Практична робота № 17. Розробка сценаріїв мовою оболонки bash (частина 2). Аргументи командного рядка, управляючі структури, цикли. Перевірка умов командою test

Аргументи командного рядка

Передача аргументів у програму відбувається таким чином:

```
<ім'я програми> [<арг.1> <арг.2>…]
```

Усередині програми доступ до аргументів здійснюється за допомогою спеціальних змінних, званих позиційними:

`$<n>, n` – номер позиції аргументу;
`$1` – перший;
`$2` – другий.

Є ще додаткові можливості підстановки.
`$0` – ім'я сценарію;
`$*` – всі аргументи одним рядком;
`$@` – всі аргументи окремо;
`$#` – кількість аргументів;
`$$` – ідентифікаційний номер поточного процесу;
`$!` – ідентифікаційний номер фонового завдання;
`$?` – ідентифікаційний номер виконуваної команди.

Команди shift і set

Команда **`shift`** зсуває номери аргументів.
Приклад:

```
echo "$0 має $# аргументів"
shift
echo "$0 has $# arguments"
```

Команда **`set`** встановлює позиційні параметри. Імена (ідентифікатори) позиційних параметрів складаються з однієї або декількох цифр, крім одиночного нуля. Значеннями позиційних параметрів є аргументи, які були задані під час запуску оболонки (перший аргумент є значенням позиційного параметра 1 та ін.). Змінити значення позиційного параметра можна за допомогою вбудованої команди **`set`**.

Таким чином, позиційними параметрами можуть бути не обов'язково аргументи, а й виведення будь-якої команди.

Приклад:

```
echo "$1 $2 $3"
set `uname -a`
echo "$1 $2 $3"
```

Команди розгалуження

Мова оболонки містить два оператори розгалуження: ***if*** і ***case***.

Оператор ***if*** виконує дії залежно від істинності заданої умови і має такий синтаксис:

```
if <list1>
then
<commands1>
elif < list2>
then
<commands2>
else
<commands3>
fi
```

У наведеному записі як ***elif***, так і ***else*** можуть бути відсутні. Якщо тіло оператора ***if*** невелике, то зазвичай використовують запис в один рядок, такий як:

```
if <list1>; then <commands1>; else <commands2>; fi;
```

При цьому важливо правильно ставити ***;***.

Алгоритм виконання оператора ***if*** такий:

1) виконується ***<list1>***;

2) якщо результат виконання ***<list1>*** – істина, то виконується ***<commands1>*** і оператор ***if*** завершується;

3) інакше виконується ***<list2>*** і перевіряється результат його виконання;

4) якщо результат виконання ***<list2>*** – істина, то виконується ***<commands2>*** і оператор ***if*** завершується;

5) якщо результат виконання ***<list2>*** – хиба, то виконується ***<commands3>***.

Як умови ***<list1>*** і ***<list2>*** можуть використовуватися звичайні команди, але переважно – це виклик однієї або декількох команд ***test*** у вигляді ***test <вираз>*** або більш стисло ***test [<вираз>]***. Детально команда ***test*** розглянута нижче.

Нижче наведено приклад використання оператора ***if***:

```
#!/bin/bash
# Виконання програми з контролем її існування
if [ -x $HOME/script ]; then $HOME/script; fi;
```

У цьому прикладі перевіряється, чи існує в домашньому каталозі користувача файл з ім'ям ***script*** і чи є він виконуваним (атрибут ***'x'***). Якщо це так, то сценарій запускає ***script*** на виконання.

Якщо необхідно зробити вибір з багатьох альтернатив, то зручніше використовувати оператор ***case***, що має такий синтаксис:

```
case значення in
шаблон1) список1 ;;
шаблон2) список2 ;;
…
esac
```

Оператор ***case*** є аналогом оператора switch у мові С і працює схожим чином. Рядок значення порівнюється з кожним із зазначених шаблонів до тих пір, поки не буде виявлена відповідність. Після виявлення відповідності виконується список, зазначений після шаблону. Два символи «**;;**» після списку мають таке ж значення, що і оператор ***break*** у програмах мовою С. Якщо ніяких відповідностей не виявлено, то оператор case завершується без виконання будь-яких дій.

Нижче наведено приклад використання оператора ***case***. Сценарій очікує як параметр назву фрукта і видає його опис.

```
#!/bin/bash
# Друк інформації про фрукти
if [ -z "$1" ]
then
echo "Помилка: пропущений параметр"
exit 1
fi

case "$1" in
apple) echo "Яблука дуже корисні" ;;
banana) echo "Банани дуже смачні" ;;
*) Echo "Нічого не можу сказати про" $1 ;;
esac
```

Синтаксис команди test

Команда **`test`** перевіряє зазначений вираз і закінчує свою роботу з кодом завершення 0 (істина) або 1 (хиба). Як вираз можуть використовуватися вирази 3 типів:

- перевірки характеристик файлів;
- порівняння рядків;
- числові порівняння.

Основний синтаксис для перевірки характеристик файлів такий:

`test <опція> <ім'я_файлу>`.

У наведеній нижче таблиці зведені можливі опції команди **`test`** для перевірки характеристик файлів.

Таблиця 1.

Опція	*Опис*
`-b`	Істина, якщо файл існує і є блоковим спеціальним файлом
`-c`	Істина, якщо файл існує і є символьним спеціальним файлом
`-d`	Істина, якщо файл існує і є каталогом
`-e`	Істина, якщо файл існує
`-f`	Істина, якщо файл існує і є звичайним файлом
`-h`	Істина, якщо файл існує і є символічним посиланням
`-r`	Істина, якщо файл існує і доступний на читання
`-s`	Істина, якщо файл існує і має ненульовий розмір
`-S`	Істина, якщо файл існує і є сокетом
`-w`	Істина, якщо файл існує і доступний за попереднім записом
`-x`	Істина, якщо файл існує і доступний на виконання

У наведеній нижче таблиці зведені можливі способи вказівки використання команди **`test`** для перевірки рядків.

Таблиця 2.

Опція	*Опис*
`-z <рядок>`	Істина, якщо рядок має нульову довжину
`-n <рядок>`	Істина, якщо рядок має ненульову довжину
`<рядок1> = <рядок2>`	Істина, якщо рядки збігаються
`<рядок1>! = <рядок2>`	Істина, якщо рядки різні

У разі використання числових порівнянь команда **`test`** має такий синтаксис:

`test <число1> <оператор> <число2>`

У наведеній нижче таблиці зведені можливі значення операторів у числових порівняннях.

Таблиця 3.

Оператор	***Опис***
`-eq`	Істина, якщо числа рівні
`-ne`	Істина, якщо числа не рівні
`-lt`	Істина, якщо менше
`-le`	Істина, якщо менше або дорівнює
`-gt`	Істина, якщо більше
`-ge`	Істина, якщо більше або дорівнюють

Організація циклів

Мова оболонки дозволяє організовувати циклічне виконання команд. У розпорядження користувачеві пропонується кілька варіантів циклів:

- ***while;***
- ***for;***
- ***select.***

Цикл while

Оператор циклу **`while`** має такий синтаксис:

```
while <команда>
do
<список>
done
```

Під час виконання циклу спочатку виконується **`<команда>`**. Якщо результат ненульовий, то відбувається вихід з циклу, в іншому випадку виконується тіло циклу **`<список>`** і відбувається перехід на наступну ітерацію. Хоча як умову команда може використовувати будь-яку допустиму в GNU/Linux команду, зазвичай це команда **`test`**. Нижче наведено приклад виведення на термінал десяти послідовних чисел від 0 до 9.

```
#! /bin/bash
# Арифметичні обчислення
x = 0
while [$x -lt 10]   # значення змінної x менше 10?
do
echo $x
x = `expr $x + 1`  # збільшення x на 1
done
```

Цикл for

Цикл **for** виконує команди з список для кожного елемента зі списку і має такий синтаксис:

```
for <ім'я> in <елемент1> <елемент2> ... <елементN>
do
<список>
done
```

Як елементи можна використовувати різні шаблони. В наведеному нижче прикладі всі файли з домашнього каталогу користувача, які закінчуються на **.bash**, копіюються в каталог **scripts** і робляться доступними на читання всім користувачам.

```
#!/bin/bash
# виконання операцій над групою файлів
for FILE in $HOME/*.bash
do
cp $FILE ${HOME}/scripts
chmod a+r ${HOME}/scripts/${FILE}
done
```

Мова програмування оболонки містить оператори, які порушують нормальне виконання циклу. Ці оператори мають назви **break** і **continue**. Вони працюють точно так само, як і їх аналоги в мові С.

Цикл select

Цикл **select** дозволяє створювати зручні меню. Він корисний, коли необхідно, щоб користувач вибрав один елемент із запропонованого списку. Оператор **select** має такий же вигляд, як і оператор **for**, за винятком ключового слова.

Під час виконання цього оператора циклу всі елементи зі списку висвічуються на екрані разом з їх порядковими номерами, після чого з'являється спеціальне запрошення для введення. Зазвичай воно має вигляд **#?**. Введений користувачем номер пункту меню записується в змінну **REPLY**. Якщо **$REPLY** містить номер пункту меню, то в змінну **<ім'я>** заноситься значення відповідного елемента зі списку. В іншому випадку список буде виведений заново.

Після того, як користувачем буде зроблений допустимий вибір, виконується команди зі **<список>**, після чого виконання циклу повторюється з самого початку (висвітиться запрошення для введення та ін.). Для повторного відображення меню у відповідь на запрошення введення слід натиснути клавішу Enter. Вихід з циклу здійснюється тими ж засобами, що і для **while** та **for**.

Нижче наводиться приклад використання оператора ***select***. У користувача запитується тип пристрою «миша», і залежно від зробленого вибору виконуються певні дії. В цьому випадку це просто виведення повідомлення, що підтверджує зроблений вибір. У наведеному сценарії звернемо увагу на другий рядок. У більшості випадків діючий за замовчуванням вигляд запрошення не влаштовує. Визначаючи змінну PS3, можна задати необхідний текст запрошення.

```
#!/bin/bash
# «Конфігурація» пристрою «миша»
PS3 = "Оберіть тип пристрою «миша»:"
select ITEM in Microsoft Logitech ps2 none
do
case $ITEM in
Microsoft) echo "дії по «миші» Microsoft" ;;
Logitech) echo "дії по «миші» Logitech" ;;
ps2) echo "дії по «миші» ps2" ;;
none) echo "вибрано – немає «миші»" ;;
esac
break
done
```

Завдання

1. Написати сценарій, що приймає три аргументи (**a**, **b**, **c**) і виводить значення ***(a+b)/c***.
2. Написати сценарій, що приймає два числові аргументи і виводить найбільший з них. У тому випадку, якщо аргументів більше 2, треба вивести повідомлення про помилку.
3. Потрібно перевірити, чи є файл звичайним або він є каталогом (аргументом є назва файлу). Якщо це звичайний файл, то сценарій повинен виводити ім'я файлу і його розмір. У разі, якщо розмір файлу перевищує кілобайт, то розмір повинен виводитися в кілобайтах. Якщо розмір перевищує мегабайт – у мегабайтах.
4. Написати сценарій, який виводить посекундно в циклі імена файлів поточного каталогу і їх порядковий номер (використовувати команду ***sleep <секунди>***).
5. Написати сценарій, який генерує 100 файлів ***1.txt ... 100.txt***, і в кожен файл записує поспіль 100 чисел N, де N = порядковий номер файлу. Потім скрипт повинен з'єднати в один файл всі файли з парними номерами (***even.txt***) і в інший файл – всі файли з непарними номерами (***odd.txt***).

Контрольні питання

1. Яким чином у програму можна передати аргументи?
2. Як у програмі можна отримати доступ до аргументів, з якими визивалася програма?
3. Які спеціальні змінні можна використовувати в скриптах командного інтерпретатора?
4. Яким чином можна запускати програми залежно від результату виконання інших програм?
5. Які оператори розгалуження існують у bash? Наведіть їх синтаксис.
6. Для чого призначена та який синтаксис має команда ***`test`***?
7. Які види циклів існують у bash? Наведіть їх синтаксис.