Міністерство освіти й науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

ЗВІТ

Про виконання лабораторної роботи №6

З дисципліни «Архітектура системного програмного забезпечення»

**Робота з процесами ОС Linux**

TI-92 Черноусова Дениса

Перевірила д.т.н. Левченко Л.О.

Київ - 2021

**Мета роботи:**

- набуття навичок управління процесами в оболонці Bash;

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

**Теоретичні відомості**

***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.

Процес можна розглядати як сукупність даних ядра системи (*Kernel*), необхідних для опису образу програми в пам'яті і управління її виконанням, або як програму в стадії її виконання, тому що усі програми Unix представлені у вигляді процесів. Ядро ОС обробляє переривання від пристроїв, виконує запити системних процесів та додатків користувача, розподіляє віртуальну пам'ять, створює і знищує процеси, забезпечує багатозадачність за допомогою перемикання між ними, містить драйвери пристроїв, обслуговує файлову систему. Процес складається з інструкцій, які виконуються процесором, даних та інформації про виконуване завдання, а саме: виділена пам'ять, відкриті файли і статус процесу.

***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 взаємодіють між собою засобами міжпроцесної взаємодії (*Interprocess Communication - IPC), а саме:*

- *канали* (*pipe*, конвейєри та іменовані канали *FIFO: First In First Out*),

- *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),

- *сокети*.

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).

Сокети представляють собою *віртуальний об'єкт*, який існує, поки на нього посилається хоча б один з процесів. *Сокети* UNIX бувають 2х типів: локальні і мережеві. *Локальному сокету* присвоюється UNIX-адреса і буде створений спеціальний файл (файл сокета) по заданому шляху, через який зможуть повідомлятися будь-які локальні процеси шляхом простого читання/запису з нього. При використанні *мережевого сокета* створюється абстрактний об'єкт, прив'язаний до порту операційної системи, який слухає, та мережевого інтерфейсу. Цьому типу сокета присвоюється INET-адреса, яка має адресу інтерфейсу і порту, який слухає.

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).

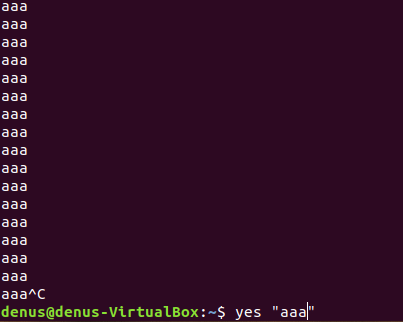
Фоновий процес після свого запуску завдяки використанню спеціальної команди командної оболонки відключається від клавіатури і екрану, тобто не очікує введення даних зі стандартного введення і не виводить інформацію на стандартне виведення, а командна оболонка не очікує закінчення запущеного процесу, що дозволяє користувачеві негайно запустіть ще один процес.

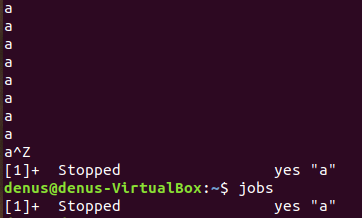
Зазвичай фонові процеси вимагають дуже великого часу для свого завершення і не потребують втручання користувача під час існування процесу. Наприклад, компіляцію програм або архівування великого обсягу інформації можна перевести у фоновий режим.

Для запуску програми в якості фонового процесу досить набрати в командному рядку ім'я програми і в кінці додати знак амперсанта (***&***), відокремлений пропуском від імені програми та її параметрів командного рядка, якщо такі є.

**Завдання :**

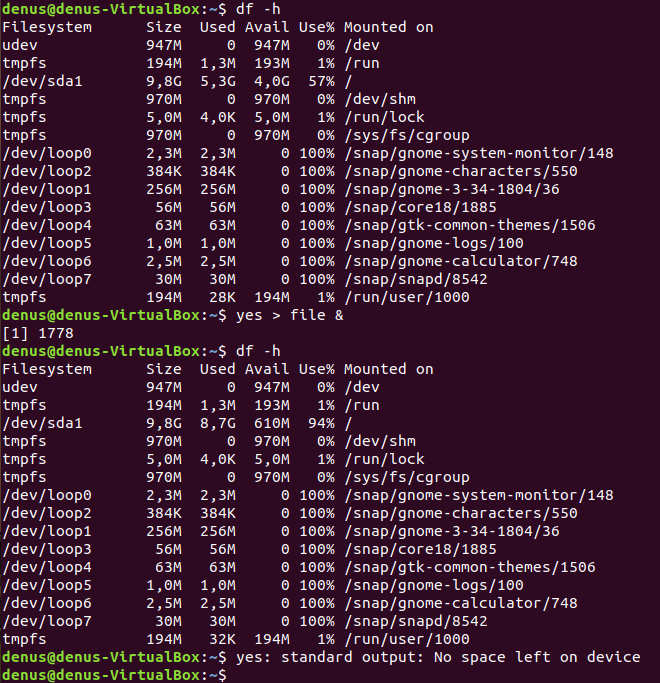
1. Ознайомтесь з роботою команд по управлінню процесами.
2. Вивести на екран лістинг характеристик (у довгому і короткому форматах) процесів, ініціалізованих з вашого терміналу. Записати їх у файл. Проаналізувати і пояснити вміст кожного поля повідомлення.
3. Вивести всю ієрархію процесів поточної оболонки разом полями *pid* та *ppid.*
4. Побудувати дерево процесів, які визначені у попередньому пункті. Результат виконання вивести на екран і дописати в файл.
5. Переглянути список процесів вашого користувача.
6. Вивести список процесів вашого користувача у вигляді дерева (команда *pstree*).
7. Переглянути список сигналів вашого користувача. Записати у окремий файл
8. За допомогою команди *history* виведіть команди, які ви використовували.

**Хід виконання роботи**

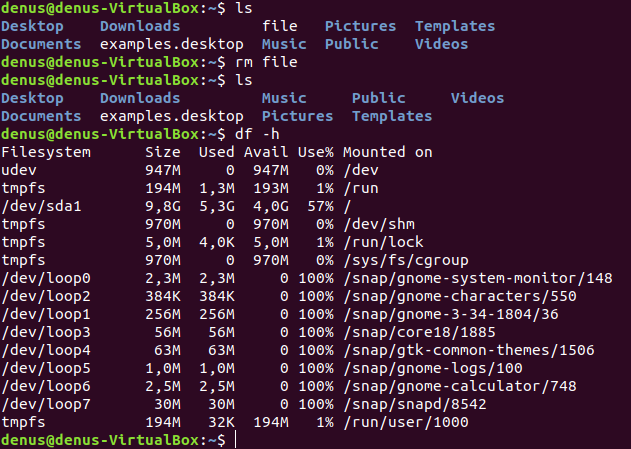
Виведення рядків за допомогою команди yes і знищення цього процесу за допомогою CTRL+C

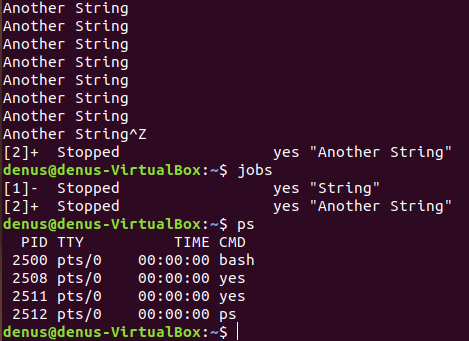
Виведення рядків за допомогою команди yes і зупинка цього процесу за допомогою CTRL+Z

Запуск процесу фонового виведення рядків

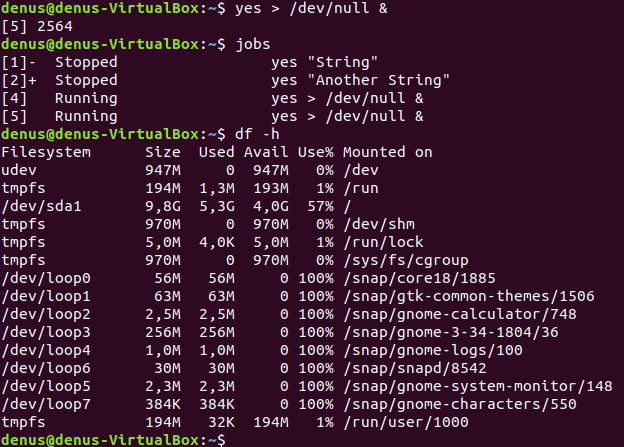
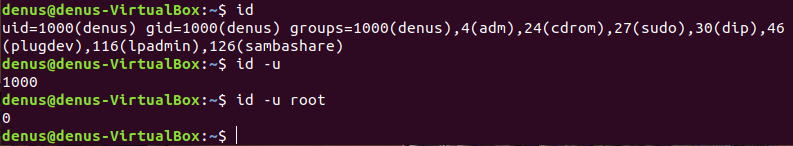


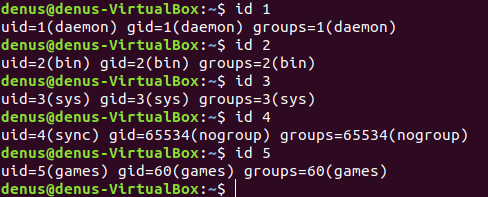
Процес заповнення файлу в домашньому каталозі. Пам'ять системи поступово зникає

Видалення файлу з гігабайтами даних

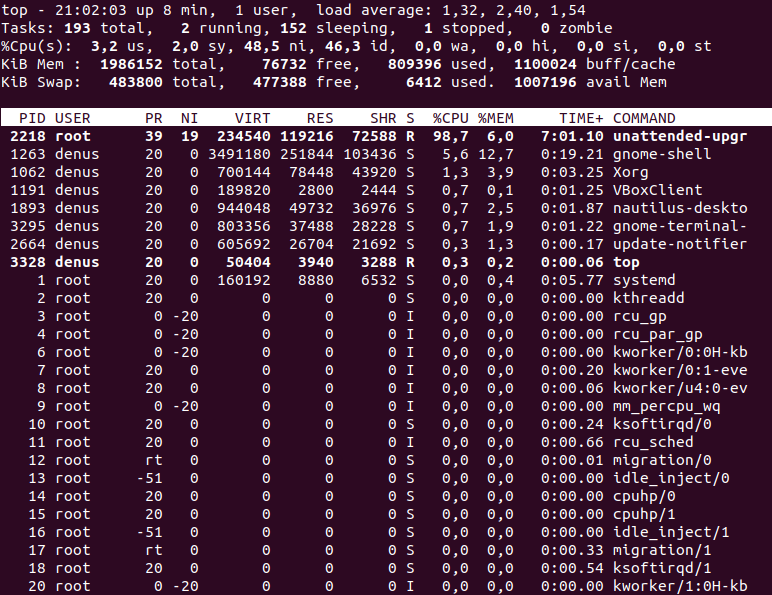


Виведення призупинених процесів

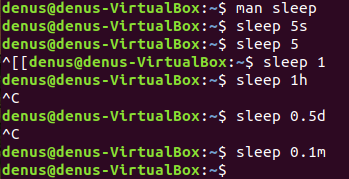
Створення нових фонових процесів. Виведення стану фонових завдань користувача

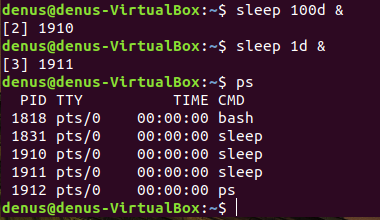
Виведення ід користувача та супер користувача

Кожен процес належить певному користувачеві та певній групі

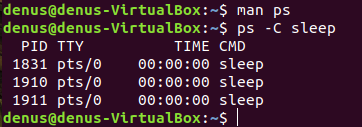


Використання команди top для моніторингу усіх процесів

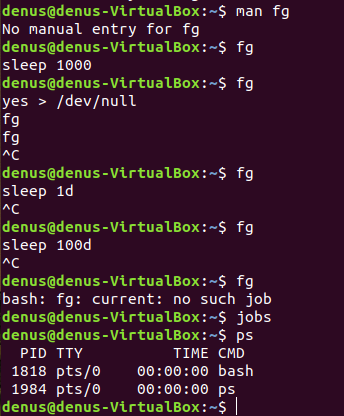
Команда sleep з різними префіксами. Команда sleep 0.1m дає затримку 0.1 хвилини = 6 секунд



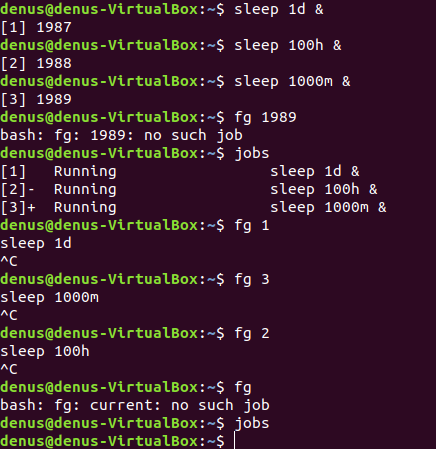
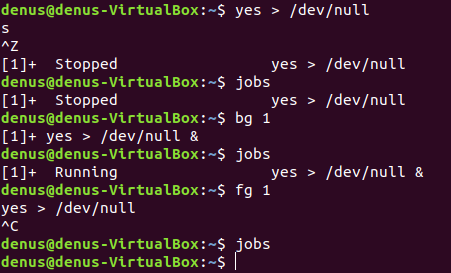
Запуск sleep в фоновому режимі



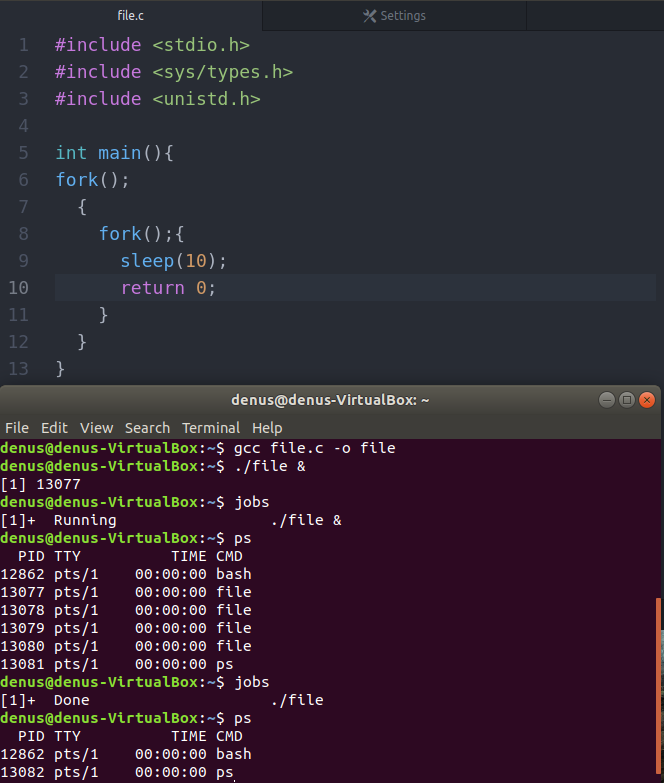
Виведення даних про конкретні команди (sleep)

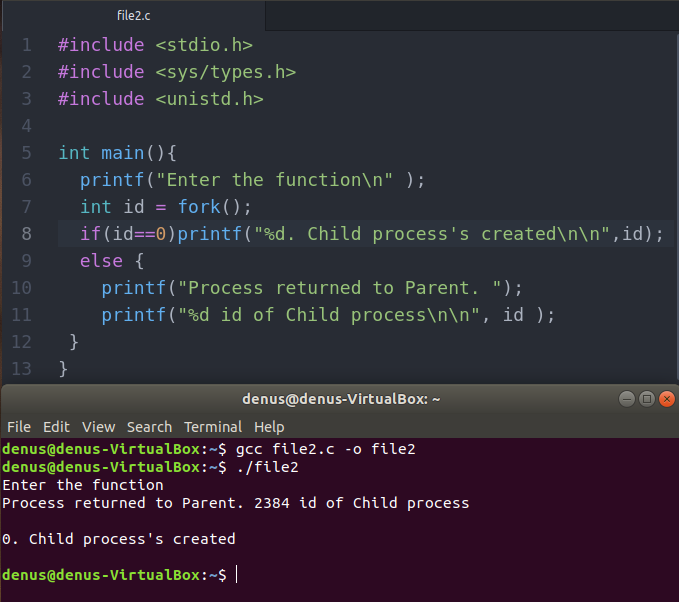
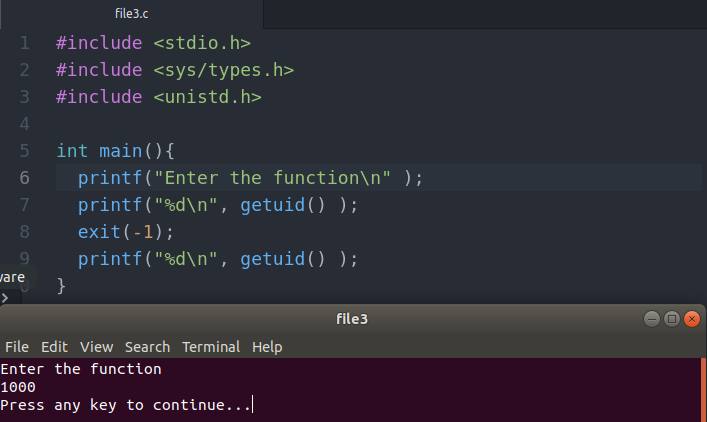
Виведення даних про процес через його id

Виводимо новостворені процеси з фонового режиму і відключаємо їх через CTRL+C

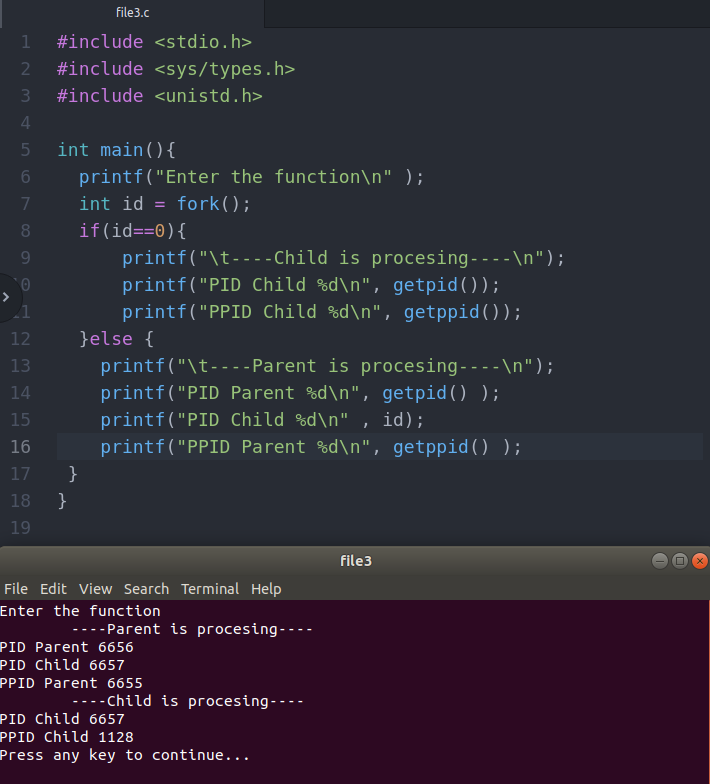
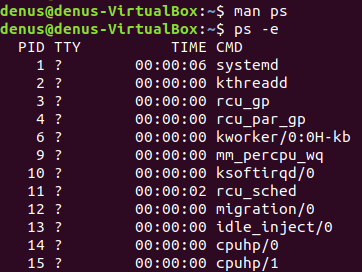
Використовуємо номер фонового процесу (номер задачі) в сеансі користувача з командою fg (foreground)

Використання команди bg

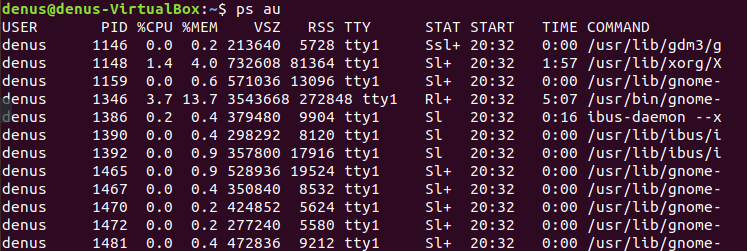
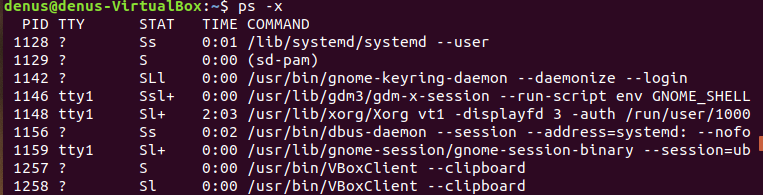
Демонстрація роботи функції fork() в мові програмування С

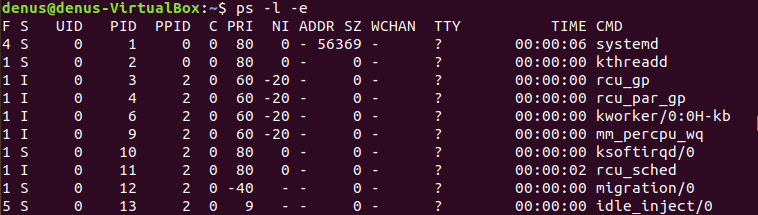
Демонстрація батьківського (Parent) і дочірнього (Child) процесів

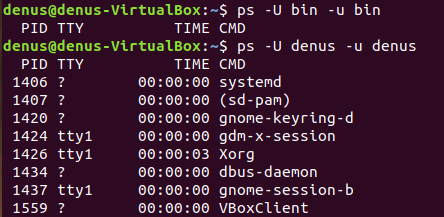
Демонстрація команд getuid() та exit()

Виводимо PID та PPID

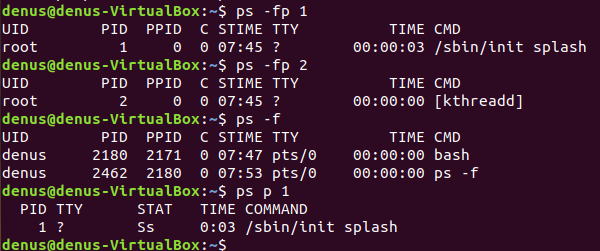
Виведення усіх процесів системи

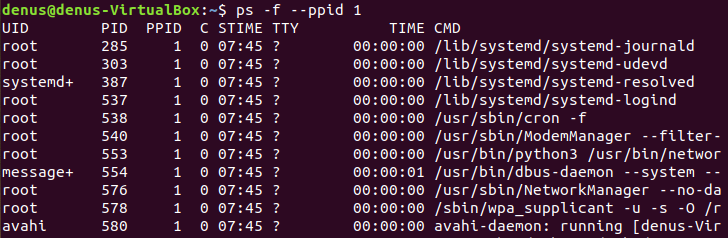
Виведення усіх процесів у форматі BCD

Виведення користувацьких процесів разом з полем TTY

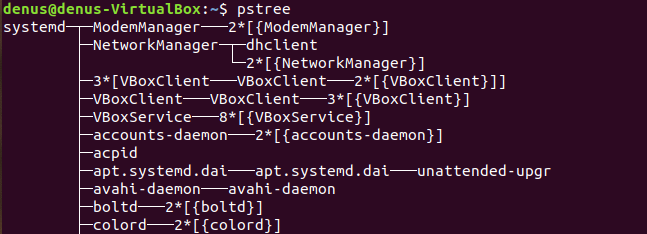
Виведення усіх процесів в довгому форматі

Виведення процесів певного користувача

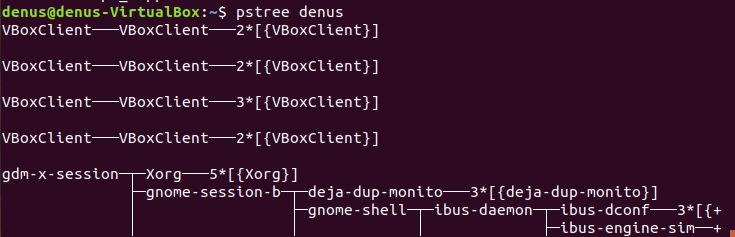
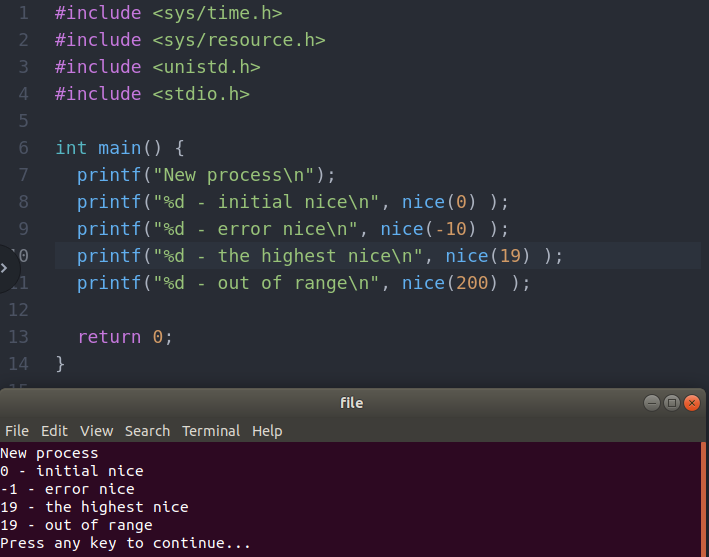
Виведення процесів в повному форматі + по PID



Виведення процесів в повному форматі по PPID

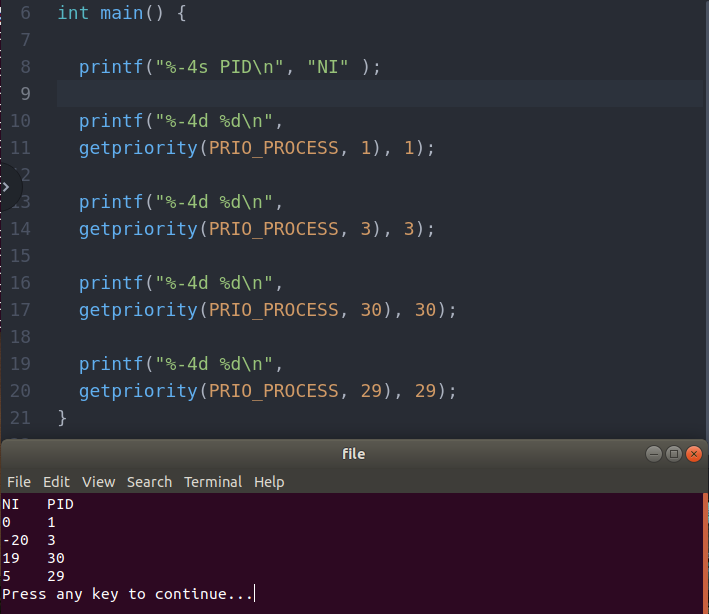
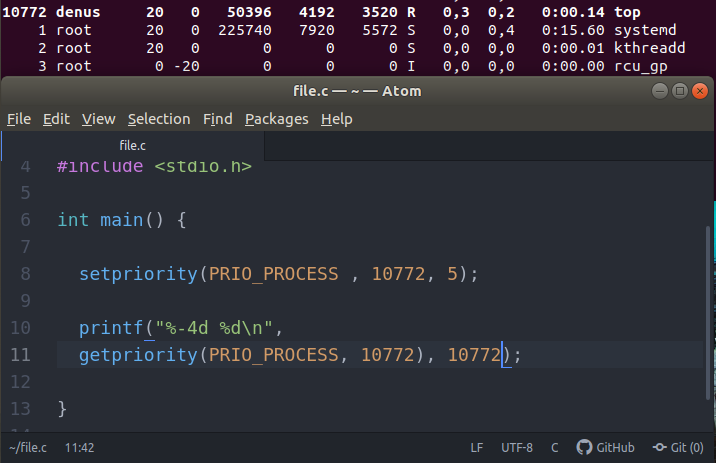


Виведення дерева процесів

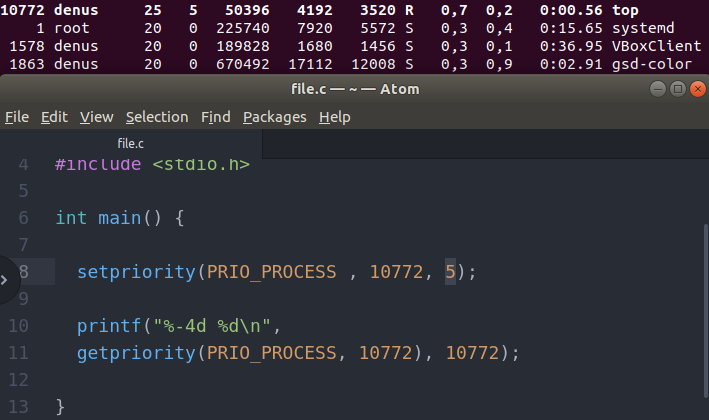
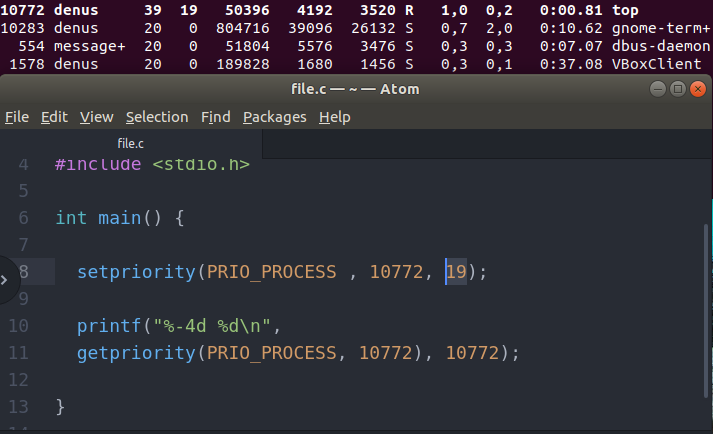
Виведення дерева процесів певного користувача

Використання команди nice для користувача denus

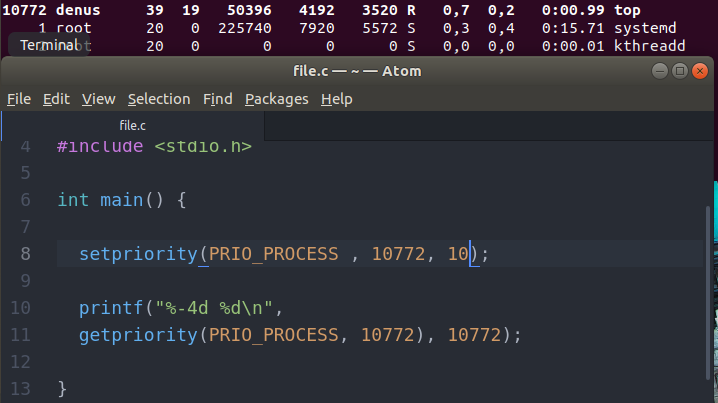
**При збільшенні пріоритету відбулася помилка оскільки тільки процес з характеристикою CAP\_SYS\_NICE (яким фактично володіє користувач *root*) може збільшувати величину пріоритетності, зменшуючи його значення ввічливості і, таким чином, підвищуючи пріоритет процесу.**

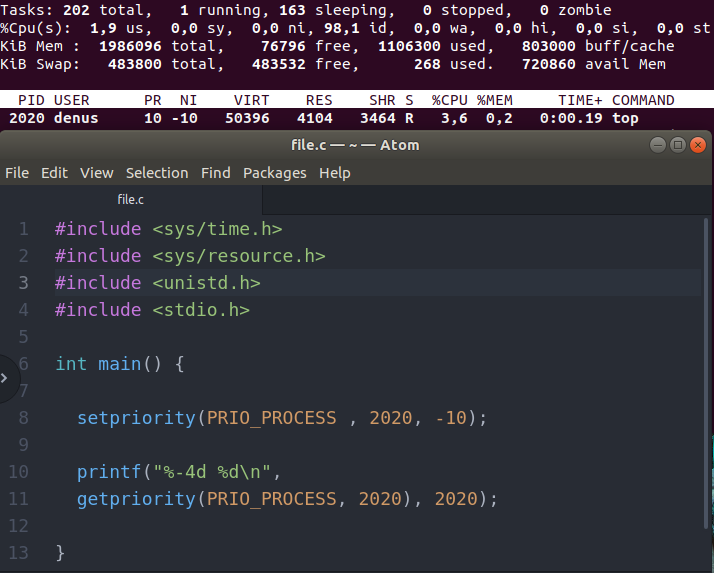
Виведення пріоритетності процесів через команду getpriority()

Встановлення меншої пріоритетності для процесу top

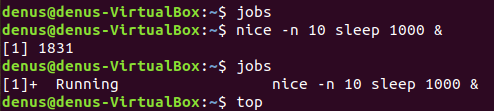
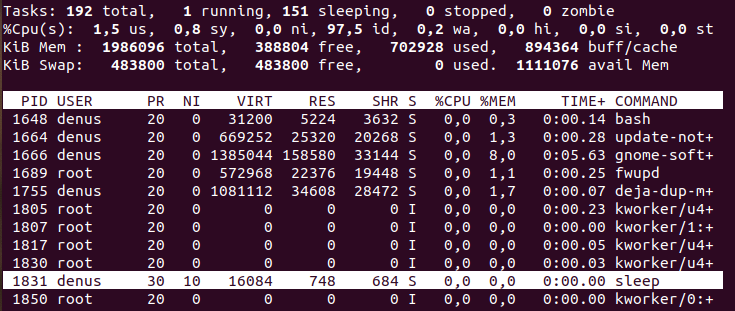
Встановлення меншої пріоритетності для процесу top

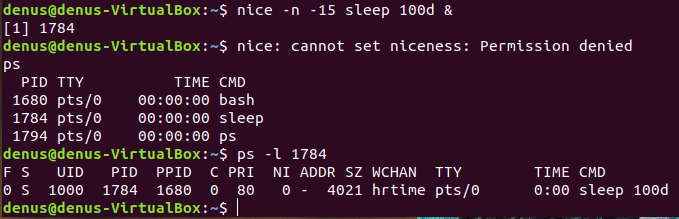
Встановлення меншої пріоритетності для процесу top

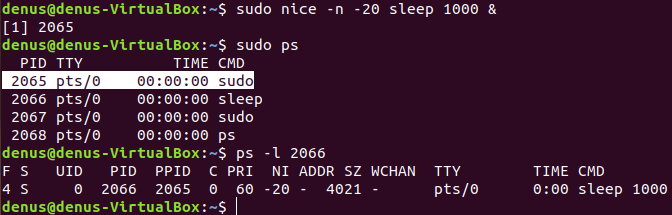
Тільки root користувач може зменшити значення ввічливості (збільшити пріоритетність)



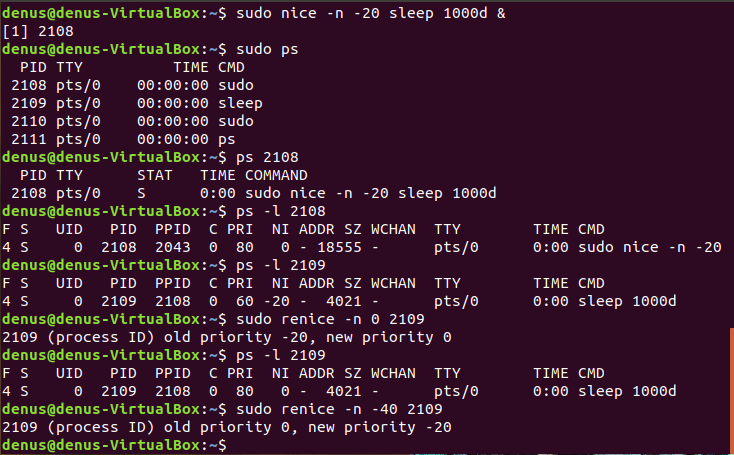
Встановлення відє’много значення nice через sudo

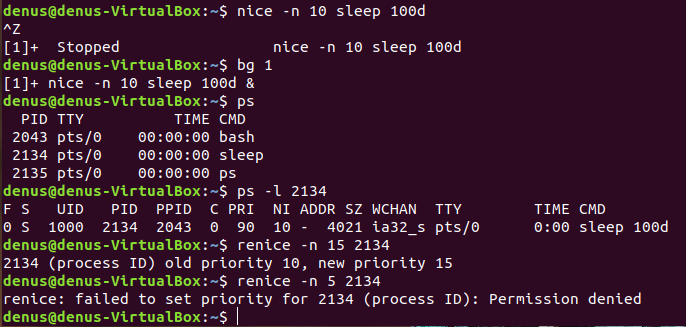
Використання nice для команд

Процес з відповідним значенням nice

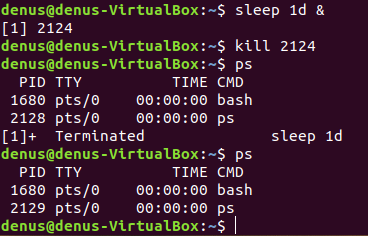
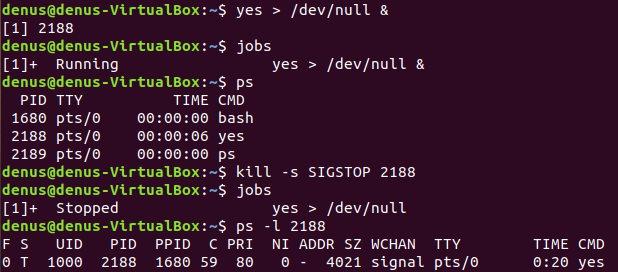
Неможливість створення користувацького процесу з від’ємним значенням nice

Cтворення процесу з від’ємним значенням nice через sudo

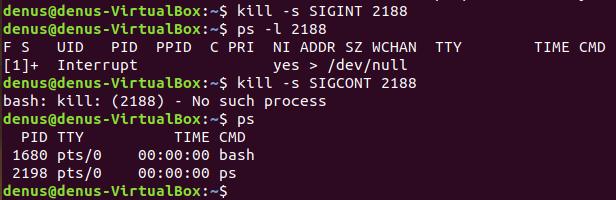
Вільна зміна пріоритетності процесу через renice + sudo



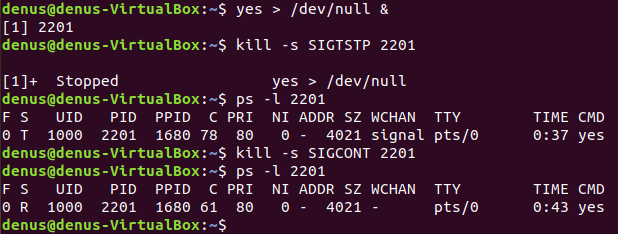
Зміна пріоритетності процесу через renice

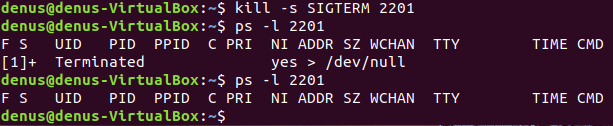
Знищення процесу за його PID

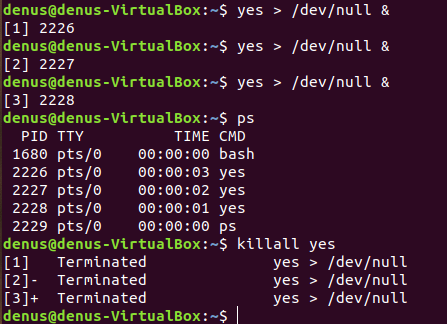
Зупинка процесу через ОС

Запуск цього процесу

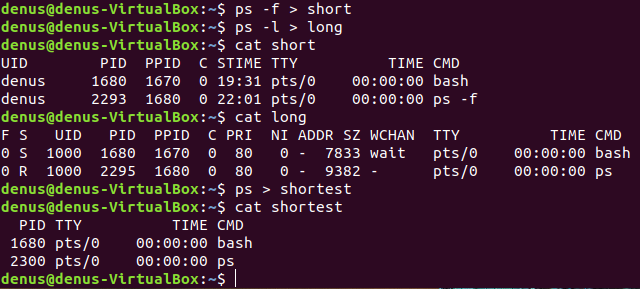
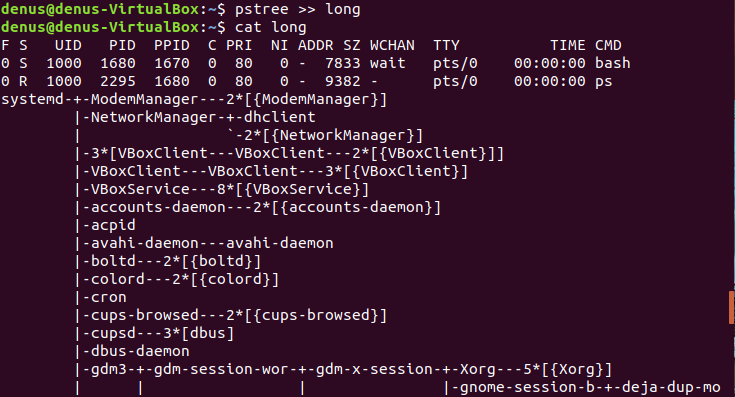
Знищення процесу через перебивання (Interruption)

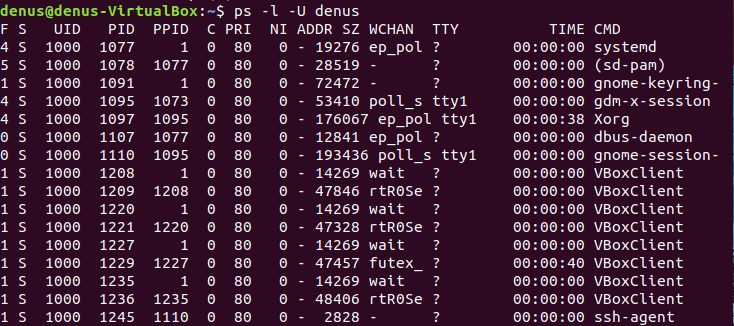
Сигнал на зупинку та продовження роботи процесу



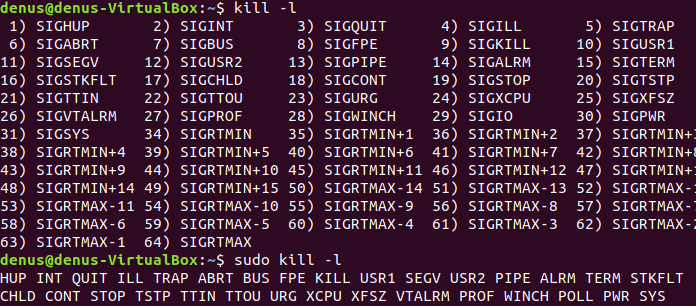
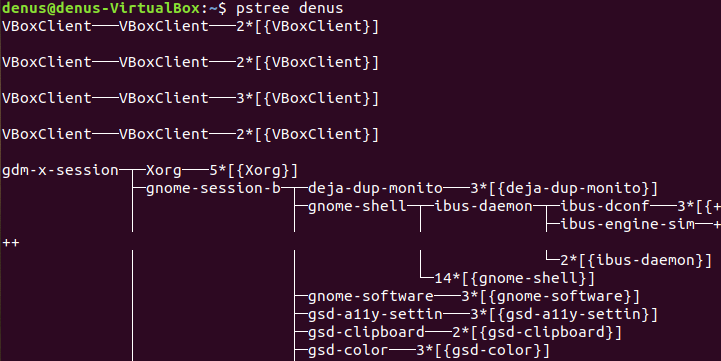
Посилання сигналу, що процес незабаром буде знищений

Знищення процесів команд yes

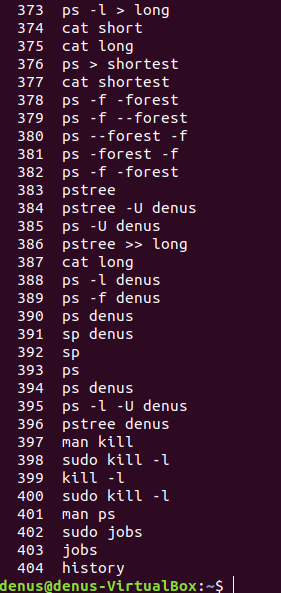
Виведення у файл таблиць процесів сеансу у різних формах

Виведення дерева процесів у файл

Виведення усіх процесів користувача у довгій формі

Виведення дерева процесів користувача

Виведення можливих сигналів для користувача та супер користувача

Історія

**Висновок**

Під час роботи над лабораторною роботою були здобуті навички створення і управління процесами в ОС Linux. Були опануванні на практиці і вивчені команди ps, top, pstree, bg, fg, nice, renice, kill, killall*.* Також здобуті навички роботи з процесами в мові програмування С.