man :wyswietla stronę manuala dla polecenia 'program'

info : podobnie jak man, wyświetla stronę pomocy dla polecenia 'program'

--help : każdy program ma opcję --help lub -h, która wyświetla któtką pomoc

logout : komenda służąca do wylogowania sie z terminala

exit : komenda służąca do zakończenia procesu powłoki: tcsh, bash, itp.

Może być używana w skryptach. Pozwala też zwrócić kod wyjścia ze skryptu:

- exit
- exit 127

passwd : zmień hasło użytkownika

^D : (Ctrl-D) wysyła komunikat EOF (end-of-file) do terminala

Oznacza koniec wprowadzania danych. W powłoce 'bash' skutkuje zazwyczaj zamknięciem terminala

whoami : wyświetla nazwę użytkownika

id : wyświetla obecną nazwę i grupę użytkowinka oraz ich numery (UID i GID)

su : komenda do przelogowania się jako inny użytkownik (su, od 'switch user')

Pozwala, w tym samym terminalu, zmienić uprawnienia do wykonywania komend chwilowo na innego użytkowika (zmienić aktualny UID i GID)

- *su username :* zmień użytkownika na użytkownika o nazwie 'username'
- *su - username :* zmień użytkownika, wyczyść zmienne środowiskowe i ustaw nowe wartości zmiennych HOME, SHELL, USER, LOGNAME, PATH

finger : wypisuje wszystkich użytkowników obecnie zalogowanych na danej maszynie (i ich terminale)

- finger
- *finger @komputer :* użytkownicy zalogowani na komputerze o nazwie 'komputer;
- *finger username :* wypisuje informacje o użytkowniku 'username' i wszystkie terminale na których jest zalogowany
- *finger xxxx :* wypisuje informacje o wszystkich użytkownikach o imieniu xxxx

w : podobnie jak finger wypisuje zalogowanych użytkowników i ich aktywne terminale

Cd : zmienia aktualny katalog (od 'change directory')

- *cd dirname :* zmienia aktualny katalog na 'dirname'
- *cd dir1/dir2/dir3 :* wchodzi do katalogu 'dir3', który jest w katalogu 'dir2', który jest w 'dir1'
- *cd :* z dowolnego miejsca, zmienia katalog na domowy
- *cd .. :* przechodzi do katalogu o jeden wyższego w drzewie katalogów niż obecny
- *cd /home/dir :* z dowolnego miejsca, przechodzi do katalogu zaczynając od początku drzewa: /
- *cd - :* przechodzi do poprzedniego katalogu

pwd : wypisuje scieżkę obecnego katalogu (od 'print working directory')

ls : listuje katalog

- *ls :* listuje katalog . (ls .)
- *ls plik1 plik2 plik3 :* listuje tylko wymienione pliki
- *ls *.txt :* wypisze wszystkie pliki o nazwie konczącej się na '.txt'
- *ls katalog1 katalog2 :* listuje wymienione katalogi
- *ls -l :* szczegółowa lista
- *ls -a :* wypisuje również ukryte pliki (czyli te których nazwa zaczyna się kropką)
- *ls -R :* listuje katalogi rekursywnie (czyli wyświetla również zawartość podkatalógow)
- *ls -d :* wyświetla tylko nazwy katalogow, tak jak zwyczajnych plików, czyli nie listuje ich zawartości

cat : wypisuje wszystkie podane mu pliki na standardowe wyjście

- *cat plik :* jeśli nie przekierujemy standardowego wyjścia do innego pliku (>, >>) lub programu (|), to wypisze plik na ekran
- *cat plik1 plik2 plik3 :* wypisze po kolei zawartość wszystkich plików

tac : wypisuje wszystkie podane mu pliki na standardowe wyjście, ale w odwraca kolejność linii

- *tac plik1 plik2 :* wypisze połączone oba pliki, od ostatniej do pierwszej lini

echo : powtarza na standardowym wyjściu słowa podane w argumencie

- *echo costam wypisz, czy echo "costam wypisz"* – wypisze 'costam wypisz' i zakończy znakiem nowej linii
- *echo -n "costam wypisz" :* po wypisaniu argumentów, nie wypisze znaku nowej linii
- *echo $HOME :* wypisuje zawartość zmiennej środowiskowej HOME

wc : liczy linie, słowa, i znaki w pliku

gdy nie podamy argumentu, czyta ze standardowego wejścia

- *cat plik1 plik2 | wc -l :* policzy wszystkie linie z połączonych plików plik1 plik2
- *wc plik :* wypisze linie słowa i znaki oraz nazwe pliku
- *wc -m :* tylko znaki (lub --chars)
- *wc -l :* tylko linie (lub --lines)
- *wc -w :* tylko słowa (lib --words)

less : wygodne i szybkie przegladanie plikow tekstowych

- *less plik :* wyświetla zawartośc pliku i pozwa przewijac strony (q-wyjscie)

touch : zmienia czas dostępu i modyfikacji pliku, lub jeśli plik nie istnieje - tworzy go.

- touch plik

cp : kopiuje plik

- *cp plik1 plik2 :* stworzy ./plik2 identyczny z plik1
- *cp plik3 ../katalog/jakis/ :* stworzy plik ../katalog/jakis/plik3
- *cp pom.* podkatalog/ :* skopiuje wszystkie pliki zaczynające się na 'pom.' do ./podkatalog/
- *cp plik5 ~/katalog/jakis/pliczek :* stworzy plik ~/katalog/jakis/pliczek

mv : przesuwa plik (tym samym służy również do zmiany nazwy)

- *mv plik1 plik2 :* zmieni nazwę pliku z ./plik1 na plik2
- *mv plik3 ../katalog/jakis/ :* przesunie plik do ../katalog/jakis/plik3
- *mv plik4 podkatalog/ :* przesunie plik ./podkatalog/plik4
- *mv plik5 ~/katalog/jakis/pliczek :* przesunie i zmieni nazwę ~/katalog/jakis/pliczek

rm : kasuje plik

- rm plik -
- *rm -r katalog :* kasuje wszystko w katalogu i wszystkie jego podkatalogi (--recursive)
- *rm -f plik :* nie pyta się czy skasować (--force)

mkdir : tworzy katalog

- mkdir moj_nowy_katalog
- mkdir /home/users/ja/moj_nowy_katalog

rmdir : usuwa pusty katalog

- rmdir katalog

chmod : zmienia prawa dostępu do pliku grupy użytkowników: u - user, g - group, o - others, a - all

prawa dostępu: r - read, w - write, x - execute

- *chmod o+r plik :* udziel innym prawo do czytania pliku
- *chmod a-x plik :* zabierz wszystkim prawo do wykonywania pliku
- *chmod g=rw plik :* ustaw prawa do czytania i pisania dla swojej grupy
- *chmod -R go+w katalog :* ustawia prawa wszystkim plikom w katalogu i jego podkatalogach (--recursive)

locate : wypisuje gdzie ostatnio, na tym komputerze, był widziany plik o podanej nazwie (lub fragmencie nazwy)

locate raport.txt : locate txt

find : przejrzyj katalog w poszukiwaniu danego pliku

- find . -name raport.txt
- find /home/user -name "rap*xt"

> : przekierowanie wyjścia z programu do pliku.

Standardowym wyjściem każdego programu jest ekran (konsola tekstowa) a standardowym wejściem jest klawiatura. Można te wejścia i wyjścia przekierowywać dowolnie.

- *echo "ala ma kota" > plik.txt :* wyjście z programu echo wpisze do pliku plik.txt
- *ls -l > lista.dat :* wypisze liste plików do pliku lista.dat

>> : doklejenie wyjścia z programu na koniec pliku

- *echo "ala ma psa" >> plik.txt :* dopisze "ala ma psa" na koniec pliku plik.txt
- *ls -l > lista.dat :* wypisze liste plików do pliku lista.dat

| : przekierowanie wyjścia jednego programu na wejście drugiego

- *cat plik.txt | wc -l :* cat wypisuje plik.txt na wyjście które przekierowujemy na program liczący wiersze.
- *ls -l | lpr :* program drukujący 'lpr' dostanie na wejście liste plików
- *cat plik.txt | tac | grep "coś" | head > cosie.txt :* wypisanie pliku.txt na program 'tac', który odwaca kolejność wierszy, wynik tego przekierwany na 'grep', który wypisze tylko linie zawierające słowo "coś", wynik tego wysłany na program 'head', który pośle dalej tylk opierwsze 10 wierszy na wyjście, które przekierowaliśmy do pliku cosie.txt.

>! : przekierowanie do pliku. Działa podobnie jak >, ale kontynuuje nawet gdy plik już istnienie. Działa w "tcsh".

- *echo "ala ma kota" > plik.txt :* gdy plik.txt istnieje, ta komenda może się nie powieść.
- *echo "ala ma kota" >! plik.txt :* konieczne będzie użycie wykrzyknika >!
- *echo "ala ma kota" >| plik.txt :* to samo tylko w 'bash'

< – przekierowanie pliku jako standardowego wejścia.

- *imie < imoiona.txt :* jeśli program imie przyjmuje imiona na standardowe wejście, możemy te imiona spisać do pliku i podać programowi w ten sposób
- *cat imiona.txt | imie :* to samo można też zrobić tak

<< : podanie na wejście nastepujących później linii.

- *imie << 'imiona'vvvv bbbb nnnn 'imiona'>> :* podanie tzw. dokumentu w miejscu. Wszystkie następujące linie między etykietami 'Imiona' będą podane na standardowe wejście programu 'szachy'.
- *echo "vvvv bbbb nnnn" | imie :* to samo można też zrobić tak
- *echo -e "vvvv\bbbb\nnnn" | imie :* to samo można też zrobić tak

2> : przekierowanie standardowego wyjścia dla błędów do pliku. Oprócz standardowego wyjścia i wejścia, każdy program ma jeszcze standardowe wyjście dla błędów. Je też możemy przekierowywać, na przykład w inne miejsce niż wyjście zwykłe. Działa w "bash", nie w "tcsh".

- *find -name "plik.*" >znalezione.log 2>bledy.log :* pliki znalezione przez 'find' wylądują w znalezione.log, komunikaty o błędach nie zaciemnią nam wyniku i wpiszą sie do innego pliku – bledy.log
- *cp -r dane/ backup/ 2>error.log :* jeśli podczas kopiowania całego katalogu wystąpią błędy, wszystkie komunikaty zostaną wpisane do error.log

- *(ls > plik.log) >& plik.err :* w 'tcsh' nie można przekierować samego wyjścia dla błędów, stąd konieczność takiej konstrukcji.

&> lub >& : przekierowanie obu wyjść do pliku.

- *ls >& plik.log :* standardowe wyjście i standardowe wyjście dla błędów jest przekierowane do plik.log
- *ls >plik.log 2>&1 :* to samo, ale działa tylko w 'bash'. Przekierowanie wyjścia, a potem skopiowanie go na wyjście dla błędów.
- *ls &> plik.log :* to samo co >& ale w notacji bardziej właściwej dla 'bash'.

ps : wypisuje procesy uruchomione na komputerze

- *ps :* wyświetla procesy uruchomione przez użytkownika
- *ps -a :* wyświetl również procesy innych użytkowników
- *ps -l, ps -f, ps -F* : wiecej informacji o procesach (od: long, full, extra full)
- *ps f* : wyswietla drzewo zaleznosci procesow (od: forest)
- *ps –help*

bg : uruchamia na nowo zatrzymane (Ctrl-Z) zadanie, ale w tle, tak jakby zostało ono uruchomione z &

- *bg :* uruchamia ostatnio zatrzymane zadanie
- *bg NUMER :* uruchamia zadanie o danym numerze na liście zatrzymanych zadań (jobs)

fg : uruchamia na nowo zatrzymane (Ctrl-Z) zadanie, na pierwszym planie

- *fg :* uruchamia ostatnio zatrzymane zadanie
- *fg NUMER :* uruchamia zadanie o danym numerze na liście zatrzymanych zadań (jobs)

jobs : wyświetla liste zatrzymanych zadań

kill : zabija podany proces

PID - jest to numer identyfikacyjny procesu (process id), można go odczytać np. używając polecenia ps

- *kill PID* wysyła do procesu o numerze PID sygnał do przerwania procesu
- *kill -KILL PID* zabija proces bez pytania

& : modyfikator który pozwala uruchomić od razu proces w tle.

- *Xcalc &* – uruchamia program xcalc w tle, dzięki temu mamy wolną konsole

top : interaktywne narzędzie do monitorowania procesów

vim : zaawansowany edytor tekstowy w trybie tekstowy

gvim : vim w trybie graficznym

macs : zaawansowany edytor tekstowy w trybie graficznym

uemacs : edytor telstowy w trybie teksowym

mcedit : edytor tekstowy w trybie tekstowym

mcedit jest wbudowanym w Midnight Commandera edytorem.

Posiada m.in. podświetlanie składni.

pine : program do obsługi poczty

Bardzo dobry, szybki i wygodny program do sprawdzania i wysyłania poczty elektronicznej. Jego największą zaletą jest to, że działa w trybie tektowym, wic można uruchomić go na zdalnym terminalu.

ssh : program do zdalnego logowania używając szyfrowanego połączenia

Najważniejszy sieciowy program. Umożliwia logowanie się na dowolny komputer na świecie, przy czym połączenie jest bezpieczne dzięki algorytmom szyfrującym opartym na kluczach RSA.

- *ssh xxxxx :* zaloguje mnie na xxxxx
- *ssh ja@xxxxx :* zaloguje mnie jako użytkownika 'ja' na xxxxx
- *ssh ja@xxxxx komenda :* zaloguje mnie tylko by wykonac na xxxxx komendę

scp : program do kopiowania plików używając szyfrowanego połączenia SSH

rlogin : prosty protokół zdalnego logowania

ping : program diagnostyczny sprawdzający czy istnieje połączenie sieciowe z danym komputerem.

df : wypisuje rozmiary i ilość dostępnego miejsca na zamontowanych dyskach (w kilobajtach i w procentach)

- df
- *df /dev/sda1* ogranicz wyniki tylko do jednej partycji
- *df -h* wyświetl rozmiary w wygodnych dla użytkownika jednostkach (human readable)
- *df -m* rozmiary w megabajtach

du : policz rozmiary katalogów i plików zawartych w podanym katalogu

- *du* rozmiar obecnego katalogu
- *du katalog* policz rozmiar podanego katalogu
- *du -s :* wypisz tylko sumę, a nie rozmiary poszczególnych podkatalogów
- *du -sm :* wypisz tylko sumę dla każdego katalogu i podaj rozmiar w megabajtach
- *du -sm dir* | sort -n :* posortuj wyniki od najmniejszego do najwiekszego z podanych katalogów

which : wypisuje gdzie znajduje się plik z programem o podanej nazwie

- *which ls :* zlokalizuj plik, który zostanie uruchomiony po wywołaniu komendy 'ls'

env :wspisuje aktualne wartości wszystkich zmiennych środowiskowych

alias : ustawia i wypisuje definicje skrótowych komend ('aliasów'), które są obecnie ustawione w środowisku

- *alias :* wypisuje aliasy
- *alias ls 'ls –color=auto'* : definiuje nowy alias o nazwie 'ls' i podanej treści

/dev/null : studnia bez dna. Urządzenie do którego możemy pisać dowoli, a wszystko co wpiszemy przepada.

- *find -name "plik.*" 2>/dev/null :* jeśli w sród wyników szukania nie chcemy oglądać komunikatów o błędach
- *latex plik.tex >/dev/null :* program wykona wszystkie czynności, ale nie zaśmieci nam konsoli logami
- *licz >/dev/null 2>/dev/null & :* jeśli chcemy puścić program w tle, a potem się wylogować (zamknąć konsole), musimy przekierować wyjścia z programu tak, aby nie próbował pisać do nieistniejącego już urządzenia. Urządzeninie 'null' jest zawsze.

/dev/zero : składnica zer. Jest to urządzenie do czytania, które nigdy sie nie kończy. Można z niego wczytać dowolną liczbe bajtów, a wszystkie będą równe zero.

- *dd if=/dev/zero of=zera.txt count=1000 :* wczyta tysiąc zer do pliku 'zera.txt'.
- *cat /dev/zero :* radzę nie próbować
- *head -c 10 /dev/zero > zera.txt :* wypisze pierwsze 10 bajtów z '/dev/zero' do pliku 'zera.txt'. W wyniku mamy plik z dziesięcioma zerami.

/dev/random : zbiór liczb losowych. Jest to urządzenie do czytania, które daje prawdziwie losowe dane. Korzysta przy tym z systemowego zbiornika entropii, który jest uzupełniany dzieki różnym przejawom aktywności użytkownika. Zbiór ten może się wyczerpać, dlatego nie należy z niego czytać wielu liczb na raz.

- *od -t x1 -N 100 /dev/random :* wypisze pierwsze 100 losowych bajtów z urządzenia /dev/random na ekran (w systemie szesnastkowym)

/dev/urandom : zbiór liczba pseudolosowych. Jest to urządzenie do czytania, które podaje liczby pseudolosowe. Ma ich do dyspozycji dowolnie dużo.

- *od -t d1 -N 100 /dev/urandom* : wypisze pierwsze 100 bajtów z urządzenia /dev/urandom na ekran (w systemie dziesiętnym)

/dev/stdin : standardowe wejście aktualnego procesu. Kazdy proces który próbuje czytać z tego urządzenia, dostanie zawoartość własnego wejścia.

- *echo "ma kota" | cat ala.txt /dev/stdin :* program cat połączy zawartość pliku ala.txt z tym co dostał na standardowe wejście

/dev/stdout : standardowe wyjście aktualnego programu. Gdy proces wypisze coś do pliku /dev/stdout, pojawi się to na jego standardowym wyjściu.

- *a2ps --output plik.ps plik.txt :* program a2ps stworzy dokument postscript w pliku plik.ps

- *a2ps --output /dev/stdout plik.txt :* program a2ps wypisze dokument na ekran (jego standardowe wyjście)

/dev/stderr : standardowe wyjście dla błędów aktualnego programu. Gdy proces wypisze coś do pliku /dev/stderr, pojawi się to na jego standardowym wyjściu dla błędów.