik`
wyświetli wszystkie linie od 3-ciej to końca pliku
`> sed '3,$d' plik`
usunie wszystkie linie od 3-ciej do końca pliku
`> sed -n '/Marek/p' /etc/passwd`
wyświetla linie zawierające słowo `Marek` z pliku `/etc/passwd`
`> sed '/UNIX/c Linux' plik`
Zamienia linie w których występuje słowo `UNIX` zwrotem `Linux`
`> sed -n '/UNIX/=' plik`
wyświetla numery linii w których występuje wyrażenie `UNIX`
`> sed 's/UNIX/Linux/g plik`
zamienia wszystkie wystąpienia słowa `UNIX` na `Linux`
`> sed -n 's/UNIX/Linux/g plik`
tak jak wyżej ale wyświetlane są wyłącznie linie w których nastąpiła zmiana

Wyrażenia regularne

Wybrane metaznaki wyrażeń rozszerzonych wyrażeń regularnych (POSIX ERE, *ang. Extended Regular Expressions*)
`[lista]` pasuje do pojedynczego znaku z danej listy
`[^lista]` pasuje do znaku nie podanego na liście
. (kropka) pasuje do dowolnego pojedynczego znaku
`\w` jest równoważne `[0-9a-zA-Z]` lub `[[:alnum:]]`, czyli zastępuje dowolna literę lub cyfrę
`\W` oznacza to samo co `$[^[[:alnum:]]]`
`^` i `$` to odpowiednio początek i koniec linii
`\<` oraz `\>` początek i koniec słowa
Po wyrażeniu regularnym mogą stać operatory powtórzenia:
`?` poprzedzający element pasuje zero lub jeden raz, np. `miark?a` pasuje do `miarka` ale też `miara`
`*` poprzedzający element pasuje zero lub więcej razy, np `W*in` pasuje zarówno do słowa `Windows` jak i do `Linux`
`+` poprzedzający element pasuje jeden lub więcej razy,
`{n}` poprzedzający element pasuje dokładnie `n` razy
`|` operator LUB, np. `Fizyka|fizyka` pasuje do `fizyka` oraz `Fizyka`
`()` grupowanie, np. `fizy(ka|cy)` pasuje zarówno do `fizyka` i `fizycy`.

Uwaga: w podstawowych wyrażeniach regularnych (POSIX BRE *ang. Basic Regular Expressions*) stosowanych w większości narzędzi UNIXowych metaznaki `?`, `+`, `{}`, `()`, `|` tracą swoje szczególne znaczenie; zamiast nich należy użyć `\?`, `\+`, `\{\}`,`\(\)`, `\|`.

Przykłady:
`grep 'bash$' /etc/passwd`
linie zakończone słowem `bash` w pliku `/ert/passwd`
`grep '\<[aA]' plik`
linie w których występuje wyraz rozpoczynający się literą `a` lub `A`
`grep '^From: ' /var/mail/$USER`
lista odebranej poczty (linie rozpoczynające się słowem `From:`)
`grep -v '^$' plik`
wszystkie linie, które nie są puste
`grep '[0-9]\{9\}' plik`
dziewięciocyfrowe ciągi liczb, np. numery telefonów
`grep '(.\+) plik'`
psuje do dowolnego ciągu składającego się przynajmniej z jednego znaku zawartego w nawiasach

Inne przydatne polecenia i narzędzia (textutils): nano, emacs, vi, vim, awk, join, paste, tac, nl, od, split, csplit, uniq, comm, ptx, tsort, tr, fold

Potoki

`polecenie1 | polecenie2`
połączenie wyjścia programu 1 z wejściem programu 2

Przykłady:

```
> cat /etc/passwd | wc -l
> grep Marek /etc/passwd | cut -f 5 -d : | sort | head -n 1 > wybraniec
```

tee

czyta standardowe wejście i przesyła je na standardowe wyjście oraz do pliku.

Postać: `tee [-a] plik`

Najważniejsze opcje:
`-a` dopisuje zawartość strumienia wyjściowego do pliku (bez tej opcji zawartość pliku zostałaby nadpisana)
Przykład:

```
> grep Marek /etc/passwd | tee plik1.txt | wc -l
```

zapisze linie z pliku `/etc/passwd` zawierające słowo `marek` w pliku `plik1.txt`, zaś na ekranie wyświetlona zostanie ilość tych linii.