***Лабораторна робота №10***

***«Розробка сценаріїв на мові 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*** здвигає 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 при запуску оболонки (перший аргумент є значенням позиційного параметра 1, і т. д.). 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 оператора розгалуження: ***if*** і ***case***.

Оператор ***if*** виконує дії в залежності від істинності заданої умови і має наступний 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*** наступний:

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>***.

У якості умов ***<list1>*** і ***<list2>*** можуть використовуватися звичайні команди, але переважно - це виклик одній 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 даному 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 значння in***

***шаблон1) список1 ;;***

***шаблон2) список2 ;;***

***…***

***esac***

Оператор ***case*** є аналогом оператора switch в 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 символу ';;' після списку мають таке ж значення, що і оператор ***break*** в програмах на мові 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*** вираз перевіряє зазначений вираз і закінчує свою роботу з кодом завершення 0 (істина) або 1 (хиба). В якості вираз можуть використовуватися вирази 3 типів:

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

Основний синтаксис для перевірки характеристик файлів наступний:

***test <опція> <ім’я\_файлу>***.

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

|  |  |
| --- | --- |
| ***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 символічною 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.

|  |  |
| --- | --- |
| ***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 |

При використанні числових порівнянь команда ***test*** має такий синтаксис:

***test <число1> <оператор> <число2>***

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

|  |  |
| --- | --- |
| ***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 наступний синтаксис:

***while <команда>***

***do***

***<список>***

***done***

При виконанні 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 в якості умови команда може використовуватися будь-яка допустима в GNU/Linux команда, звичайно 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 наступний 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 в точності 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.

При виконанні даного 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 і т. д.). 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 в залежності від зробленого вибору виконуються певні дії. В даному випадку це просто виведення повідомлення, що підтверджує зроблений вибір. У наведеному сценарії звернемо увагу на другий рядок. В 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 (***а***, ***b***, ***с***) і виводить значення ***(a+b)/c***.

2. Написати сценарій, що приймає два числові аргументи і виводить найбільший з них. У випадку якщо аргументів більше 2 – 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 - в мегабайтах.

4. Напиcати 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 - все файли з непарними номерами (***odd.txt***).

***Контрольні питання***

1. Яким чином в програму можна передати аргументи?

2. Як в 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 в залежності 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.