Wstęp

Czym są skrypty powłoki? Są to zwykłe pliki tekstowe, które zawierają instrukcje oraz polecenia systemowe wykonujące dane zadanie. Wyobraź sobie sytuację, że musisz wykonać jakąś czynność w systemie – na przykład wykonać kopię zapasową (z wykorzystaniem programu tar) trzech katalogów, w których finalnej nazwie zawarta będzie aktualna data. Dodatkowo chciałbyś wykonywać to co tydzień w poniedziałek. Pomijając mechanizm cron, za każdym razem musiałbyś wpisywać te same komendy.

Istnieje jednak szybsze rozwiązanie. Tworzysz skrypt, do którego wpisujesz po kolei Twoje polecenia, zapisujesz i później za każdym razem uruchamiasz tylko swój skrypt. Proste i ułatwiające życie? No jasne, także zobaczmy jak to wygląda w praktyce :)

Hello World

Ulubionym edytorem tworzymy nowy plik tekstowy z rozszerzeniem sh:

nano hello_world.sh

Każdy skrypt rozpoczynamy od zdefiniowania powłoki (shell) jakiej on dotyczy. W naszym przypadku jest to bash, więc:

#!/bin/bash

Standardowo pierwszy program podczas nauki danego języka programowania rozpoczynamy od wypisania klasycznego *Hello World!*. W systemie Linux za wypisanie ciągu znaków odpowiada komenda echo.

echo „Hello World!”

To tyle. Zapisujemy i wychodzimy z edycji pliku.

Uruchamianie skryptów

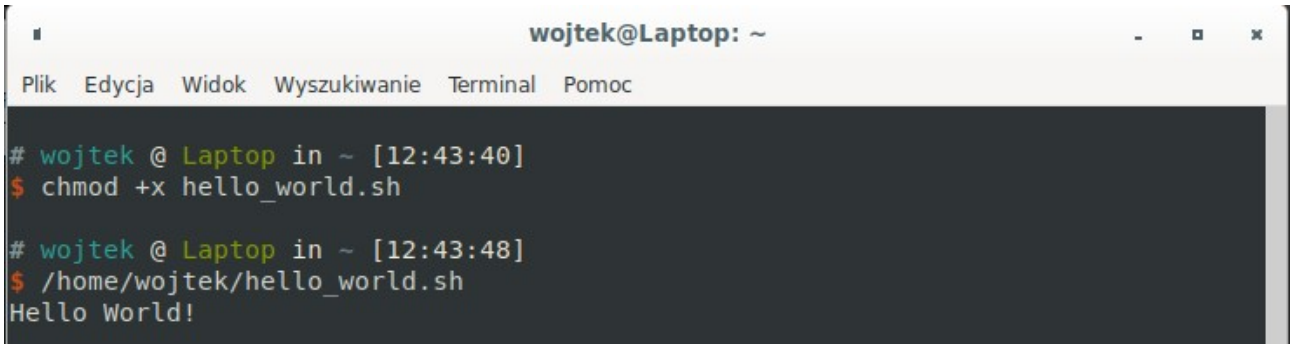
W tym momencie jest to jedynie zwykły plik tekstowy. Aby stał się skryptem musimy nadać mu uprawnienia wykonywania:

chmod +x hello_world.sh

Skrypty uruchamiamy poprzez podanie ich ścieżki. Pierwszym ze sposobów (wykorzystywanym w mechanizmie cron) jest pełna ścieżka. Wpisujemy:

/home/wojtek/hello_world.sh

i wciskamy enter. W tym momencie na naszym ekranie powinien wyświetlić się napis *Hello World!*



Uruchomienie skryptu hello_world.sh

Drugim ze sposobów, zdecydowanie szybszym (i częściej wykorzystywanym) jest podanie bezpośredniej ścieżki do skryptu:

./hello_world.sh

W powyższym zapisie kropka oznacza dany katalog (w którym się obecnie znajdujemy).

Zmienne

W skryptach (tak jak to występuję w językach programowania) możemy definiować zmienne. W przypadku skryptów bash definiujemy je jako:

nazwa_zmiennej=wartość

Mogą one zawierać różne typy: tekst, liczba, wynik danej komendy, lokalizacja pliku/katalogu. W przypadku danych typu tekst wartość definiujemy wewnątrz „”, bądź ‘’. Jeżeli chcemy, aby w danej zmiennej była wartość zwróconej komendy systemowej definiujemy ją wewnątrz `` (obok cyfry 1 na klawiaturze, pod ~).

Odwołanie się do zmiennej odbywa się poprzez podanie *$nazwa_zmiennej*. Tutaj należy zauważyć, że podczas definiowania podawaliśmy jedynie jej nazwę. W momencie odwołania musimy przed jej nazwą wstawić znak dolara.

```
# wojtek @ Laptop in ~ [12:56:31]
$ cat zmienne.sh
#!/bin/bash
tekst="To jest przykładowy tekst"
liczba=13
wynik_komendy=`date`
lokalizacja=/home/wojtek

echo $tekst
echo $liczba
echo $wynik_komendy
echo $lokalizacja

# wojtek @ Laptop in ~ [12:56:34]
$ ./zmienne.sh
To jest przykładowy tekst
13
pon, 8 cze 2020, 12:56:36 CEST
/home/wojtek
```

Operacje na zmiennych

Pętla if

Najpopularniejszą pętlą jest pętla sprawdzająca if. Działa ona tak samo jak w innych przypadkach: sprawdza warunek, jeżeli się zgadza wykonuje działanie, jeżeli się nie zgadza wykonuje inne. Jej schemat w przypadku basha wygląda następująco:

if [warunek]

then

działanie_jeżeli_prawda

else

działanie_jeżeli_fałsz

fi

Należy jednak zwrócić uwagę, iż początek możemy zapisać również trochę inaczej:

if [warunek]; then

Uwaga! Spacje obok warunku są nieprzypadkowe. Należy pamiętać, aby pomiędzy warunkiem, a [] zawsze była spacja (przed oraz za warunkiem).

```
# wojtek @ Laptop in ~ [13:13:46]
$ cat if.sh
#!/bin/bash
liczba=13
if [ $liczba == 12 ]
then
        echo 12
else
        echo $liczba
fi

if [ $liczba == 13 ]; then
echo 13
else
echo $liczba
fi
```

Działanie pętli if

** Tabulator użyty dla przejrzystości kodu, nie jest wymagany.*

Sprawdzanie warunków

Istnieje wiele metod sprawdzania warunków. W powyższym przypadku wykorzystano najprostszy, porównanie wartości ze zmienną (==) - jeden znak równości służy do przypisania wartości do zmiennej. Oprócz tego możemy zastosować następujące:

Operator porównania	**Opis**	**Przykład użycia**
-e	Plik istnieje	if [-e /*home/wojtek/plik]*
-d	Plik istnieje i jest katalogiem	if [-d /*home/*wojtek]
-lt	Mniejsze niż	if [$zmienna -lt 1]
-gt	Większe niż	if [$zmienna -gt 2]
-le	Mniejsze lub równe	if [$zmienna -le 3]
-ge	Większe lub równe	if [$zmienna -ge 4]
!	**Zaprzeczenie**	**if [! -e /*home/wojtek/plik]*** *(tutaj : plik nie istnieje)*

Parametry i zmienne specjalne

W momencie uruchamiania skryptu możemy przekazać dodatkowe parametry, które będą wykorzystywane wewnątrz. Aby tego dokonać po nazwie skryptu podajemy je kolejno (maksymalnie 9):

./skrypt.sh parametr1 parametr2 parametr3

Wewnątrz skryptu występują zmienne specjalne, które to odwołują się m. in. do tych parametrów.

$0 – jest to nazwa uruchomione skryptu, możemy wyświetlić ją poprzez echo.

Zmienne od $1 do $9 są to przekazane parametry. Możemy je wykorzystać na przykład w pętlach, do podjęcia odpowiednich zadań w zależności od wskazań użytkownika.

Zmiennna $# zawiera ilość przekazanych wszystkich parametrów. Jeżeli natomiast chcemy jednocześnie wyświetlić wszystkie parametry wówczas musimy zrobić echo na zmiennej $@.

```
# wojtek @ Laptop in ~ [13:51:50]
$ cat parametry.sh
#!/bin/bash
echo "Nazwa skryptu to: $0"
echo "Pierwszy argument to: $1"
echo "Trzeci argument to: $3"
echo "Ilość przekazanych argumentów wynosi: $#"
echo "Poszczególne argumenty to: $@"


# wojtek @ Laptop in ~ [13:51:53]
$ ./parametry.sh pierwszy 2 3 pięć
Nazwa skryptu to: ./parametry.sh
Pierwszy argument to: pierwszy
Trzeci argument to: 3
Ilość przekazanych argumentów wynosi: 4
Poszczególne argumenty to: pierwszy 2 3 pięć
```

Zmienne specjalne

Przykładowe skrypty

Na sam koniec tego wprowadzenia do skryptów powłoki bash chciałbym zaprezentować Ci trzy proste skrypty.

1. Skrypt wypisujący informacje o użytkowniku oraz danym hoście.

```
#!/bin/bash
echo "Moja nazwa użytkownika" `whoami`
echo "Moja nazwa hosta to" `hostname`
echo -e "\nIlość wolnej pamięci RAM/SWAP"
free -h
```

```
# wojtek @ Laptop in ~ [14:03:50]
$ ./przykladowy1.sh
Moja nazwa użytkownika wojtek
Moja nazwa hosta to Laptop

Ilość wolnej pamięci RAM/SWAP
               total        used        free      shared  buff/cache   available
Mem:           4,5Gi       2,7Gi       451Mi       489Mi       1,4Gi       1,1Gi
Swap:          2,7Gi       159Mi       2,6Gi
```

Wynik działania przykładowego skryptu nr 1

2. Prosty kalkulator

```
# Prosty kalkulator, który najpierw pobiera od użytkownika rodzaj działania,
# a następnie pobiera dwie liczby, na których będzie operował
#!/bin/bash
while true; do
echo "\nJaki rodzaj działania chcesz wykonać? [+ - * /] \nAby zakończyć wciśnij Ctrl + C"
read dzialanie
echo "Podaj dwie liczby, po każdej naciskając ENTER"
read pierwsza
read druga
echo "Wynik działania to:"
case $dzialanie in
"+") echo $(($pierwsza + $druga)) ;;
"-") echo $(($pierwsza - $druga)) ;;
"*") echo $(($pierwsza * $druga)) ;;
```

"/") echo $(($pierwsza / $druga)) ;;

**) echo "Wybrałeś zły rodzaj działania."*

esac

done

```
# wojtek @ Laptop in ~ [14:27:48] C:130
$ ./przykladowy2.sh

Jaki rodzaj działania chcesz wykonać? [+ - * /]
Aby zakończyć wciśnij Ctrl + C
+
Podaj dwie liczby, po każdej naciskając ENTER
2
3
Wynik działania to:
5

Jaki rodzaj działania chcesz wykonać? [+ - * /]
Aby zakończyć wciśnij Ctrl + C
/
Podaj dwie liczby, po każdej naciskając ENTER
6
2
Wynik działania to:
3

Jaki rodzaj działania chcesz wykonać? [+ - * /]
Aby zakończyć wciśnij Ctrl + C
^C
```

Wynik działania przykładowego skryptu nr 2

3. Skrypt ze wstępu (tar trzech katalogów, data w nazwie)

#!/bin/bash

 # Deklaracja zmiennych

data=`date +"%Y-%m-%d"`

kat1=/home/wojtek/Pulpit

kat2=/home/wojtek/Dokumenty

kat3=/home/wojtek/Obrazy

 # Pakowanie zawartości

tar -czf pulpit-$data.tar.gz $kat1

tar -czf dokumenty-$data.tar.gz $kat2

tar -czf obrazy-$data.tar.gz $kat3

Daj znać czy powyższy e-book był dla Ciebie przydatny. Może masz jakieś pytania? Czekam na Twoją wiadomość pod adresem mailowym biuro@wojst.pl

Trzymaj się :)