“Київський фаховий коледж зв’язку”

Циклова комісія Комп’ютерної інженерії

ЗВІТ ПО ВИКОНАННЮ

ЛАБОРАТОРНОЇ РОБОТИ №3

з дисципліни: «Операційні системи»

Тема: «Знайомство з базовими командами CLI-режиму в Linux»

Виконала

студентка

групи БІКС-13

Лось Злата Володимирівна

Перевірив викладач

Сушанова В.С.

Київ 2024

**Мета роботи:**

1. Знайомство з базовими командами CLI-режиму в Linux.

2. Знайомство з базовими текстовими командами в термінальному режимі роботи в різних ОС.

**Матеріальне забезпечення занять:**

1. ЕОМ типу IBM PC.

2. ОС сімейства Windows та віртуальна машина Virtual Box (Oracle).

3. ОС GNU/Linux (будь-який дистрибутив).

4. Сайт мережевої академії Cisco netacad.com та його онлайн курси по Linux.

**Завдання для попередньої підготовки.**

1. \*Прочитайте короткі теоретичні відомості до лабораторної роботи та зробіть невеликий словник базових англійських термінів з питань призначення команд та їх параметрів.

**Shell (або «шелл», «командна оболонка») —** це не тільки командний інтерпретатор, який забезпечує інтерфейс взаємодії між користувачем та ядром операційної системи, але й своєрідна мова програмування, в якій присутні такі конструкції, як оператори умовного розгалуження, цикли, змінні та багато іншого.

**Команда** — це назва програми, яку користувач вводить у терміналі для виконання певного завдання. Команди можуть мати додаткові дані та параметри/опції. Команди в Linux/Unix чутливі до регістру. Виконання введеної команди здійснюється лише після натискання клавіші Enter.

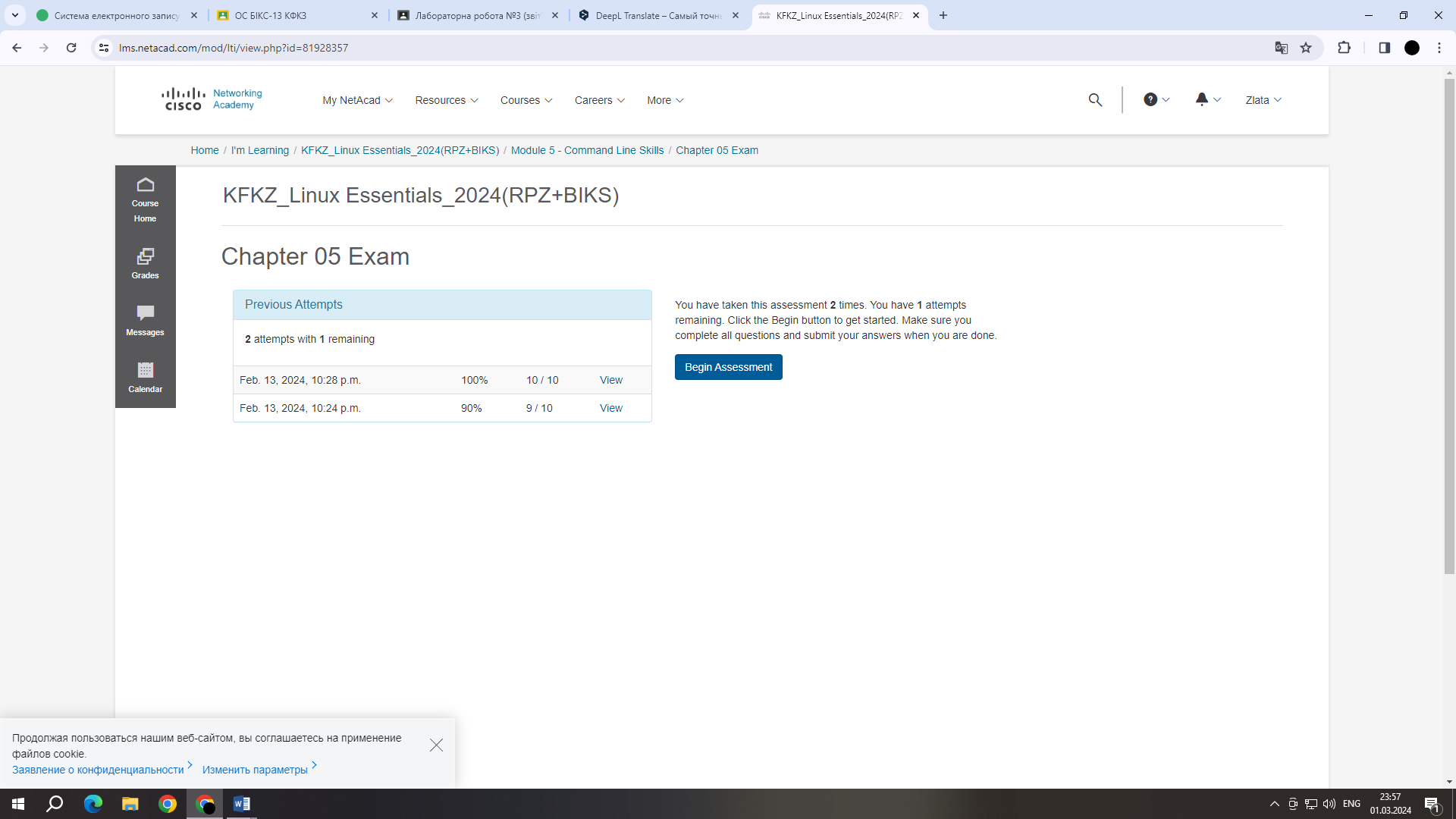
**Аргументи командного рядка** - це параметри, які передаються скрипту під час його виконання в оболонці bash. У Linux вони також відомі як позиційні параметри.

2. Вивчіть матеріали онлайн-курсу академії Cisco “NDG Linux Essentials”:

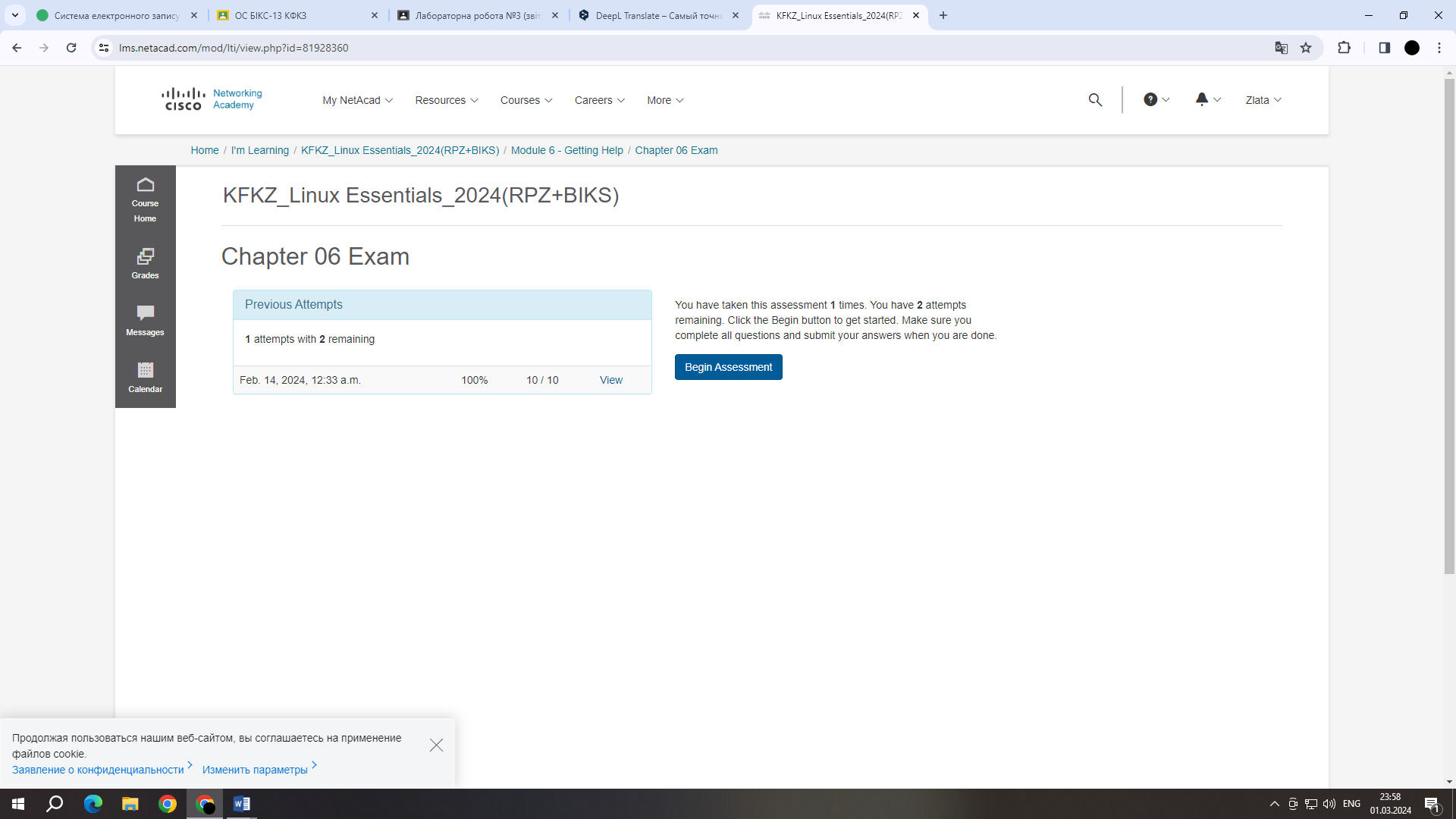
* Chapter 5 - Command Line Skills
* Chapter 6 - Getting Help

3. Пройдіть тестування у курсі NDG Linux Essentials за такими темами:

- Chapter 05 Exam



- Chapter 06 Exam



4. \*Дайте визначення наступним поняттям:

- **Командний інтерпретатор** - програма, яка забезпечує взаємодію користувача з операційною системою.

- **Оболонка** - це програма, яка забезпечує текстове спілкування між операційною системою та користувачем. Зазвичай це програма в текстовому режимі, яка зчитує введені користувачем дані та інтерпретує їх як команди системі.

- **Команда** - це назва програми, яку користувач вводить у терміналі для виконання певного завдання. Команди можуть мати додаткові дані та параметри/опції.

**5. \*\*Дайте відповіді на наступні питання:**

**- Яку базову інформацію надає рядок запрошення prompt?**

Рядок запрошення prompt надає таку базову інформацію:

* поточний користувач.
* назва хоста (комп'ютера).
* поточний робочий каталог.
* символ запрошення: зазвичай `$` для звичайних користувачів і `#` для користувачів з правами адміністратора.

**- Для чого команді потрібні параметри та аргументи?**

Параметри та аргументи в командах використовуються для того, щоб надати командам додаткову інформацію або інструкції, які вони повинні виконати. Вони дозволяють користувачам здійснювати більш гнучке та точне керування поведінкою програм, скриптів або команд у командному рядку. Ось декілька ключових аспектів, для чого потрібні параметри та аргументи:

* Кастомізація дій: Дозволяють користувачу вказати, яку саме дію або набір дій виконати. Наприклад, при копіюванні файлів можна вказати, з якого місця копіювати та куди копіювати.
* Гнучкість: Надають можливість використовувати одну й ту ж команду для різних сценаріїв, змінюючи її поведінку за допомогою різних параметрів та аргументів.
* Ефективність: Дозволяють виконувати складні завдання однією командою, замість використання довгої послідовності команд.
* Точність: Параметри та аргументи допомагають визначити точні критерії або об'єкти для операцій, зменшуючи ймовірність помилок.
* Автоматизація: У скриптах або програмах вони дозволяють передавати специфічні дані або налаштування, що робить можливим автоматизацію задач без необхідності вручну змінювати код.
* Універсальність: Параметри та аргументи дозволяють одній команді або програмі служити багатьом користувачам з різними потребами, надаючи кожному можливість налаштувати її під свої завдання.

**- Яке призначення команд ls, які параметри та аргументи вона може мати? Наведіть 3 приклади.**

Команда ls використовується для відображення списку файлів та директорій у поточному каталозі або вказаному каталозі. Ця команда дозволяє користувачам переглядати вміст директорій, включаючи інформацію про файли та папки, таку як права доступу, кількість посилань, власника, групу, розмір та час останньої модифікації.

Параметри та аргументи команди ls:

Команда ‘ls’ має численні параметри та аргументи, що дозволяють змінювати її поведінку або формат виведення. Ось деякі з найбільш використовуваних параметрів:

* ‘-l’: Виводить детальний список файлів та директорій, включаючи права доступу, кількість посилань, власника, групу, розмір і час останньої модифікації.
* ‘-a’: Виводить усі файли та директорії, включаючи приховані (те, що починається з крапки ‘.’, наприклад, ‘.bashrc’).
* ‘-h’: Використовується з ‘-l’ для відображення розміру файлів у зручному для читання форматі (наприклад, КБ, МБ).

**Приклади використання команди ls:**

* Простий перегляд вмісту каталогу: ‘ls’

Виводить список файлів та папок у поточному каталозі.

* Детальний перегляд вмісту каталогу: ‘ls –l’

Показує детальну інформацію про файли та папки, таку як права доступу, кількість посилань, власника, розмір, і час останньої модифікації.

* Перегляд усіх файлів, включаючи приховані, з детальною інформацією та в зручному форматі: ‘ls –lah’

Комбінує параметри ‘-l’ для детального перегляду, ‘-a’ для включення прихованих файлів, і ‘-h’ для відображення розміру файлів у зручному для читання форматі.

**- Яким чином можна використати історію команд, які переваги це надає?**

Як використовувати історію команд:

* Перегляд історії команд: Використовуйте команду ‘history’ для відображення списку останніх команд, які ви виконували.
* Пошук у історії: Ви можете шукати конкретні команди, використовуючи комбінацію ‘Ctrl + R’ та вводячи фрагмент команди для пошуку.
* Використання команд за номером: За допомогою ‘!номер’ можна виконати команду за її номером у списку історії.
* Повторення останньої команди: Команда ‘!!’ дозволяє повторно виконати останню команду.

Переваги:

1. Ефективність

* Швидкий доступ до раніше виконаних команд: Замість повторного введення довгих або складних команд, ви можете швидко знайти та повторно використати їх, що значно економить час.
* Можливість редагування команд: Легко модифікувати попередні команди для нових завдань, не вводячи їх з нуля.

2. Вивчення та аналіз

* Аналіз історії команд: Дозволяє переглянути команди, виконані протягом сесії або за певний період, щоб краще зрозуміти виконані операції.
* Вивчення нових команд: Перегляд раніше використаних команд може допомогти вивчити та запам'ятати їх.

3. Відновлення і виправлення помилок

* Швидке виправлення помилок: Якщо попередня команда містила помилку, ви можете швидко отримати до неї доступ і виправити.
* Відновлення після збоїв: Якщо сесія раптово завершилася, ви можете легко знайти та повторити необхідні команди з історії.

**- Яке призначення команди echo?**

Команда echo використовується для відображення рядка тексту на стандартний вихід, який зазвичай є терміналом. Ця команда має широкий спектр використання, від виведення простих повідомлень до складніших сценаріїв, таких як передача значень у скриптах оболонки.

**- Охарактеризуйте поняття змінної в оболонці Bash, які типи змінних вона підтримує?**

В оболонці Bash, змінна є іменованим об'єктом, якому присвоюється значення. Змінні використовуються для зберігання та маніпулювання даними в скриптах та командному рядку. Основні характеристики змінних в оболонці Bash включають:

**Оголошення змінної:**

* Ім'я змінної: Змінна може мати будь-яке ім'я, але воно повинно починатися з літери або символу підкреслення ’ \_’.
* Ім'я змінної не повинно містити пробілів.

**Присвоєння значення змінній:**

* Використовується знак рівності ‘=’ для присвоєння значення змінній.
* Немає пробілів між ім'ям змінної, знаком рівності та значенням.

`name="Zlata"`

**Типи змінних:**

* Рядкові змінні (Strings): Зберігають текстові дані. Наприклад: `name="Zlata"`.
* Числові змінні (Integers): Можуть зберігати цілі числа. Наприклад: `count=10`.
* Змінні з масивами (Arrays): Дозволяють зберігати кілька значень у вигляді масиву. Наприклад: `numbers=(1 2 3)`.
* Змінні з іншими оболочковими об'єктами (наприклад, командами або функціями).

**Отримання значення змінної:**

* Використовується `$` перед іменем змінної для отримання її значення. Наприклад: `echo $name`.

**Змінні та лапки:**

* Рядкові значення можна визначати як водночас у подвійних `"` або одинарних `'` лапках.
* У подвійних лапках дозволяється використання спеціальних символів (наприклад, змінних), у той час як у одинарних лапках текст трактується буквально.

**- Яке призначення команд env, export та unset?**

**Команда env** — дозволяє запускати іншу програму в користувацькому оточенні без зміни в поточному оточенні. У разі використання без аргументу виведе список змінних поточного оточення.

**Команда export** — створює змінну оточення.

**Команда unset** — видаляє змінні оболонки та змінні оточення.

**- Які команди для отримання довідки по командам в терміналі ви знаєте?**

**1. ‘man’ (довідкові сторінки):**

Призначення: Використовується для відображення довідкових сторінок для команд та програм.

Синтаксис: ‘man команда’.

Приклад: ‘man ls’ виведе довідкову інформацію для команди ‘ls’.

2. ‘--help’ або ‘-h’ опції:

Призначення: Багато команд підтримують опції ‘--help’ або ‘-h’, які виводять коротку довідку по використанню команди.

Синтаксис: ‘команда –help’ або ‘команда –h’.

Приклад: ‘ls –help’ або ‘ls –h’ виведе довідкову інформацію для команди ‘ls’.

3. ‘info’:

Призначення: Деякі програми надають інформацію через систему ‘info’ . ‘info’ дозволяє переглядати докладну довідку в текстовому режимі.

Синтаксис: ‘info команда’.

Приклад: ‘info ls’ виведе інформацію в текстовому режимі для команди ‘ls’.

**6. Підготувати в електронному вигляді початковий варіант звіту:**

- Титульний аркуш, тема та мета роботи

- Словник термінів

- Відповіді на п.4 та п.5 з завдань для попередньої підготовки

**Хід роботи**

1. Опрацюйте всі приклади команд, що представлені у лабораторній роботі курсу NDG Linux Essentials - Lab 5: Command Line Skills та Lab 6: Getting Help. Створіть таблицю для опису цих команд

|  |  |
| --- | --- |
| Назва команди | Її призначення та функціональність |
| ls | Виводить інформації про каталоги та файли. За замовчуванням без аргументів відображає інформацію для поточного каталогу |
| ls -l | Використанні параметру -l в команді ls дозволяє відобразити інформацію про файли, розташовані в поточному робочому каталозі, у довгому форматі, який надає більш розширену додаткову інформацію |
| ls -l /tmp | Використання аргументу /tmp в поєднанні з параметром -l в команді ls дозволяє відобразити детальну інформацію про файли в каталозі /tmp. |
| cd | Змінює контекст поточного робочого каталогу оболонки. |
| cd .. | Переміщує користувача на один рівень вгору в ієрархії директорій. |
| cd /tmp | Переміщує користувача в директорію /tmp, яка зазвичай використовується для тимчасових файлів. |
| pwd | Виводить повний шлях до поточного робочого каталогу. |
| mkdir | Створює новий каталог. |
| rmdir | Видаляє каталог. Він має бути порожнім для видалення. |
| rmdir /tmp/newdir | Видаляє каталог newdir у директорії /tmp, якщо він порожній. |
| touch | Змінює час останнього доступу та/або модифікації файлу. Якщо файл не існує, створює порожній файл. |
| touch file.txt | Створює або оновлює час файлу file.txt. |
| cat file.txt | Виводить вміст файлу file.txt на екран. |
| more file.txt | Показує вміст файлу file.txt сторінка за сторінкою. |
| head file.txt | Виводить перші 10 рядків файлу file.txt. Кількість рядків можна змінити за допомогою параметра -n. |
| tail file.txt | Виводить останні 10 рядків файлу file.txt. Кількість рядків можна змінити за допомогою параметра -n. |
| grep "pattern" file.txt | Шукає рядки, що містять "pattern" у файлі file.txt, і виводить їх. |
| sort file.txt | Сортує рядки у файлі file.txt та виводить результат. |
| uniq file.txt | Виводить або зберігає тільки унікальні рядки з вхідного файлу. Зазвичай використовується після sort. |
| man ls | Виводить інструкцію (man page) для команди ls, яка використовується для виводу вмісту каталогу. |
| help | Виводить допомогу по вбудованим командам оболонки, наприклад, в Bash або іншій оболонці. Використання команди з назвою, наприклад, help cd, надасть інформацію про команду cd. |
| mkdir -p /tmp/newdir | Створює каталог newdir у директорії /tmp, включно з усіма необхідними батьківськими каталогами (опція -p). Якщо каталоги вже існують, помилка не виводиться. |

**Контрольні запитання**

**1. Які типи команд існують в оболонці Bash?**

*1.1 Вбудовані (Built-in) команди:*

Вбудовані команди обробляються безпосередньо самою оболонкою Bash. Вони виконуються швидше зовнішніх команд, оскільки не потрібно створювати новий процес. Приклади вбудованих команд включають ‘cd’ (зміна директорії), ‘history’ (історія команд), ‘echo’ (виведення тексту) та ‘export’ (експорт змінних середовища).

*1.2 Зовнішні команди:*

Зовнішні команди — це програми, які знаходяться за межами оболонки Bash, зазвичай розміщені в стандартних каталогах, таких як ‘/bin’, ‘/usr/bin’, ‘/sbin’, ‘/usr/sbin’ тощо. При виклику зовнішньої команди Bash створює новий процес для її виконання. Приклади зовнішніх команд включають ‘ls’ (виведення списку файлів), ‘grep’ (пошук у тексті) та ‘wget’ (завантаження файлів з мережі).

*1.3 Аліаси (Aliases):*

Аліаси дозволяють користувачам визначати скорочення для довгих команд або груп команд. Аліас створюється за допомогою команди ‘alias’ та діє лише в поточній сесії оболонки, якщо його не додано до файлу конфігурації, наприклад, ‘.bashrc.’ Наприклад, ‘alias ll='ls -la'’ створює аліас ‘ll’ для команди ‘ls –la’.

*1.4 Функції оболонки (Shell Functions):*

Функції оболонки дозволяють користувачам визначати складніші команди з використанням комбінацій вже існуючих команд. Функції ведуть себе подібно до міні-скриптів всередині поточної оболонки і можуть приймати аргументи та повертати значення. Вони оголошуються за допомогою синтаксису ‘function\_name() { commands; }’.

*1.5 Ключові слова (Keywords):*

Ключові слова — це слова, які мають спеціальне значення в Bash. Вони контролюють поведінку оболонки та використовуються для визначення умовних конструкцій, циклів та інших контрольних структур. Приклади ключових слів включають ‘if’, ‘else’, ‘fi’, ‘while’, ‘do’, ‘done’, ‘for’, ‘in’, ‘case’, ‘esac’, ‘function’.

*1.6 Перевірка типу команди:*

Для визначення типу конкретної команди в Bash можна використати команду ‘type’. Наприклад, ‘type cd’ покаже, що ‘cd’ є вбудованою командою оболонки.

**2. Що таке змінні оточення? Які вони бувають. Як їх можна переглянути в терміналі?**

Змінні оточення в операційних системах є спеціальними змінними, які зберігають інформацію про робоче середовище користувача та системи. Ці змінні можуть включати шляхи до важливих директорій, деталі про поточного користувача, інформацію про використання мережі тощо. Вони використовуються як системними процесами, так і запущеними користувачем програмами для отримання конфігурації та властивостей середовища.

*Змінні оточення можна класифікувати на дві основні категорії:*

* *Системні змінні оточення:* Це глобальні змінні, які визначені для всієї системи і всіх користувачів. Вони включають шляхи до системних директорій, версії ОС, деталі про систему і т.д. Приклади системних змінних оточення включають ‘PATH’, ‘HOME’, ‘OS’, ‘USER’.
* *Користувацькі змінні оточення:* Ці змінні визначені на рівні окремого користувача і можуть бути налаштовані для зміни поведінки або конфігурації програм для конкретного користувача. Наприклад, користувач може визначити змінну оточення для зберігання шляху до своїх скриптів або конфігураційних файлів.

*Існує кілька способів перегляду змінних оточення:*

* printenv: Команда ‘printenv’ виводить список всіх змінних оточення разом з їх значеннями. Якщо вам потрібно переглянути конкретну змінну, ви можете використати ‘printenv VARIABLE\_NAME’, де ‘VARIABLE\_NAME’ — назва змінної, яку ви хочете переглянути.
* env: Команда ‘env’ також виводить всі змінні оточення, але крім того, вона може використовуватися для запуску програми в модифікованому середовищі.
* echo: Для перегляду конкретної змінної оточення можна використати команду ‘echo’ разом з назвою змінної, наприклад, ‘echo $HOME’ виведе домашній каталог поточного користувача.
* set: В Bash, команда ‘set’ використовується для виведення списку всіх змінних (включно з змінними оточення) і функцій, доступних у поточній сесії оболонки. Вона виводить більше інформації, ніж ‘env’ чи ‘printenv’, оскільки включає не лише змінні оточення, але й локальні змінні оболонки.

**3. \*Опишіть змінну $PS1. Як в терміналі переглянути її вміст?**

Змінна ‘$PS1’ (Prompt String 1) є однією з вбудованих змінних оболонки Bash, яка визначає основний формат запрошення командного рядка, що відображається користувачам перед вводом кожної команди. Змінна ‘$PS1’ дозволяє налаштувати вигляд командного рядка для підвищення його інформативності та зручності використання, включаючи такі дані як ім'я користувача, ім'я хоста, поточний робочий каталог та інше.

*Вміст змінної $PS1*

Змінна $PS1 може містити звичайний текст та спеціальні ескейп-секвенції, які інтерпретуються оболонкою. Ось деякі з найпоширеніших ескейп-секвенцій:

* **‘\u’** - Ім'я поточного користувача.
* **‘\h’** - Частина імені хоста до першої крапки.
* **‘\W’** - Поточний робочий каталог (тільки назва каталогу, без шляху).
* **‘\w’** - Повний шлях поточного робочого каталогу.
* **‘\d’** - Дата у форматі "день тижня, місяць, число".
* **‘\t’** - Поточний час у 24-годинному форматі HH:MM:SS.
* **‘\n’** - Новий рядок.
* **‘\s’** - Назва оболонки.
* **‘\#’** - Номер поточної команди в сесії.
* **‘\!’** - Номер поточної команди в історії команд.
* **‘\$’** - Показує символ $ для звичайних користувачів та # для користувача root.

Для перегляду поточного значення змінної $PS1 у терміналі, використовуйте команду echo: ‘echo $PS1’

**4. \*Як можна змінити значення змінної $PS1? Що при цьому відбудеться в рядку запрошенні в bash (рядок запрошення перед початком кожної команди). Як змінити значення цієї змінної не на поточний сеанс, а за замовчуванням?**

Щоб змінити значення змінної $PS1 і тим самим змінити вигляд рядка запрошення в Bash, ви можете виконати команду присвоєння в терміналі. Наприклад: **‘PS1='[\u@\h \W]\$ '’**

Ця команда встановить рядок запрошення таким чином, що він буде відображати ім'я користувача (‘\u’), ім'я хоста до першої крапки (‘\h’), поточний робочий каталог (‘\W’), а потім символ ‘$’ для звичайного користувача або ‘#’ для користувача з привілеями суперкористувача (root). Все це буде відображено у квадратних дужках.

Зміни, внесені за допомогою команди вище, тимчасові і діятимуть лише до закінчення поточної сесії терміналу. Щоб зробити зміни постійними, вам потрібно додати команду присвоєння до одного з файлів налаштувань оболонки, які виконуються при старті. Для Bash зазвичай це файл ‘.bashrc’ в домашньому каталозі користувача.

1. Відкрийте файл ‘.bashrc’ у текстовому редакторі. Наприклад, використовуючи ‘nano’: ‘nano ~/.bashrc’

2. Додайте команду присвоєння змінної ‘$PS1’ в кінець файлу. Наприклад: ‘PS1='[\u@\h \W]\$ '’

3. Збережіть файл і вийдіть з редактора. Для ‘nano’ це можна зробити, натиснувши ‘Ctrl+O’, ‘Enter’, а потім ‘Ctrl+X’.

4. Щоб застосувати зміни, перезапустіть термінал або перечитайте файл налаштувань за допомогою команди ‘source’: ‘source ~/.bashrc’.

**5. \*Для чого використовують лапки в оболонці Bash?**

В оболонці Bash лапки використовуються для керування інтерпретацією символів та рядків. Існують три основні типи лапок:

**Одинарні лапки (')**

* Одинарні лапки використовуються для вказівки рядків, в яких всі символи між ними інтерпретуються буквально. Це означає, що жоден символ в межах одинарних лапок не вважається спеціальним і не підлягає інтерпретації оболонкою.

**Подвійні лапки (")**

* Подвійні лапки дозволяють розширення змінних (тобто заміну змінних їх значеннями), виконання команд в зворотних лапках або ‘$()’ та інтерпретацію спеціальних символів (наприклад, ‘\n’ для нового рядка), але при цьому ігнорують інші спеціальні символи, як-от символ долара (‘$’), якщо він не використовується для позначення змінної.

**Зворотні лапки (`)**

* Зворотні лапки використовуються для виконання команд і заміни виводу цих команд в місці їх використання. Однак, цей синтаксис старий і його часто замінюють на сучасніший синтаксис ‘$()’, який має ті ж самі функції, але краще підтримує вкладеність команд.

**6. \*\*Для чого використовують інструкції керування, які їх види Ви знаєте?**

Інструкції керування в оболонці Bash використовуються для створення умовних конструкцій, циклів, перевірки результатів виконання команд, та іншого контролю за ходом виконання скрипта. Вони дозволяють створювати складні скрипти, які можуть приймати рішення, повторювати операції та обробляти різноманітні сценарії виконання. Ось деякі основні види інструкцій керування в Bash:

**Цикли**

* ‘for’ - використовується для ітерації по списку значень або по діапазону чисел
* ‘while’ - виконує блок команд, поки умова істинна
* ‘until’ - виконує блок команд, поки умова не стане істинною

**Вибір**

* ‘case’ - спрощує множинний вибір і порівняння з фіксованими значеннями

**Керування виконанням**

* ‘break’ і ‘continue’ - використовуються для виходу з циклу або пропуску ітерації циклу відповідно

**7. \*\*В чому різниця якщо в кінці рядку запрошення bash стоїть символ $ чи #? Наприклад на екрані ми бачимо наступні записи**

* **[centos@localhost Desktop]$**

Якщо ви бачите символ ‘$’, це означає, що ви увійшли в оболонку як звичайний користувач (не root або інший користувач з привілеями).

Такий символ у кінці рядка запрошення вказує, що ваші команди виконуються з обмеженими привілеями, і вам необхідно користуватися командою ‘sudo’ або входити в режим root, якщо вам потрібні адміністративні привілеї.

* **[root@localhost Desktop]#**

Якщо ви бачите символ ‘#’, це означає, що ви увійшли в оболонку як користувач з привілеями (зазвичай root або інший адміністратор).

Такий символ вказує, що ваші команди виконуються з повними адміністративними привілеями, і вам не потрібно використовувати ‘sudo’.

**8. \*\*Яке призначення команд whereis та locate? Яка між ними відмінність?**

Команди whereis та locate використовуються для пошуку файлів та директорій в системі, але вони працюють трішки по-різному.

* ‘whereis’:

Команда ‘whereis’ призначена для швидкого знаходження бінарних файлів, існуючих в системних шляхах виконання. Вона шукає файли в стандартних системних місцях, таких як ‘/bin’, ‘/sbin’, ‘/usr/bin’, ‘/usr/sbin’ та інші.

* ‘locate’:

Команда ‘locate’ шукає файли та директорії в базі даних, яка зазвичай побудована на основі індексів файлової системи. Це дозволяє їй працювати швидше, оскільки вона не перескановує файлову систему в реальному часі. Однак, важливо зауважити, що індексація файлової системи не оновлюється в режимі реального часу, індекси часто оновлюються періодично за допомогою фонових процесів або планувальників завдань.

**Відмінність між ‘whereis’ та ‘locate’:**

Джерело інформації:

* ‘whereis’ використовує стандартні системні шляхи для пошуку бінарних файлів.
* ‘locate’ використовує індексовану базу даних, що зазвичай побудована з файлівої системи.

Частота оновлення:

* ‘whereis’ використовує актуальні дані про файлову систему в реальному часі.
* ‘locate’ може показати застарілі результати, оскільки база даних може бути оновлена періодично, а не в режимі реального часу.

Швидкодія:

* ‘locate’ може бути швидшою для пошуку, оскільки вона використовує індексовану базу даних.
* ‘whereis’ проводить пошук в реальному часі і може вимагати більше часу для виконання, але вона відображає більш точні та актуальні результати.

Висновок: ознайомилась з базовими командами CLI-режиму в Linux, ознайомилась з базовими текстовими командами в термінальному режимі роботи в різних ОС.