2.2 Polecenia i cechy powłoki bash

Współczesne programy muszą być graficzne, posiadać kolorowe przyciski, funkcje „przeciągnij i opuść” (drag&drop), być łatwe i przyjemne.

Powłoka **bash** wydaje się, na pierwszy rzut oka, nieprzyjazna i nieelegancka w porównaniu chociażby z interfejsem systemu operacyjnego Microsoft Windows. Pisanie na klawiaturze wydaje się trudniejsze, mniej wygodne niż klikanie myszką.

> Czy kiedykolwiek próbowałeś/aś zmienić uprawnienia dla 120 plików w Microsoft Windows?

Bezdyskusyjnie, **bash** dostarcza funkcjonalność upraszczająca pracę administracyjne w systemie.

W tym rozdziale zajmiemy się najciekawszymi możliwościami powłoki **bash**.

2.2.1Uzupełnianie wpisywanych poleceń i nazw plików

Powłoka bash ma funkcję uzupełniania (kompletacji) wprowadzanych w wierszu poleceń nazw plików oraz nazw poleceń. Po prostu wprowadź pierwsze znaki i naciśnij klawisz **Tab** (-->|). bash uzupełni nazwę polecenia lub pliku.

Przykładowo, w katalogu domowym, za znakiem zachęty naciśnij kolejno klawisze...

- M, K, D, Tab, pojawi się → mkdir
- C, D, spacja, /, H, Tab → cd /home
- V, I, spacja, ., B, A, Tab → vi .bash → _, Tab → vi .bash_history

Jeżeli jest więcej niż jedna możliwość, po powtórnym naciśnięciu klawisza **Tab** - bash pokaże wszystkie możliwości.

Przykładowo wprowadzamy kolejno: M, K, Tab, Tab → pojawi się

mkallcomposecac	mkfontdir	mkmanifest
hes	mkfontscale	mknod
mkcfm	mkhtmlindex	mkpasswd
mk_cmds	mkhybrid	mktemp
mkcomposecache	mkinfodir	mkzftree
mkdir	mkinodedb2	mkzimage_cmdline
mkdirhier	mkisofs	
mkfifo		

Ta cecha ułatwia wprowadzanie długich nazw pliku (i ścieżek dostępu).

2.2.2 Stosowanie historii poleceń w bash

bash zapamiętuje wszystkie wprowadzane polecenia. Domyślnie są one zapisywane w ukrytym pliku **.bash_history** w katalogu domowym użytkownika. Maksymalnie plik ten przechowuje 1000 wierszy poleceń.

By wyświetlić zawartość pliku, należy użyć polecenia **history.**

Przykład:

```
geeko@da51:~> history
    1  ls -i
    2  ls -li
    3  history
geeko@da51:~>
```

By wyświetlić polecenia przechowywane w pamięci podręcznej (*cache*), należy użyć klawiszy strzałek. Wyświetlane jest jedno polecenie. Naciśnięcie klawisza ↑ wyświetli poprzednie polecenie, naciśnięcie klawisza ↓ pokaże następne polecenie z historii poleceń. Po znalezieniu poszukiwanego polecenia – można zmienić jego zawartość (edycja) i wykonać naciskając klawisz **Enter.**

Przy przeglądaniu zapisów historii – można wybrać określone polecenia (filtrowanie poleceń z historii). Wprowadzenie jednej lub więcej liter lub naciśnięcie klawisza **PageUp** lub **PageDown** spowoduje wyświetlenie poprzedniego lub następnego polecenia zaczynającego się od tych liter z pamięci podręcznej historii poleceń.

Jeżeli wprowadzimy część polecenia (niekoniecznie początkową), a następnie naciśniemy kombinację klawiszy Ctrl i R, to zostaną wyszukane i wyświetlone polecenia zawierające podany ciąg znaków. Proces wyszukiwania zaczyna się od ostatniego polecenia w historii poleceń.

Ćwiczenie. Korzystanie z historii poleceń w bash

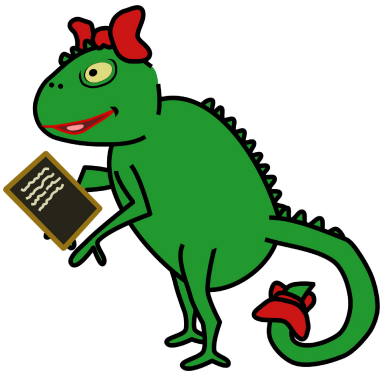
Podaj ostatnie polecenia z historii zawierające następujące słowa:

test _____

mkdir _____

mv _____

cp _____



2.2.3Używanie zmiennych bash

Zmienne środowiskowe oraz zmienne powłoki **bash** umożliwiają jego konfigurację oraz dostosowanie.

Zmienne środowiskowe kontrolują zachowanie programu uruchamianego z powłoki. Zmienne powłoki kontrolują zachowanie samej powłoki.

Zmienne powinno się pisać dużymi literami (np. PATH). Jeżeli definiujemy własne zmienne, dobrze również przestrzegać tej zasady.

W skryptach powłoki używa się nazw zmiennych pisanych małymi literami.

Najważniejsze zmienne środowiskowe, to:

PATH – przy startowaniu każdego programu, system szuka go kolejno w katalogach podanych tutaj (katalogi rozdzielone są znakiem „:”). Ważna jest więc kolejność (porządek) przechowywanych ścieżek.

HOME – katalog domowy użytkownika.

USER – nazwa logowania użytkownika.

By wyświetlić wartość zmiennej powłoki lub środowiska, napisz

```
echo $zmienna
```

Przykład:

```
geeko@da51:~ > echo $HOME
/home/geeko
geeko@da51:~ >
```

Dla jasności będziemy również w tekście podręcznika pisać znak $ przed nazwą zmiennej

By ustawić wartość zmiennej lub zdefiniować nową zmienna, należy w wierszu poleceń napisać:

zmienna=wartość

Przykład:

```
geeko@da51:~ > MYVAR=myvalue
geeko@da51:~ > echo $MYVAR
myvalue
geeko@da51:~ >
```

Wartość może być liczbą, znakiem lub łańcuchem (ciągiem). Jeżeli ciąg znaków zawiera spację, należy podać go w cudzysłowie, jak w przykładzie:

```
geeko@da51:~ > MYVAR="my value"
geeko@da51:~ > echo $MYVAR
my value
geeko@da51:~ >
```

Polecenie echo można też użyć do wyświetlenia na ekranie określonego tekstu (komunikatu):

```
geeko@da51:~> echo Hello Geeko
Hallo Geeko
geeko@da51:~>
```

Ćwiczenie. Używanie zmiennych bash - część I

Zapisz wartość zmiennej $PATH na Twoim komputerze:

_____

_____

_____

_____

Jeżeli chcemy na ekranie wyświetlić coś takiego:

Zmienna $HOME jest ustawiona na zawartość*_zmiennej_$HOME*

podana powyżej metoda nie zadziała poprawnie. Otrzymamy:

```
geeko@da51:~> echo Variable $HOME is set to $HOME
Variable /home/geeko is set to /home/geeko
```

```
geeko@da51:~>
```

Wstawienie tekstu w cudzysłów – również nie zadziała:

```
echo "Variable $HOME is set to $HOME"
Variable /home/geeko is set to /home/geeko
geeko@da51:~>
```

Pojedynczy cudzysłów da natomiast następujący rezultat:

```
geeko@da51:~> echo 'Variable $HOME is set to $HOME'
Variable $HOME is set to $HOME
geeko@da51:~>
```

Jest jednak parę dobrych rozwiązań tego problemu.

Przykłady:

- kombinacja cudzysłowów

```
geeko@da51:~> echo 'Variable $HOME is set to' "$HOME"
Variable $HOME is set to /home/geeko
geeko@da51:~>
```

- kombinacja mieszana znaków cudzysłowia i nie

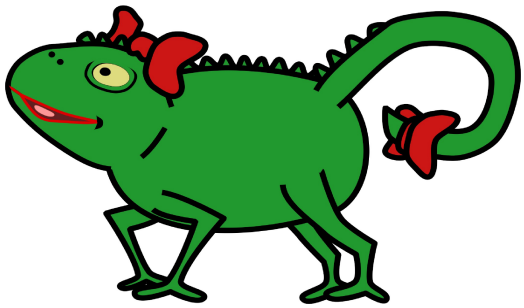
```
geeko@da51:~> echo 'Variable $HOME is set to' "$HOME"
Variable $HOME is set to /home/geeko
geeko@da51:~>
```

- maskowanie znaku $ ukośnikiem \

```
geeko@da51:~> echo "Variable \$HOME is set to $HOME"
Variable $HOME is set to /home/geeko
geeko@da51:~>
```

Ćwiczenie. Używanie zmiennych bash - część II

Przećwicz wszystkie przykłady z tego rozdziału używając zmiennej $PATH zamiast zmiennej $HOME.



2.2.4 Używanie aliasów w bash

Definiowanie aliasów pozwala na tworzenie skrótów klawiszowych dla poleceń oraz ich opcji lub na tworzenie poleceń o całkowicie innych nazwach.

W SUSE Linux Enterprise Desktop 10 używając poleceń dir, md, ls – używamy aliasów.

Polecenie **alias** pokazuje zdefiniowane w systemie aliasy.

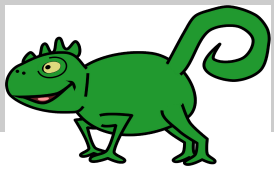
Przykładowy wynik polecenia alias:

```
alias +='pushd .'
alias -='popd'
alias ..='cd ..'
alias ...='cd ../..'
alias beep='echo -en "\007"'
alias dir='ls -l'
alias l='ls -alF'
alias la='ls -la'
alias ll='ls -l'
alias ls='/bin/ls $LS_OPTIONS'
alias ls-l='ls -l'
alias md='mkdir -p'
alias o='less'
alias rd='rmdir'
alias rehash='hash -r'
alias unmount='echo "Error: Try the command: umount" 1>&2; false'
alias which='type -p'
alias you='su - -c "/sbin/yast2 online_update"'
```

Widzimy, że **dir** jest aliasem polecenia **ls -l** , a **md** to alias polecenia **mkdir -p**.

Możemy zapytać o znaczenie pojedynczego aliasu:

```
geeko@da51:~> alias md
alias md='mkdir -p'
geeko@da51:~> alias dir
```



```
alias dir='ls -l'
```

Celem sprawdzenia, czy dane polecenie jest aliasem, należy użyć polecenia **type**.

type powie nam, czy dane polecenie jest wbudowanym poleceniem powłoki, zwykłym poleceniem, funkcją czy aliasem.

Dla zwykłych poleceń, **type** wyświetli ścieżkę dostępu do programu.

Dla aliasów – wyświetli element, którego nazwa jest aliasem.

```
geeko@da51:~> type -a ls
ls is aliased to `/bin/ls $LS_OPTIONS'
ls is /bin/ls
```

Powyższy przykład pokazuje, że ls jest aliasem użytym dla dodania pewnych opcji do polecenia. Opcja **-a** pozwala na wyświetlenie zarówno zawartości aliasu jak i ścieżki do oryginalnego polecenia ls. ls używa opcji przechowywanych w zmiennej $LS_OPTIONS. Te zmienne są odpowiedzialne za wyświetlanie przez ls różnych typów plików różnymi kolorami.

Wynik działania zmodyfikowanego i zastąpionego aliasem polecenia ls:

```
malgosia@mp:~> ls
3074_postfix_ilias_Lucy  maius
AYA                      Malgosi_witryna
bin                      Matcomp
c.jpg                    My Music
C.jpg                    nauczyciele-unet-oeizik
cni_materials            nowe_rozne_kursy
collectNWData.sh         obrazek.png
```

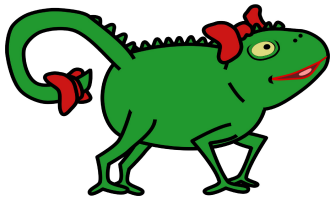
Większość systemowych aliasów jest zdefiniowanych w pliku **/etc/bash.bashrc**.

Do definiowania aliasów również użyjemy polecenia alias.

Składnia polecenia: alias *nazwa_aliasu=”polecenie z opcjami”*

Do usunięcia z kolei – użyjemy polecenia unalias.

Składnia polecenia: unalias *nazwa_aliasu*



Aliasy zdefiniowane w ten sposób działają wyłącznie w bieżącej powłoce, jak

w poniższym przykładzie:

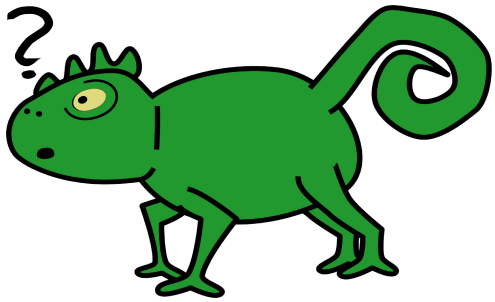
```
geeko@da51:~> alias ls="echo Hello"
geeko@da51:~> ls
Hello
geeko@da51:~> bash
geeko@da51:~> ls
bin  Desktop  Documents  public_html
geeko@da51:~>
```

Po zakończeniu sesji pracy z powłoką, aliasy te znikają. By zachować je na stałe, należy zapisać je w jednym z plików konfiguracyjnych powłoki. W SUSE Linux Enterprise Desktop, plik **~/.alias** jest tworzony dla aliasów definiowanych przez każdego z użytkowników.

Plik ten jest wczytywany przez **~/.bashrc**, gdzie znajduje się odpowiednie do tego polecenie. Aliasy nie zastąpią skryptów, ale mogą znacznie zaoszczędzić czasu przy pracy w linii poleceń powłoki.

Ćwiczenie. Używanie aliasów w bash

Utwórz nowy alias **spy** wyświetlający zawartość archiwum tar spakowanego przez gzip.



2.2.5 Używanie wzorców i zmiennych globalnych przy wyszukiwaniu

Czasami chcemy wykonywać operację na grupach plików lub katalogów, bez wymieniania wszystkich po kolei. W takich przypadkach można skorzystać ze wzorców wyszukiwania:

- zmienna globalna „?” - zastępuje pojedynczy znak (z wyjątkiem ukośnika „/”)

```
geeko@da51:~> ls /bin/s?
/bin/sh  /bin/su
```

```
geeko@da51:~>
```

- zmienna globalna „*” zastępuje dowolnej długości ciąg znaków (pusty też) za wyjątkiem kropki „.” na początku nazwy pliku lub ukośnika „/”:

```
geeko@da51:~> ls /bin/s*
/bin/sash     /bin/setkeycodes  /bin/sh              /bin/sort
/bin/scsidev  /bin/setleds      /bin/showconsolefont  /bin/stty
/bin/sed      /bin/setmetamode  /bin/showkey         /bin/su
/bin/setfont  /bin/setserial    /bin/sleep           /bin/sync
geeko@da51:~>
```

- [0-9] zastępuje dowolny ze znaków podanych wewnątrz nawiasu kwadratowego (tutaj: cyfry od 0 do 9).

```
geeko@da51:~> ls /bin/s[e-h]*
/bin/sed              /bin/setleds          /bin/sh
/bin/setfont          /bin/setmetamode      /bin/showconsolefont
/bin/setkeycodes      /bin/setserial        /bin/showkey
geeko@da51:~>
```

- [abcd] zastępuje jeden ze znaków a, b, c, d.

```
geeko@da51:~> ls /bin/s[abcd]*
/bin/sash  /bin/scsidev
```

- [a-ew-z] zastępuje dowolny znak z zakresów a-e i w-z

```
geeko@da51:~> ls /bin/s[a-cm-t]*
/bin/sash  /bin/scsidev  /bin/sort  /bin/stty
geeko@da51:~>
```

- [!abc] zastępuje dowolny znak z wyjątkiem a,b oraz c.

```
geeko@da51:~> ls /bin/s[!eh]*
/bin/sash  /bin/scsidev  /bin/sleep  /bin/sort  /bin/stty  /bin/su  /bin/sync
geeko@da51:~>
```

Uwaga. Znaczenie niektórych wzorców wyszukiwania różni się od znaczenia tych wzorców jako wyrażeń regularnych (→ w dalszej części podręcznika)

Jeżeli używamy wzorców wyszukiwania (zmiennych globalnych), powłoka porównuje je ze wszystkimi nazwami plików, a następnie zastępuje wyrażenie wszystkimi znalezionymi nazwami.

Uwaga. Porządek znaków we wzorcu jest inny dla zwykłych użytkowników i użytkownika root, ponieważ domyślnie używają oni innego kodowania znaków.

Dla zwykłych użytkowników (kodowanie UTF-8) znak małej litery jest tuż za znakiem dużej litery (między A i C będzie aBa → AaBaC). Dla użytkownika root (kodowanie POSIX) wszystkie małe litery są za dużymi (między A i C będzie tylko B → ABC).

Zmienna $LANG odpowiada za język. W poniższym przykładzie język jest ustawiony na US English z kodowaniem UTF-8:

```
geeko@da51:~> echo $LANG
en_US.UTF-8
geeko@da51:~>
```

A tutaj mamy język polski, również z kodowaniem UTF-8:

```
malgosia@mp:~> echo $LANG
pl_PL.UTF-8
malgosia@mp:~>
```

Po przełączeniu się na konto roota, widzimy, że kodowanie jest inne:

```
malgosia@mp:~> su -
Password:
mp:~ # echo $LANG
POSIX
mp:~ #
```

Uwaga. UTF-8 (Unicode) obsługuje wszystkie rodzaje liter, nie tylko łacińskie, podczas gdy POSIX (ASCII) ma tylko litery łacińskie

Ćwiczenie. Używanie wzorców wyszukiwania

Ile programów w katalogu /bin ma nazwę...

... która zaczyna się od liter „sa”? _____

... która zaczyna się od litery „x”, „y” lub „z” ? _____

... która kończy się literą „s”? _____

... trzyliterową? _____

... która ma w nazwie przynajmniej jedną cyfrę?_____