Bash

Bash для усіх

Автор: Савенко Вадим Анатолійович @2024

Зміст:

1. Коротко про Bash

Bash був розроблений як покращена версія оригінального Bourne Shell (sh), яка включає нові можливості та вдосконалення. Він сумісний з більшістю команд та скриптів, написаних для sh, але також надає безліч додаткових функцій, таких як покращені можливості обробки рядків, роботи з масивами та розширені механізми управління потоками вводу-виводу. В основному він використовується як командний інтерпретатор у дистрибутивах Linux. Зазвичай файл для Bash проектів має формат *.sh*, але файли можуть і не мати такого формату. Достатньо ввести в терміналі команду: “sh /шлях/до/файлу”. Але якщо не хочеться постійно вводити таку команду, можна написати команду: “chmod +x /шлях/до/файлу”. Тепер, при введенні в консолі “./той_самий_файл” або “/шлях/до/файлу”, система зрозуміє, що це Bash проект і треба його запускати через команду: “sh /шлях/до/файлу”.

2. Команди

Для написання коду Bash можна використовувати будь-який редактор коду. Код, який відображає текст в консолі:

```
echo “Hello World”
```

В результаті в консолі буде:

```
Hello World
```

Там, де було написано "Hello World", можна вказати дані різних значень. Також є команда, яка приймає дані від користувача. Виглядає це наступним чином:

```
read var
```

Де було написано "var", вказувалося, що створюється змінна "var". Розмова про змінні буде пізніше.

Створимо код, який виведе на екран наше введене ім'я та прізвище, за допомогою виведених нами команд:

```
echo “Your first name:”
read firstName
echo “Your last name:”
read lastName
echo “Your are: $firstName $lastName”
```

Якщо змінні були написані зі знаком долара ($), він має бути обов'язково, якщо ми його використовуємо (якщо створюємо змінні, то знак долара не потрібно ставити).

Після запуску цього коду вам пропонується ввести ваше ім'я та прізвище, і після введення ваших даних ви отримуєте результат у терміналі, хто ви є.

Також можна створювати коментарі в коді. Це призначено для пропуску виконання команди. Коментарі створюються через хештег (#) в початку рядка команди. Ось приклад:

```
# echo Коментар
```

Є й інші команди, але у цій книзі будуть показані основні коди для Linux. Словник з базовими командами знаходиться на сторінці 15.

3. Змінні

На сторінці 4 було згадано про змінні. Іноді стають випадки, коли змінні повинні мати свої дані, а не ті, які ввів користувач. Ось приклад створення змінних:

```
var1="Hi" # Тип данних: текстові
var2=50 # Тип данних: числові
var3=(50 "Hi" 30) # Тип данних: массив
echo "Vars: 1st=$var1; 2nd=$var2; 3rd=${var3[2]}"
```

Як було зауважено, всі змінні мають значення, яке має свій тип даних. У Bash існують 3 типи даних:

- **Текстові**: там, де зберігаються будь-які набори символів, вони зберігаються в подвійних лапках (“ ”).
- **Числові**: там, де зберігаються лише цифри.
- **Массиви**: там, де зберігається набір різних значень з різними типами даних.

Якщо ми знаємо, як відображати змінні числових та текстових типів даних, то масиви також мають індекс, який є цілим числом та зберігається в квадратних дужках ([]). Індекс - це номер послідовності елемента, перший елемент якого починається не з 1 (оскільки це вже другий елемент), а з 0.

Якщо індекс менший за 0 (наприклад: -1, -2, -3 і так далі), це означає, що елементи беруться в зворотньому порядку. Наприклад: [-1] - останній елемент; [-2] - передостанній елемент; і так далі.

Також можна додавати числові та текстові змінні в масив. Ось приклад:

```
var=50
array=($var "Hi" 30)
```

У Python після кожного елемента (крім останнього) потрібно ставити кому та пробіл, підтверджуючи створення елемента у списку. Для Bash, як масиву, достатньо просто поставити пробіл.

4. Математика

Для виконання математичних операцій в Bash використовується команда let, арифметичне розширення $(( )), або команда expr. Тип даних змінних повинен бути числовим. Ось код для виконання математичних операцій:

```
let "a = 5 + 3"
b=$((5 * 3))
c=$(expr 10 - 3)
```

Якщо ми помножимо текстові дані на числові, то отримаємо повторення послідовності символів текстових даних. Якщо додати до текстових ще текстові дані, то це буде об'єднання текстових даних.

Крім множення, додавання та віднімання, існують інші арифметичні операції. Ось приклад коду з іншими арифметичними діями:

```
quotient=$((first_number / second_number)) # Ділення
power=$(echo "$first_number^$second_number" | bc) # Степінь
sqrt_first=$(echo "scale=2; sqrt($first_number)" | bc) # Корінь
```

До речі!!! Було забуто розповісти важну річ. Ім'я змінної повинно бути:

1) Написано англійським алфавітом;
2) Не збігається з ключовими словами в Bash (такими як: *as*, *from*, *with*, *if* і інші. Ми будемо вивчати всі ключові слова в Bash);
3) Повинно починатися з літери;
4) Цифри не можуть бути першими символами назви змінної;
5) Не має містити інших символів, крім: букв англійського алфавіту, цифр, дефісів та символу підкреслення (_);

5. Конвертація

Коли ми хочемо мати дані змінної у іншому форматі, потрібно здійснити конвертацію. Ось приклад коду:

```
# Числові у текстові
number=789
text="$number" # Зберігає number у числовому форматі

# Текстові у числові
text="1011"
number=$((text)) # Зберігає text у текстовому форматі
```

6. Умови

Якщо сьогодні понедіок, вівторок, середа, четверг или п'ятниця — ми працюємо. Якщо субота — працюємо менша. Інакше — ми відпочиваємо. Цей абзац є про — прикладом умови.

У Bash можна задавати умови. Ось приклад коду по умовам:

```
if [ $VAR -gt 10 ]; then # Если это истина
    echo "VAR больше 10"
elif [ $VAR -eq 10 ]; then # Иначе, если это истина
    echo "VAR равно 10"
else # Иначе (Если всё было - ложь)
    echo "VAR меньше 10"
fi
```

-gt позначає "больше чем", а *-eq* позначає "равно", а *-lt* позначає "меньше чем". Одним словом, це все — логічні оператори. У Bash є ще більше логічних операторів. Загалом, вони діляться для текстових або для числових.

Усі логічні оператори:

Для чилових данних:

- *-gt*: більше за (greater than)
- *-lt*: менше за (less than)
- *-ge*: більше або дорівнює (greater than or equal to)
- *-le*: менше або дорівнює (less than or equal to)
- *-eq*: дорівнює (equal to)
- *-ne*: не дорівнює (not equal to)

Для строковых данных:

- >: більше за (greater than)
- <: менше за (less than)
- =: дорівнює (equal to)
- !=: не дорівнює (not equal to)

7. Цикли

Було наведено приклад на сторінці 8, а тепер уявімо, що це цикл, який постійно повторюється до тих пір, поки ми не йдемо на пенсію.

У Bash є цикли, які повторюють дії до тих пір, поки їхнє значення не буде false, і цикли, які виконують набір команд кілька разів. Ось приклад цикла для виконання набору кодів кілька разів:

```
for i in {1..5}; do
    echo "Число: $i"
done
# Показує результат від 1 до 5.
# Таким чином, виконується команда 5 разів.
# {1..5} - там задаємо кількість виконань.
```

Ось приклад постійного циклу:

```
COUNTER=0

while [ $COUNTER -lt 5 ]; do
    echo "COUNTER: $COUNTER"
    COUNTER=$((COUNTER + 1))
done
```

Там, де квадратні дужки (*[ ]*), ми вказуємо умови для розумін-ня істини та брехні.

8. Функції

Функції в програмуванні, включаючи скрипти Bash, використовуються з декількох важливих причин. Давайте розглянемо приклад створення коду з функцією.:

Тепер розберемо код.

```
my_function() {
    local a=$1
    local b=$2
    echo "Это функция"
    echo "Аргумент a: $a"
    echo "Аргумент b: $b"
}

# Виклик функції з аргументом
my_function 3 5
```

1. Шебанг (Shebang):

Це перший рядок скрипта, відомий як "шебанг". Він вказує системі, що цей скрипт має виконуватися за допомогою інтерпретатора Bash.

2. Оголошення функції:

Тут визначається функція з ім'ям my_function. У Bash функції оголошуються за допомогою назви функції, після якої йде пара круглих дужок, а також фігурних дужок {} для визначення тіла функції.

Усередині функції:

- *local a=$1*: Створюється локальна змінна a і присвоюється значення першого аргумента функції (*$1*).
- *local b=$2*: Створюється локальна змінна b і присвоюється значення другого аргумента функції (*$2*).
- *echo "Это функция"*: Виводиться рядок "*Это функция*".
- *echo "Аргумент a: $a"*: Виводиться значення змінної *a*.
- *echo "Аргумент b: $b"*: Виводиться значення змінної *b*.

3. Виклик функції з аргументами:

Тут викликається функція *my_function* з двома аргументами: 3 і 5. Ці аргументи передаються функції, де вони стають доступними як *$1* і *$2*.

Детальний опис роботи:

1. **Коли викликається функція** *my_function* 3 5:

- Число 3 стає першим аргументом і присвоюється змінній a усередині функції.
- Число 5 стає другим аргументом і присвоюється змінній b усередині функції.
- **Усередині функції**:
- *local a=$1*: Змінна a приймає значання 3.
- *local b=$2*: Змінна b приймає значання 5.
- *echo "Это функция"*: Виводиться рядок "*Это функция*".
- *echo "Аргумент a: $3"*: Виводиться рядок "*Аргумент a: 3*".
- *echo "Аргумент b: $5"*: Виводиться рядок "*Аргумент b: 5*".

Повний процес виконання:

1. Скрипт починається з рядка #!/bin/bash, який вказує системі використовувати інтерпретатор Bash..
2. Визначається функція *my_function*.
3. Функція my_function викликається з аргументами 3 і 5.
4. Усередині функції *my_function*:

- Аргументи 3 і 5 присвоюються локальним змінним a і b.
- Функція виводить рядки з текстом та значеннями аргументів.

9. Відкриття файлів

Відомо, що можна зберігати дані за допомогою змінної, але можна зберігати, змінювати, додавати та читати дані з файлу. Ось приклад коду Bash для зчитування, додавання та зміни файлів:

```
# Приклад читання з файлу
while IFS= read -r line; do
    echo "$line" # $line - змінна, що зберігає дані з файлу.
done < "/шлях/до/файлу"

# Приклад додавання даних до файлу
echo "Новий рядок" >> "им’яФайла"

# Приклад запису у файл (перезапис)
echo "Перезаписаний рядок" > "им’яФайла"
```

10. Словник базових змінних Linux

Призначення змінної	Ім'я змінної	Опис змінної
Шлях до виконуваних файлів	$PATH	Список каталогів, у яких Bash шукає виконувані файли.
Домашня тека користувача	$HOME	Шлях до домашньої теки поточного користувача.
Ім'я поточного користувача	$USER	Ім'я поточного користувача.
Команда для оболонки за замовчуванням	$SHELL	Шлях до виконуваного файлу оболонки за замовчуванням.
Каталог тимчасових файлів	$TMPDIR (или $TMP)	Путь к директории, используемой для временных файлов.
Версія операційної системи	$OSTYPE	Тип операційної системи .
Стандартний текстовий редактор	$EDITOR	Ім'я стандартного текстового редактора.
Персональний ідентифікатор користувача	$UID	Числовий ідентифікатор поточного користувача.
Персональний ідентифікатор групи	$GID	Числовий ідентифікатор групи поточного користувача.
Стандартна кількість рядків на екрані	$LINES	Кількість рядків у терміналі.
Стандартна кількість стовпців на екрані	$COLUMNS	Кількість стовпців у терміналі.

11. Словник базових команд Linux

Команда	Призначення
`sudo su`	Перейти в режим суперкористувача
`echo` *текст*	Вивести *текст* у терміналі
`nano` *шлях/до/файлу* Пакет: *nano*	Увійти до редактора тексту для редагування *шлях/до/файлу.*
`ip addr` Пакет: *ip*	Отримати IP-адресу хоста (ПК)
`cd` *шлях/до/директорії*	Перейти в каталог за адресою: *шлях/до/директорії*
`ls`	Переглянути вміст каталогу
`wget` *https://посилання.на.файл* Пакет: *wget*	Скачать файл с интернета
`rm` *шлях/до/файлу*	Видалення файлу
`rm –r` *шлях/до/директорії*	Видалення директорії
`cp` *путь/к/файлу/файл путь/к/вставлению*	Копіювання файлу
`cp –r` *шлях/до/директорії шлях/до/вставлення*	Копіювання директорії
`mkdir` *им'яПапки*	Створення папки “*им'яПапки*”
`kill` *процес*	Зупиняє процес
`neofetch`	Інформація про систему

Команда	Призначення
`chmod +x` *шлях/до/файлу*	Повідомляє системі, що це не текстовий файл, а код для виконання команд Bash
`git clone` *https://шлях.до.репозиторию* Пакет: *git*	Завантажує всі файли з репозиторія (наприклад, з GitHub)
`sudo ssh –X` *користувачЦьогоПК@IPкомп'ютера* Пакет: *openssh*	Підключення віддаленого доступу і робота через консоль
`sudo ssh –X` *юзерЦьогоПК@IPкомп'ютера програма* Пакет: *openssh*	Підключення віддаленого доступу і робота з програмою (це може бути графічна або консольна)
`mv` *файл1 файл2 … /переміщення/у/директорію*	Переміщення файлу
`mv –r` */шлях/директорії /переміщення/у/директорію*	Переміщення директорії
`scp` */шлях/до/локального/файлу username@remote_host:/шлях/до/відаленого/ПК* Пакет: *openssh*	Копіювання файлу на інший ПК
`gzip` *файл.txt* Пакет: *gzip*	Стиснення архіву *файл.txt.tar.gz*
`gzip –l` *архів* Пакет: *gzip*	Інформація про архів
`bzip2 –d` *архів* Пакет: *bzip2*	Розпакування архіву

Bash для усіх

Автор: Савенко Вадим Анатолійович @2024