Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій у енергетиці

Лабораторна робота №5

з дисципліни «Операційна система UNIX»

«Створення сценаріїв в оболонці Bash»

Виконав:

студент 2-го курсу, НН ІАТЕ

групи ТР-23

Брукалюк С.П.

Перевірила:

д.т.н. Левченко Л.О.

КИЇВ 2023

**Мета роботи**: набуття навичок управління процесами в оболонці *Bash* та опанування команд *ps, top, pstree, bg, fg, nice, renice, kill, killall*

**Порядок роботи:**

1. Ознайомитися з теоретичними відомостями.

2. Виконати завдання за варіантом.

3. Представити скріншоти виконання завдання.

**Короткий теоретичний опис роботи**

***Управління процесами в Linux***

Процеси - це одна з найбільш фундаментальних абстракцій в системах UNIX після файлів. Від оптимального налаштування підсистеми управління процесами та числа одночасно виконуваних процесів залежить завантаження ресурсів процесора, що безпосередньо впливає на продуктивність системи в цілому. Задача ядра – управління процесами. Необхідно чітко розуміти відмінності між процесом і програмою.

*Процес* - це *середовище виконання завдання (оточення)*, яке містить виконуваний код, системні дані, дані користувача і, а також набір додаткових ресурсів, отриманих під час виконання (ресурси пам'яті, можливість доступу до пристроїв введення/виведення та різних системних ресурсів, включаючи послуги ядра. *Програма* - це файл, який містить виконуваний код, дані для ініціалізації та дані користувача.

***Типи процесів***

*Системні процеси* в Unix завжди розташовані в оперативній пам'яті. Їх виконувані інструкції та дані знаходяться в ядрі, тому такі процеси є складовою ядра. Системні процеси можуть викликати функції, а також звертатися до даних, які не мають доступу до інших процесів, наприклад, *системний процес init,* який запускається ядром системи при завантаженні і є одним з ключових процесів для нормального функціонування системи. Приклади системних процесів системні - *vmdaemon, pagezero, bufdaemon, syncer*.

*Демони – неінтерактивний процес*, який працює у фоновому режимі і не прив'язаний ні до якого керуючого терміналу. Зазвичай демони запускаються при ініціалізації системи, однак після ініціалізації ядра забезпечують роботу різних підсистем Unix: системи термінального доступу, системи друку, системи мережевого доступу, мережевих послуг і т.п. Демони не пов'язані із жодним користувачем, тобто не мають ніякого відношення до користувацьких процесів. Як правило, демони знаходяться у стадії очікування, поки для певного процесу не виникне потреба виконати певну послугу (звернення до архіву файлу, друк документу). Прикладами процесів-демонів слугують сервери протоколів НТТР (*httpd*) та FТР (*ftpd*), сервер системного журналу (*syslogd)*, інші приклади - *usbd, sshd.* Зазвичай демони в кінці назви містять літеру «d».

Усі інші *процеси*, які виконуються в системі, вважаються *прикладними або інтерактивними*. Практично це процеси, які запускаються під час роботи користувача. Наприклад, під час реєстрації користувача в системі запускається командний інтерпретатор (*shell*), який надає можливість працювати користувачу в Unix. Інші приклади інтерактивних процесів - *ls, sh, fsck*

Користувацькі процеси можуть виконуватися як в інтерактивному, так і у фоновому режимі, але виключно в рамках сеансу користувача. При виході з системи усі користувацькі процеси знищуються.

Процеси можуть виконуватися на передньому плані (*foreground*) - режим за замовчуванням і у фоновому режимі (*background*). На передньому плані в кожний момент для поточного термінала може виконуватися тільки один процес. Однак користувач може перейти в інший віртуальний термінал і запустити на виконання ще один процес, а на іншому терміналі ще один і т. д. Процес переднього плану - це процес, з яким ви взаємодієте, він отримує інформацію з клавіатури (стандартне введення) і посилає результати на ваш екран (стандартне виведення).

З точки зору ядра процес являє собою запис в ***таблиці процесів***. Цей запис містить відомості про стан процесу і дані, що існують протягом усього часу його життя. Розмір таблиці процесів дозволяє запускати кілька сотень процесів одночасно. Процес також використовує таблицю усіх відкритих процесом файлів, які зберігається в його адресному просторі. *Запис в таблиці процесів і простір процесу разом складають контекст, або оточення, процесу*.

Контекст кожного процесу містить:

- ***pid*** (*process ID* - ідентифікатор процесу), примусово призначається планувальником при запуску процесу;

- ppid (*parent process identifier)) -* ідентифікатор батьківського процесу, UID і GID - ідентифікатори прав процесу (UID - *user identifier, ідентифікатор користувача* і GID – *group* identifier, ідентифікатор групи процесів);

- ***tty*** - ім'я керуючого терміналу (термінал, з якого запущений процес);

- ***nice*** - пріоритет процесу або показник ввічливості;

- статус/стан процесу ***stat*** (*R*=Running*, виконується; S=Sleeping* – у стану очікування; *D=Direct* процес очікує певного сигналу виключно від апаратної частини; *T=Tracing* – процес знаходиться у режимі тросування/відладка програми; *Z=Zombie*, у стані зомбі, тобто процес закінчився, але з деяких причин не звільнений з ядра; «<» підвищений пріоритет; «+» знаходиться в інтерактивному режимі);

- таблиця відкритих (використовуваних) файлів процеса;

- змінні оточення.

***Пріоритети процесів***

Кожному процесу планувальник ОС Linux призначає *пріоритет*, який впливає на те, як довго буде працювати процес. Це пояснюється тим, що частка ресурсів процесора, яка призначається процесу, зважується відповідно до його значення «ввічливості» (*niceness*). Пріоритети називаються в Linux *значеннями ввічливості (NI),* оскільки їх основна ідея - «бути ввічливим» по відношенню до інших процесів шляхом проходження процесної пріоритетності, дозволяючи іншим процесам споживати більше системного процесорного часу. Ядро Linux виділяє завданням з більш високим пріоритетом більший квант часу, а завданням з меншим пріоритетам - менший квант часу, який в середньому становить 200 і 10 мс відповідно.

Діапазон значень ввічливості - від «-20» (найвищий пріоритет) до «19» (нижчий пріоритет) включно, а значення за замовчуванням – 0 (процеси, запущені звичайними користувачами, зазвичай мають пріоритет 0). Таким чином, чим нижче значення ввічливості процесу, тим вище його пріоритет і більше квант часу.

Linux надає декілька викликів для отримання і установки значення ввічливості процесу. Найпростіший з них - ***nice():***

#include <unistd.h>

**int nice(int inc);**

***Управління сигналами***

Одним із способів взаємодії між процесами в Linux є сигнали; сигнали - це програмні переривання, які можуть бути послані процесу за допомогою системного виклику. Кожному виду сигналів в системі відповідає назва і номер. Багато сигнали мають спеціальний сенс. Зокрема, сигнал SIGTERM повідомляє процес про необхідність завершення, сигнал SIGTERM використовується для безумовного завершення процесу, сигнали SIGSTOP і SIGCONT, відповідно, зупиняють і відновлюють виконання процесу. Сигнали SIGKILL (примусове завершення процесу) і SIGSTOP неможливо ні перехопити, ні ігнорувати, тому що вони адресовані не процесу, а планувальнику ОС.

Для того щоб послати сигнал процесу з програми-оболонки:

***kill [-s <назва сигналу>] <список ідентифікаторів процесів>***

**Завдання**

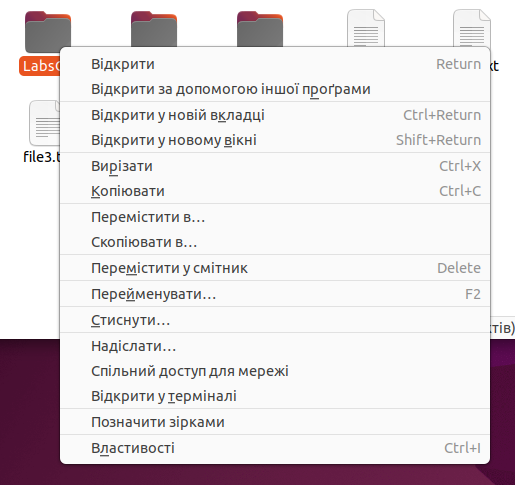
1. Ознайомитися з теоретичними матеріалом по лабораторній роботі. Набути навичок по роботі з процесами.

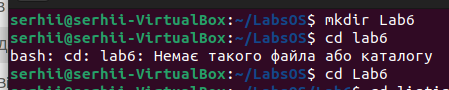
2. Опанувати команди для управління процесами.

3. Підготувати звіт для викладача про виконання лабораторної роботи і представити його.

**Результат виконання роботи виконання роботи**

1.Відкриємо теку LabsOS та створимо у ній теку для лабораторної №6.





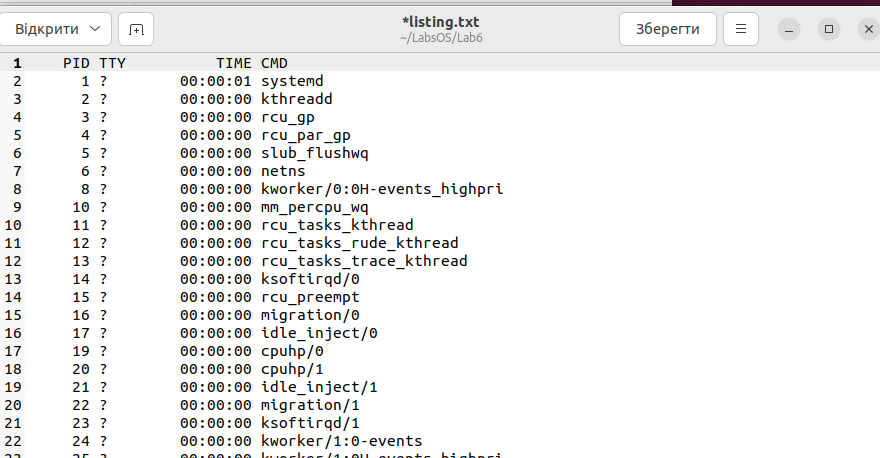
2.Створимо listing.txt, в який згодом запишемо лістинг.



3.Запишемо лістинг у цей файл, за допомогою команди ps, ключа –e та оператора >>.



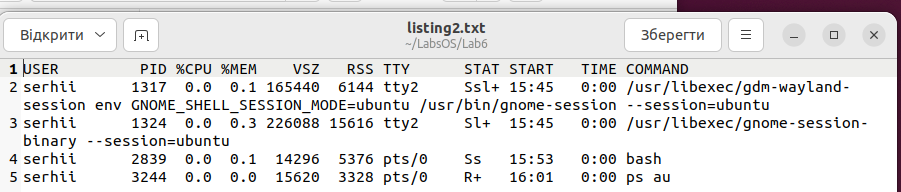
4.Виведення короткого формату активних процесів.



5.Створимо listing2.txt та запишемо туди лістинг усіх процесів у форматі BSD.Використаємо новий ключ au.



6.Виведення повного формату.



USER: Ім'я користувача, який є власником процесу.

PID: Ідентифікатор процесу (Process ID), унікальний номер, призначений процесу.

%CPU: Відсоток використання процесора процесом з моменту останнього оновлення.

%MEM: Відсоток фізичної пам'яті (ОЗУ), якою користується процес.

VSZ: Віртуальний розмір пам'яті процесу в кількості кілобайт (KB).

RSS: Розмір Resident Set, частина пам'яті процесу, яка знаходиться в ОЗУ, в кількості кілобайт

TTY: Тип терміналу, пов'язаний з процесом.

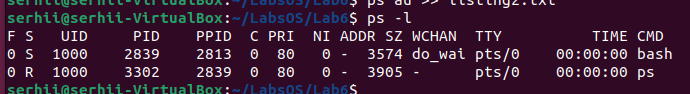
STAT: Статус процесу.

START: Час запуску процесу.

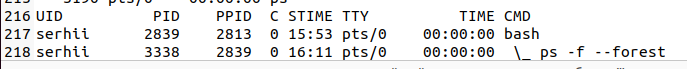
TIME: Сумарний час використання процесора процесом з моменту запуску.

COMMAND: Команда та її аргументи, які запустили процес.

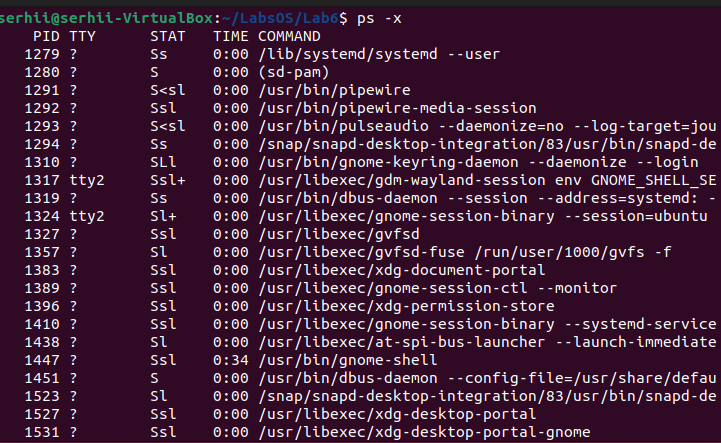
7. Команда ps -lу форматі BSD виводить розширений лістинг та водночас ієрархію процесів разом з ідентифікатором процесу команди та його батьківським процесом.



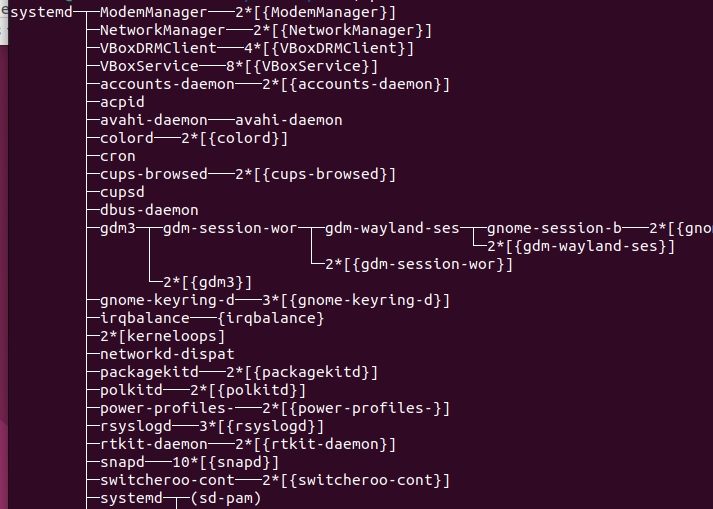
8. Для виведення дерева процесів разом з полями попереднього пункту використаємо команду ps -f –forest та запишемо у файл listing.txt оператором >>.



9.Використаємо команду ps з ключем –x, щоб відобразити запущені користувацькі процеси.

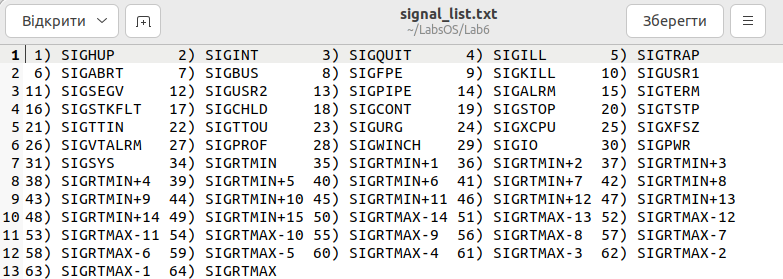


10.Щоб вивести дерево процесів користувача використаємо команду pstree.

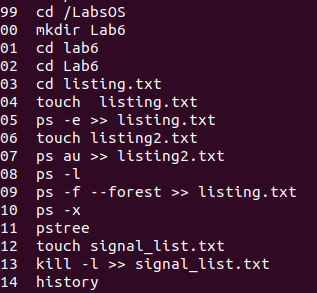


11.Створимо новий файл у теці signal\_list.txt.Запишемо у нього список всіх сигналів, які підтримує система , за допомогою команди kill ключа –l та оператора >>.





12.Список усіх команд за допомогою команди history.



**Висновок**: В ході виконання лабораторної роботи було здобуто теоретичні та практичні вміння у керуванні процесами в оболонці Bash. Отримано поглиблені знання з використання команд ps, top, pstree, bg, fg, nice, renice, kill, killall, як теоретично, так і на практиці. У рамках виконаної лабораторної роботи були демонстровані приклади використання цих команд, а також зафіксовані їхні результати в різні текстові файли. Додатково були освоєні команди для роботи з сигналами та установки пріоритетів для процесів у операційній системі Linux.