Skrypty basha

Skrypty basha pozwala nam wykonać serię akcji (poleceń) terminala, nie wymagając od nas równocześnie wchodzenia w ogóle w terminal. Podstawowa zasada jest taka: wszystko, co możemy odpalić z terminala, możemy również wykonać przy użyciu skryptu. Odwrotnie też działa: wszystko co możemy zrobić w skrypcie, zadziała również w terminalu.

Konfiguracja

Zasadniczo, zwykle nie ma nic do konfigurowania. Pierwsza linia skryptu powinna zawierać wskazanie dla systemu, którego interpretera używać, aczkolwiek większość powłok domyślnie założy, ze używamy jej i wszystko wykona prawidłowo. Jednak na wszelki wypadek, warto to sobie zawsze skonfigurować.

```
which bash
```

Powie nam, jaka jest lokalizacja naszego interpretera. Na 99% odpowiedzią będzie /bin/bash. Używanego interpretera oznaczamy w pierwszej linii skryptu w następujący sposób:

```
#!/bin/bash
```

Nasz pierwszy skrypt

Stwórz pusty plik o dowolnej nazwie. Na pewno wszyscy pamiętają, że Linux działa bez rozszerzeń, ale jak poznać, że plik `jajco` jest plikiem tekstowym, a `asdfasdf123` skryptem? No nie da rady, dlatego zwykle skryptom dajemy rozszerzenie `.sh`

```
geany skrypt1.sh
```

> Teraz przydatna rzecz: jak po poleceniu, chociażby takim jak geany, damy znak &, to program wystartuje nam w tle, nie blokując nam terminala. Polecam.

main.cpp | jajco.txt | skrypt1.sh

```
1  #!/bin/bash
2
3  echo poniżej pojawią się wszystkie pliki z tego katalogu
4  ls
5  echo told ya\'
6
```

taki skrypt odpalimy poleceniem

```
./skrypt1.sh
```

I wynik działania tego skryptu, do przewidzenia generalnie.

```
michal@michal-VirtualBox:~/skrypty$ ./skrypt1.sh
poniżej pojawią się wszystkie pliki z tego katalogu:
jajco.txt
najlepsze_przepisy_na_lasagne.doc
najlepsze_teksty_do_masakrowania_lewactwa.txt
skrypt1.sh
told ya'
michal@michal-VirtualBox:~/skrypty$
```

Jeżeli skrypt się nie odpali, to należy mu ustawić odpowiednie uprawnienia. Oczywiście, zależy, czego oczekujemy od skrypt i kto ma mieć do niego dostęp, ale na potrzeby dzisiejszych zajęć wystarczy nam

```
chmod 755
```

Zmienne

Wszyscy, którzy uważali na programowaniu wiedzą, czym są zmienne. W Linuksie działają prawie dokładnie tak samo. Plus jest taki, że nie musimy deklarować typu zmiennej. Składnia deklaracji też jest podobna.

```
zmienna='jajco'
```

Zwrócić należy uwagę, że nie mamy tutaj żadnych spacji, ani przed, ani po znaku =. Jest tak dlatego, że shell każdą linijkę traktuje jako polecenie. Dlatego linijka

```
zmienna = jajco
```

zostanie potraktowana jako próba uruchomienia polecenia „zmienna” z parametrami „=” i „jajco”. Jakkolwiek wspaniałe nie byłyby to argumenty, tak polecenie „zmienna” raczej się nie odpali.

Aby gdzieś dalej w skrypcie odwołać się do zmiennej, znów, prawie jak w C, używamy jej nazwy, ale poprzedzonej znakiem dolara

```
$zmienna
```

```
1  #!/bin/bash
2
3  zmienna='jajco'
4  echo Zmienna, którą zadeklarowalem to $zmienna
5
```

I wynik działania tego wspaniałego kawałka kodu

```
michal@michal-VirtualBox:~/skrypty$ ./zmienne.sh
Zmienna, którą zadeklarowalem to jajco
michal@michal-VirtualBox:~/skrypty$
```

W skryptach basha możemy też wykonywać operacje matematyczne. Tu już nie jest podobnie, jak w C. Pracować będziemy tylko na integerach, ponieważ bash nie wspiera bezpośrednio liczb innych niż całkowite. Wynik działania arytmetycznego zapisujemy do zmiennej.

```
wynik=$(($zmienna1[+-*/]$zmienna2))
```

```
1  #!/bin/bash
2  zm1=2
3  zm2=4
4  wynik=$(($zm1+$zm2))
5  echo Wynik dodawania to $wynik
6
```

Do operacji arytmetycznych możemy użyć również polecenia let, które również zapisze nam wynik do zmiennej

```
let a=1+8
```

Alternatywnie, możemy użyć polecenia expr, które zamiast zapisywać wynik do zmiennej wydrukuje je na terminalu

```
expr 5 + 2
```

Wiemy jak zapisac do zmiennej wynik działania, ale do zmiennej możemy również zapisać wynik jakiegoś polecenia. Aby to osiągnąć, zamknąć to polecenie musimy w znakach backticks, czyli `. To ten znaczek, co dzieli klawisz z tyldą ~.

```
1  #!/bin/bash
2  wynik=`grep ^[Aa] jajco.txt | wc -l`
3  echo w pliku jajco.txt jest $wynik linii zaczynających sie od A
4
```

Wynik działania tego skryptu:

```
michal@michal-VirtualBox:~/skrypty$ ./backtick.sh
w pliku jajco.txt jest 5 linii zaczynających sie od A
```

Argumenty w skryptach

Gdy odpalimy skrypt, to niektóre zmienne są ustawiane automatycznie.

- $0 – nazwa skryptu
- $1-9 – argumenty z wiersza poleceń
- $# - ilość argumentów podanych w wierszu poleceń

I w ten sposób możemy na przykład wykonać proste dodawanie:

```
1  #!/bin/bash
2  wynik=$(($1+$2))
3  echo Wynik dodawania to $wynik
4
```

```
michal@michal-VirtualBox:~/skrypty$ ./dodawanie.sh 3 5
Wynik dodawania to 8
```

If

Instrukcja warunkowa if w bashu też działa inaczej niż w C. Bliżej mu do pseudokodu, niż normalnego kodu.

```
if [ warunek ]
then
      cośtam
fi
```

Budowanie tych warunków to już w ogóle inna bajka. Inaczej będzie wyglądało porównywanie stringów, inaczej integerów

Dla liczb porównania wyglądają następująco. Dla uproszenia porównamy je do C:

- `-eq - == (equals)`
- `-gt - > (greater than)`
- `-lt - < (less than)`
- `-ge - >= (greater or equal)`
- `-le <= (less or equal)`

Stringi natomiast porównujemy tak:

- `= - == (równa się)`
- `!= - != (nie równa się)`

Czyli warunek, który wykona się dla zmiennej1 większej od zmiennej2 wyglądać będzie następująco:

```
[ zmienna1 -gt zmienna2 ]
```

Tutaj, inaczej niż w moich poprzednich skryptach, nawiasów kwadratowych nie pomijamy.

Operacje logiczne działają, dla odmiany, identycznie jak w C.

```
[ zmienna1 -eq zmienna2 ] && [ zmienna2 -lt zmienna3 ]
```

Możemy też użyć else'a lub elifa(elseif), jeśli taka jest nasza wola

```
if [ warunek ]
then
     cośtam
elif [ warunek ]
then
     cośtamcośtam
else
     cośtam innego
fi
```

Kilka zadań, bo tak

1. napisz skrypt, który wyświetla jeden komunikat, jeśli przekazano do niego jakiekolwiek argumenty, a inny, jeśli nie przekazano żadnego.
2. napisz skrypt, który wypisze zawartość katalogu Downloads, a następnie wypisze ile jest w nim plików.
3. napisz skrypt, który porówna 3 parametry i wydrukuje największy z nich. Jeśli poda się do niego więcej lub mniej niż 3 parametry, musi wydrukować error.
4. napisz skrypt, który jako argument przyjmie trzy litery sektorów oraz liczbę x , a następnie wypisuje pierwszych x komputerów z tych sektorów z pliku komputery.txt z lab6.
5. napisz skrypt, który zapisze do pliku log[data_utworzenia] wynik polecenia top.