**Work-case 2**

Ланковський Ерік

## Практичне завдання “Work-case 2”

**Створення нової віртуальної машини:**

**VBoxManage createvm:** Ця команда використовується для створення нової віртуальної машини.

**--name <name>:** Задає ім'я віртуальної машини.

**--ostype <OS>:** Визначає тип операційної системи, яку буде встановлено на віртуальній машині.

**--register:** Реєструє віртуальну машину в VirtualBox.

**VBoxManage createvm --name "MyVM" --ostype "Linux" --register**

**Вибір/додавання доступного для віртуальної машини обладнання:**

**VBoxManage modifyvm:** Ця команда використовується для зміни налаштувань віртуальної машини, включаючи додавання обладнання.

**--memory <size>:** Встановлює об'єм оперативної пам'яті для віртуальної машини.

**--cpus <count>:** Встановлює кількість процесорів для віртуальної машини.

**--hda <path>:** Додає віртуальний жорсткий диск до віртуальної машини.

**--nic<number> <type>:** Додає мережевий адаптер до віртуальної машини.

**VBoxManage modifyvm "MyVM" --memory 2048 --cpus 2 --hda "path/to/disk.vdi" --nic1 bridged**

**Налаштування мережі та підключення до точок Wi-Fi:**

**VBoxManage modifyvm:** Як зазначено вище, ця команда також використовується для налаштування мережі.

**--nic<number> <type>:** Визначає тип мережевого підключення (наприклад, bridged, NAT, host-only).

**--bridgeadapter<number> <adapter>:** Для bridged-з'єднання, вказує мережевий адаптер, який буде використано для підключення до мережі.

**VBoxManage modifyvm "MyVM" --nic1 bridged --bridgeadapter1 "Wi-Fi Adapter"**

**Можливість роботи з зовнішніми носіями (flash-пам'ять):**

**--vmname <name>:** Вказує ім'я віртуальної машини.

**--storagectl <name>:** Вказує контролер зберігання.

**--port <port>:** Вказує порт підключення.

**--device <device>:** Вказує номер пристрою.

**--type <type>:** Вказує тип носія.

**--medium <path>:** Вказує шлях до файлу.

**VBoxManage storageattach "MyVM" --storagectl "IDE Controller" --port 1 --device 0 --type hdd --medium /dev/sdb**

Порівняння FluxBox та Gnome.

|  |  |  |
| --- | --- | --- |
| Feature | Fluxbox | GNOME |
| Тип | Window Manager | Desktop Environment |
| Ресурси | Дуже легкий, споживає мало ресурсів | Важкий, споживає багато ресурсів |
| Швидкість | Швидкий та чуйний | Може бути повільним на слабкому обладнанні |
| Налаштування | Висока гнучкість налаштування через текстові файли конфігурації | Менш гнучкий, налаштування через графічний інтерфейс |
| Зовнішній вигляд | Мінімалістичний, простий | Сучасний, з різноманітними візуальними ефектами |
| Функціональність | Базова функціональність, фокус на керуванні вікнами | Повноцінне середовище з багатьма вбудованими програмами |
| Простота використання | Потребує певних знань для налаштування | Інтуїтивно зрозумілий для новачків |
| Встановлення | Зазвичай встановлюється окремо | Часто встановлюється за замовчуванням в дистрибутивах Linux |
| Підтримка | Менша спільнота, менше документації | Велика спільнота, багато документації та підтримки |
| Програми | Не включає в себе багато програм, користувач сам обирає потрібні | Має великий набір вбудованих програм для різних потреб |
| Оновлення | Оновлюється рідше | Оновлюється часто, з новими функціями та покращеннями |
| Висновок | Ідеально підходить для користувачів, які цінують швидкість та гнучкість | Підходить для користувачів, які хочуть сучасне та функціональне середовище |

**Відповіді на контрольні запитання:**

**Типи команд в оболонці Bash**

В оболонці Bash існує кілька типів команд:

* **Вбудовані команди (Built-in commands)**: Це команди, які вбудовані безпосередньо в оболонку Bash і виконуються нею безпосередньо. Вони зазвичай швидші за зовнішні команди. Приклади: cd, echo, export, set.
* **Зовнішні команди (External commands)**: Це команди, які знаходяться у вигляді окремих виконуваних файлів в файловій системі. Bash запускає ці файли для виконання команди. Приклади: ls, grep, find, man.
* **Псевдоніми (Aliases)**: Це скорочення для інших команд або послідовностей команд. Вони дозволяють користувачам створювати власні зручні імена для часто використовуваних команд. Приклад: alias la='ls -la'.
* **Функції (Functions)**: Це блоки коду, які можуть бути визначені користувачем для виконання певних завдань. Функції можуть приймати аргументи та повертати значення.

**Змінні оточення**

Змінні оточення - це динамічні іменовані значення, які впливають на поведінку програм та процесів в операційній системі. Вони зберігаються в пам'яті та доступні для всіх запущених програм.

**Типи змінних оточення**

* **Системні змінні**: Встановлюються операційною системою та впливають на роботу всієї системи. Приклади: PATH, HOME, USER, TERM.
* **Користувацькі змінні**: Встановлюються користувачем та впливають на роботу його програм та процесів.

**Перегляд змінних оточення**

Для перегляду всіх змінних оточення в терміналі можна скористатися командою env або printenv.

**Змінна $PS1**

$PS1 - це спеціальна змінна оточення, яка визначає формат рядка запрошення в терміналі Bash. Вона може містити різні символи та escape-послідовності, які відображають інформацію про поточний стан системи, наприклад, ім'я користувача, ім'я хоста, поточний каталог тощо.

**Перегляд вмісту $PS1**

Для перегляду вмісту змінної $PS1 можна скористатися командою echo $PS1.

**Зміна значення $PS1**

Для зміни значення $PS1 можна скористатися командою export PS1="нове значення". Наприклад, export PS1="\[\e[32m\]\u@\h \w\$ \[\e[0m\]". Після цього рядок запрошення в терміналі зміниться відповідно до нового значення.

**Зміна значення $PS1 за замовчуванням**

Для змінення значення $PS1 не на поточний сеанс, а за замовчуванням, потрібно змінити значення цієї змінної в файлі конфігурації Bash. Зазвичай це файл ~/.bashrc або ~/.bash\_profile.

**Використання лапок в оболонці Bash**

В оболонці Bash використовуються різні типи лапок для різних цілей:

* **Одинарні лапки (' ')**: Використовуються для захисту тексту від інтерпретації оболонкою. Всередині одинарних лапок всі символи сприймаються буквально.
* **Подвійні лапки (" ")**: Використовуються для захисту тексту від інтерпретації оболонкою, за винятком деяких символів, таких як $, `` та`. Всередині подвійних лапок можна використовувати змінні та команди.
* **Зворотні лапки (`` `)**: Використовуються для виконання команд та підстановки їх результату в текст.

**Інструкції керування**

Інструкції керування використовуються для організації виконання команд в оболонці Bash. Вони дозволяють створювати цикли, розгалуження та інші складні конструкції.

**Види інструкцій керування**

* **Умовні оператори**: if, elif, else. Дозволяють виконувати команди залежно від виконання певної умови.
* **Цикли**: for, while, until. Дозволяють повторювати виконання блоку команд кілька разів.
* **Оператори перенаправлення**: > (перенаправлення виводу), < (перенаправлення введення), | (конвеєр). Дозволяють змінювати стандартні потоки введення та виведення команд.

**Символи $ та # в кінці рядка запрошення**

* **$**: Означає, що користувач має звичайні права доступу.
* **#**: Означає, що користувач має права суперкористувача (root).

**Команди whereis та locate**

* **whereis**: Шукає виконувані файли, вихідні тексти та man-сторінки для заданої команди.
* **locate**: Шукає файли та каталоги за їх іменем або частиною імені в базі даних.

**Відмінність між whereis та locate**

* whereis шукає інформацію про команди в конкретних каталогах, тоді як locate шукає файли по всій файловій системі за допомогою бази даних.
* whereis надає більш точну інформацію про місцезнаходження файлів, тоді як locate може знаходити файли, які вже не існують, оскільки його база даних може бути неактуальною.

**Висновки:**

Виконавши цю роботу, я отримав глибше розуміння принципів роботи з командним інтерфейсом **Bash** та його можливостей. Для мене було особливо цікаво дізнатися про:

* Різноманітність команд: Існує безліч команд для виконання різноманітних завдань, від навігації файловою системою до управління процесами та налаштування системи.
* Змінні оточення: Змінні оточення відіграють ключову роль у налаштуванні поведінки програм та процесів. Особливо цікавим було дізнатися про змінну $PS1 та її вплив на вигляд рядка запрошення.
* Гнучкість налаштування: Bash надає широкі можливості для налаштування під свої потреби, включаючи створення псевдонімів, функцій та зміну значення змінних оточення.
* Інструкції керування: Інструкції керування дозволяють створювати складні конструкції для автоматизації виконання завдань та управління потоком виконання програм.
* Робота з файлами: Команди whereis та locate є потужними інструментами для пошуку файлів та інформації про команди.

Ця робота допомогла мені не тільки ознайомитися з основними командами та поняттями Bash, але й зрозуміти, як використовувати їх для вирішення практичних завдань. Я також навчився користуватися документацією та шукати необхідну інформацію в інтернеті.

В цілому, я вважаю, що Bash є незамінним інструментом для кожного, хто хоче ефективно працювати з операційною системою Linux. Розуміння принципів роботи з Bash відкриває широкі можливості для автоматизації завдань, налаштування системи та управління нею.

Я впевнений, що отримані знання та навички будуть корисними мені в подальшому навчанні та роботі.