

## **Podstawy skryptów systemowych w Linux – zmienne, parametry i instrukcje sterujące.**

Automatyzacja pracy - skrypty w Linux (Bash)

### **Cel ogólny**

Poznanie zasad tworzenia i działania skryptów Bash oraz możliwości ich wykorzystania do automatyzacji pracy w systemie Linux.

### **Cele szczegółowe**

Po zakończeniu lekcji uczeń:

1. Definiuje pojęcie skryptu Bash.
2. Wyjaśnia zastosowanie skryptów w systemie Linux.
3. Charakteryzuje rolę interpretera Bash w wykonywaniu skryptów.
4. Opisuje zasady tworzenia i uruchamiania skryptów.
5. Wyjaśnia znaczenie deklaracji interpretera `#!/bin/bash`.
6. Rozróżnia zmienne użytkownika i zmienne środowiskowe.
7. Wymienia i opisuje podstawowe zmienne systemowe wykorzystywane w Bash.
8. Wyjaśnia rolę polecenia `read` oraz parametrów przekazywanych do skryptu.
9. Rozróżnia parametry specjalne skryptów Bash (`$0`, `$1`, `$2`, `$#`, `$*`).
10. Wyjaśnia zasadę działania instrukcji warunkowej `if`.
11. Charakteryzuje zastosowanie instrukcji wyboru `case`.
12. Rozróżnia rodzaje pętli (`for`, `while`, `until`) oraz wskazuje ich zastosowania.
13. Wyjaśnia przeznaczenie instrukcji `break` i `continue`.
14. Omawia możliwości wykonywania operacji matematycznych w skryptach Bash.
15. Rozpoznaje podstawowe polecenia administracyjne wykorzystywane w skryptach Linux.
16. Rozpoznaje polecenia służące do pracy z plikami i katalogami.
17. Wyjaśnia znaczenie debugowania skryptów oraz działania polecenia `bash -x`.
18. Omawia budowę prostego skryptu Bash.

### **Cele w języku ucznia**

Po lekcji:

- wiem, czym jest skrypt Bash,
- znam zastosowania skryptów w Linuxie,

- wiem, jak tworzy się i uruchamia skrypty,
- rozróżniam zmienne użytkownika i systemowe,
- wiem, do czego służą parametry skryptu,
- rozumiem działanie instrukcji if i case,
- rozróżniam pętle for, while i until,
- wiem, jak wykonywane są operacje matematyczne w Bash,
- rozpoznaję podstawowe polecenia administracyjne Linux,
- wiem, do czego służy debugowanie skryptów.

## 1. Skrypty w systemie Linux

W codziennej pracy administratora lub użytkownika systemu Linux często wykonuje się te same czynności wielokrotnie. Mogą to być kopie zapasowe, wyszukiwanie plików, zbieranie informacji o systemie czy zarządzanie użytkownikami. Aby nie wpisywać za każdym razem tych samych poleceń, można zapisać je w postaci skryptu.

**Skrypt** jest plikiem tekstowym zawierającym zestaw poleceń wykonywanych przez interpreter powłoki. W systemach Linux najczęściej wykorzystywany jest interpreter **Bash** (**Bourne Again Shell**).

Dzięki skryptom możliwe jest automatyzowanie pracy oraz wykonywanie złożonych operacji za pomocą jednego polecenia.

## 2. Tworzenie i uruchamianie skryptów

Skrypt jest zwykłym plikiem tekstowym. Aby system wiedział, jaki interpreter ma go uruchomić, pierwsza linia powinna zawierać tzw. shebang:

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

# (hash) bywa nazywany w slangu programistycznym **sharp** lub **she**

! to **bang** (wykrzyknik)

W skryptach można umieszczać komentarze rozpoczynające się znakiem:

```
#
```

Po zapisaniu pliku należy nadać mu prawo wykonywania:

```
chmod +x skrypt.sh
```

Opcja +x nadaje plikowi uprawnienie wykonywania, dzięki czemu może zostać uruchomiony jak program.

Uruchomienie skryptu następuje poleceniem:

```
./skrypt.sh
```

lub

bash skrypt.sh

### **3. Zmienne w skryptach**

Zmienne służą do przechowywania danych wykorzystywanych podczas działania programu.

Przykładowe deklaracje:

IMIE="Jan"

WIEK=18

Aby odczytać wartość zmiennej, należy poprzedzić jej nazwę znakiem \$.

echo \$IMIE

Zmienne mogą przechowywać tekst, liczby, wyniki działania poleceń oraz dane wprowadzone przez użytkownika.

### **4. Zmienne systemowe**

Bash udostępnia wiele gotowych zmiennych środowiskowych zawierających informacje o systemie i użytkowniku.

Najczęściej wykorzystywane są:

- USER - nazwa zalogowanego użytkownika,
- HOME - katalog domowy użytkownika,
- PWD - bieżący katalog roboczy,
- PATH - ścieżki wyszukiwania poleceń,
- TERM - typ używanego terminala.

Zmienne środowiskowe są często wykorzystywane przez skrypty administracyjne.

Zmienne systemowe (środowiskowe) są tworzone automatycznie przez system lub powłokę Bash i przechowują informacje o użytkowniku, środowisku pracy oraz konfiguracji systemu.

### **5. Wprowadzanie danych i parametry skryptu**

Skrypt może pobierać dane od użytkownika przy pomocy polecenia:

read

Polecenie read pobiera dane wpisane przez użytkownika i zapisuje je do zmiennej.

Przykład:

read imie

Innym sposobem przekazywania danych jest użycie parametrów podawanych podczas uruchamiania skryptu.

./skrypt.sh Ala Kowalska

W skrypcie parametry są dostępne jako:

- \$1 - pierwszy parametr,
- \$2 - drugi parametr,
- \$3 - trzeci parametr.

Dodatkowo:

- \$0 oznacza nazwę skryptu,
- \$# oznacza liczbę parametrów,
- \$\* oznacza wszystkie parametry.

Parametry umożliwiają dostosowanie działania skryptu bez konieczności jego modyfikowania.

## 6. Instrukcja warunkowa IF

Instrukcja if umożliwia podejmowanie decyzji na podstawie spełnienia określonego warunku.

Można za jej pomocą sprawdzać:

- wartości liczbowe,
- teksty,
- istnienie plików,
- istnienie katalogów.

Przykładowo skrypt może sprawdzić, czy katalog istnieje, a następnie wykonać odpowiednie działanie.

Instrukcja ta jest podstawowym mechanizmem podejmowania decyzji w skryptach Bash.

## 7. Instrukcja CASE

Instrukcja case służy do realizowania wyboru wielowariantowego.

Znajduje zastosowanie przede wszystkim w:

- menu tekstowych,
- analizie parametrów skryptu,
- wyborze sposobu działania programu.

Jest odpowiednikiem instrukcji wyboru znanej z innych języków programowania.

## 8. Pętle w skryptach

Pętle umożliwiają wielokrotne wykonywanie tych samych poleceń.

### Pętla FOR

Stosowana wtedy, gdy liczba powtórzeń jest znana.

Przykład zastosowania:

- przetwarzanie listy plików,
- wykonywanie operacji dla wielu użytkowników.

### Pętla WHILE

Wykonuje instrukcje tak długo, jak spełniony jest określony warunek.

**Pętla UNTIL** działa odwrotnie niż pętla WHILE i wykonuje instrukcje, dopóki warunek jest fałszywy.

Do sterowania pętlami wykorzystuje się także polecenia:

- break – natychmiast kończy działanie pętli,
- continue – przerywa bieżącą iterację i przechodzi do następnej.

Krótkie porównanie:

- for - gdy znamy liczbę powtórzeń,
- while - gdy warunek ma być sprawdzany przed każdą iteracją,
- until - gdy instrukcje mają być wykonywane, dopóki warunek jest fałszywy.

## 9. Operacje matematyczne

Skrypty Bash umożliwiają wykonywanie podstawowych działań arytmetycznych:

- dodawania,
- odejmowania,
- mnożenia,
- dzielenia.

Współcześnie najczęściej używa się składni:

`$((a+b))`

Konstrukcja `$(( ))` służy do wykonywania obliczeń arytmetycznych w Bash.

Możliwe jest również porównywanie wartości liczbowych w instrukcjach warunkowych.

## **10. Komendy administracyjne wykorzystywane w skryptach**

Skrypty bardzo często korzystają z poleceń administracyjnych systemu Linux.

Do najczęściej używanych należą:

### **Informacje o systemie**

- `date` - wyświetla aktualną datę i czas,
- `hostname` - wyświetla nazwę komputera,
- `uname` - wyświetla informacje o systemie operacyjnym,
- `uptime` - wyświetla czas pracy systemu od ostatniego uruchomienia.

### **Informacje o użytkownikach**

- `who` - wyświetla zalogowanych użytkowników,
- `whoami` - wyświetla nazwę bieżącego użytkownika,
- `id` - wyświetla identyfikator użytkownika i grupy.

### **Zarządzanie procesami**

- `ps` - wyświetla listę procesów,
- `top` - monitoruje procesy w czasie rzeczywistym,
- `kill` - kończy działanie procesu.

### **Informacje o dyskach**

- `df` - wyświetla wykorzystanie systemów plików,
- `du` - wyświetla rozmiar plików i katalogów,
- `mount` - montuje system plików,
- `umount` - odmontowuje system plików.

Polecenia te pozwalają pobierać informacje niezbędne do monitorowania i zarządzania systemem. Polecenia administracyjne są często wykorzystywane do monitorowania systemu, zarządzania użytkownikami, procesami oraz zasobami komputera.

## 11. Operacje na plikach i katalogach

Jednym z najczęstszych zastosowań skryptów jest zarządzanie plikami.

Najważniejsze polecenia to:

- ls - wyświetlanie zawartości katalogu,
- cp - kopiowanie plików,
- mv - przenoszenie i zmiana nazw,
- rm - usuwanie plików,
- mkdir - tworzenie katalogów,
- find - wyszukiwanie plików,
- cat - wyświetlanie zawartości pliku.

## 12. Debugowanie skryptów

Podczas tworzenia skryptów mogą pojawiać się błędy składniowe lub logiczne.

Do wyszukiwania błędów można wykorzystać tryb diagnostyczny:

```
bash -x skrypt.sh
```

Pozwala on śledzić wykonywanie kolejnych instrukcji i ułatwia odnalezienie problemu.

Opcja -x powoduje wyświetlanie kolejnych poleceń wykonywanych przez skrypt, co ułatwia odnalezienie błędów.

### Najważniejsze pojęcia do zapamiętania:

- Bash
- skrypt
- shebang (`#!/bin/bash`)
- zmienna
- zmienna środowiskowa
- parametr (`$1`, `$2`)
- if
- case

- for
- while
- until
- read
- chmod +x
- bash -x

### **Podsumowanie**

Skrypty Bash są podstawowym narzędziem automatyzacji pracy w systemie Linux. Umożliwiają tworzenie programów wykonujących powtarzalne zadania administracyjne i użytkowe. Podczas tworzenia skryptów wykorzystuje się zmienne, parametry, instrukcje warunkowe, instrukcje wyboru, pętle, operacje matematyczne oraz polecenia systemowe. Umiejętność pisania i analizowania skryptów jest jedną z podstawowych kompetencji administratora systemów Linux oraz technika informatyka.