LABO 03

Przekierowanie wejścia wyjścia:

• program ma trzy podstawowe strumienie wejścia-wyjścia

- standardowe wejście
- standardowe wyjście
- standardowe wyjście diagnostyczne

• przekierowanie standardowego wyjścia

ls -1>katalog.txt – zapisze listę plików katalogu do pliku

wc -l katalog.txt – wypisze liczbę linii w pliku oraz jego nazwę na terminal

wc -l katalog.txt >>katalog.txt - dopisze powyższą informacje do pliku

• przekierowanie standardowego wejścia

ls -1>katalog.txt – zapisze listę plików katalogu do pliku

wc -l <katalog.txt – wypisze liczbę linii w pliku na terminal

wc -l <katalog.txt >>katalog.txt - możemy samą liczbę plików dopisać do pliku

• przekierowanie standardowego wyjścia diagnostycznego

find / -name "sk*" -type f >pliki.txt 2>bledy.txt

• przekierowanie standardowego wyjścia i wyjścia diagnostycznego

find . -name ala.txt >wszystko.txt 2>&1

• przetwarzanie potokowe

cat >imiona.txt

ala
ola
ala
ola
ela
hela
ola
ala
ela
stefan
olek
roman

^D

sort <imiona.txt >posortowane.txt

more posortowane.txt

uniq <posortowane.txt >rozne.txt

more rozne.txt

wc -l <rozne.txt

LUB sort imiona.txt | uniq|wc -l LUB

sort <imiona.txt | uniq|wc -l LUB cat imiona.txt|sort|uniq|wc -l LUB ...

++

Polecenie echo i znaki cytowania

Do wyświetlenia tekstu lub wartości zmiennych na ekranie służy polecenie echo.

Polecenie to może być stosowane z kilkoma opcjami :

-n - nie wyświetlaj nic i przejdź do następnej linii

-e - włącz interpretację następujących komend występujących po \ :

\a - dzwonek

\b - kasuj ostatni znak

\c - przechodź do następnej linii

\n - nowa linia

\t - tabulator

\\ - znak \

Powłoka Bash odróżnia i inaczej interpretuje trzy rodzaje znaków cytowania :

- cudzysłów " " - umożliwia wyświetlenie tekstu, zmiennej ($), zacytowanie polecenia (``) lub użycie znaków specjalnych występujących po \
- apostrof ' ' - wszystko to co zawarte jest między apostrofami interpretowane jest jako zwykły tekst
- odwrotny apostrof ` ` - przydaje się w sytuacji gdy chcemy podstawić lub zacytować jakieś polecenie

---

Zmienne środowiskowe *(ang.environment variables)* można przyrównać do aliasów dla podstawowych lokalizacji w systemie takich jak dyski, ścieżki, foldery czy pliki. Kontrolują zachowanie różnych programów np. zmienna TEMP określa lokalizację, w której programy umieszczają pliki tymczasowe.

Niektóre przykłady zmiennych środowiskowych:

```
$HOME          #ścieżka do twojego katalogu domowego
$USER          #twój login
$HOSTNAME      #nazwa twojego hosta
$OSTYPE        #rodzaj systemu operacyjnego
```

SET

Aby wyświetlić listę zmiennych istniejących w systemie wystarczy także w oknie wiersza poleceń wpisać komendę SET

Ważniejsze zmienne to :

BASH=/bin/bash - nazwa naszej powłoki
BASH_VERSION=1.14.7(1) - wersja naszej powłoki
COLUMNS=80 - liczba kolumn znaków na naszym ekranie
LINES=25 - liczba linii na ekranie
HOME=/home/student - nasz katalog domowy
LOGNAME=student - nasz login
OSTYPE=Linux - typ systemu
PATH=/usr/bin:/sbin:/bin:/usr/sbin - domyślna ścieżka dostępu
PS1=[\u@\h \W]\$ - ustawienie znaku zachęty
PWD=/home/student - nasz aktualny katalog roboczy
SHELL=/bin/bash - nazwa naszej powłoki
USERNAME=student - nazwa użytkownika, który jest aktualnie zalogowany w systemie
EDITOR=vim - nazwa domyślnego edytora tekstu
HISTFILE=/home/student/.bash_history - plik zawierający historię poleceń
MAIL=/var/spool/mail/$USER - określa plik, do którego dopisywane są listy przychodzące na nasze konto

Zmienna przechowuje jakąś wartość. Odwołujemy się do niej (odczytujemy ją) pisząc znak $ bezpośrednio przed nazwą zmiennej.

Zmienne można je podzielić na:

globalne - widoczne w każdym podshellu

lokalne - widoczne tylko dla tego shella w którym został ustawione

Aby bardziej uzmysłowić sobie różnicę między nimi zrób mały eksperyment: otwórz xterma (widoczny podshell) i wpisz:

x="napis"

echo $x

xterm

x="napis" zdefiniowałeś właśnie zmienną x, która ma wartość "napis"

echo $x wyświetli wartość zmiennej x

xterm wywołanie podshella

wpisz więc jeszcze raz:

echo $x nie pokaże nic, bo zmienne lokalne nie są widoczne w podshellach

Możesz teraz zainicjować zmienną globalną:

export x="napis"

Teraz zmienna x będzie widoczna w podshellach, jak widać wyżej służy do tego polecenie export, nadaje ono wskazanym zmiennym atrybut zmiennych globalnych. Jeśli napiszesz samo export, opcjonalnie export -p uzyskasz listę aktualnie eksportowanych zmiennych. Na tej liście przed nazwą każdej zmiennej znajduje się zapis:

declare-x

To wewnętrzne polecenie BASH-a, służące do definiowania zmiennych i nadawania im atrybutów, -x to atrybut eksportu czyli jest to, to samo co polecenie export. Ale tu uwaga! Polecenie declare występuje tylko w BASH-u, nie ma go w innych powłokach, natomiast export występuje w ksh, ash i innych, które korzystają z plików startowych /etc/profile. Dlatego też zaleca się stosowanie polecenia export.

export -n zmienna

spowoduje usunięcie atrybutu eksportu dla danej zmiennej

dostępne zmienne środowiskowe można wyświetlić za pomocą polecenia:

printenv | more

env

set

usuwamy

unset

Wyrażenia regularne

Znaczenia meta-znaków :

* - oznacza dowolną sekwencję, dowolną ilość znaków

? - oznacza dowolny pojedynczy znak

[qwA1] - oznacza dowolny pojedynczy znak, wymieniony w nawiasach

[a-z] - oznacza dowolną małą literę

[A-G] - oznacza dowolną dużą literę od A do G

[a-zA-Z0-9] - oznacza dowolną małą lub dużą literę i dowolną cyfrę

Meta-znaki najczęściej wykorzystywane są do listowania zawartości katalogów :

ls *

wyświetla wszystkie pliki w katalogu z wyjątkiem zaczynających się od '.' kropki (są to pliki ukryte)

ls *.*

wyświetla wszystkie pliki zawierające w nazwie '.'

ls [A-B]*

wyświetla wszystkie pliki zaczynające się od A lub B

ls ??1

wyświetla pliki, których nazwa składa się z dwóch dowolnych liter i jedynki na końcu.

Wyrażenia regularne można podzielić na:

Podstawowe, których możemy używać np. za pomocą komendy grep:

^ lub \A - początek ciągu znaków

$ lub \Z - koniec ciągu znaków

. - każdy znak oprócz znaku nowej linii

(a|b) - a lub b

(...) - grupa znaków

[abc] - zakres a lub b lub c

[^abc] - zakres nie a lub b lub c

[a-q] - zakres małe a do małe q

[A-Q] - zakres wielkie A do wielkie Q

[0-7] - liczby 0 do 7

\< - początek słowa

\> - koniec słowa

Typu POSIX, których również możemy używać np. za pomocą komendy grep:

[:upper:] - wielka litera

[:lower:] - mała litera

[:alpha:] - litery

[:alnum:] - litery i cyfry

[:digit:] - cyfry

[:punct:] - znak interpunkcyjny

[:blank:] - spacja lub tabulator

[:space:] - spacja

[:word:] - litery, cyfry i znaki podkreślenia

Rozszerzone, których możemy używać np. za pomocą komendy “grep -E” lub egrep:

* - zero lub więcej

+ - jeden lub więcej

? - zero lub jeden

{3} - trzy

{3,} - trzy lub więcej

{3,5} - trzy, cztery lub pięć

Typu PERL, których możemy używać np. za pomocą komendy “grep -P” **jeżeli jest wkompilowana obsługa Perla lub programując w Perlu** itp.:

\s - spacja

\S - nie spacja

\d - cyfra

\D - nie cyfra

\w - słowo

\W - nie słowo

\n - znak powrotu do początku linii

\r - znak nowej linii

\t - tabulator

\v - tabulator pionowy

\b - granica słowa

\B - nie granica słowa

Tryb “ungreedy” – w wyrażeniach typu PERL, jeżeli dodamy po znakach * + ? znak ? – oznaczać to będzie tryb “ungreedy” – minimalny – czyli najmniejsze pasujące wyrażenie będzie brane pod uwagę przy dopasowywaniu.

Znaki specjalne – ^ $ ( ) < . * + ? [{ \ | > – są to znaki które są interpretowane i aby ich używać jako zwykłe znaki w wyrażeniach regularnych należy je poprzedzać znakiem \ .

Poniżej kilka przydanych przykładów jak wykorzystać wyrażenia regularne:

Aby przetestować poprawność wyrażenia regularnego możemy przekierować wyjście z jakiegoś programu np echo, do programu któryy obsługuje wyrażenia regularne, np. grep. Przydatna będzie tu opcja “-o” programu grep która sprawi że program wyświetli tylko to co pasuje do wyrażenia regularnego. Na przykład by wyszukać czy wyrażenie pokaże trzy pierwsze kolumny z adresu IP:

```
echo 123.456.789.1|grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.'
```

lub:

```
echo 123.456.789.1|grep -Eo '[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.'
```

Aby wyszukać w katalogu nazwy w których jest 6 cyfr:

```
ls /katalog |grep -E '[[:digit:]]{6}'
```

Aby wyszukać wewnątrz pliku plik.txt linie które mają wielką literę lub małe a lub małe b (opcja -r oznacza szukanie wewnątrz):

```
grep -r '[[:upper:]ab]' plik.txt
```

Aby wyszukać wewnątrz pliku plik.txt adresy email:

```
egrep -r '([a-zA-Z0-9\._\-]+@[a-zA-Z_]+\.[a-zA-Z]{2,6})' plik.txt
```