Systemy operacyjne

wykład 3 - Praca z interpreterem poleceń c.d

dr Marcin Ziółkowski

Instytut Matematyki i Informatyki
Akademia im. Jana Długosza w Częstochowie

31 marca 2016 r.

Przekierowywanie wejścia

W interpreterze poleceń standardowym wejściem oraz wyjściem jest konsola (terminal). Na poprzednim wykładzie poznaliśmy sposoby przekierowania wyniku na niestandardowe wyjście - na przykład zapis do pliku. Istnieje też możliwość, aby program pobierał dane nie z konsoli, ale z pliku. W tym przypadku wykorzystujemy znak $<$. Oto przykład skryptu, który zapisuje dane do pliku, sortuje plik, pokazuje dane związane z zawartością pliku oraz zapisuje wynik do nowego pliku.

```
echo "Say" >> 1.txt
echo "Hello" >> 1.txt
echo "Linuxers!" >> 1.txt
sort < 1.txt >> 2.txt
wc < 2.txt
```

Przekierowywanie standardowego wyjścia diagnostycznego

W pewnych sytuacjach jest wygodnie zbierać informacje o błędach w osobnych plikach. Służy do tego fraza 2 >, która przekierowuje standardowe wyjście diagnostyczne. Oto przykład skryptu:

```
mkdir katalog1
cd katalog1
touch 1.txt
cd ..
rmdir katalog1 2> blad.txt
```

Praca w tle i przetwarzanie potokowe

Jeżeli chcemy, aby pewne operacje w terminalu odbywały się w tle, tak aby nadal móc pracować w tej samej konsoli należy polecenie zakończyć symbolem &.
Często istnieje też potrzeba przetwarzania danych przez kilka programów. Można wtedy użyć tzw. przetwarzania potokowego. Wtedy standardowe wyjście jednego programu możemy przekazać na standardowe wejście drugiego programu. Oto przykład skryptu:

```
mkdir katalog1
cd katalog1
touch 1.txt
echo "Hello" > 1.txt
echo "Boys!" >> 1.txt
echo "Boys!" >> 1.txt
sort 1.txt >> 2.txt
uniq < 2.txt > 3.txt
wc < 3.txt
```

Praca w tle i przetwarzanie potokowe

Inaczej i krócej:

```
mkdir katalog1
cd katalog1
touch 1.txt
echo "Hello" > 1.txt
echo "Boys!" >> 1.txt
echo "Boys!" >> 1.txt
sort 1.txt| uniq | wc
```

O zastosowaniu znaków cytowania

Szczególne znaczenie w BASHU mają pewne symbole specjalne tzw. znaki cytowania. Pierwszym z takich symboli jest odwrócony apostrof - ‘ ‘. Używamy go w przypadku, gdy chcemy aby wynik instrukcji stał się częścią składową nowego polecenia. W ten sposób możemy polecenia zagnieżdżać. Oto prosty skrypt:

```
‘echo da‘‘echo te‘
```

I nieco ciekawszy:

```
echo "date" > 1.txt
echo "rm" > 2.txt
‘cat 1.txt‘
‘cat 2.txt‘ 1.txt
```

O zastosowaniu znaków cytowania

Drugim ze znaków cytujących jest cudzysłów " ". Pozwala on na umieszczanie tekstu oraz wartości zmiennych zawierających spacje. Cudzysłowy zachowują znaczenie specjalne trzech znaków:

`$, /, ‘ .`

Oto przykład dwóch skryptów:

skrypt 1

```
cd Pulpit
mkdir ala ma kota
```

Działanie tego skryptu polega na utworzeniu trzech katalogów : **ala, ma** oraz **kota**.

skrypt 2

```
cd Pulpit
mkdir "ala ma kota"
```

Ten skrypt tworzy jeden katalog o nazwie **ala ma kota**.

O zastosowaniu znaków cytowania

Trzecim ze znaków cytujących jest apostrof ' '. Pozwala on na umieszczanie tekstu oraz wartości zmiennych zawierających spacje, nie zachowuje jednak znaczenia specjalnego żadnych znaków. Używając apostrofu można tworzyć np. katalogi o dowolnych nazwach - nawet zastrzeżonych, co pokazuje kolejny przykład skryptu.

```
cd Pulpit
mkdir '$HOME'
```

Skrypt ten tworzy katalog o nazwie $*HOME* na pulpicie.

Zmienne i operatory arytmetyczne

W BASHU, jak w każdym języku programowania, występuje możliwość używania zmiennych oraz wykonywania operacji arytmetycznych. Wywołanie zmiennej *x* uzyskujemy poprzez poprzedzenie jej znakiem $. Jeśli chcemy ją tylko wyświetlić musimy wykorzystać polecenie **echo**. Dodatkowo możemy wykonywać na zmiennych oraz na stałych operacje arytmetyczne, jednak liczby te mogą być tylko całkowite. Oto pierwszy z przykładów:

```
x=5
y=6
echo "$x + $y = $[x+y]"
```

Zmienne i operatory arytmetyczne

Przykład pokazujący wykorzystanie operacji arytmetycznych (na stałych):

```
echo "$[3] + $[5] = $[3+5]"
echo "$[12] - $[3] = $[12-3]"
echo "$[8] * $[7] = $[8*7]"
echo "$[45] : $[5] = $[45/5]"
```

Zmienne i operatory arytmetyczne

Oczywiście zmiennymi mogą być też polecenia. Oto kolejny przykład:

```
x=‘mkdir NOWY‘
y=‘touch 1.txt‘
z=‘mv 1.txt /home/marionesta/Pulpit/NOWY‘
$x
$y
$z
```