МЕТОДИЧНА РОЗРОБКА
для проведення лабораторного заняття з дисципліни
«Клієнт-серверне програмування»

Лабораторна робота 4

Робота в командній оболонці bash, середовище оточення.

Мета роботи: Знайомство із текстовою операційною оболонкою bash, робота зі змінними оточення.

1. Теоретичні відомості

Операційна система дозволяє записати необхідну послідовність команд в спеціальний файл, який називається сценарієм оболонки. Далі цей сценарій можна виконувати подібно виконанню звичайної команди Linux, набравши ім'я файла. Такий принцип організації файлів існує в MS DOS – це так звані командні файли.

Створювати сценарії надають спеціальні програми – оболонки, які виступають посередником між користувачем та операційною системою.

Основними функціями командних оболонок є:

• організація діалогу з користувачем (введення команд);
• виконання внутрішніх команд;
• запуск зовнішніх програм;
• виконання командних файлів.

При вході в командний інтерпретатор відкривається вікно, в якому відображається: root – ім'я облікового запису користувача, ksp – ім'я комп'ютера, «~» - каталог виконання команди (домашній каталог).

```
[root@ksp ~]#
```

Команди терміналу, як правило, складаються з назви програми, ключа і значення. В загальному вигляді виглядають так:

назва_ програми [-ключ] [значення].

назва_програми - це ім'я виконуваного файлу з каталогів, записаних у змінну $PATH (/bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin та ін.);

[ключ] - опції програми, які може приймати виконувана програма;

[значення] - даний параметр може приймати в якості аргументу цифри, текст, спеціальні символи і навіть змінні.

Оболонка має свої налаштування та забезпечує доступ до ресурсів системи, ористувач має свої налаштування при роботі з системою та своїми додатками. Для цього використовується середовище оточення.

Середовище оточення - це область, яка містить визначальні властивості системи у вигляді змінних, і це середовище оболонка будує при кожному запуску сесії.

2. Перегляд історії команд

Усі команди, які набрані користувачем, bash запам’ятовує і дозволяє звертатися до них у подальшому. По стрільці уверх (можна використовувати і «^P», previous), список поданих команд «провертається» від останньої до першої, а по стрільці униз («^N», next) – назад.

Відповідна команда відображається у командному рядку, як така яку тільки що набрали, її можна редагувати і виконати.

Якщо необхідно добути з історії якусь давню команду, простіше здійснювати пошук за допомогою команди «^R» (reverse search). При цьому виводиться підказка спеціального виду («(reverse -i-search)»), підрядок пошуку (оточений символами `n’) і остання команда з історії, в якій ця підказка присутня

```
[root@ksp ~]#
(reverse-i-search)`n': nano .bash_profile
```

Набирається ^R, потім набирається початок команди в історії, яку необхідно знайти.

Історія команд між сеансами роботи користувача зберігається у файлі .bash_history, який знаходиться у домашньому каталозі користувача. Робиться це у момент завершення роботи оболонки: історія, яка накопичена за час роботи, дописується у кінець цього файлу. При наступному запуску bash зчитує .bash_history повністю. Історія зберігається не вічно, кількість команд, які запам’ятовуються у .bash_history, обмежена (зазвичай, 500 командами, але це можна переналаштувати).

На практиці для перегляду історії команд треба: **cat .bash_history**.

```
[root@ksp ~]# cat .bash_history
who
pwd
ls
cd /
ls
ls -al
whoami
last
logout
```

Утиліти

Ніяка програма – хай навіть і оболонка – не може самостійно розбиратися у всіх задокументованих командах. Для цього є **утиліти**, тобто, корисні програми. Командний інтерпретатор не зобов’язаний вміти виконувати усе, що вводить користувач. Йому достатньо розібрати командний рядок, виділити з нього команду і параметри, а потім запустити утиліту – програму, ім’я якої співпадає з іменем команди.

В дійсності, особистих команд у командному інтерпретаторі не багато. В основному, це – оператори мови програмування і інші засоби управління самим

інтерпретатором. Усі знайомі нам команди, навіть echo, існують в Linux вигляді окремих утиліт. shell займається тільки тим, що підготовлює набір параметрів у командному рядку (наприклад, розкриває шаблони), запускає програми і обробляє результати їх роботи.

В bash типи команд можна визначити за допомогою команди **type.** Особисті команди bash називаються builtin (вбудовані команди), а для утиліт виводиться шлях, який вміщує назву каталогу, в якому лежить файл з відповідною програмою, і ім'я цієї програми. Деякі самі необхідні команди вбудовані у bash, навіть не дивлячись на те, що вони є у вигляді утиліт (наприклад. echo).

```
[root@ksp ~]# type cat
cat is /usr/bin/cat
[root@ksp ~]# type echo
echo is a shell builtin
[root@ksp ~]#
```

3. **Змінні оточення.**

Кожний процес, який запускається, система наділяє деяким інформаційним простором, який цей процес має право змінювати як йому заманеться. Правила користування цим простором прості: в ньому можна задавати іменовані сховища даних (***змінні оточення***), у якому записувати яку завгодно інформацію (присвоювати значення змінній оточення), а подалі, цю інформацію зчитувати (підставляти значення змінної).

Змінні оточення дозволяють простим і надійним способом передавати налаштування відразу для безлічі додатків.

Таким чином, змінні оточення в Linux - це спеціальні змінні, які визначені оболонкою і використовуються програмами під час виконання. Вони можуть визначатися системою і користувачем. Системні змінні оточення Linux визначаються системою і використовуються програмами системного рівня.

Системні змінні оточення (змінні середовища)

Ці змінні доступні у всій системі та для всіх користувачів. Вони завантажуються при старті системи з системних файлів конфігурації: **/etc/environment, /etc/profile, /etc/profile.d/, ~/.bash_profile, /etc/bash.bashrc**:

- **/etc/environment** – системний файл конфігурації, який означає, що він використовується всіма користувачами. Він належить root, тому необхідно бути адміністратором і використовувати його sudo для зміни;

- **~/.bash_profile** є одним із сценаріїв ініціалізації особистої оболонки вашого власного користувача. Кожний користувач має один такий файл і може його редагувати.

- **/etc/profile та /etc/profile.d/*.sh** є сценаріями глобальної ініціалізації, які є еквівалентними **~/.bash_profile** для кожного користувача. Глобальні сценарії виконуються раніше користувацьких сценаріїв; і main /etc/profile виконує усі *.sh сценарії з директорії /etc/profile.d/ безпосередньо перед виходом;

- **/etc/bash.bashrc** - це загальносистемна версія файлу ~/.bashrc. ОС Ubuntu налаштована за замовчуванням для виконання цього файлу, коли користувач вводить оболонку або середовище робочого столу.

Змінні середовища - це змінні, які були визначені для поточної оболонки і успадковуються усіма дочірніми оболонками або процесами. Змінні середовища використовуються для передачі інформації процесам, які запущені з оболонки.

Змінні оболонки - це змінні, які містяться виключно в оболонці, в якій вони були встановлені або визначені. Вони часто використовуються для відстеження поточних даних (наприклад, поточного робочого каталогу).

Зазвичай такі змінні позначаються за допомогою великих літер. Це допомагає користувачам розрізняти змінні середовища в інших контекстах.

У сценаріях зазвичай використовуються оголошення виду:

ім'я_змінної="значення змінної".

Для отримання значення змінної необхідно перед її ім'ям поставити символ долара **$.**

Змінна середовища	Значення
USER	поточний користувач
PWD	поточна директорія
HOSTNAME	ім’я хоста(комп’ютера)
PS1	це підказка, яку shell показує, коли виводить командний рядок
OLDPWD	попередня робоча директорія. Використовується оболонкою для того, щоб повернутися в попередній каталог при виконанні команди cd -
HOME	домашня директорія поточного користувача
SHELL	шлях до оболонки поточного користувача (наприклад, bash або zsh).
EDITOR	заданий за замовчуванням редактор. Цей редактор викликається у відповідь на команду edit.
LOGNAME	ім’я користувача, яке використовується для входу в систему
PATH	шляхи до каталогів, у яких здійснюватиметься пошук викликаних команд. При виконанні команди система проходитиме по даним каталогам у вказаному порядку і вибере перший з них, в якому знаходитиметься виконуваний файл шуканої команди.
LANG	поточні налаштування мови та кодувань
TERM	тип поточного емулятора терміналу
MAIL	місце зберігання пошти поточного користувача

LS_COLORS	задає кольори, що використовуються для виділення об’єктів (наприклад, різні типи файлів у виводі команди ls будуть виділені різними кольорами).

Виведення змінних оболонки і середовища

Якщо ви виконаєте команду **printenv** або **env** без будь-яких аргументів, то вони покажуть список всіх змінних середовища:

```
[root@ksp ~]# printenv
XDG_SESSION_ID=3
HOSTNAME=ksp
SELINUX_ROLE_REQUESTED=
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
SSH_CLIENT=192.168.65.1 58441 22
SELINUX_USE_CURRENT_RANGE=
SSH_TTY=/dev/pts/0
USER=root
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=
```

```
MAIL=/var/spool/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
PWD=/root
LANG=en_US.UTF-8
SELINUX_LEVEL_REQUESTED=
HISTCONTROL=ignoredups
SHLVL=1
HOME=/root
LOGNAME=root
SSH_CONNECTION=192.168.65.1 58441 192.168.65.128 22
LESSOPEN=||/usr/bin/lesspipe.sh %s
XDG_RUNTIME_DIR=/run/user/0
 =/usr/bin/printenv
```

Приклад 1. Використання команди **printenv** (виводить значення окремих змінних):

```
[root@ksp ~]# printenv SHELL
/bin/bash
[root@ksp ~]# printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@ksp ~]#
```

Приклад 2. Команда **env** дозволяє змінювати середовище, в якій запущені програми, передаючи набір значень змінних в команду:

env STR1="hello"

```
[root@ksp ~]# env STR1="hello"
XDG_SESSION_ID=3
HOSTNAME=ksp
_=/usr/bin/env
STR1=hello
[root@ksp ~]#
```

Приклад 3. Використання команди **set.** Якщо ви хочете отримати список всіх змінних, включаючи змінні (та функції) оболонки, то можете використати команду **set**:

```
[root@ksp ~]# set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extquote:force_fignore:histappend:h
ostcomplete:interactive_comments:login_shell:progcomp:promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
XDG_SESSION_ID=3
_=set
colors=/root/.dircolors
[root@ksp ~]#
```

Приклад 4. Щоб знайти всі змінні, що містять вказаний рядок, використовуйте команду **grep**:

printenv | grep [ІМ'Я_ЗМІННОЇ]

```
[root@ksp ~]# printenv | grep HOME
HOME=/root
[root@ksp ~]# printenv | grep PATH
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@ksp ~]# printenv | grep USER
USER=root
[root@ksp ~]#
```

Приклад 5. Використання команди ***echo*** для перегляду значення змінної.

```
[root@ksp ~]# echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/root/bin
[root@ksp ~]# echo $UID
0
[root@ksp ~]# echo $HOSTNAME
ksp
[root@ksp ~]# echo $PS1
[\u@\h \W]\$
```

Змінні користувача

Змінні користувача визначаються самим користувачем під час написання сценарію оболонки. В своєму власному сценарії користувач може довільно використовувати та змінювати їх.

Ім'я змінної bash має обов'язково починатися з літери. Далі можна використати і цифри. У bash змінні не мають типу, тому оголошувати змінну попередньо не потрібно, а відразу можна надавати значення. Система самостійно “здогадується” про тип змінної за тим значенням, яке їй присвоюється.

Для присвоєння змінній користувача цілого значення або одного текстового слова досить записати оператор присвоєння в загальноприйнятій формі.

Приклад 6. Присвоєння значень змінним.

S1=5

S2=HELLO

S3=process

```
[root@ksp ~]# S1=5 S2=HELLO S3=process
[root@ksp ~]# echo $S1 $S2 $S3
5 HELLO process
```

Характерною особливістю сценаріїв є те, що для доступу до значення змінної, перед її іменем ставиться знак долара ($).

Приклад 7. Використання одинарних та подвійних лапок

Якщо текстовий рядок складається з кількох слів, тобто містить пропуски, тоді використовуються одинарні лапки.

```
[root@ksp ~]# VAR1='long text string'
[root@ksp ~]# echo $VAR1
long text string
[root@ksp ~]#
```

При використанні подвійниих лапок забезпечується підстановка значень змінних всередині рядків.

```
[root@ksp ~]# VAR2="Value of VAR1 is $VAR1"
[root@ksp ~]# echo $VAR2
Value of VAR1 is long text string
```

Приклад 8. Програмування арифметичних виразів.

Для програмування арифметичних виразів можна застосувати такі знаки операцій:

+ сума,
- різниця,
* множення,
/ ділення,
% ділення з остачею.

Оператор **let**.

```
[root@ksp ~]# let y=5+6/3; echo $y
7
```

Оператор **expr.**

```
[root@ksp ~]# expr y=5+6/3; echo $y
y=5+6/3
7
```

```
[root@ksp ~]# expr 'y=$a+$b/$c'; a=5 b=6 c=3; echo $y
y=$a+$b/$c
7
[root@ksp ~]# expr y=$a+$b/$c; a=5 b=6 c=3; echo $y
y=5+6/3
7
```

Приклад 9. Порівняння чисел

a **–eq** b визначення рівності чисел a і b;
a **–ne** b визначення нерівності чисел a і b;
a **–gt** b визначення того, чи число a більше числа b ;
a **–ge** b визначення того, чи число a більше або дорівнює числу b;

a **–lt** b визначення того, чи число a менше числа b;
a **–le** b визначення того, чи число a менше або дорівнює числу b.

```
[root@ksp ~]# [ $a -gt $b ]; echo $?
1
[root@ksp ~]# [ $a -lt $b ]; echo $?
0
```

Приклад 10. Порівняння рядків.
str1 = str2 визначення рівності рядків str1 і str2;
str1 != str2 визначення нерівності рядків str1 і str2;
-n перевірка ненульової довжини рядка;
-z перевірка нульової довжини рядка.

```
[root@ksp ~]#  [ -n hello ]; echo $?
0
[root@ksp ~]#  [ -z ]; echo $?
0
[root@ksp ~]#  [ -z hello ]; echo $?
1
```

Приклад 11. Файлові операції порівняння
Такі операції можуть використовуватись для перевірки файлів:
-d перевірка того, чи є файл каталогом;
-f перевірка того, чи є файл звичайним файлом;
-r перевірка того, чи є право доступу для читання файла;
-w перевірка того, чи є право доступу для запису у файл;
-x перевірка того, чи є право доступу для виконання файла;
-s перевірка того, чи є файл з ненульовою довжиною.

```
drwxr-xr-x. 2 root root  110 Mar  9 14:42 cpp
[root@ksp ~]# [ -f cpp ]; echo $?
1
[root@ksp ~]# [ -d cpp ]; echo $?
0
[root@ksp ~]# [ -x cpp ]; echo $?
0
```

4. **Завдання на виконання**.

1. Ознайомитися з теоретичними матеріалом по лабораторній роботі.
2. Виконати перегляд історії команд, добути з історії якусь давню команду.
3. За допомогою команди **type** визначити тип чотирьох довільних команд.
4. За допомогою команди **printenv** вивести значення змінних середовища на вибір.
5. Використовуючи команду **grep,** визначити значення декількох змінних на вибір (зробити скрін).
6. Присвоїти значення змінним:
S1= HELLO S2= FIRSTNAME S3= SECONDNAME (*власні імена*).

Вивести на екран.

7. Використовуючи одинарні та подвійні лапки, вивести на монітор текст із використанням змінної S2.

8.Виконайте програмування довільного арифметичного виразу за допомогою оператора **let**.

9. Виконайте програмування закону Ома за допомогою оператора **expr,** задайте довільні значення напруги та опору.

10. Задайте два числа та виконайте всі можливі порівняння.

11. Використовуючи значення змінних S2 та S3, виконайте порівняння рядків.

12. Використовуючи наявні файли та каталоги, виконайте їх перевірку.

Підготувати звіт про виконання лабораторної роботи та зробіть висновки.

5. Результати виконання роботи та оформлення звіту

Зміст звіту:

1) титульний аркуш;
2) короткі теоретичні відомості;
3) скріни з виконаними завданнями та висновки.

Контрольні питання

1. Команди для перегляду історії команд.
2. Змінні середовища.
3. Присвоєння значень змінним користувача.
4. Призначення вбудованих змінних.
5. Команди програмування арифметичних виразів.
6. Команди порівняння.

Методичну розробку склав
доцент кафедри інформаційних технологій
та систем електронних комунікацій Юрій БОРЗОВ