Kyiv College of Communications

PERFORMANCE REPORT

Тема: “Знайомство з базовими командами CLI-режиму в Linux”

Discipline: "Operating Systems"

Performed by students of Group CSN-33 *(Computer Systems and Networks)*.:

Kuznetsov Artur Serhiyovych

Finchuk Alina Oleksiivna

Checked by:  
Suchanova Victoria Sergiivna

словник базових англійських термінів

| **English term** | **Ukrainian meaning** |
| --- | --- |
| **Command Line Interface (CLI)** | Інтерфейс командного рядка |
| **Shell** | Оболонка |
| **Command** | Команда |
| **Prompt** | Рядок запрошення |
| **Argument** | Аргумент |
| **Option / Parameter** | Параметр (опція) |
| **Directory** | Каталог (папка) |
| **Hidden file** | Прихований файл |
| **History** | Історія команд |
| **Environment variable** | Змінна середовища |
| **Local variable** | Локальна змінна |
| **Export** | Експорт (зробити змінну доступною іншим процесам) |
| **Unset** | Видалення змінної |
| **Alias** | Псевдонім |
| **Function** | Функція |
| **Quoting** | Лапки (цитування) |
| **Man pages (Manual pages)** | Довідкові сторінки Linux |
| **Help command** | Команда довідка |
| **uname** | Команда для отримання інформації про систему |
| **whereis** | Команда пошуку розташування виконуваних файлів, вихідного коду та документації |
| **locate** | Команда швидкого пошуку будь-яких файлів у системі (за базою даних) |
| **$ (dollar sign)** | Символ у Bash-запрошенні, що вказує на **звичайного користувача** |
| **# (hash sign)** | Символ у Bash-запрошенні, що вказує на **користувача root (адміністратора)** |

### Визначення понять

*Командний інтерпретатор*  
 Це програма, яка «розуміє» те, що я пишу в терміналі, і перетворює це на дії для операційної системи. По суті, він перекладає мої команди з людського вводу у щось, що система може виконати.

*Оболонка*  
 Оболонка — це середовище, де я спілкуюся з системою. Вона може бути текстова (як Bash) або графічна (як робочий стіл). Якщо говорити про Linux, то оболонка — це саме той інтерфейс у терміналі, через який ми вводимо команди.

*Команда*  
 Команда — це інструкція, яку я пишу в оболонці, щоб виконати якусь дію. Наприклад, показати файли, створити каталог чи запустити програму.

### Відповіді на питання

1. Яку базову інформацію надає рядок запрошення prompt?  
 Prompt показує, що система чекає від мене команду. У ньому зазвичай є ім’я користувача, ім’я комп’ютера і поточний каталог. Ще в кінці стоїть символ $ для звичайного користувача або # для адміністратора.

2. Для чого команді потрібні параметри та аргументи?

* Параметри (опції) змінюють роботу команди, додають якісь деталі.
* Аргументи вказують, з чим саме працювати — наприклад, ім’я файлу чи папки.  
   Тобто без аргументів команда часто не знає, над чим діяти, а без параметрів вона працює у стандартному режимі.

3. Призначення команди ls.  
 ls показує вміст каталогу, тобто список файлів і тек.  
Найчастіші параметри:

* + -l — детальний список (права доступу, розмір, дата).
  + -a — показати приховані файли (ті, що з крапкою).
  + -h — розмір у зручному форматі (KB, MB).

Приклади:

1. ls — показати файли в поточному каталозі.
2. ls -la — показати всі файли з деталями.
3. ls /etc — показати файли у каталозі /etc.

4. Яким чином можна використати історію команд, які переваги це надає?  
 Історія дозволяє подивитися, що я вводив раніше (history), і швидко повторити команду без повторного набору. Це економить час, зручно, якщо команда довга, і зменшує шанс зробити помилку при повторному вводі.

5. Яке призначення команди echo?  
 echo просто виводить текст на екран або значення змінної. Це корисно для перевірки чи для відображення інформації в скриптах.

6. Змінна в оболонці Bash.  
 Змінна — це як коробочка, куди можна покласти значення (текст, число, шлях). Bash підтримує:

* локальні змінні (тільки в цьому терміналі),
* змінні середовища (успадковуються іншими програмами),
* спеціальні змінні (наприклад, $? чи $$).

7. Призначення команд env, export, unset.

* env показує всі змінні середовища, які зараз існують.
* export робить змінну доступною для інших програм, а не тільки в терміналі.
* unset видаляє змінну.

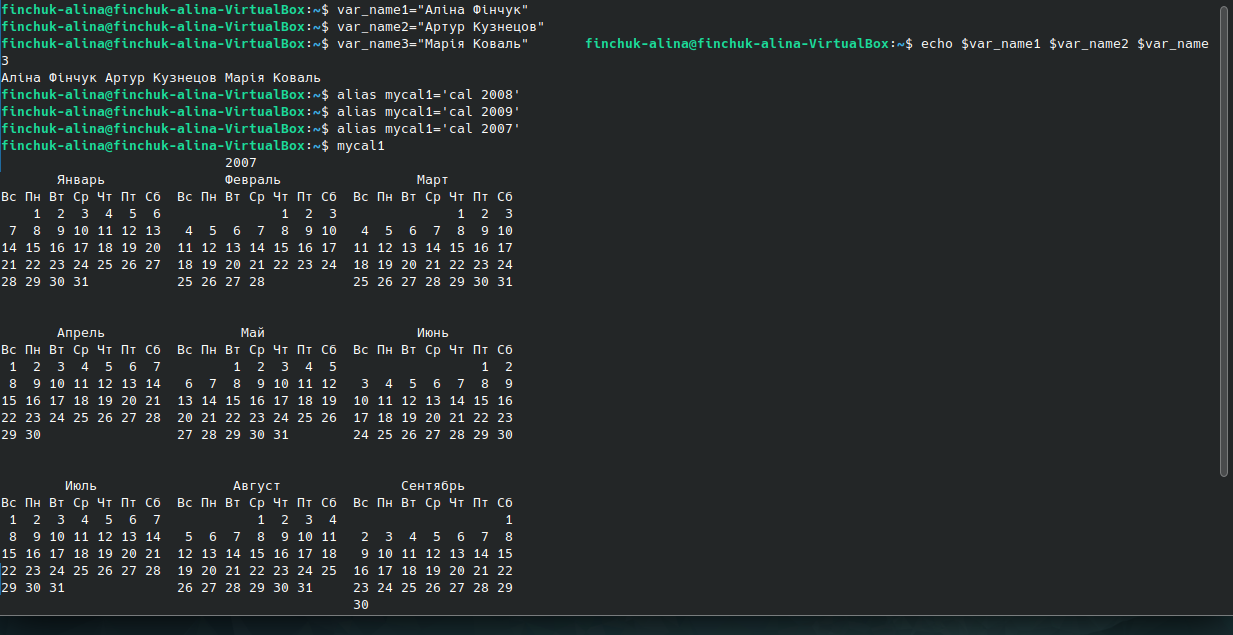
8. Які команди для отримання довідки по командам в терміналі ви знаєте?

* man <команда> — повна документація.
* <команда> --help — коротка довідка.
* info <команда> — ще один варіант довідки.
* help <команда> — для вбудованих команд Bash.

Практична частина: Робота в в терміналі (закріплення практичних навичок) **обов'язково представити свої скріншоти:**

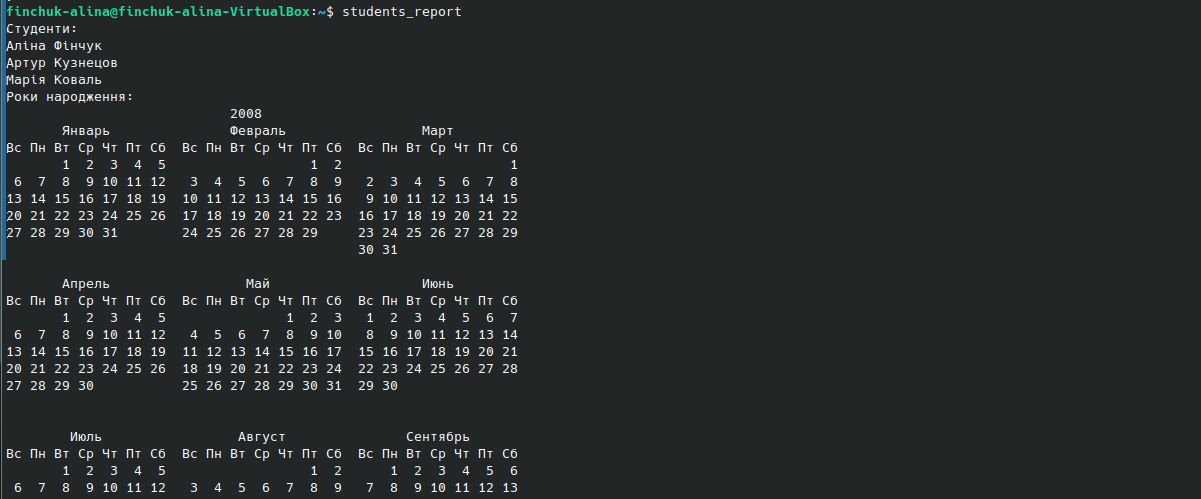
2.1. Робота зі змінними (Variables) та псевдонімами (Aliases) в терміналі:

* Створіть змінні, що будуть містити Ваші імена та прізвища $var\_name1, $var\_name2, $var\_name3
* За допомогою команди echo виведіть імена студентів вашої команди
* Створіть псевдоніми mycal1, mycal2, mycal3 для команди cal для автоматичного виведення календарю вашого року народження



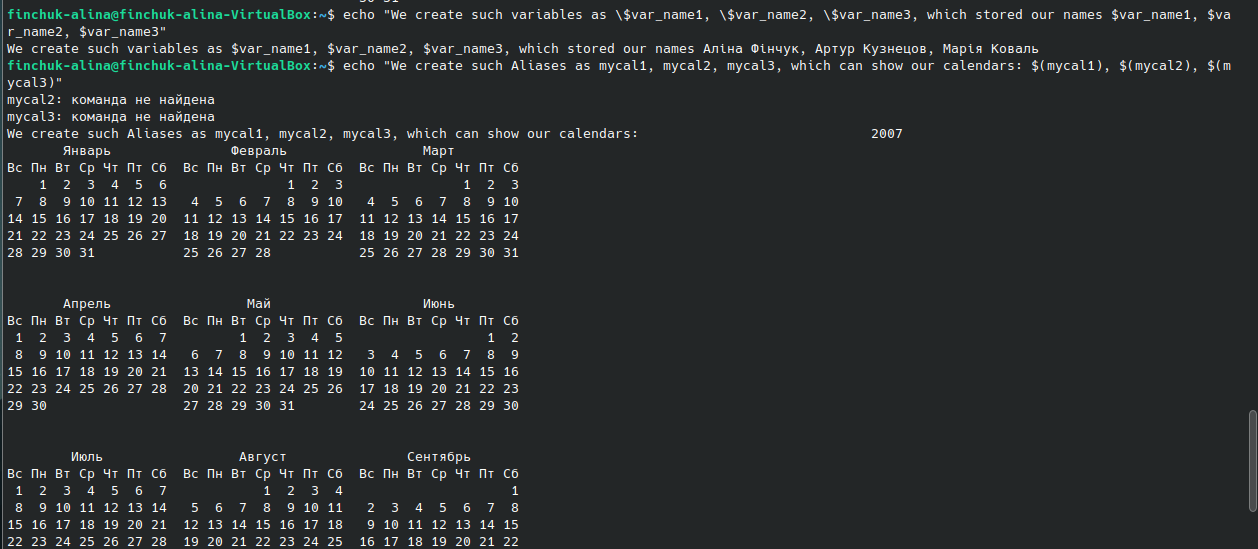
2.2. \*Робота з функціями (Functions) в терміналі:

* Створіть функцію students\_report, що порядково буде виводити спочатку імена студентів Вашої команди, а потім роки їх народження



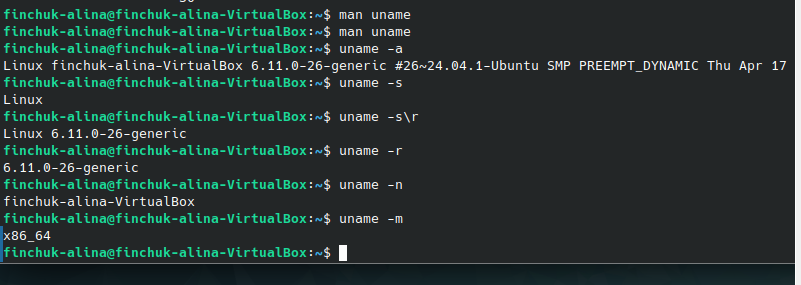
2.3. \*Робота з лапками (Quoting) в терміналі. Виведіть в командному рядку наступні речення:

* “We create such variables as $var\_name1, $var\_name2, $var\_name3, which stored our names Name1, Name2, Name3” (у реченні спочатку виводимо назви змінних, а потім їх вміст)
* “We create such Aliases as mycal1, mycal2, mycal3, which can show our calendars: Calendar1, Calendar2, Calendar3” (у реченні спочатку виводимо назву команди-псевдонімів, потім вивід цих команд).



2.5. Робота з командами довідки (Man Pages) в терміналі:

* На прикладі команди uname продемонструйте як отримати довідку. На основі отриманої додаткової інформації наведіть 5 різних варіантів виводу результату інформації по даній команді з використанням 5 різних параметрів (Options)



### Відповіді на контрольні запитання

1.Які типи команд існують в оболонці Bash?

вбудовані команди (включені в саму оболонку), зовнішні команди (окремі програми у файловій системі) та функції оболонки, які є блоками коду, що визначаються користувачем для виконання певних завдань.

Приклад

Built-in commands:pwd, cd, echo, true, help.

External commands:ls, grep, uname, chmod, ps.

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

**Змінні середовища** — змінні, набір динамічних значень, які можуть вплинути на те як запущені процеси будуть себе поводити на комп'ютері.

Системні змінні середовища — впливають на всю систему і всі користувачі системи мають

до них доступ. Вони налаштовуються адміністратором системи і зберігають важливу

інформацію (наприклад, шлях до системних папок).Користувацькі (локальні) змінні

середовища — встановлені для конкретного користувача і діють лише в його сеансі. Вони

можуть перезаписувати системні змінні для конкретного користувача.

У терміналі змінні середовища можна переглянути кількома способами: Команда printenv

виведе список усіх змінних середовища з їхніми значеннями. Якщо вказати ім'я змінної як

аргумент, виведеться лише її значення, наприклад: printenv HOME

Аналогічна команда — env, теж виведе всі змінні середовища. Для перегляду значення

конкретної змінної можна використати команду з echo, наприклад: echo $PATH

покаже змінну середовища PATH.

Якщо список дуже великий, корисно використовувати фільтрацію через grep. Наприклад:

printenv | grep USER

виведе змінні, у назві яких є "USER".Ще одна команда для перегляду всіх (включно зі

змінними оболонки) — set.

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

$PS1 — це змінна середовища в оболонках Unix/Linux (особливо в bash), яка відповідає за вигляд командного рядка (підказки, prompt), що відображається перед введенням команди. Це основне запрошення, де можна вказати, яку інформацію показувати (наприклад, ім’я користувача, назву хоста, поточний каталог, час тощо). У терміналі Unix/Linux переглянути поточне значення $PS1 можна командою:echo $PS1 Це виведе рядок, який визначає вигляд підказки командного рядка.Для PowerShell (в Windows) $PS1 не використовується. Там інші змінні та механізми налаштування prompt. Отже, $PS1 — це змінна оболонки bash, яка налаштовує вигляд командного рядка, а її вміст подивитися можна через echo $PS1 в терміналі Linux/Mac.

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.

Змінити можна прямо в терміналі командою:

PS1="MyPrompt$ "

У цьому випадку після виконання команда **рядок запрошення (prompt)** зміниться на "MyPrompt$ " замість стандартного вигляду.

Що при цьому відбудеться ?

Bash буде показувати новий формат запрошення.

Наприклад, якщо задати:

PS1="\u@\h:\w$ "

то замість простого $ з’явиться щось типу:

user@hostname:/home/user$

Як змінити $PS1 не на поточний сеанс, а **за замовчуванням**?

Щоб зміни зберігалися після виходу з терміналу:

1. Відкрити файл конфігурації оболонки (для Bash зазвичай це ~/.bashrc):

nano ~/.bashrc

1. Додати рядок:

PS1="\u@\h:\w$ "

1. Зберегти файл і оновити налаштування командою:

source ~/.bashrc

Після цього новий вигляд запрошення буде використовуватися **за замовчуванням при кожному запуску Bash**.

5. Для чого використовують лапки в оболонці Bash?

У оболонці Bash лапки використовуються для контролю над інтерпретацією спеціальних символів та групування тексту, що дозволяє запобігти небажаному розширенню змінних, обробці пробілів та командній підстановці. Одинарні лапки (' ') блокують розширення змінних та команд, тоді як подвійні лапки (" ") дозволяють розширювати змінні та команди, але захищають від розбиття на окремі слова за пробілами.

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

Інструкції керування застосовують для того, щоб:

**керувати порядком виконання команд,**

**створювати умови (якщо/інакше),**

**організовувати цикли (повторення дій),**

**реалізовувати логіку програм** у Bash-скриптах.

Вони дозволяють автоматизувати роботу та робити скрипти більш гнучкими.

### Основні види інструкцій керування:

1. **Умовні конструкції**

if ... then ... else ... fi — виконує команди залежно від умови.

case ... esac — багатоваріантний вибір (аналог switch).

1. **Цикли (повторення)**

for ... do ... done — перебір елементів.

while ... do ... done — цикл з перевіркою умови на початку.

until ... do ... done — цикл до виконання умови.

1. **Оператори керування потоком**

break — достроковий вихід з циклу.

continue — перехід до наступної ітерації циклу.

exit — завершення виконання скрипта.

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

[centos@localhost Desktop] $ та [root@localhost Desktop] #

$ - означає, що у терміналі працює **звичайний користувач**. Такий користувач має обмежені права і не може виконувати критично небезпечні дії з системою без sudo.

#— означає, що у терміналі працює **користувач root (адміністратор)**. Root має повні права на систему: може змінювати будь-які файли, встановлювати/видаляти програми, керувати користувачами тощо.

$- обмежений користувач, # — адміністратор з повними правами.

1. Яке призначення команд whereis та locate? Яка між ними відмінність?

Команди використовуються whereisдля locateпошуку файлів у Linux. whereisшукає файли програм (виконувані, вихідні коди, документацію) у заздалегідь визначених стандартних каталогах, що робить його швидким, але обмеженим за призначенням. Натомість locateвикористовує базу даних, яка оновлюється за розкладом, щоб знайти будь-які файли за назвою по всій системі, що робить його універсальнішим, але результати можуть бути неактуальними одразу після встановлення нового файлу.

conclusions

Conclusion 1 - Artur Kuznetsov:

"Completing the series of terminal tasks has significantly enhanced my understanding and practical skills in working with the Linux CLI. Creating variables and aliases helped me better manage repetitive commands and personalized my working environment. Developing shell functions enabled automation of sequential outputs, which improved task efficiency. Using quoting techniques refined my command line outputs, and exploring control statements deepened my scripting knowledge without relying solely on functions. Lastly, consulting man pages of the uname command taught me to independently seek help and utilize command options for diverse system information. Overall, these exercises strengthened both my theoretical insights and command-line proficiency."

Conclusion part 1- Alina Finchuk:

"Through these terminal exercises, I have made substantial progress in mastering various Bash features. Applying quoting conventions allowed precise control over text and variable interpretation in the shell. Investigating control flow statements demonstrated alternative scripting methods that simplify command sequences. Learning to read and apply manual pages empowered me to understand command functionalities more deeply. Together, these tasks enriched my ability to adapt and optimize terminal workflows for everyday use."

Conclusion part 2 - Alina Finchuk:

"Engaging with these Bash terminal operations has deepened my comprehension of shell scripting components such as quoting, control statements, and accessing help documentation. The practical experience of using aliases and variables alongside exploring scripting logic through control commands has enhanced my scripting versatility. Utilizing man pages demonstrated a critical skill in solving problems independently. This holistic approach to learning Bash has fortified my command line expertise and prepared me to tackle more complex tasks."