Лабораторна робота № 5 Створення сценаріїв в оболонці Bash

Мета роботи - набути навичок створювання bash-скриптів в ОС Linux.

Теоретичні відомості

Bourne-again shell (GNU Bash) - це реалізація Unix shell, написана на C в 1987 році Брайаном Фоксом (Brian Fox) для GNU Project. Синтаксис мови Bash є надбудовою синтаксису мови Bourne shell. Переважна більшість скриптів для Bourne shell можуть бути виконані інтерпретатором Bash без змін, за винятком скриптів, які використовують спеціальні змінні або вбудовані команди Bourne shell. Також синтаксис мови Bash включає ідеї, запозичені з Korn shell (ksh) і C shell (csh): редагування командного рядка, історія команд, стек директорій, змінні $ RANDOM і $ PPID, синтаксис POSIX *для підстановки команд: $* (...).

Командна оболонка *BASH* дозволяє створювати скрипти за допомогою групування декількох команд, які виконують певну дію.

Скрипт - це звичайний текстовий файл, що містить системні або вбудовані команди оболонки. Такий файл може бути запущений на виконання наступним чином: ***$bash ім'я_файла.***

Оболонка послідовно інтерпретує і виконує команди, задані в сценарії. Ці ж команди можуть бути виконані простим послідовним викликом їх в командному рядку оболонки. Для файлів сценаріїв оболонки *bash* прийнято встановлювати розширення ***.sh***. Тобто, для виконання скрипту необхідно запустити командну оболонку, передавши їй як параметр ім'я відповідного файлу. Є другий варіант запуску скрипту - вказати його ім'я в командній оболонці (тобто зробити з нього якийсь вид програми). Для цього треба в параметрах доступу визначити файл як *виконуваний*, і в перших рядках цього файлу явно вказати оболонку, для якої призначений цей скрипт, в такий спосіб: **#!оболонка**

Будь-який сценарій для Bash починається з вказівки в першому рядку комбінації:

#!/bin/bash

Символи **#!** повідомляють системі про те, що наступний за ними аргумент – це програма, яка застосовується для виконання даного файлу. В даному випадку програма ***/bin/sh*** - командна оболонка, що застосовується за замовчуванням. Ця послідовність вказує на програму, яка використовується для обробки сценарію в командну оболонку bash. У загальному випадку символ «**#**» в скрипті означає коментар, що означає ігнорування рядка. Але якщо він є першим символом файлу і за ним слідує символ «**!**» та шлях до файлу (наприклад, /bin/bash, /bin/perl, /bin/ sh і т.д), командна оболонка запускає відповідний файл і передає йому його ім'я в якості аргументу.

Створення сценарію

Створити файл, який містить команди, можна допомогою будь-якого текстового редактора. У даній роботі рекомендується використовувати вбудований в ***mc*** редактор. Для створення нового файлу в *mc* використовуйте комбінацію клавіш Shift + F4. Створіть файл з ім'ям *first* з таким вмістом:

```
#!/bin/sh
# first
```

```
# Цей файл переглядає всі файли в поточному каталозі для пошуку рядка
# POSIX, а потім виводить імена знайдених файлів в стандартне виведення.
for file in *
do
  if grep -q POSIX $file
  then
    echo $ file
  fi
done
exit 0
```

Виконання скрипту відбувається по рядкам.

Перетворення сценарію у виконуваний файл

Файл сценарію можна виконати двома способами. Перший - запустити оболонку з ім'ям файлу сценарію як параметром: **$ /bin/sh first.** Цей варіант працює.

Другий спосіб (більш доцільний) - запускати сценарій, ввівши його ім'я і тим самим присвоївши йому статус інших команд Linux. Зробити це можна за допомогою команди **chmod**, змінивши режим файлу (**file mode**) і зробивши його виконуваним для всіх користувачів:

$ chmod u+x <ім'я сценарія>

$ chmod u+x first

Додані режими: для власника (User), виконання (x - eXecutable).

Аналогічну функціональність реалізує наступна команда:

chmod 744 <ім'я_файла_сценарія>

Після цього можна виконувати файл за допомогою **команди $ first**

При цьому може з'явитися повідомлення про помилку, яке говорить про те, що команда не знайдена.

Запуск сценарію на виконання з поточного каталогу проводиться за допомогою наступної команди: **./ <ім'я_файла_сценарія>**

Виправити помилку можна запровадивши з клавіатури в **командному рядку ./first** в каталозі, що містить сценарій, щоб задати командній оболонці повний відносний шлях до файлу.

Зазначення шляху, який починається з символів «**./**», дає ще одну перевагу - ви випадково не зможете виконати іншу команду з тим же ім'ям, що і у вашого файлу сценарію. Після того як ви переконаєтеся в коректній роботі вашого сценарію, можете перемістити його в більш відповідне місце, ніж поточний каталог.

Синтаксис команд для створення сценарію

Змінні та підстановка їх значень. Всі змінні у мові – текстові. Їх імена повинні починатися з літери і складатися з латинських букв, цифр і знаку підкреслення (_). Щоб скористатися значенням змінної, треба перед нею поставити символ **$**. Використання значення змінної називається *підстановкою*.

Розрізняють два класи змінних: позиційні та з іменем.

Позиційні змінні - це аргументи командних файлів, їх іменами слугують цифри:

$0 - ім'я виконуваної команди (для скрипту - це шлях, вказаний при його виклику, для функції - ім'я оболонки).
$1 - перший аргумент, другий - $2 і т.д. - змінні, які відповідають аргументам, заданим при виклику сценарію (n - десяткове число, що відповідає номеру аргументу).

Значення змінній присвоюється наступним чином: **ім'я змінної=значення**.

Наприклад, X=1, або X=a, або X="f" і т.п. Але до і після знаку «=» немає пропуску!

Прийнято називати змінні буквами верхнього регістру, наприклад:

```
#!/bin/bash
TERM=vt100
CONTER=0
```

При виконанні команд використовується *підстановка змінних*. В команду «підставляється» будь що (змінна, виведення іншої команди і т.п.). Для підстановки використовується або символ «$», або вираз, розміщений у зворотні апострофи ***(`вираз`).***

Якщо в тексті команди зустрічається символ «$», то наступний за ним текст до пробілу або кінця команди інтерпретується як ім'я змінної, значення якої підставляється в текст команди.

Наприклад:
```
$FRUIT=Яблуко
 $echo "Фрукт "$FRUIT
```
Команда *echo* виведе на екран Фрукт Яблуко.

Команди
```
$DATE=`date`
$echo $DATE
```
Виведуть системну дату на екран.

Командна оболонка дозволяє виконувати арифметичні операції. Для цього вираз, яке необхідно інтерпретувати як арифметичний, записати у подвійні круглі дужки, і перед ними ставиться знак долара.

Наприклад:
```
$foo=$(((5+3*2)-4)/2))
echo $foo
```
Для отримання значення змінної використовуються наступний синтаксис:
```
ALFA=$BETA+$GAMMA
```
Виконувану команду необхідно записати у зворотні апострофи і присвоїти змінній: **`FILES=`/bin/ls -a/home/student`**

Користувач може надати значення змінній через командну оболонку за допомогою команди **read**, якої в якості аргументу передається ім'я необхідної змінної. Наприклад:
```
$read CHOICE
Привіт !!!
$echo "Ви ввели "${CHOICE}
Привіт !!!
$echo "Ви ввели "${CHOICE}
```

Умовний оператор if | then та else

При написанні скриптів часто виникає є необхідність у перевірці (розгалуженні) будь-яких процесів. Оператори **if | then** перевіряють код завершення

переліку команд на «успішне завершення (істина)», це означає «0». Якщо це так, то виконується одна або більше команд, які записані після оператора *then*. Якщо перевірка повертає «Не успішне завершення (невірно)», це означає «1», виконується оператор *else* «інакше» як того вимагає умова. На завершення умови обов'язково закриваємо його **«fi»!**

Використання дужок

Квадратні дужки «**[**» є спеціальною вбудованою командою ***test*** , яка сприймає свої аргументи як вираз порівняння або файлову перевірку **[.....]**.

Подвійні дужки «**[[**» є розширеним варіантом від «[», є зарезервованим словом, а не командою, його bash виконує як один елемент з кодом повернення. Всередині «**[[....]]**» дозволяється виконання операторів **&&**, ||, які призводять до помилки в звичайних дужках «[....]» тим самим варіант з подвійною дужкою більш універсальний.

Круглі дужки «**((**»- є арифметичними виразами, яке так само повертають код 0. Тим самим такі вирази можна використовувати в операціях порівняння.

Список логічних операторів, які використовуються для ***if | then | else:***

«**-z**» - рядок порожній, «**-n**» - рядок не порожній,

«**=, (==)**» рядки рівні, «**!=**» - рядки нерівні,

«**-eq**» - дорівнює, «**-ne**» - не дорівнює, «**-lt, (<)**» - менше,

«**-le, (<=)**» - менше або дорівнює, «**-gt, (>)**» - більше,

«**-ge, (> =)**» - більше або дорівнює, «!» - заперечення логічного виразу,

Наприклад:

«-a, (&&)» - логічне «І», «**-o, (||)**» - логічне «АБО».

Ключі для роботи з файлами	
-d файл	True (істина), якщо файл - каталог
-e файл	True (істина), якщо файл існує.
-f файл	True (істина), якщо файл - звичайний
-r файл	True (істина), якщо файл доступний для читання
-s файл	True (істина), якщо файл ненульового розміру
-w файл	True (істина), якщо файл доступний для запису
-x файл	True (істина), якщо файл - виконуваний файл

Конструкції перевірки if|then

if умова; then
команди
fi

Наприклад:

```
#!/bin/bash
 if echo Тест; then
    echo 0
 fi

 #!/bin/bash
 if [-f $HOME/.bashrc]; then
    echo "Файл існує!"
 fi
```

де $HOME/.bashrc - шлях до файлу, echo - друкує повідомлення в консоль.

Конструкції перевірки if|then|else

if умова; then
команди 1
else
команди 2
fi

Наприклад:
```
if [-d /bin/bash]
then
   echo "/bin/bash  is a directory"
else
   echo "/bin/bash is NOT a directory"
fi
```

Умови для арифметичних виразів

Якщо оператор > використовувати всередині [[....]], він розглядається як оператор порівняння рядків, а не чисел. Тому для порівняння чисел треба використовувати оператори «<» або «>», які розташовані у круглі дужки.

Наприклад:
```
#!/bin/bash
if ((3 <6)); then
   echo Так
fi
```

Використання команди test, якою є квадратні дужки. Якщо «3» менше «6» друкуємо «Так».

```
#!/bin/bash
if [3 -lt 6]; then
  echo Так
fi
```

Можна використовувати і подвійні квадратні дужки, це розширений варіант команди test ,еквівалентом якої є «[]». Якщо «3» менше «6» друкуємо «Так».

```
#!/bin/bash
if [[3 -lt 6]]; then
    echo Так
fi
```

Використовуємо подвійні квадратні дужки, тому що застосовуємо оператор «**&&**». Якщо перший вираз 2=2 (істина), тоді виконуємо другий, й якщо він теж 2=2 (істина), друкуємо «Вірно».

```
#!/bin/bash
a="2"
b="2"
if [[2="$a" && 2="$b"]]; then
    echo Вірно
else
    echo Не вірно
fi
```

Якщо перший вираз 2=2 (істина), тоді виконуємо другий, й якщо змінна «b» не дорівнює двом (неправда), друкуємо «Не вірно».

```
#!/bin/bash
a="2"
b="3"
if [[2="$a" && 2="$b"]]; then
    echo Вірно
else
    echo Не вірно
fi
```

Вкладання кількох перевірок

Bash дозволяє вкладати в блок кілька блоків

```
#!/bin/bash
if [Умова 1]; then
   if [Умова 2]; then
      команда 1
   else
      команда 2
   fi
else
   команда 3
fi
```

Побудова багатоярусних конструкцій

Для побудови багатоярусних конструкції, коли необхідно виконувати безліч перевірок краще використовувати ***elif*** - це коротка форма запису конструкції ***else if.***

```
if [Умова 1]; then
   команда 1
   команда 2
elif [Умова 2]; then
   команда 3
   команда 4
else
   команда 5
fi
```

Цикл ПОКИ

Якщо потрібно повторити виконання послідовності команд, але заздалегідь не відомо, скільки разів слід їх виконати, застосовується цикл ***while***:

```
while <умова> do
   <оператори>
done
```

До тих пір, поки код завершення останньої команди <списка1> є 0, виконуються команди <списка2>. При заміні службового слова while на until умова виходу з циклу змінюється на протилежне.

Приклад. Програма перевірки паролів

```
#!/bin/sh
echo "Enter password"
read trythis
while ["$trythis"!="Secret"]; do
```

```
   echo "Sorry, try again"
   read trythis
done
exit 0
```

Наступні рядки - приклад виведення даного сценарію:

```
Enter password
password
Sorry, try again
secret
$
```

Цикл until

Цикл виконується, поки умова не стане істинною (true):

```
until <умова> do
   <оператори>
done
```

Запис дуже схожа на синтаксис запису циклу while, але перевірка зі зворотною умовою.

Приклад. Обчислення суми цілих чисел, що вводяться з клавіатури. Ознака закінчення введення - число 0.

```
#!/bin/sh
sum=0
read num
until [$num -eq 0]; do
   sum = $((sum+num))
   read num
done
echo $ sum
exit 0
```

Як правило, якщо потрібно виконати цикл хоча б один раз, застосовують цикл while; якщо такої необхідності немає, використовують цикл *until*.

Оператор циклу for

Оператор циклу *for* призначений для обробки в циклі низки значень, які можуть являти собою будь-яку множина рядків. Рядки можуть бути перераховані в програмі або являти собою результат виконаної командною оболонкою підстановки імен файлів. Синтаксис оператора циклу:

```
for змінна in значення
do
   оператори
done
```

Приклад. Вивести на екран всі імена файлів сценаріїв в поточному каталозі, що починаються з літери "f", та імена всіх сценаріїв, які закінчуються символами *.sh*. Це можна зробити наступним чином:

```
#!/bin/sh
for file in $ (ls f*.sh); do
```

```
  echo $ file
done
echo $ file
exit 0
```

Функції

Синтаксис оголошення функції:

```
    function <ім'я> ()
  {
    <список>;
  }
```

Функція визначається іменем <ім'я>. Тіло функції - <список> записується між дужками { }.

Приклади обчислення факторіала.

Рекурсивне визначення факторіала:

```
function factorial
{
    typeset -i n=$1
    if [$n=0]; then
        echo 1
        return
    fi
    echo $((n*$(factorial $((n-1)))))
}
for i in {0..16}
do
   echo "$i!=$(factorial $i)"
done
```

Ітеративне визначення факторіала:

```
f=1
for ((n=1; $n<=17; $((n++)) );
do
  echo "$((n-1))!=$f"
  f=$((f*n))
done
```

Відладка сценаріїв

При виявленні помилки при виконанні сценарію командна оболонка виводить на екран номер рядка, що містить помилку. Якщо помилку відразу не видно, потрібно додати кілька додаткових команд *echo* для виведення значень змінних, протестувати фрагменти програмного коду, вводячи їх в командній оболонці в інтерактивному режимі. Основний спосіб відстеження помилок, які найбільш складно виявляються - використання опцій відладки командної оболонки.

Опції відладки командного рядка:

Опція	Призначення
sh -n <сценарій>	Тільки перевіряє синтаксичні помилки

sh -v <сценарій> Виводить на екран команди перед їх виконанням
sh -x <сценарій> Виводить на екран команди після обробки командного рядка
sh -u <сценарій> Видає повідомлення про помилку при використанні невизначеної змінної

Завдання:

1. Ознайомитися з теоретичними матеріалом по лабораторній роботі.
2. Опанувати команди, які використовують при написанні сценарію.
3. Підготувати звіт для викладача про виконання лабораторної роботи і представити його.

Хід виконання роботи

Розробити скрипт, який виконує зазначені дії згідно варіанту по списку журналу групи. Створити файл з назвою власного прізвища, записати до нього ПІБ студента, групу навчання, за бажанням додаткову інформацію щодо ваших уподобань. Створити скрипт, який виводить ПІБ студента, групу навчання, коментарі щодо майбутніх дій, необхідно робити затримку після видачі результатів на екран. Скрипт повинен містити функції, які виконують зазначені дії, а також додатково записати дію відповідно до варіанту.

Варіанти:

1. Сценарій перейменування власного файлу.
2. Вивести коротку довідку щодо команди *read*.
3. Відобразити список процесів.
4. Вивести дерево вашого домашнього каталогу.
5. Знайти текстовий рядок у вашому файлі.
6. Вивести інформацію про систему.
7. Змінити поточний каталог.
8. Відобразити режим доступу до вашого файлу.
9. Змінити режим доступу до власного файлу групі користувачів для встановлення дозволу тільки на читання.
10. Виконати копіювання власного файлу зі збереженням атрибутів.
11. Створити каталог та виконати його копіювання.
12. Перемістити власний файл у новий каталог.
13. Створити каталог, вивести його назву на екран та видалити.
14. Вивести на екран вміст вашого файлу.
15. Скопіювати власний файл, видалити у ньому назву групи.
16. Вивести версію системи.
17. В каталозі віднайти виконуваний файл.
18. Вивести імена файлів, які починаються на буку «f».
19. Вивести на екран дату створення файлу (власного або будь-якого іншого).
20. Вивести довідку про команду *printf*.
21. Створити архів для вашого файлу або каталогу, де він зберігається.
22. Вивести список інстальованих програм менеджера пакетів ***dpkg.***
23. Вивести довідку про менеджер пакету ***aptitude.***
24. Відобразити стан поточної конфігурації мережі.

25. Вивести інформацію про поточного користувача.
26. Вивести інформацію про поточний каталог.
27. Створити новий файл, записати до нього вміст вашого файлу.
28. Створити новий каталог, скопіювати до нього ваш файл.
29. Ввести з командного рядка значення двох змінних, обчислити їх суму, записати у ваш файл.
30. Ввести з командного рядка вашу дату народження і записати її до вашого файлу.

Підготувати звіт

1. Описати хід виконання поставлених завдань, надаючи знімок екрану (screenshot).
2. Висновки по роботі.

Контрольні питання

1. Що таке скрипт?
2. Як перетворити скрипт у виконуваний файл?
3. Що означає символ «\», введений в командному рядку перед натисканням Enter?
4. Для чого використовується команда read?
5. Як працює умовний оператор if-fi?
6. Які конструкції застосовуються для організації циклу?

Література

1. Роббинс А. Bash. Карманный справочник системного администратора. СПб.: ООО "Альфа-книга". 2017. 152 с.
2. Колисниченко Д. Н. Руководство по командам и shell-программированию в Linux. СПб.: БХВ-Петербург. 2011. 288 с.
3. Тейлор Д., Перри Б. Сценарии командной оболочки. Linux, OS X и Unix. – СПб: Питер, 2017. – 448 с.