Skrypty powłoki

dr inż. Krzysztof Konopko
e-mail: k.konopko@pb.edu.pl

Plan na dziś

- Tworzenie i uruchamianie skryptów.
- Zmienne.
- Instrukcje warunkowe.
- Pętle.
- Wybrane polecenia.
- Funkcje.

Dla tych co lubią poczytać:

- M. Lach, „Bash. Praktyczne skrypty”, Helion 2015.
- A. Robbins, „Bash. Leksykon kieszonkowy. Przewodnik dla użytkowników i administratorów systemów”, Helion, 2017.
- C. Albing, JP Vossen, C. Newham, „Bash. Receptury”, Helion 2012.
- D. Taylor, B. Perry, „Genialne skrypty powłoki”, Helion 2017.
- W. E. Shotts Jr., „Linux. Wprowadzenie do wiersza poleceń”, Helion 2015.
- S. Lakshman, „Skrypty powłoki systemu Linux. Receptury”, Helion 2012.
- Ł. Sosna, „Linux. Komendy i polecenia”, Helion 2014.
- M. G. Sobell, „Linux. Programowanie w powłoce”, Helion 2013.

Skrypty powłoki

Start powłoki (dla powłoki logowania bash)

- najpierw wykonuje polecenia zawarte w pliku /etc/profile,
- następnie powłoka poszukuje plików ~/.bash_profile, ~/.bash_login lub ~/.profile, w takiej właśnie kolejności i wykonuje polecenia z pierwszego napotkanego pliku,
- Kiedy rozpoczynasz proces wylogowania, powłoka bash wykonuje polecenia zapisane w pliku ~/.bash_logout.

Start powłoki (dla powłoki interaktywnej bash)

- Interaktywne powłoki typu non-login po uruchomieniu wykonują polecenia zapisane w pliku ~/.bashrc . W wielu systemach Linux skrypt ten uruchamia plik /etc/bashrc.

Tworzenie i uruchamianie skryptów powłoki

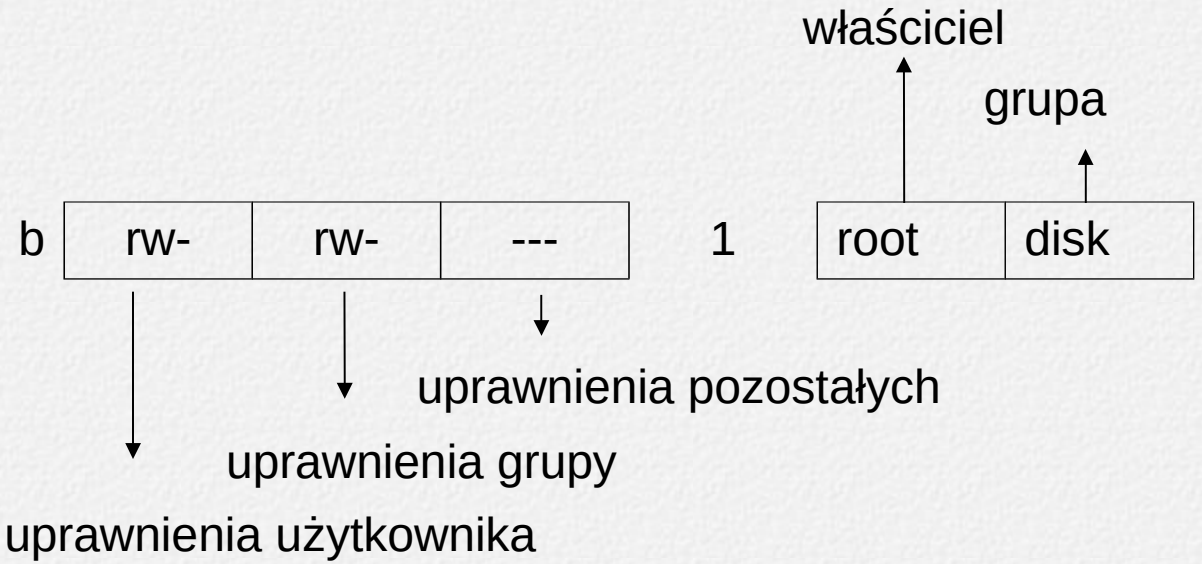
Skrypt powłoki to plik testowy o dowolnej nazwie i ustawionym bicie wykonywalności. Skrypt może zawierć dowolne polecenia powłoki i może być uruchamiany na jeden z dwóch sposobów:

```
./skrypt
bash –x skrypt.
```

Pierwsza linia skryptu zawiera ścieżkę do interpretera w postaci:

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

Prawa dostępu do plików



	Plik	**Katalog**
r	Odczyt zawartości pliku	Odczyt listy plików i podkatalogów
w	Modyfikacja zawartości plików	Modyfikacja zawartości katalogów
x	Wykonywanie pliku	Możliwość uczynienia katalogiem bieżącym

Podstawowe polecenia:

Polecenia związane z zarządzaniem plikami i katalogami:

- chmod - polecenie służące do administrowania uprawnieniami,
- chown - zmiana właściciela pliku lub katalogu.

Zmienne programowe:

Powłoka pozwala użytkownikowi na definiowanie zmiennych. Można deklarować i inicjować zmienne poprzez przypisywanie im wartości:

```
zmienna=”wartość”
```

Aby usunąć wybraną zmienną, powinieneś użyć polecenia `unset`:

```
unset zmienna
```

Zmienne powłoki są lokalne dla danego procesu. Zmienne środowiskowe są globalne i zostają umieszczone w środowisku przy użyciu wbudowanego polecenia `export`:

```
zm1=war1
export zm1
```

Pod zmienną możemy podstawić wynik jakiegoś polecenia poprzez:

- użycie odwrotnych apostrofów: `` `polecenie` ``
- rozwijania zawartości nawiasów: `$(polecenie)`

Zmienne specjalne:

Prywatne zmienne powłoki, udostępniane użytkownikowi tylko do odczytu np.:

- **$0** - nazwa bieżącego skryptu lub powłoki,
- **$1..$9** - wartości parametrów przekazywanych do skryptu,
- **$@** - wartości wszystkie parametró przekzywanych do skryptu.
- **$?** - kod powrotu ostanio wykonywanego polecenia,
- **$$** - PID procesu bieżącej powłoki.

Zmienna	Wartość
BASH_ENV	Ścieżka do pliku startowego dla powłok nieinteraktywnych.
CDPATH	Ścieżka wyszukiwania dla polecenia cd.
COLUMNS	Szerokość ekranu wykorzystywana przez polecenie select.
HISTFILE	Ścieżka do pliku przechowującego historię poleceń (domyślnie ~/.bash_history).
HISTFILESIZE	Maksymalna liczba pozycji zapisywanych w pliku HISTFILE (domyślnie 1000-2000).
HISTSIZE	Maksymalna liczba poleceń zapisywana w historii poleceń (domyślnie 1000).
HOME	Ścieżka do katalogu domowego użytkownika wykorzystywana jako domyślny argument polecenia cd podczas rozwijania ścieżek zawierających znak tyldy (~).
IFS	Wewnętrzny separator pól (ang. Internal Field Separator), używany do dzielenia słów.
INPUTRC	Ścieżka do pliku startowego biblioteki READLINE (domyślnie ~/.inputrc).
LANG	Zmienna przechowująca ustawienia językowe dla elementów, które nie zostały ustawione przez odpowiednie zmienne LC_*.
LC_	Grupa zmiennych przechowujących ustawienia językowe dla różnych elementów, w skład grupy wchodzą takie zmienne jak LC_COLLATE, LC_CTYPE, LC_MESSAGES czy LC_NUMERIC aby wyświetlić pełną listę ustawień, powinieneś użyć polecenia locale.
LINES	Wysokość ekranu wykorzystywana przez polecenie select.
MAIL	Ścieżka do pliku przechowującego skrzynkę pocztową użytkownika.
MAILCHECK	Rozmiar interwału czasowego, z jakim powłoka bash sprawdza, czy użytkownik otrzymał nową wiadomość (domyślnie 60 sekund).
MAILPATH	Lista oddzielonych od siebie dwukropkami ścieżek do plików, które powłoka bash sprawdza pod kątem nowych wiadomości.
OLDPWD	Ścieżka do poprzedniego katalogu roboczego.
PATH	Lista oddzielonych od siebie dwukropkami ścieżek do katalogów, w których powłoka bash poszukuje poleceń.
PROMPT_COMMAND	Polecenie, które powłoka bash wykonuje przed wyświetleniem podstawowego znaku zachęty.
PS1	Zmienna przechowująca podstawowy znak zachęty.
PS2	Zmienna przechowująca wtórny znak zachęty.
PS3	Zmienna przechowująca znak zachęty dla menu polecenia select.
PS4	Zmienna przechowująca znak zachęty dla trybu śledzenia wykonywania poleceń.
PWD	Ścieżka do bieżącego katalogu roboczego.
REPLY	Przechowuje wiersz ze standardowego wejścia, odczytany poleceniem read wykorzystywana przez polecenie select.

Zmienne tablicowe:

BASH pozwala na stosowanie zmiennych tablicowych jednowymiarowych. Kolejne wartości zmiennej tablicowej indexowane są przy pomocy liczb całkowitych, zaczynając od 0.

- `zmienna=(wartość1 wartość2 wartość3 wartośćn)`

Do elementów tablicy odwołuje się za pomocą wskaźników. Wskaźnikiami są indexy elementów tablicy, począwszy od 0 do n oraz @, *. Gdy odwołując się do zmiennej nie poda się wskaźnika: ${nazwa_zmiennej} to nastąpi odwołanie do elementu tablicy o indexie 0.Jeśli wskaźnikiem będą: @ lub * to zinterpretowane zostaną jako wszytskie elementy tablicy.

Można też uzyskać długość (liczba znaków) danego elementu tablicy:

- `${#nazwa_zmiennej[wskaźnik]}`

Możliwe jest dodawanie i usuwanie, z zastosowaniem komendy `unset`, elementów tablicy.

Obliczanie wyrażeń arytmetycznych:

Do przeprowadzenia obliczeń można skorzystać z polecenia `let`.

```
let wynik=liczba1*liczba2
```

obliczenia dokonywane są na liczbach całkowitych.

Można skożystać także bezpośrednio z następujacej składni:
`$((wyrażenie)) lub $[wyrażenie]`

Instrukcja warunkowa if:

Sprawdza czy warunek jest prawdziwy, jeśli tak to wykonane zostanie polecenie lub polecenia znajdujące się po słowie kluczowym `then`. Instrukcja kończy się słowem `fi`.

```
if warunek
then
  polecenie
fi
```

W sytuacji gdy test warunku zakończy się wynikiem negatywnym można wykonać inny zestaw poleceń, które umieszczamy po słowie kluczowym `else:`

```
if warunek
then
  polecenie1
else
  polecenie2
fi
```

Instrukcja warunkowa if:

Można też testować dowolną ilość warunków, jeśli pierwszy warunek nie będzie prawdziwy, sprawdzony zostanie następny, kolejne testy warunków umieszczamy po słowie kluczowym `elif:`

```
if warunek
then
  polecenie1
elif warunek
then
  polecenie2
fi
```

Warunki:

Do sprawdzania warunków służy polecenie `test`:

```
test wyrażenie1 operator wyrażenie2
```

którą można zapisać w postaci nawiasów kwadratowych:

```
[ wyrażenie1 operator wyrażenie2 ]
```

Między nawiasami a treścią warunku muszą być spacje.

Polecnie `test` zwraca wartość **0** (true) jeśli warunek jest spełniony i wartość **1** (false) jeśli warunek nie jest spełniony. Wartość warunku jest umieszczana w zmiennej specjalnej **$?**.

Operatory polecenia `test`:

- -a operator and
- -o operator or
- -b plik istnieje i jest blokowym plikiem specjalnym
- - plik istnieje i jest plikiem znakowym
- -e plik istnieje
- -h plik istnieje i jest linkiem symbolicznym
- = sprawdza czy wyrażenia są równe
- != sprawdza czy wyrażenia są różne
- -n wyrażenie ma długość większą niż 0
- -d wyrażenie istnieje i jest katalogiem
- -z wyrażenie ma zerową długość
- -r można czytać plik
- -w można zapisywać do pliku
- -x można plik wykonać
- -f plik istnieje i jest plikiem zwykłym
- -p plik jest łączem nazwanym
- -N plik istnieje i był zmieniany od czasu jego ostatniego odczytu
- plik1 -nt plik2 plik1 jest nowszy od pliku2
- plik1 -ot plik2 plik1 jest starszy od pliku2
- -lt mniejsze niż
- -gt większe niż
- -ge większe lub równe
- -le mniejsze lub równe

Instrukcja case:

Pozwala na dokonanie wyboru spośród kilku wzorców. Najpierw sprawdzana jest wartość zmiennej po słowie kluczowym case i porównywana ze wszystkimi wariantami po kolei. Oczywiście musi być taka sama jak wzorzec do którego chcemy się odwołać. Jesli dopasowanie zakończy się sukcesem wykonane zostanie polecenie lub polecenia przypisane do danego wzorca. W przeciwnym wypadku użyte zostanie polecenie domyślne oznaczone symbolem gwiazdki

```
case zmienna in
  "wzorzec1") polecenie1 ;;
  "wzorzec2") polecenie2 ;;
  "wzorzec3") polecenie3 ;;
  *) polecenie_domyślne
esac
```

Przy zastosowaniu ;; po przypasowaniu do warunku i wykonaniu instrukcji wychodzi z konstrukcji case. Przy zastosowaniu ;& wykonuje instrukcje także kolejnego warunku.

Pętla for:

Wykonuje polecenia zawarte wewnątrz pętli, na każdym składniku listy.

```
for zmienna in lista
do
  polecenie
done
```

Pętla `for` jest bardzo przydatna w sytuacjach, gdy chcemy wykonać jakąś operację na wszystkich plikach w danym katalogu.

Pętla **while**:

Najpierw sprawdzany jest warunek, jeśli jest on prawdziwy to wykonywane jest polecenie lub lista poleceń zawartych wewnątrz pętli, gdy warunek stanie się fałszywy pętla zostanie zakończona.

```
while warunek
do
polecenie
done
```

Pętla **until**:

Sprawdza czy warunek jest prawdziwy, gdy jest fałszywy wykonywane jest polecenie lub lista poleceń zawartych wewnątrz pętli, miedzy słowami kluczowymi do a done. Pętla until kończy swoje działanie w momencie gdy warunek stanie się prawdziwy

```
until warunek
do
polecenie
done
```

Polecenie read:

Czyta ze standardowego wejścia pojedynczy wiersz umożliwiając przypisanie kilku wartości kilku zmiennym.

Wybrane opcje:

-p Pokaże znak zachęty bez kończącego znaku nowej linii.

-a Kolejne wartości przypisywane są do kolejnych indeksów zmiennej tablicowej.

-e Jeśli nie podano żadnej nazwy zmienej, wiersz trafia do $REPLY.

-t timeout czas wygaśnięcia w sekundach

-s nie wyświetlaj znaków wpisanych przez użytkownika.

Pętla select:

Generuje z listy słów proste ponumerowane menu, każdej pozycji odpowiada kolejna liczba od 1 wzwyż. Poniżej menu znajduje się znak zachęty PS3, gdzie wprowadzana jest wybrana pozycja z menu. Jeśli nic nie zostanie wybrane (wciśnięty ENTER), menu będzie wyświetlone ponownie. Wartość wyboru zachowywana jest w zmiennej REPLY. Pętla działa dotąd dopóki nie wykonane zostaje polecenie break lub return.

```
select zmienna in lista
do
  polecenie
done
```

Polecenie dialog:

Tworzy okna w skryptach shellowych za pomocą, których można tworzyć listy wyboru, zadawać pytania, pobierać dane od użytkownika czy informować go o przebiegu jakichś operacji, które skrypt w danej chwili wykonuje.

`dialog opcje okno_dialogowe`

Wybrane rodzaje okien dialogowych:

yesno, menu, inputbox, textbox, infobox, checklist, radiolist, gauge

Opcje:

--clear czyści ekran

--createrc plik Możemy użyć tej opcji do wygenerowania przykładowego pliku konfiguracyjnego.

--separate-output Tą opcję stosuje się przy widgetach checklist, drukuje wyjście w osobnych liniach, co umożliwia przetwarzanie uzyskanych danych przez inny program.

--title Tytuł, na górze okna dialogowego.

--backtitle Podtytuł, w tle okna dialogowego, znajduje się w lewym górnym rogu ekranu.

Polecenie **alias**:

`alias` to komenda, która umożliwia zastąpienia wielu słów jednym słowem. Poleceniem `alias` często posługuje się w celu skrócenia komend systemowych lub dodania parametrów do istniejących komend. Zdefiniowane aliasy obowiązują w sesji powłoki. Jeżeli są wpisane do plików konfiguracyjnych powłok (np. ~/.bashrc, /etc/bashrc), to są też dostępne przy każdym uruchomieniu powłoki.

```
alias [nazwa[=wartość]]
alias       # Bez parametrów wydaje wszystkie aliasy
alias -p    # Tak samo jak powyżej
alias kopia # Wyświetla komendę, która zastąpiona
jest słowem kopia
```

Aliasy usuwa komenda `unalias`:

Funkcje:

Stosuje się je gdy w skrypcie powtarza się grupa poleceń. Do danej funkcji odwołujemy się podając jej nazwę, a wykonane zostanie wszystko co jest wpisane między nawiasy { }.

```
function nazwa_funkcji
{
polecenie1
polecenie2
polecenie3
}
```

lub:

```
function nazwa_funkcji()
{
polecenie1
polecenie2
polecenie3
}
```

Funkcje:

Funkcje moga się znajdować w innym pliku, których dołączenie odbywa się przez podanie nazwy pliku poprzedzonej kropk ą i spacją:

```
. ~/naszplik_z_funkcjami
nazwa_funkcji
```

Przekazanie parametrów do funkcji następuje dokładnie tak samo jak do poleceń:

```
nazwa_funkcji parametr_1 parametr_2
```

Zmienna specjalna **$0** przechowujaca nazwę skryptu, a nie nazwę funkcji.

Dziękuję za uwagę

Zapraszam
za tydzień :)