Міністерство освіти і науки України

НТУУ «КПІ ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

Лабораторна робота №6 з дисципліни «Операційна система UNIX»

**Робота з процесами ОС Linux**

**Варіант: 16**

Виконав студент групи ТР-12

Каркушевський Владислав

Перевірила д.т.н., проф. Левченко Л. О.

КИЇВ 2022

**Мета роботи :**

- набуття навичок управління процесами в оболонці Bash;

- опанування команд ps, top, pstree, bg, fg, nice, renice, kill, killall.

**Теоретична частина**:

Необхідно чітко розуміти відмінності між процесом і програмою.

Програма - це файл, який містить виконуваний код, сегменти з даними для ініціалізації та з даними користувача.

Процес - це одна з найбільш фундаментальних абстракцій в системах UNIX після файлів. Процес, як об'єкт виконуваного коду, - це екземпляр активної програми, що виконується під управлінням операційної системи.

Процес - це середовище виконання, яке містить сегмент виконуваного коду, сегменти даних користувача і системних даних, а також набір додаткових ресурсів, отриманих під час виконання.

Завдання – це набір інструкцій, завантажений в пам’ять комп’ютера.

Кожний процес працює у своєму віртуальному адресному просторі, отримати доступ до адресного простору іншого процесу можуть тільки деякі системні процеси. Користувацькі процеси можуть взаємодіяти з ядром за допомогою механізму системних викликів.

Потік - це одна з дій всередині процесу. Кожний потік має власний віртуалізований процесор, що містить стек, стан процесора, наприклад регістри, а також командні вказівники. Якщо в процесі тільки один потік, то процес і є потоком. У нього тільки один екземпляр віртуальної пам'яті і один віртуалізований процесор. В багатопоточних процесах потоків декілька. Віртуалізація пам'яті пов'язана з процесом, тому всі потоки одночасно використовують один і той же адресний простір пам'яті.

Образ процесу в оперативній пам'яті складається з:

- даних користувача,

- програми користувача,

- системного стеку (простору ядра),

- блоку управління процесом.

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:

- ідентифікатор процесу (pid - process ID): унікальний номер для позначення процесу;

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

- пріоритет процесу користувача (число nice - пріоритет процесу або показник ввічливості);

- термінал (tty): термінал, до якого запущений процес;

- 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; «<»

- таблиця відкритих (використовуваних) файлів процеса;

- змінні оточення.

Категорії процесів:

- Процеси ядра. Виконуються у просторі ядра, розташовуються в оперативній пам'яті і не мають відповідних програм у вигляді виконуваних файлів. Відповідний код знаходиться в модулях ядра. Приклад - процес упорядкування процесів в черзі (scheduling).

- Фонові неінтерактивні процеси. Мають відповідні виконувані файли. Призначені для обслуговування з'єднань з будь-якого мережевого протоколу або регулярне виконання рутинних операцій в системі.

- Прикладні процеси. Всі інші процеси, які породжуються в рамках сеансу роботи користувача. Процеси можна запускати в інтерактивному режимі і в фоновому режимі. В інтерактивному режимі може працювати тільки один процес. У фоновому режимі процес не підключений до потоків введення-виведення поточного терміналу.

Сигнали - це асинхронне повідомлення процесу про будь-яку подію. Коли сигнал посланий процесу, операційна система перериває його виконання. Якщо процес встановив власний обробник сигналу, операційна система запускає цей обробник, передавши йому інформацію про сигнал. Якщо процес не встановив обробник, то виконується оброблювач за замовчуванням.

Сигнали утворюються:

• з терміналу, натисканням спеціальних клавіш або комбінацій (наприклад, натискання Ctrl-C генерує SIGINT, а Ctrl-Z SIGTSTP);

• ядром системи:

- при виникненні апаратних виключень (неприпустимих інструкціях, порушеннях при зверненні до пам'яті, системних збоях і т. п.);

- помилкових системних викликах;

- для інформування про події введення-виведення;

• одним процесом іншому (або самому собі), за допомогою системного виклику kill(), в тому числі з оболонки shell утилітою /bin/kill.

Сигнали посилаються наступними засобами:

- розділювана пам'ять (це пам’ять, яка дозволяє здійснювати обмін інформацією не через ядро, в через певну частину віртуального адресного простору, в яку розміщують та зчитують дані).

- черги повідомлень (обмін повідомленнями здійснюється наступним чином: один процес поміщає повідомлення в чергу за допомогою деяких системних викликів, а будь-який інший процес може прочитати його звідти, за умови, що і процес-джерело повідомлення і процес-приймач повідомлення використовують один і той же ключ для отримання доступу до черги).

**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

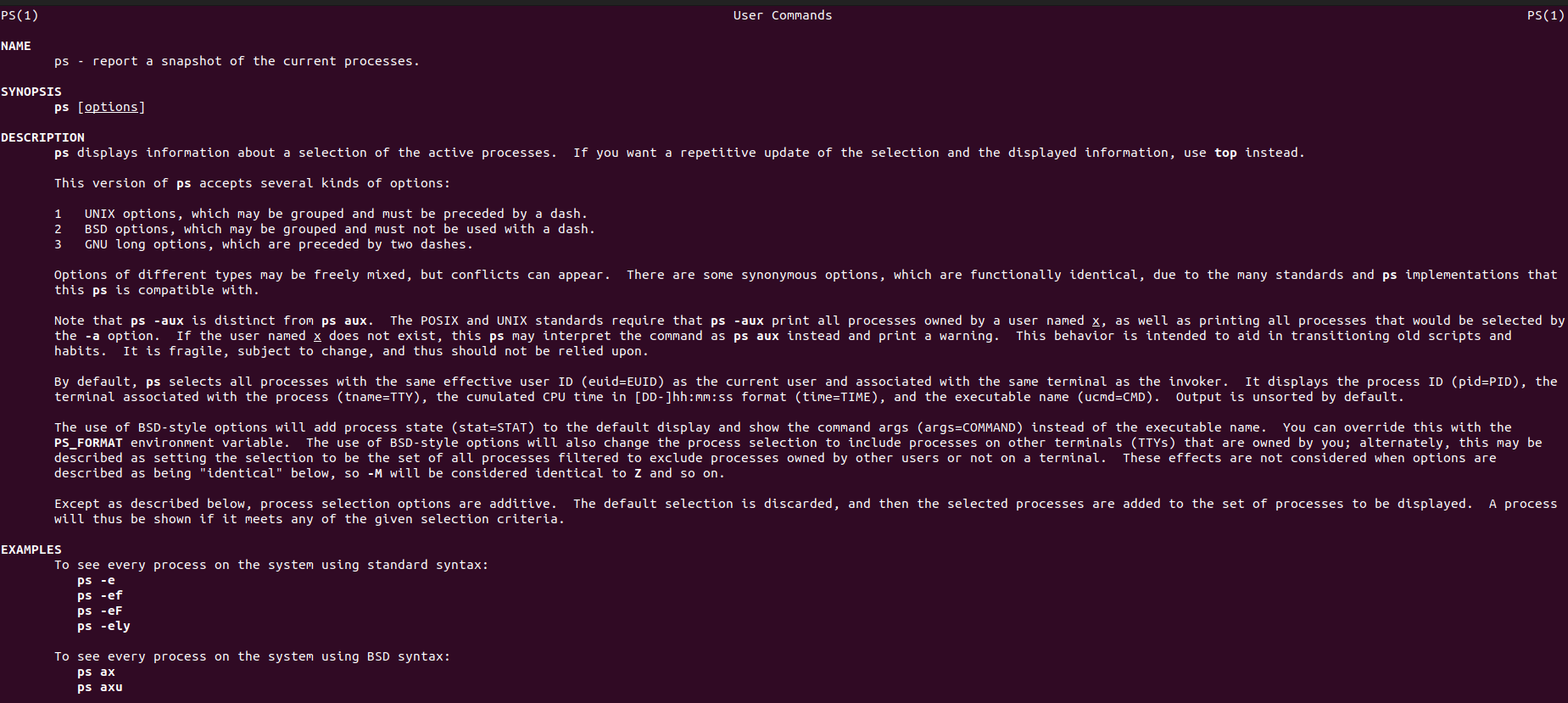
представити його.

**Хід виконання роботи**

Перед тим як виконувати лабораторну роботу, потрібно детальніше ознайомитися з основною командою, яка буде використовуватися у наступних пунктах, з командой ps. Для того аби вивести детальну інформацію про цю команду потрібно ввести man ps

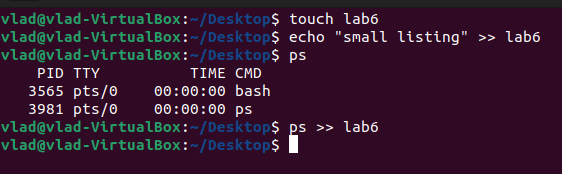


У результаті виведеться інформація про команду, різні ключі використання і тд.

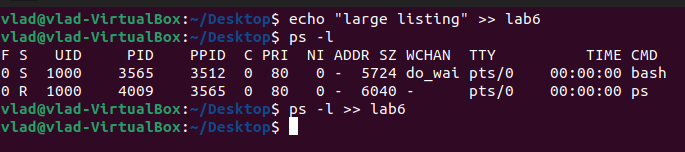


**Вивести на екран лістинг характеристик (у довгому і короткому форматах) процесів, ініціалізованих з вашого терміналу. Записати їх у файл. Проаналізувати і пояснити вміст кожного поля повідомлення.**

Для початку створимо файл lab6, за допомогою команди touch, у яку будемо поміщати усю інформацію. Далі за допомогою команди перенаправлення >> помістимо у файл текст “small listing”. Далі введемо команду ps, яка виведе лістинг характеристик процесів, ініціалізованих з терміналу. Також перенаправимо результат цієї команди у файл.



За допомогою команди перенаправлення >> помістимо у файл текст “large listing”. Далі введемо команду ps -l, яка виведе лістинг характеристик процесів у довгому форматі, ініціалізованих з терміналу. Також перенаправимо результат цієї команди у файл.



Розберемо детальніше усі поля

- 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);

- C - час обробки процесів у процесорі

- PRI - пріоритет процесу Linux на рівні ядра (зазвичай NI+20)

- NI - пріоритет виконання процесу від -20 до 19

- ADDR – адреса процесу

- SZ - розмір на фізичних сторінках

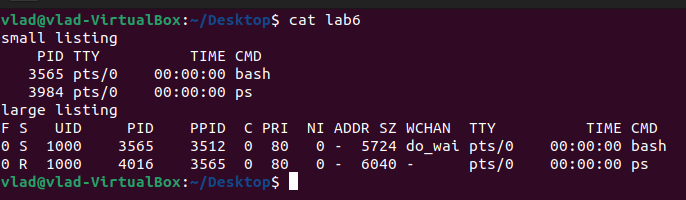
- WCHAN - це адреса події, якого очікує процес. У активного процеса ця колонка порожня.

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

- TIME - час виконання на процесорі.

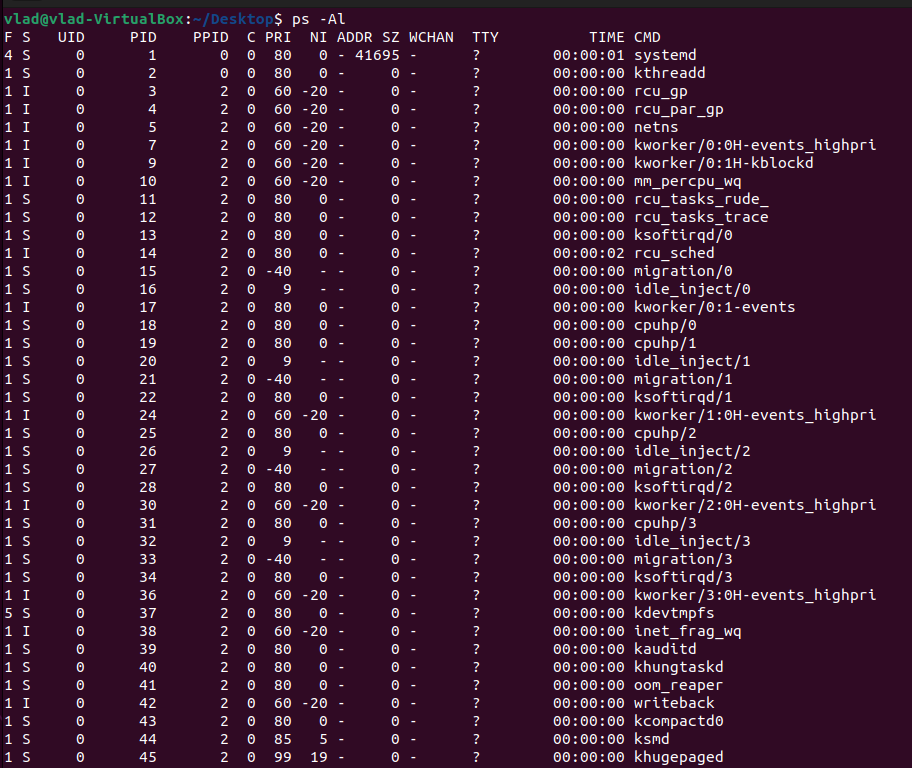
- CMD - ім’я команди

За допомогою команди cat виведемо зміст файлу lab6

****

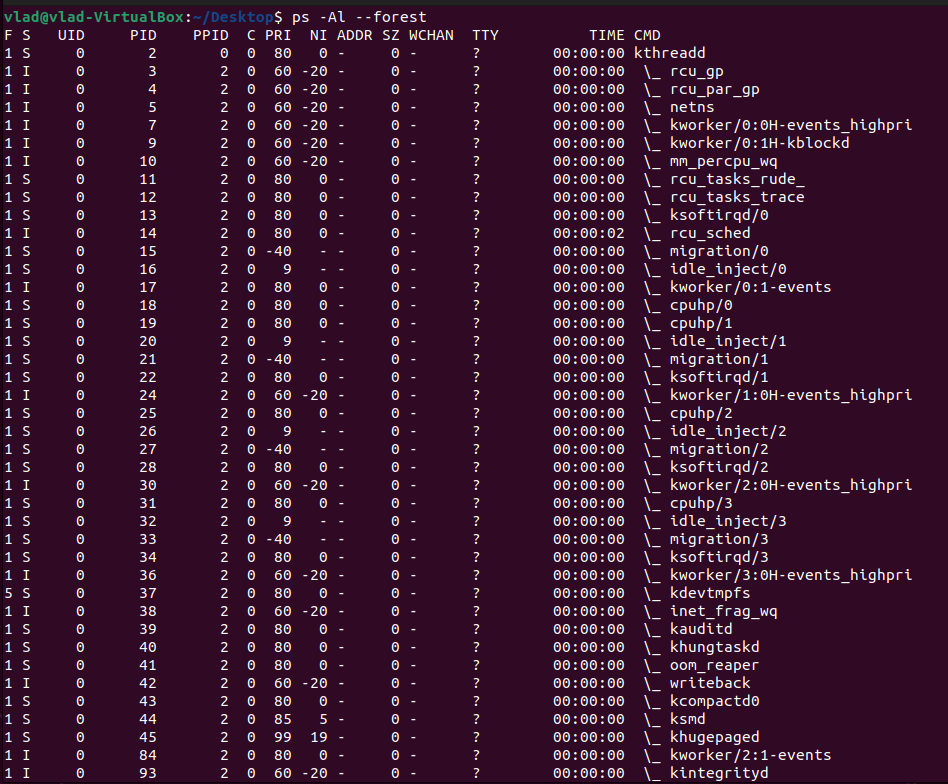
**Вивести всю ієрархію процесів поточної оболонки разом полями pid та ppid.**

Для того аби вивести усю ієрархію потрібно використовувати команду ps з ключем -A або з -e (ідентичні ключи). Для того аби вивести всю ієрархія разом з полями pid та ppid потрібно викорисати ключ -f (середній формат) або -l (повний формат). У даному випадку я використав команду ps -Al.

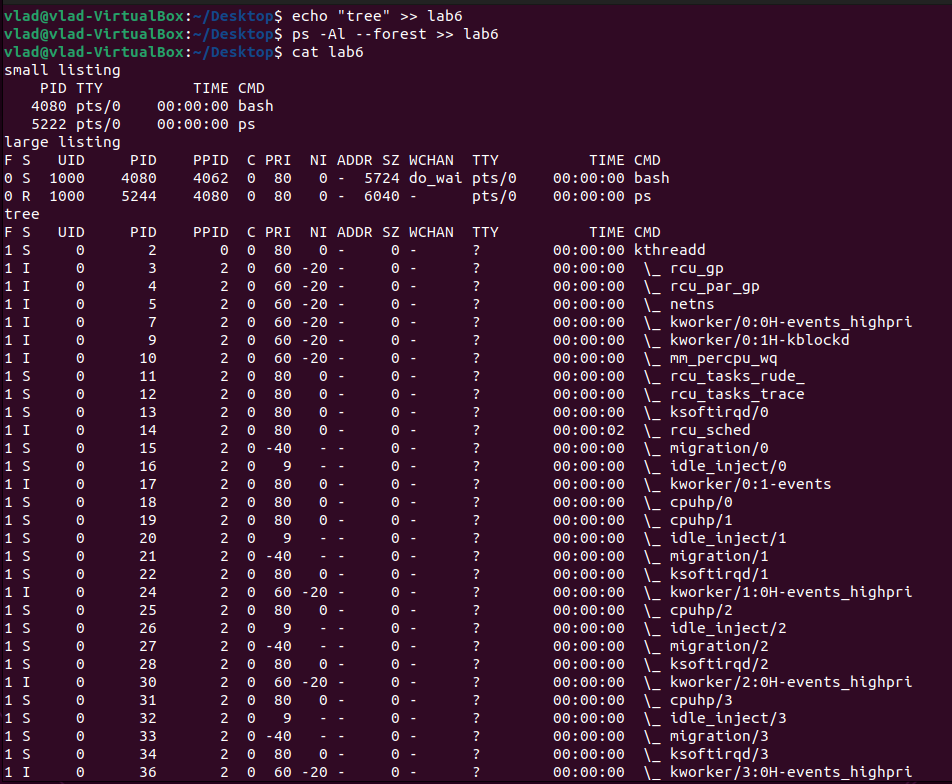
****

**Побудувати дерево процесів, які визначені у попередньому пункті. Результат виконання вивести на екран і дописати в файл.**

Для того аби побудувати дерево процесів, які були визначені у попередньому пункті, потрібно додатково до попередньої команди ps -Af добавити --forest

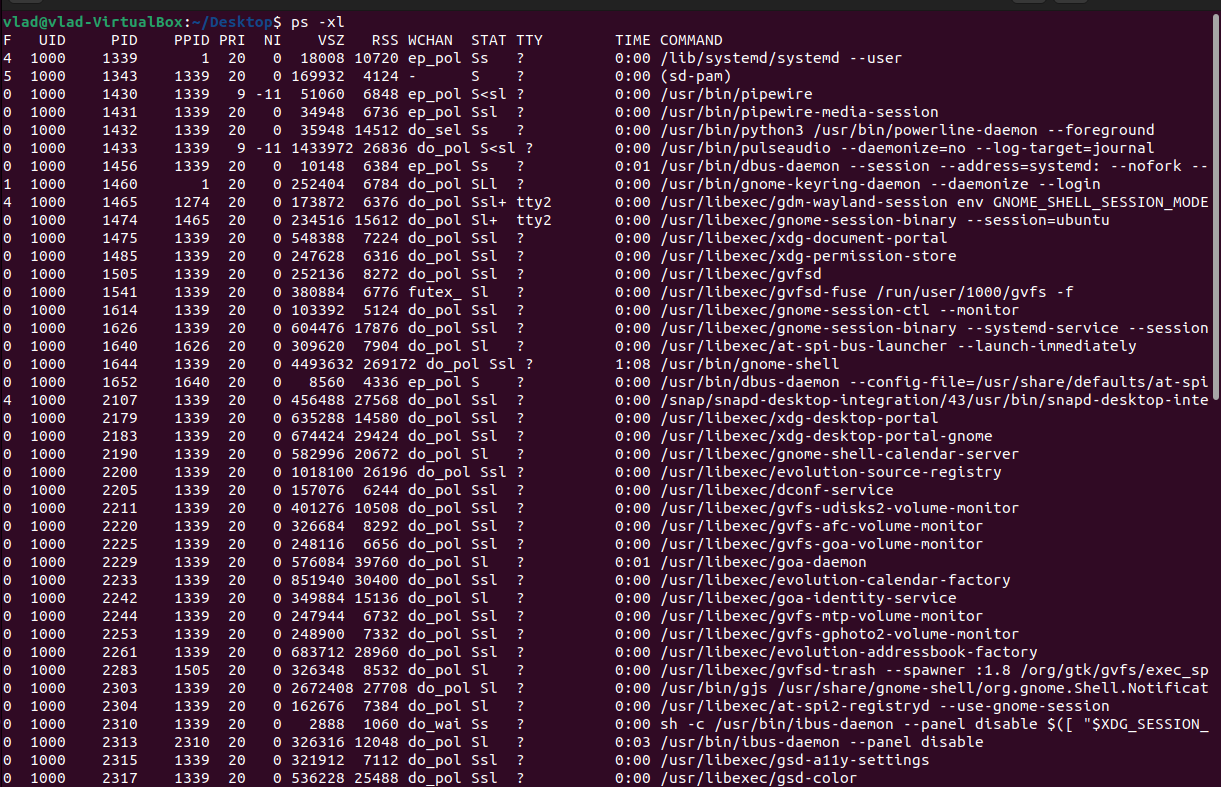
****

**Також до файлу добавимо текст “tree” та вивід команди ps -Al --forest до файлу. Та переглянемо зміст файлу за допомогою команди cat.**

****

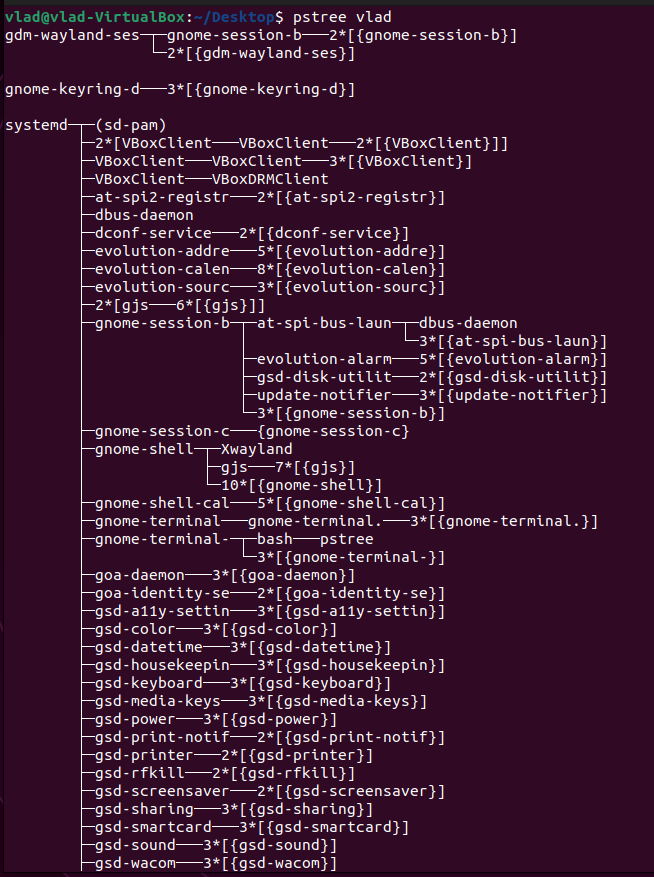
**Переглянути список процесів вашого користувача.**

Для того аби переглянути список процесів користувача потрібно використати ключ -x. Також можна додатково використати ключ -l який виведе інформацію у більш розгорнутому форматі.

****

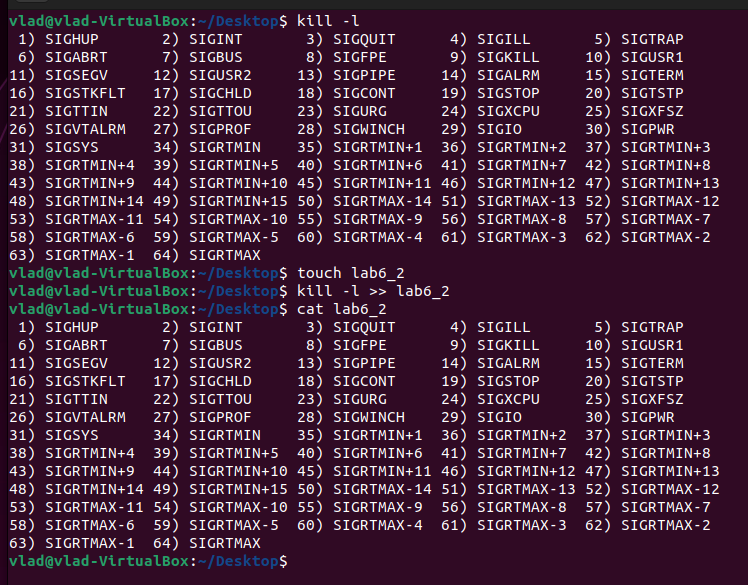
**Вивести список процесів вашого користувача у вигляді дерева (команда pstree).**

Для того аби вивести список процесів користувача у вигляді дерева потрібно використати команду pstree username, тобто pstree vlad

****

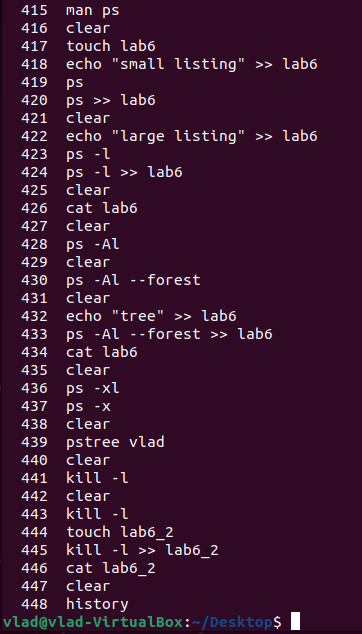
**Переглянути список сигналів вашого користувача. Записати у окремий файл**

Для того аби переглянути список сигналів користувача потрібно використати команду kill з ключем -l, тобто kill -l. Також додатково створимо новий файл lab6\_2 за допомогою команди touch. Та перенаправимо результат команди kill -l у файл lab6\_2. Також виведемо зміст файлу lab6\_2 за допомогою команди cat lab6\_2.

****

**За допомогою команди history виведіть команди, які ви використовували**

За допомогою команди history, виведемо команди які використовувалися у цій лабораторній роботі.

****

**Висновок:** На цій лабораторній роботі було ознайомлено з теорію по лабораторній роботі. Набуто навички управління процесами в оболонці Bash. Опанованно команд ps з різними ключами, top, pstree, bg, fg, nice, renice, kill, killall. Виконано завдання лабораторної роботи та наведено знімки екрану.