Практична робота № 16. Розробка сценаріїв мовою оболонки bash (частина 1). Змінні, командні файли, файли ініціалізації

Основи розробки сценаріїв оболонки

Взаємодія користувача з операційною системою здійснюється через оболонку, яка представляє собою зовнішню програму. Відразу після запуску оболонки проводиться її ініціалізація для установки ряду параметрів. При цьому оболонкою проводиться читання двох файлів: ***/etc/profile*** і ***~/.profile***. У першому з них містяться налаштування параметрів, що є загальними для всіх користувачів. У другому файлі кожен користувач може розмістити власні налаштування для роботи з оболонкою.

Призначена для користувача оболонка може бути запущена на виконання в 2 режимах – інтерактивному і не інтерактивному. Коли оболонка видає користувачеві запрошення, вона працює в **інтерактивному** режимі. Це означає, що оболонка приймає введення від користувача і виконує команди, які користувач вказує. Оболонка називається інтерактивною, оскільки вона взаємодіє з користувачем. Завершення роботи з оболонкою в цьому випадку відбувається по команді користувача.

У **не інтерактивному** режимі оболонка не взаємодіє з користувачем. Замість цього вона читає команди з деякого файлу і виконує їх. Коли буде досягнуто кінець файлу, оболонка завершиться. Запуск оболонки в не інтерактивному режимі можна здійснити таким способом:

```
$ /bin/bash <ім'я_файлу>
```

Тут *<ім'я файлу>* – ім'я файлу, що містить команди для виконання. Такий файл називається **сценарієм оболонки**. Він є текстовим файлом і може бути створений будь-яким доступним текстовим редактором.

Користувачеві може представляти незручність кожен раз вказувати ім'я програми-оболонки */bin/bash* для виконання сценарію. Для того, щоб мати можливість виконувати сценарій, набираючи тільки його ім'я, перш за все необхідно зробити його виконуваним. Для цього необхідно встановити відповідні права доступу до файлу за допомогою команди ***chmod*** у такий спосіб:

```
$ chmod u+x <ім'я_файлу>
```

Крім цього, необхідно явно вказати, яка оболонка повинна використовуватися для виконання такого сценарію. Це можна зробити, розмістивши в першому рядку сценарію послідовність символів ***#!/bin/bash***. Тут вказується, що для виконання сценарію необхідно використовувати оболонку ***/bin/bash*.**

Сценарій може містити коментарі. Коментар – це оператор, який може розміщуватися в сценарії оболонки, але оболонка не виконується. Коментар починається з символу **#** і триває до кінця рядка. Нижче наведено приклад короткого сценарію:

```
#!/bin/bash
# Приклад короткого сценарію
date
who
```

Складені команди

Крім простих команд можна формувати складові команди. Вони являють собою упорядкований набір простих команд, пов'язаних між собою за допомогою спеціальних **операторів управління (ОУ).** Складена команда має такий синтаксис:

<команда> <ОУ> <команда> <ОУ> ... <ОУ> <команда>

У таблиці 1 наведено найбільш часто використовувані оператори управління та наведені пояснення щодо виконання відповідних складених команд.

Таблиця 1.

Оператори управління у складених командах

Оператор управління в складеній команді	*Правило виконання*
команда1 ; команда2	«команда2» виконується після завершення виконання «команди1» у будь-якому разі
команда1 && команда2	«команда2» виконується, якщо «команда1» була виконана успішно
команда1 || команда2	«команда2» виконується, якщо «команда1» не була виконана успішно

Змінні

Змінна – це «слово», якому присвоєне значення. Оболонка дозволяє створювати і видаляти змінні, а також присвоювати їм значення. У більшості випадків розробник відповідальний за управління змінними в сценаріях. Використання змінних дозволяє створювати гнучкі сценарії, що легко адаптуються. Визначаються змінні таким чином (відсутність проміжків до і після символу «**=**» істотно!):

<ім'я_змінної>=<значення>

Наприклад, ***MY_NAME=Sergey*** визначає змінну з ім'ям ***MY_NAME*** і привласнює їй значення ***Sergey***. Імена змінних визначаються за тими ж правилами, що і в мові програмування С. У змінній можна зберігати будь-яке потрібне значення. Особливий випадок – коли це значення містить проміжки. Для правильного виконання такої дії вказане значення достатньо взяти в лапки.

Для того, щоб отримати значення змінної, перед її ім'ям необхідно поставити знак ***$***. У тому випадку, коли деяка змінна стає непотрібною, її можна видалити командою:

unset <ім'я_змінної>.

Нижче наведено приклад, який ілюструє роботу зі змінними в сценаріях.

```
#!/bin/bash
#приклад операцій над змінними
MY_NAME=Sergey
MY_FULL_NAME="Sergey B. Sidorov"
echo name = $MY_NAME and full name = $MY_FULL_NAME
unset MY_NAME
```

У результаті виконання команди ***echo*** на термінал буде видано повідомлення:

```
name = Sergey and full name = Sergey B. Sidorov
```

Усі розглянуті вище приклади змінних – це приклади скалярних змінних, тобто таких, які можуть зберігати лише одне значення. Поряд з ними можна використовувати змінні-масиви. Доступ до елементів масиву здійснюється операцією індексування ***[]***. Мова програмування оболонки не вимагає попереднього оголошення змінної масиву із зазначенням його розмірності. При цьому окремі елементи масиву створюються в міру доступу до них. Нижче наведено приклад роботи з масивом ***NAME.***

```
#!/bin/bash
# Приклад використання масиву
NAME[0] = Сергій
NAME[10] = Катя
NAME[2] = Ліза
echo всі імена: ${NAME[*]}
echo ${NAME[10]} і ${NAME[2]} сестри
```

Якщо замість індексу елемента використовувати *, результатом виразу буде список значень всіх елементів масиву (в нашому випадку таких три).

Область видимості змінних

Кожна змінна має свою зону видимості. Припустимо, у сценарії визначена деяка змінна. Що буде з нею після завершення цього сценарію, чи доступна вона з інших сценаріїв, що викликаються вихідним сценарієм? У мові оболонки всі змінні діляться на три категорії: **локальні змінні**, **змінні оточення** і **змінні оболонки.**

Локальна змінна – це така змінна, яка існує тільки всередині конкретного екземпляра оболонки. Вона не доступна програмам, що запускаються на виконання з цієї оболонки. Усі розглянуті раніше змінні були локальними.

Змінна оточення – це змінна, яка доступна будь-якій програмі, запущеної з цієї оболонки. Деякі програми потребують певних змінних оточення. У такому випадку у сценарії їх можна визначити.

Змінна оболонки – це спеціальна змінна, яка встановлюється оболонкою і необхідна їй для коректної роботи. Деякі з них є змінними оточення, а деякі – локальними. У змінних з іменами ***1,2,3,…*** зберігаються параметри командного рядка. Тобто якщо якийсь сценарій ***script*** запустити у вигляді ***script something***, то в його оболонці ***$1 = something***.

Змінну можна розмістити в оточенні, виконавши команду експорту:
export <ім'я_змінної>.

У результаті виконання наступного сценарію на термінал будуть видані значення всіх змінних оточення оболонки і, в тому числі, змінної ***MY_NAME***.

```
#! /bin/bash
# Приклад розміщення змінної в оточенні
MY_NAME = Sergey; export MY_NAME;
set
```

Деякі стандартні змінні оточення та їх призначення вже розглядались у лабораторній роботі №6.

Засоби введення-виведення

Задачі введення-виведення можуть бути вирішені за допомогою використання спеціальних команд ***echo, printf, read***.

Команда ***echo*** видає на стандартний пристрій виведення значення всіх своїх параметрів. Команда ***printf*** подібна своєму аналогу – функції

printf у програмах мовою С. Першим параметром вказується форматний рядок, а далі перераховуються значення, що видаються.

Команда ***read <ім'я1> <ім'я2> ... <ім'яN>*** зчитує зі стандартного пристрою введення рядок та виділяє з нього окремі слова (групи символів, відокремлювані проміжками) і кожне слово заносить у зазначені як параметри відповідні змінні. При цьому якщо змінних менше, ніж виділених слів, то в останню з них буде занесена решта рядка.

Як приклад наведено сценарій, в якому від користувача запитується рядок, після чого він виводиться на термінал. Ключ ***–n*** команди ***echo*** забороняє перехід на наступний рядок після закінчення виведення, що дозволений за замовчуванням.

```
#!/bin/bash
#приклад використання засобів введення-виведення
echo –n Введіть рядок:
read INPUT
printf "%s\n" "$INPUT"
```

Підстановки

Виділяють такі види підстановок:

– **підстановка імен файлів** – дозволяє виконувати перетворення рядка, що містить шаблони, в список імен файлів;

– **підстановка змінної** – дозволяє управляти значенням змінної, ґрунтуючись на її стані;

– **підстановка команди** – дозволяє захопити виведення команди і підставити його в іншу команду;

– **арифметична підстановка** – дозволяє виконувати прості математичні обчислення, використовуючи оболонку.

Підстановка імен файлів ґрунтується на використанні різних шаблонів для завдання групи файлів у стислому вигляді замість повного їх перерахування. Шаблон ***** ототожнюється з будь-якою кількістю будь-яких символів в імені файлу. Шаблон ***?*** відповідає одному довільному символу, а шаблон ***[группа_символів]*** явно визначає набір символів, допустимих в місці його розташування. Нижче наведені приклади використання шаблонів:

****.cpp*** – список файлів, що закінчуються на **.cpp**

module?.h – список файлів з одним довільним символом замість ?

module[aA].cpp – **modulea.cpp** і / або **moduleA.cpp**

Механізм підстановки команди полягає у виконанні оболонкою вказаного набору команд і подальшої підстановки їх виведення замість

самих команд. Підстановка команди виконується під час запису команди у зворотних апострофах. Наприклад, у результаті виконання команди ***DATE = `date`*** змінна ***DATE*** прийме значення виведення команди ***date***.

Для обчислення значення арифметичного виразу ***expression*** необхідно використовувати команду такого вигляду:

$((expression))

При цьому у виразі можуть використовуватися операції додавання, віднімання, множення і цілочисельного ділення, а також круглі дужки для завдання порядку обчислень. Наприклад, у результаті виконання наступної команди змінна ***foo*** прийме значення **3.**

```
foo = $((((5+3*2)-4)/2))
```

Спочатку в командному інтерпретаторі Bourne була передбачена спеціальна команда, призначена для обробки математичних виразів. Команда **expr** дозволяла обробляти вираження, задані в командному рядку, але для цього застосовувалися надзвичайно складні конструкції:

```
$ expr 1+5
6
```

Командний інтерпретатор **bash** продовжує підтримувати команду **expr**, оскільки він повинен залишатися сумісним з командним інтерпретатором Bourne. Однак у **bash** передбачений більш простий спосіб виконання математичних обчислень. У командному інтерпретаторі **bash** у разі використання операції присвоювання результату математичних обчислень змінної можна включити математичний вираз у конструкцію, що складається зі знака долара зі квадратних дужок (***$[operation]***):

```
$ var1=$[1+5]
$ echo $var1
6
$ var2=$[$var1*2]
$ echo $var2
12
```

Кольорове виведення тексту у сценаріях bash

Для виведення у різних кольорах використовуються спеціальні послідовності, що визначають колір тексту та колір фону (див. ЛР № 6). У коді сценаріїв **bash** можливе використання даних послідовностей для виведення інформації у різних кольорах.

```
#!/bin/bash
red="\033[0;31m" #змінна, що визначає червоний колір тексту
blue="\033[0:34m" #змінна, що визначає синій колір тексту
```

normal="\033[0m" #змінна, що визначає звичайний колір тексту командного рядка

#оголошення змінної ***user***, що містить назву поточного користувача (змінна середовища оточення ***USER***), але червоного кольору
user="redUSER"

#оголошення змінної ***hello***, що містить рядок ***"Hello"*** синього кольору
#тут екранується назва змінної, оскільки здійснюється конкатенація значення змінної з літеральною константою
hello="${blue}Hello"

#виведення змінної та повернення до звичайного кольору тексту
echo –e "$hello, $user $normal"

Команда ***echo*** з ключем ***–e*** розпізнає escape-послідовності всередині рядків, завдяки чому текст у зазначеному вище прикладі буде виведений відповідним кольором.

Файли ініціалізації

Файли ініціалізації – це сценарії командної оболонки, які викликаються автоматично. Під час реєстрації користувача в bash запускається файл ***$HOME/.bash_profile***. Якщо ***.bash_profile*** відсутній, його роль виконує файл ***$HOME/.profile***. У кожній командній оболонці є файл ініціалізації самої оболонки ***$HOME/.bashrc***. У разі виходу із системи запускається файл ***$HOME/.bash_logout***.

Існують конфігурації файлів командної оболонки, які є загальними для всіх користувачів: ***/etc/bashrc, /etc/profile***.

Файли ініціалізації є текстовими файлами і, маючи відповідні права, можна їх редагувати.

$HOME/.bash_profile (***$HOME/.profile***) зазвичай містить команди, що формують інтерфейс. Наприклад, вітання, рядок запрошення, колірні настройки, значення глобальних змінних та ін.

$HOME/.bashrc містить команди, що розширюють або налаштовують командний інтерпретатор. Наприклад, виклик необхідних програм, оголошення псевдонімів тощо.

$HOME/.bash_logout містить команди, які виконуються під час виходу із системи. Наприклад, очистка екрана, рядок повідомлення і команда ***sleep***, яка задає паузу. Така пауза дозволяє прочитати виведені повідомлення.

Завдання

1. Створити сценарій, що видає таке повідомлення: «У моєму домашньому каталозі <n> підкаталогів: <підкаталоги>, та <m> файлів: <файли>».
2. Створити сценарій, що прочитує з екрана деяке слово і що виводить кількість символів у цьому слові.
3. Написати свій ***.bash_profile*** (***.profile***), що виконує такі функції:
   – виводить рядок вітання з вказівкою поточної дати;
   – дозволяє виконати програми, що зберігаються в ***myscripts*** як звичайні команди (за допомогою змінної середовища оточення PATH);
   – виводить рядок, що включає ім'я користувача, ім'я комп'ютера і поточний каталог, використовувати різні кольори.
4. Створити свій ***.bashrc***, в якому задаються такі команди:
   – команда резервного копіювання текстових файлів, створених власником за поточний день, окрім порожніх;
   – команда видалення всіх порожніх файлів з каталога.
5. Написати свій ***.bash_logout***, що виводить прощальне повідомлення.

Контрольні питання

1. Яке призначення командного інтерпретатора?
2. За що відповідає перший рядок файлу сценарію?
3. У чому різниця локальних і глобальних змінних?
4. Що представляють собою змінні командної оболонки?
5. Як у змінну записати результат виконання команди?
6. Які стандартні змінні ви знаєте? Їх призначення?
7. Як можна запустити скрипт на виконання?