***Лабораторна робота №11***

***«Розробка сценаріїв на мові оболонки bash (частина 3). Використання функцій у сценаріях»***

***Створення функцій***

При написанні сценаріїв командного інтерпретатора часто виникає необхідність використовувати один і той же код в декількох місцях. Якщо це лише невеликий фрагмент коду, то труднощів не виникне. Але якщо в сценарії командного інтерпретатора необхідно знову і знову вводити великі шматки однакового коду, то істотно зростають витрати часу на написання програми.

У командному інтерпретаторі **bash** передбачений спосіб дії в подібних ситуаціях, який заснований на застосуванні функцій, визначених користувачем. Повторювані фрагменти коду сценарію командного інтерпретатора можна оформляти у вигляді функції, а потім використовувати вже саму функцію в сценарії стільки разів, скільки буде потрібно.

Для створення функції в сценаріях командного інтерпретатора **bash** можна скористатися одним з двох форматів. У першому форматі використовується ключове слово ***function*** поряд з ім'ям функції, яким позначається блок коду:

***function <ім’я>***

***{***

***<команди…>***

***}***

Атрибут ***<ім’я>***  визначає унікальне ім'я, присвоєне функції. Кожна функція, яка визначається в сценарії, повинна отримати унікальне ім'я.

У цьому визначенні ***<команди…>*** - це одна або кілька команд командного інтерпретатора **bash**, з яких складається дана функція. Після виклику функції командний інтерпретатор **bash** виконує кожну з цих команд в тому порядку, в якому вони присутні у визначенні функції, точно так, як і при виконанні звичайного коду сценарію.

Другий формат визначення функції в сценарії командного інтерпретатора **bash** більшою мірою нагадує формат, який застосовується для визначення функцій в інших мовах програмування:

***<ім’я> ()***

***{***

***<команди…>***

***}***

Порожні круглі дужки після імені функції вказують на те, що далі йде визначення функції. На цей формат поширюються такі ж правила іменування функцій, як і на вихідний формат функцій сценаріїв командного інтерпретатора.

***Використання функцій***

Щоб скористатися функцією в сценарії, необхідно вказати в рядку коду ім'я функції за таким же принципом, як відбувається виклик будь-якої іншої команди командного інтерпретатора:

**Приклад 1. Файл example1.sh**

***#!/bin/bash***

***# використання функції у сценарії***

***func1()***

***{***

***echo "Приклад функції"***

***}***

***count=1***

***while [ $count -le 5 ]***

***do***

***func1***

***count=$[ $count + 1 ]***

***done***

***echo "Кінець циклу"***

***func1***

***echo "Кінець сценарію"***

**Виконання сценарію**

***$ ./example1***

***Приклад функції***

***Приклад функції***

***Приклад функції***

***Приклад функції***

***Приклад функції***

***Кінець циклу***

***Приклад функції***

***Кінець сценарію***

В даному випадку при кожному посиланні на ім'я функції ***func1*** командний інтерпретатор **bash** звертається до визначення ***func1()*** і виконує всі команди, передбачені у цьому визначенні.

Визначення функції не обов'язково має бути приведене у сценарії командного інтерпретатора в першу чергу, однак необхідно дотримуватися обережності. **При спробі використання функції до її визначення з'являється повідомлення про помилку.**

Необхідно також дотримуватися певних правил, що стосуються імен функцій. Кожне ім'я функції має бути унікальним, оскільки в іншому випадку виникає проблема. Якщо в сценарії зустрічається ще одне визначення однієї і тієї ж функції, то відбувається так зване перевизначення, і нова версія функції перекриває її вихідну версію, не викликаючи жодних повідомлень про помилку.

**Приклад 2. Файл example2.sh**

***#!/bin/bash***

***# перевірка використання повторюваного імені функції***

***func1()***

***{***

***echo "*** ***Перше визначення імені функції"***

***}***

***func1***

***func1()***

***{***

***echo "*** ***Повторення одного і того ж імені функції"***

***}***

***func1***

***echo "*** ***Кінець сценарію"***

**Виконання сценарію**

***$ ./example2***

***Перше визначення імені функції***

***Повторення одного і того ж імені функції***

***Кінець сценарію***

Початкове визначення функції ***func1*** діяло цілком задовільно, але після обробки другого визначення функції ***func1*** у всьому подальшому коді сценарію виклик функції призводить до застосування другого визначення.

***Повернення значення з функції***

У командному інтерпретаторі **bash** функції розглядаються як свого роду міні-сценарії, зі своїм статусом виходу. Передбачено три способи, за допомогою яких можна формувати статус виходу для користувацьких функцій.

**1. Статус виходу, що заданий за замовчуванням.** За замовчуванням статус виходу функції визначається як статус виходу, повернутий останньою командою в функції. Після завершення виконання функції можна скористатися стандартною змінною ***$?*** для визначення статусу виходу функції.

**Приклад 3. Файл example3.sh**

***#!/bin/bash***

***# перевірка статусу виходу функції***

***func3()***

***{***

***echo "*** ***Намагання відобразити неіснуючий файл"***

***ls -l badfile***

***}***

***echo "*** ***Тестування функції:"***

***func3***

***echo "*** ***Статус виходу: $?"***

**Виконання сценарію**

***$ ./example3***

***Тестування функції:***

***Намагання відобразити неіснуючий файл***

***ls: badfile: No such file or directory***

***Статус виходу: 1***

В даному випадку статус виходу функції дорівнює 1, оскільки виконання останньої команди в функції закінчилося невдачею. Але це не дозволяє дізнатися, чи закінчилось виконання всіх інших команд у функції успішно. Змінимо тіло функції ***func3*** наступним чином:

***func3()***

***{***

***ls -l badfile***

***echo " Намагання відобразити неіснуючий файл"***

***}***

Цього разу останньої інструкцією до функцій був виклик команди відлуння, який завершився успішно, тому функція має статус виходу 0, незважаючи на те, що виклик однієї з команд у функції закінчився невдачею. Таким чином, підхід, який передбачає використання заданого за замовчуванням статусу виходу функції, не завжди виправданий. На щастя, передбачені інші способи формування статусу виходу функції.

**2. Використання команди *return*.** У командному інтерпретаторі **bash**, для виходу з функції з конкретним статусом може застосовуватися команда ***return***, яка дозволяє задати одне цілочисельне значення для визначення статусу виходу функції, що може служити простим способом завдання статусу виходу функції програмним шляхом.

**Приклад 4. Файл example4.sh**

***#!/bin/bash***

***# використання команди return у функції***

***func4()***

***{***

***Echo "Введіть значення: "***

***read value***

***echo "Подвоєння значення"***

***return $[$value\*2]***

***}***

***func4***

***echo "Нове значення дорівнює $?"***

Функція ***func4*** подвоює значення, щр міститься в змінній ***$value***, отримане в рядок введення даних користувачем. Потім в цій функції відбувається повернення сформованого результату за допомогою команди ***return***, а повернене значення відображається в сценарії з використанням змінної ***$?***.

Однак при використанні даного способу повернення значення з функції слід дотримуватись обережності. Щоб уникнути проблем, необхідно керуватися двома рекомендаціями:

* обов'язково виконувати вибірку значення, що повертається, відразу після завершення функції;
* не забувати про те, що статус виходу повинен знаходитися в межах від 0 до 255.

**3. Використання виведення з функції.** За аналогією з тим, що можна перехоплювати виведення команди за допомогою змінної командного інтерпретатора, можна також перехоплювати за допомогою змінної командного інтерпретатора виведення функції. Цей спосіб може використовуватися для отримання висновку будь-якого типу з функції для присвоювання його змінній:

***result = 'dbl'***

Ця команда присвоює висновок функції ***dbl*** змінної командного інтерпретатора ***$result***. Нижче наведено приклад використання даного способу в сценарії.

**Приклад 5. Файл example5.sh**

***#!/bin/bash***

***# використання команды echo для повернення значення***

***func5() {***

***echo "Введіть значення: "***

***read value***

***echo $[$value\*2]***

***}***

***result=‛func5‛***

***echo "Нове значення дорівнює $result"***

**Виконання сценарію**

***$./example5***

***Введіть значення: 200***

***Нове значення дорівнює 400***

***$./example5***

***Введіть значення: 1000***

***Нове значення дорівнює 2000***

У новій версії функції подвоєння значення тепер застосовується інструкція ***echo*** для відображення результату обчислення. У сценарії просто відбувається перехоплення виведення функції ***func5***, а не перегляд статусу виходу для отримання відповіді.

***Передача параметрів у функцію***

У командному інтерпретаторі **bash** функція розглядається як свого роду міні-сценарій. Це, зокрема, означає, що у функцію можна передавати параметри, як і у звичайний сценарій.

У функціях можна використовувати стандартні змінні середовища параметрів для представлення будь-яких параметрів, переданих у функцію в командному рядку. Наприклад, ім'я функції визначено в змінній ***$0***, а всі параметри в командному рядку функції визначаються з використанням змінних ***$1, $2*** і т.д. Крім того, для визначення кількості параметрів, що передаються в функцію, можна використовувати спеціальну змінну ***$#***.

Задаючи функцію у сценарії, необхідно приводити параметри в тому ж командному рядку, як і функцію, приблизно так:

***func1 $value1 10***

Потім в функції можна здійснити вибірку значень параметрів з використанням змінних середовища параметрів. Нижче наведено приклад застосування даного способу для передачі значень в функцію.

**Приклад 6. Файл example6.sh**

***#!/bin/bash***

***# передача параметрів у функцію***

***addem() #функція додавання 2 чисел***

***{***

***if [$# -eq 0] || [$# -gt 2]***

***then***

***echo -1***

***elif [$# -eq 1]***

***then***

***echo $[$1+$1]***

***else***

***echo $[$1+$2]***

***fi***

***}***

***echo "Сума 10 та 15: "***

***value=‛addem 10 15‛***

***echo $value***

***echo "Спробуємо знайти суму тільки одного числа: "***

***value=‛addem 10‛***

***echo $value***

***echo "Тепер спробуємо знайти суму жодного числа: "***

***value=‛addem‛***

***echo $value***

***echo "Наприкінці спробуємо знайти суму трьох чисел: "***

***value=‛addem 10 15 20‛***

***echo $value***

**Виконання сценарію**

***$./example6***

***Сума 10 та 15: 25***

***Спробуємо знайти суму тільки одного числа: 20***

***Тепер спробуємо знайти суму жодного числа: -1***

***Наприкінці спробуємо знайти суму трьох чисел: -1***

У сценарії ***example6*** функція ***addem*** спочатку перевіряє кількість параметрів, переданих зі сценарію. Якщо параметри не задані або кількість параметрів більше двох, функція ***addem*** повертає значення -1. Якщо заданий тільки один параметр, функція ***addem*** складає значення цього параметра з самим собою для отримання результату. Якщо задані два параметри, функція ***addem*** для формування результату складає отримані значення.

***Завдання***

1. Написати функцію, яка:

* вводить символьний рядок, що містить маршрутне ім'я деякого файлу;
* якщо файл існує, то виводить на екран його вміст;
* якщо файл не існує, то створює його і записує в нього рядок, передану в якості параметра;

1. Написати функцію, яка:

* у заданому першим параметром каталозі знаходить всі прості файли, в яких містяться задані другим і третім параметрами символьні рядки;
* у знайдених файлах видаляє всі повторювані рядки;
* виводить на екран імена всіх отриманих файлів.

1. Написати функцію, яка:

* у каталозі, ім'я якого передається першим параметром, знаходить все прості файли розміром більше заданого другим параметром;
* створює в зазначеному каталозі 3 нових каталоги;
* поміщує в створені каталоги файли з вихідного каталогу: в перший - файли, що містять один рядок із заданим словом, у другій - файли з двома такими рядками, в третій - з трьома;
* імена всіх файлів, які не включені в нові каталоги, виводить на екран.

***Контрольні питання***

1. Синтаксис оголошення функції у bash.
2. Які існують способи повертання значення з функції?
3. Як передаються параметри у функцію?