**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи №2

На тему: «Ознайомлення та керування процесами в операційних системах для персонального комп’ютера. Linux та MacOS»

З дисципліни: «Операційні системи»

**Лектор** : ст.викл каф.ПЗ

Грицай О.Д.

**Виконала:** ст.гр.ПЗ-23

Кохман О.В.

**Прийняла:** ст.викл каф.ПЗ

Грицай О.Д.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2022 р.

\_ \_ \_ \_ \_ \_ \_ \_ .

Львів – 2022

**Тема:** Ознайомлення та керування процесами в операційних системах для персонального комп’ютера. Linux та MacOS.

**Мета:** Ознайомитись з процесами та потоками в операційних системах Linux та MacOS. Навчитись працювати із системними утилітами, що дають можливість отримувати інформацію про процеси, потоки, використовувану ними пам’ять та іншу необхідну інформацію.

**Теоретичні відомості**

**Моніторинг процесів в ос Linux**

Для перегляду запущених процесів в ОС Linux використовуються утиліти.

● **top** - вивести список процесів.

● **ps** - інтерактивно спостерігати за процесами (в реальному часі).

● **uptime** - подивитися завантаження системи.

● **w** - вивести список активних процесів для всіх користувачів.

● **free** - вивести обсяг вільної пам'яті.

● **pstree** - відображає всі запущені процеси у вигляді ієрархії.

**Управління процесами:**

До команд управління процесами відносяться команди **nice , renice** , а

також:

**kill** - Завершити процес (або послати йому сигнал).

**pkill** - Відправлення сигналу процесу по імені або іншому атрибуту.

**killall** - Завершити процес по імені.

**pgrep** - Переглядає запущені процеси і виводить на стандартний вивід

список ідентифікаторів процесів, які відповідають критеріям відбору. Всі

критерії повинні збігатися.

**sleep** - Зупиняє виконання на вказану КІЛЬКІСТЬ секунд.

Менеджери середовища Carbon є загальносистемними і забезпечують

низькорівневий сервіс для всіх прикладних середовищ. У число цих

менеджерів входять, наприклад:

● Collection Manager - забезпечення абстрактних типів для колекцій

даних.

● Component Manager - забезпечення для додатка можливості

знаходити під час виконання різні програмні об'єкти (компоненти), а

також створювати компоненти.

● Date, Time and Measurement Utilities - робота з датою, часом,

географічними місцями, часовими зонами і т.п.

● File Manager - файловий API для всіх файлових систем.

● Folder Manager - забезпечення роботи з папками.

● Memory Manager - виділення пам'яті в віртуальному адресному

просторі задачі та інші функції керування віртуальною пам'яттю.

● Multiprocessing Services - засоби для створення потоків, керування

ними та синхронізації.

Core Foundation - фреймворк, який забезпечує деякі базові програмні

служби, корисні для більш високих рівнів програмного забезпечення. Core

Foundation використовує об'єктно-орієнтовану парадигму "непрозорих" типів,

"чорних ящиків" для таких програмних об'єктів як числа, рядки, масиви,

словники, дерева і т.д. Цей компонент також забезпечує роботу з

підключеннями (plug-in) і ряд інших сервісів. Деякі з сервісів, які

забезпечуються Core Foundation:

● String Services - набір інструментів для маніпулювання рядками,

включаючи підтримку Unicode.

● Bundle Services - засоби організації і пошуку різних типів програмних

ресурсів (виконуваних кодів, графічних і звукових образів і т.п.).

● Plug-in Services - забезпечення архітектури підключень.

● Collection Services - високорівневі абстракції колекцій.

● URL Services - засоби доступу до локальних або віддалених ресурсів

через URL.

● Notification Services - механізм обміну повідомленнями

(нотифікаціями) між процесами.

● Open Transport - основні модулі користувальницького рівня для

забезпечення роботи в мережі і комунікацій в MacOS.

**Індивідуальне завдання**

1. Встановити операційні системи Linux та MacOS

2. За допомого консольних засобів ОС Linux отримати повну інформацію про процеси.

3. За допомогою утиліт top, htop, qps, System Monitor отримати повну інформацію про процеси в ОС Linux та MacOS.

4. Використовуючи консольні засоби ОС Linux та утиліти змінити пріоритет виконання процесу.

5. Використовуючи консольні засоби ОС Linux та сторонні утиліти змінити стан виконання процесу, завершити виконання заданого процесу.

6. Скомпілювати файл main.cpp представлений у лабораторній роботі № 1

(на MacOS і Linux можна командою: g++ main.cpp -pthread) і запустити

виконуваний файл на різній кількості активних процесорів (ядер). Знайти для даної програми величини *A* , *S* , *p* при різних вхідних значеннях величини *n* . Порівняти результати для різних операційних систем.

7. Результати лабораторної роботи оформити у звіт, у висновку надати

порівняння моніторингу процесів у різних системах різними утилітами,

відповідно до індивідуального варіанту.

**9)** Конвертування з одного відео формату в інший (наприклад з використанням ffmpeg)

**Протокол роботи**

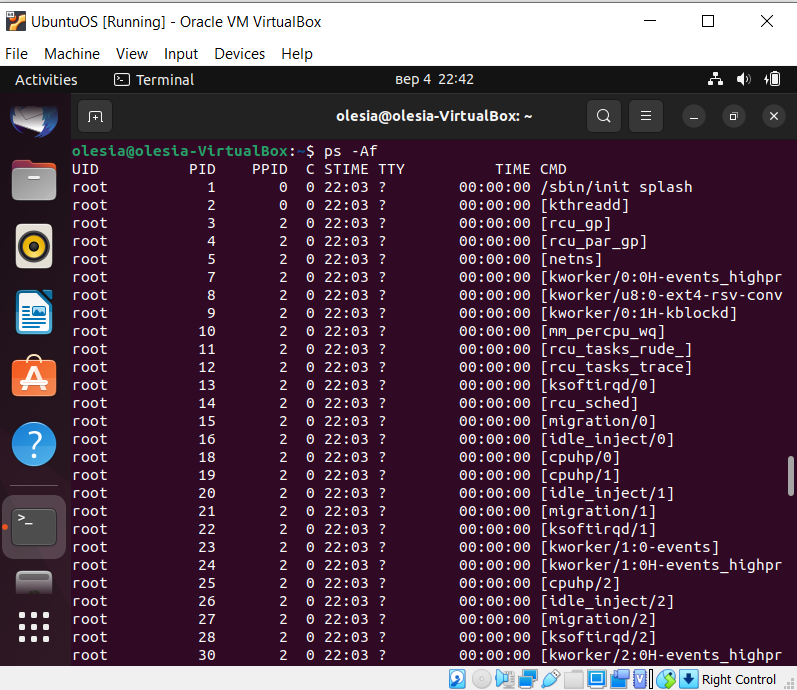
****

Рис. 1 Повна інформація про процеси за допомогою консольних засобів Linux

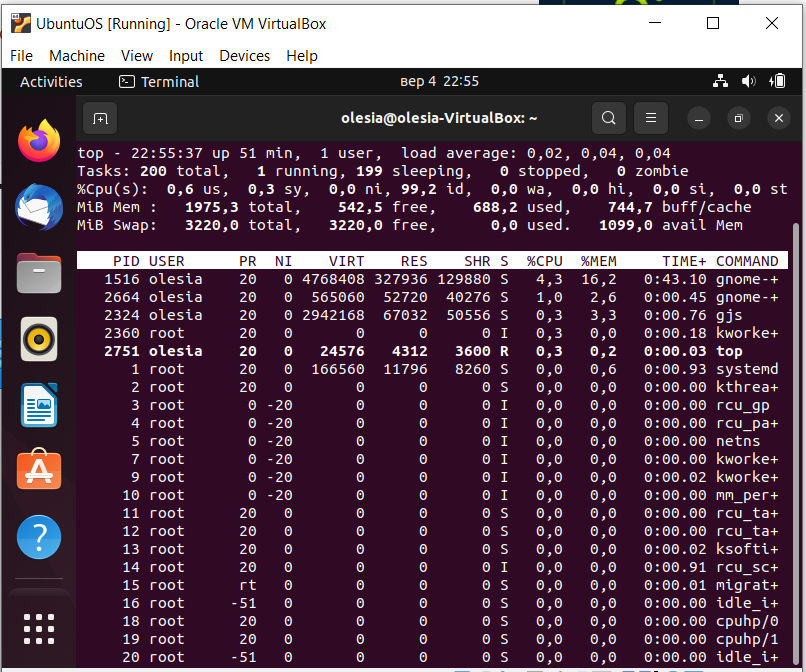
****

Рис. 2 Повна інформація про процеси за допомогою top в Linux

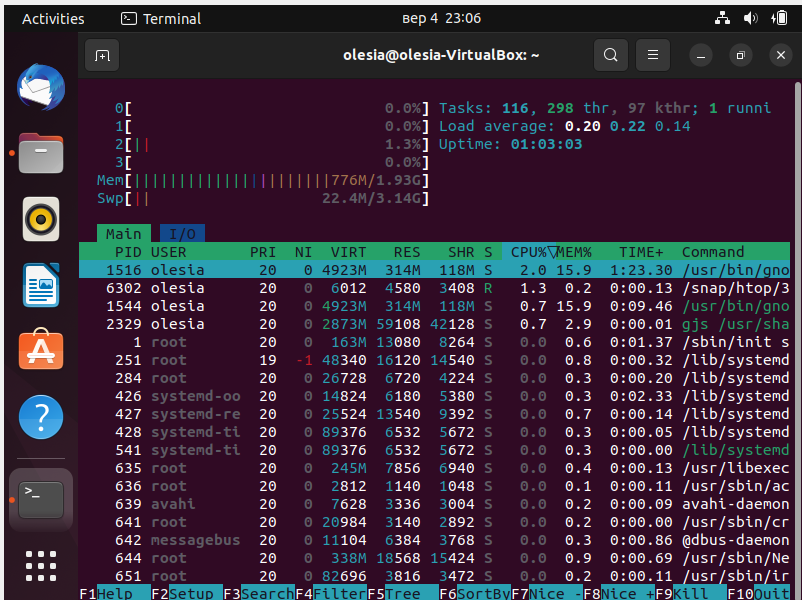
****

Рис. 3 Повна інформація про процеси за допомогою htop в Linux

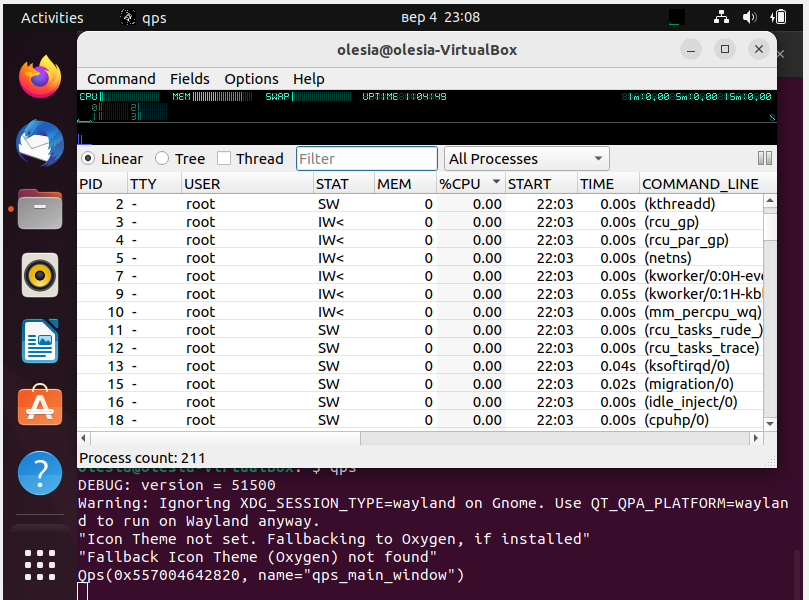
****

Рис. 4 Повна інформація про процеси за допомогою qps в Linux

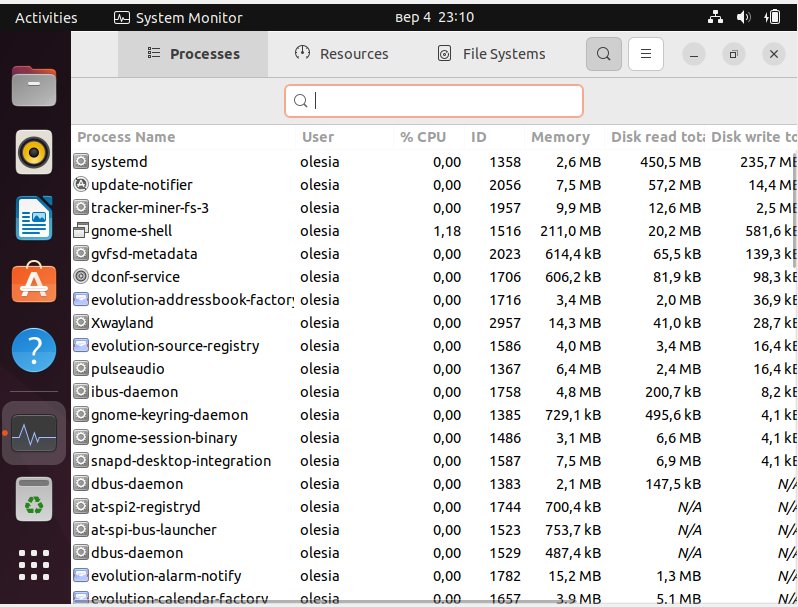
****

Рис. 5.1 Повна інформація про процеси за допомогою System Monitor в Linux

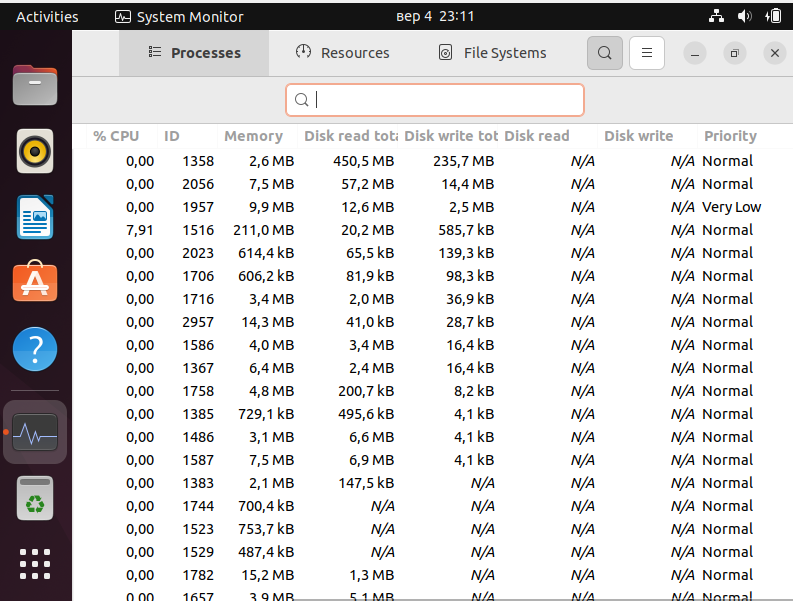
****

Рис. 5.2 Повна інформація про процеси за допомогою System Monitor в Linux

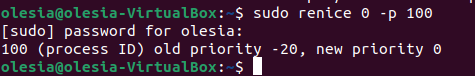
****

Рис. 6 Зміна пріоритету в консолі Linux

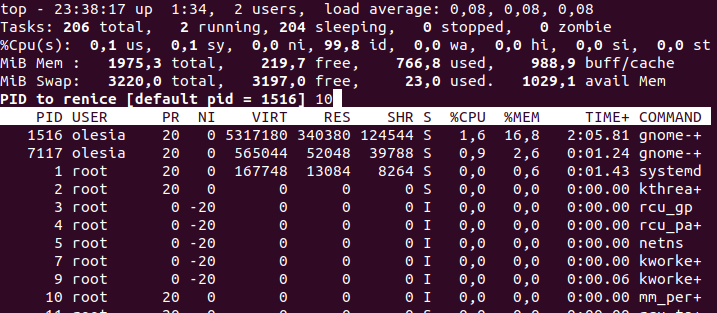
****

Рис. 7 Зміна пріоритету за допомогою top в Linux

****

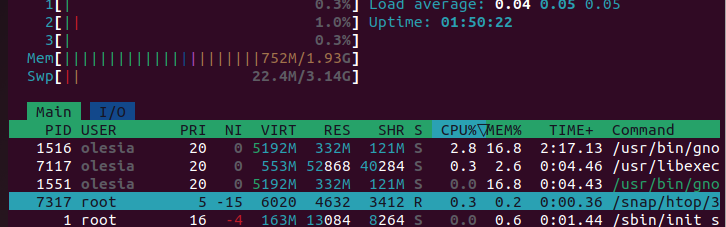
****

Рис. 8 Зміна пріоритету за допомогою htop в Linux

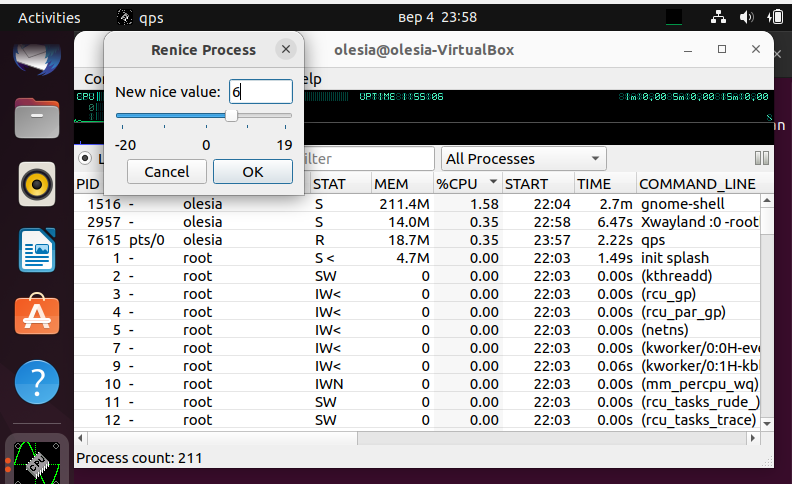
****

Рис. 9 Зміна пріоритету за допомогою qps в Linux

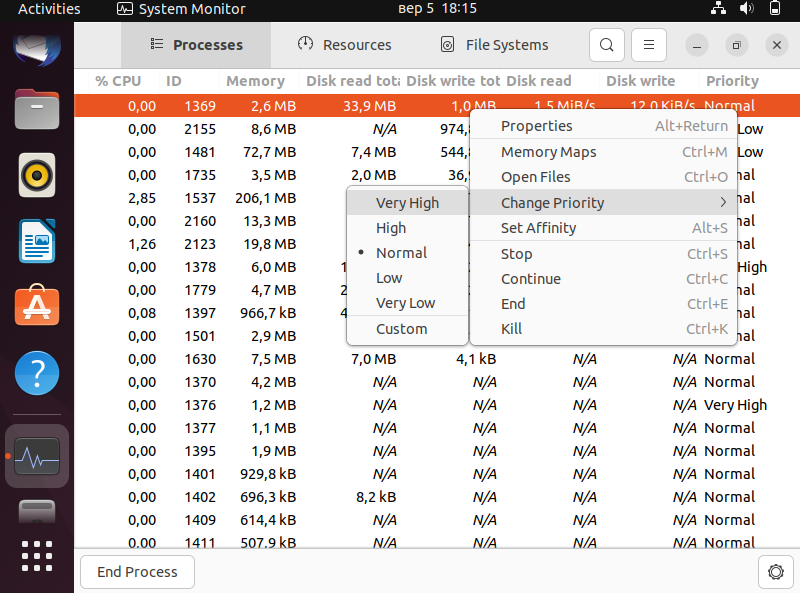


Рис. 10 Зміна пріоритету за допомогою System Monitor в Linux

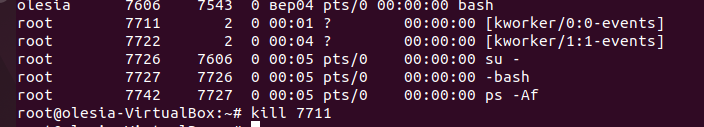
****

Рис. 11 Завершення процесу за допомогою консольних засобів в Linux

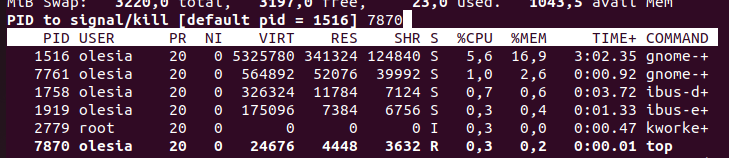
****

Рис. 12 Завершення процесу за допомогою top в Linux

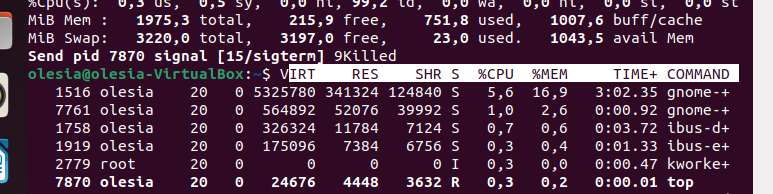
****

Рис. 13 Завершення процесу за допомогою htop в Linux

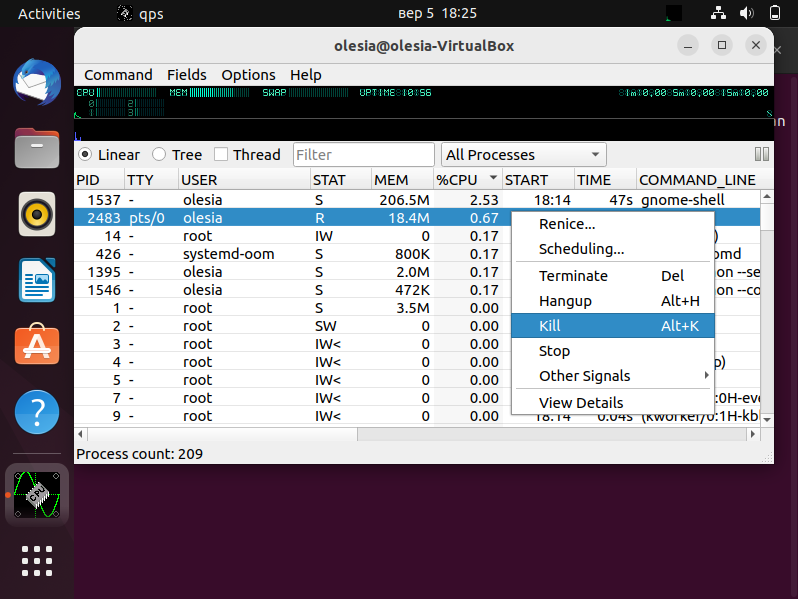


Рис. 14 Завершення процесу за допомогою qps в Linux

****

Рис. 15 Завершення процесу за допомогою System Monitor в Linux

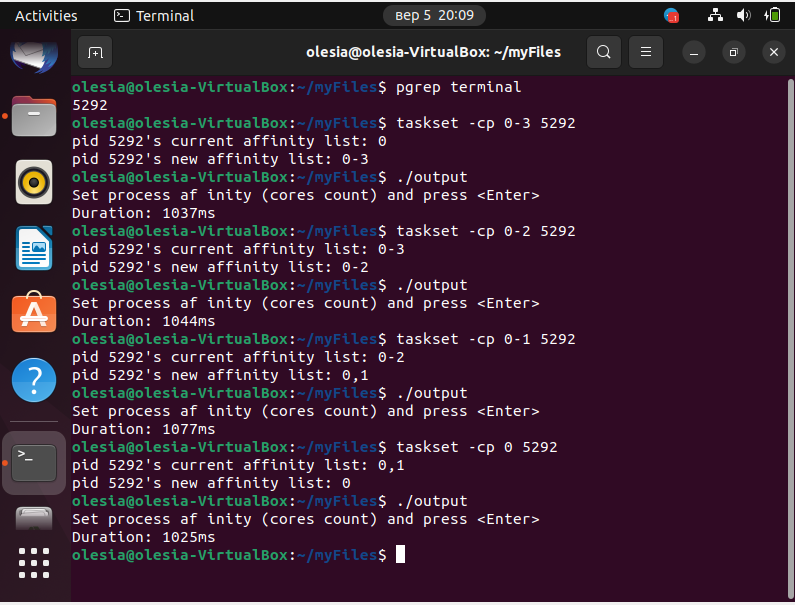
****

Рис. 16 Зміна ядер в консолі в Linux

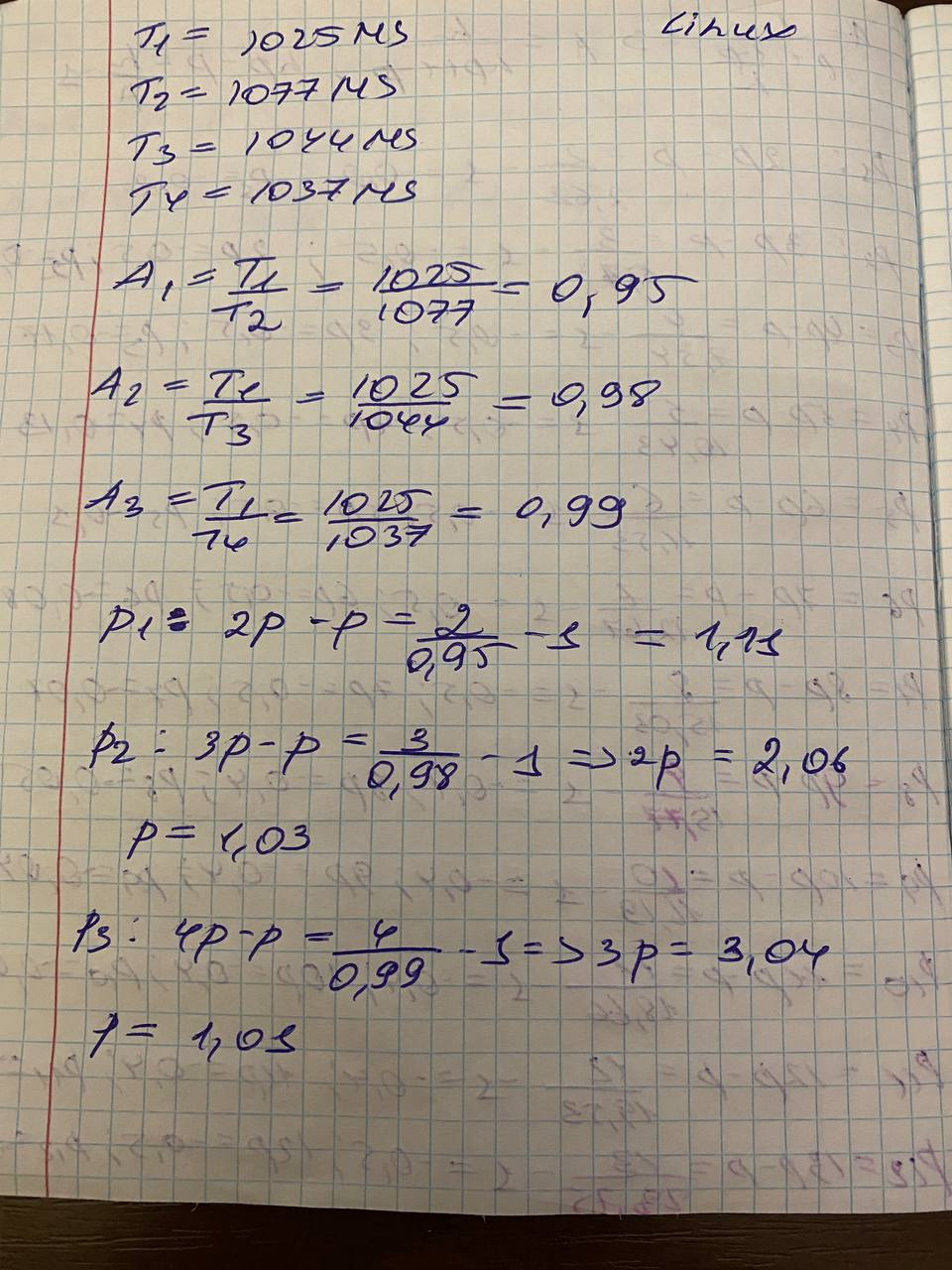


Рис. 17 Величини A, S, p для різної кількості ядер

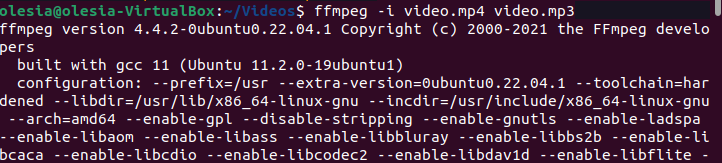


Рис. 18 Конвертування відео з mp4 до mp3 в Linux

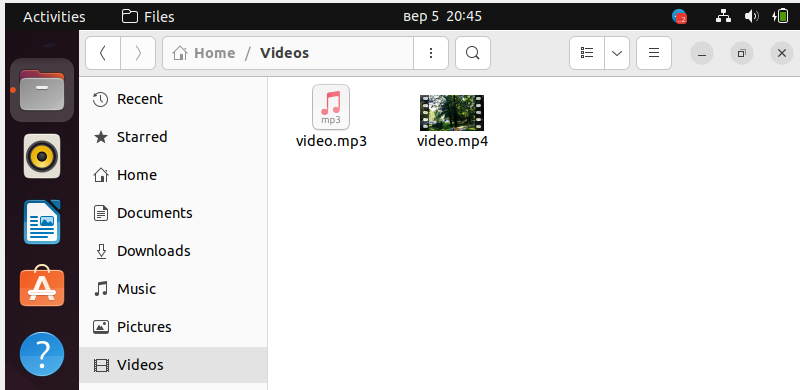


Рис. 19 Підтвердження конвертування в Linux

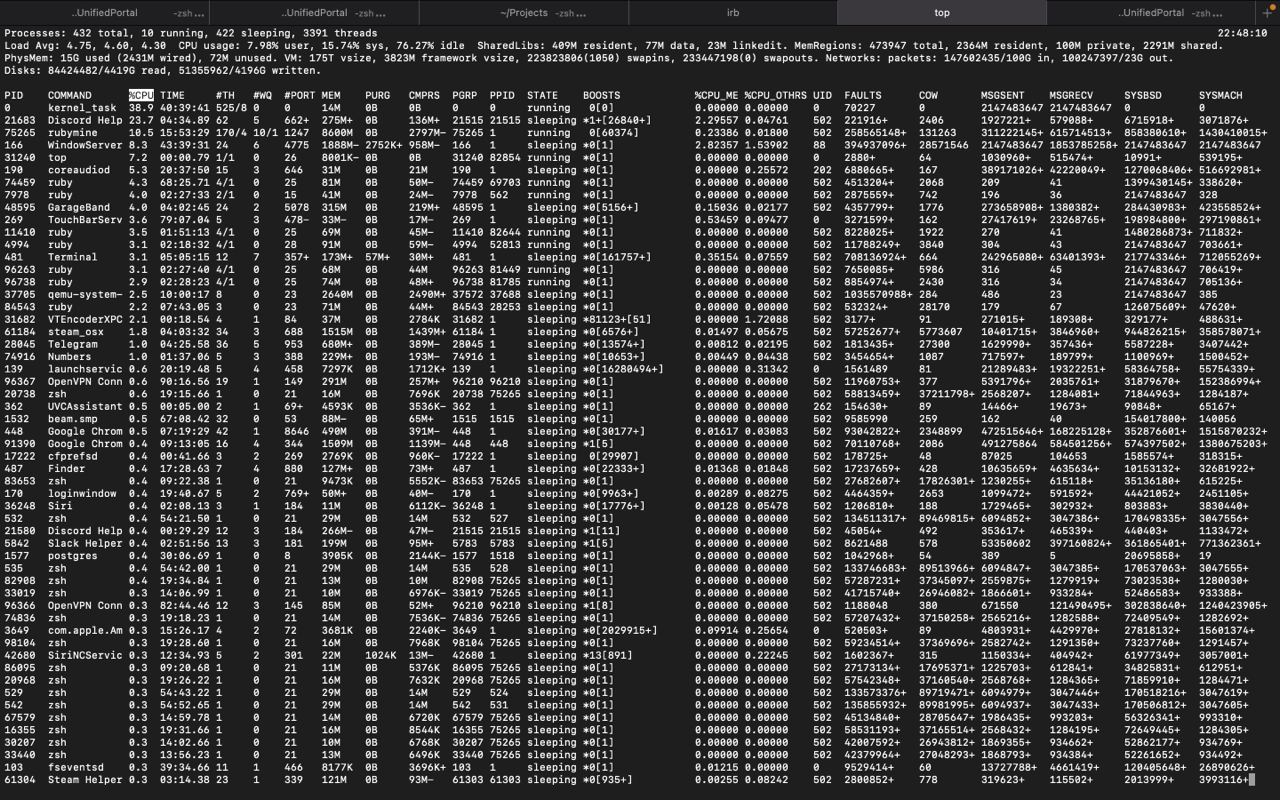


Рис. 20 Повна інформація про процеси за допомогою top в MacOS

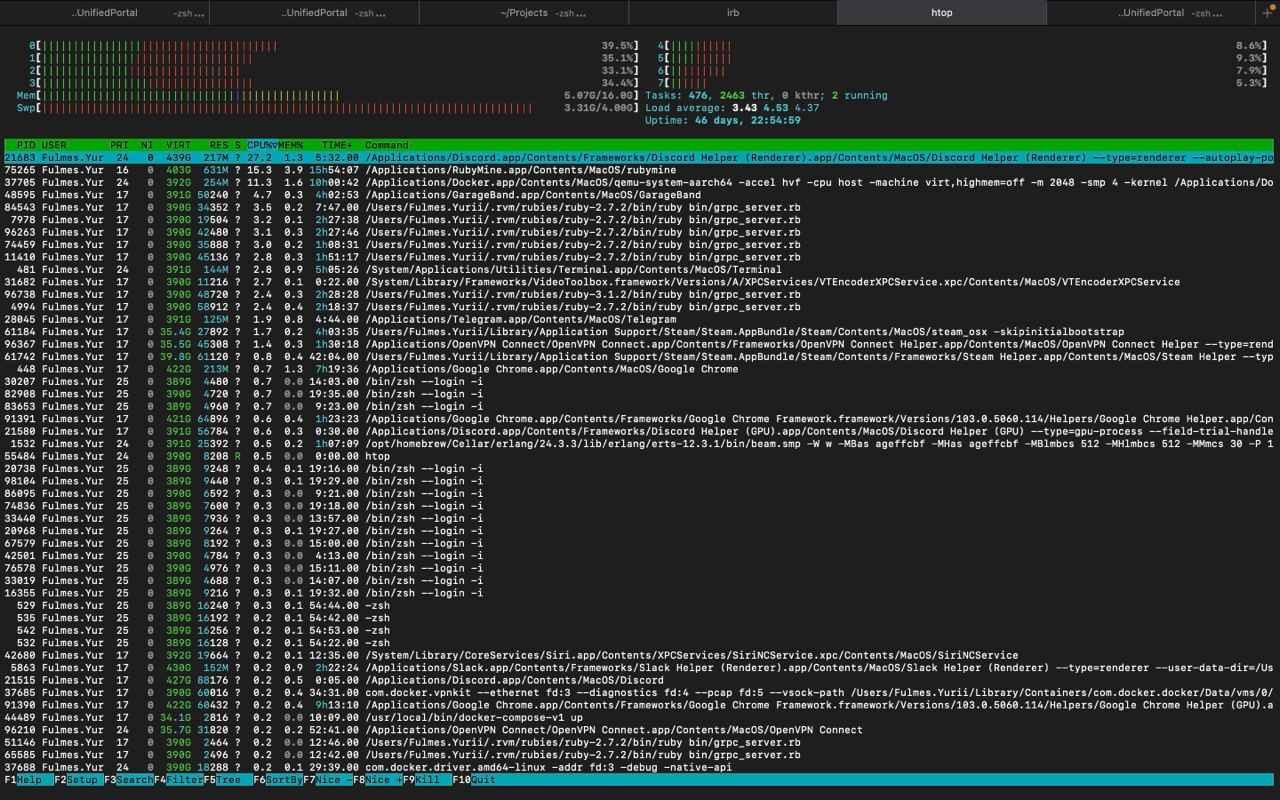


Рис. 21 Повна інформація про процеси за допомогою htop в MacOS

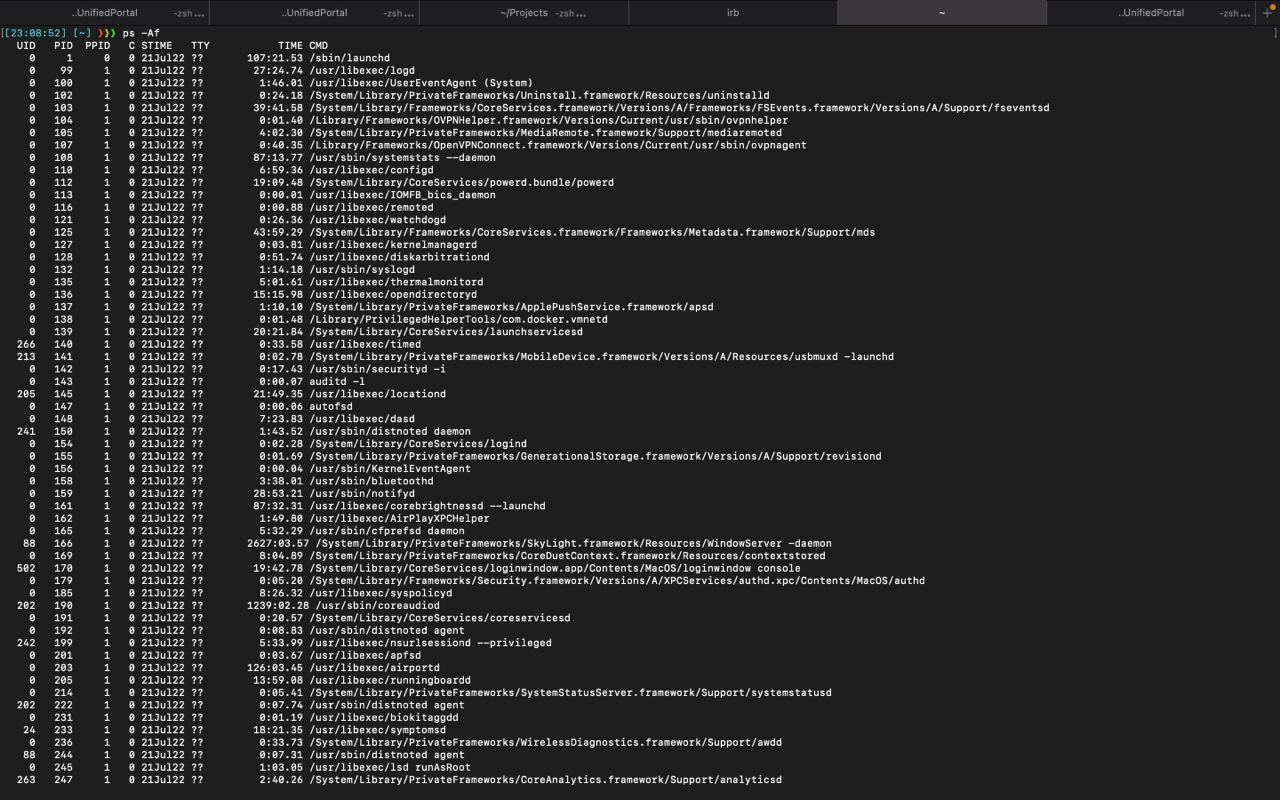


Рис. 22 Повна інформація про процеси за допомогою ps в MacOS

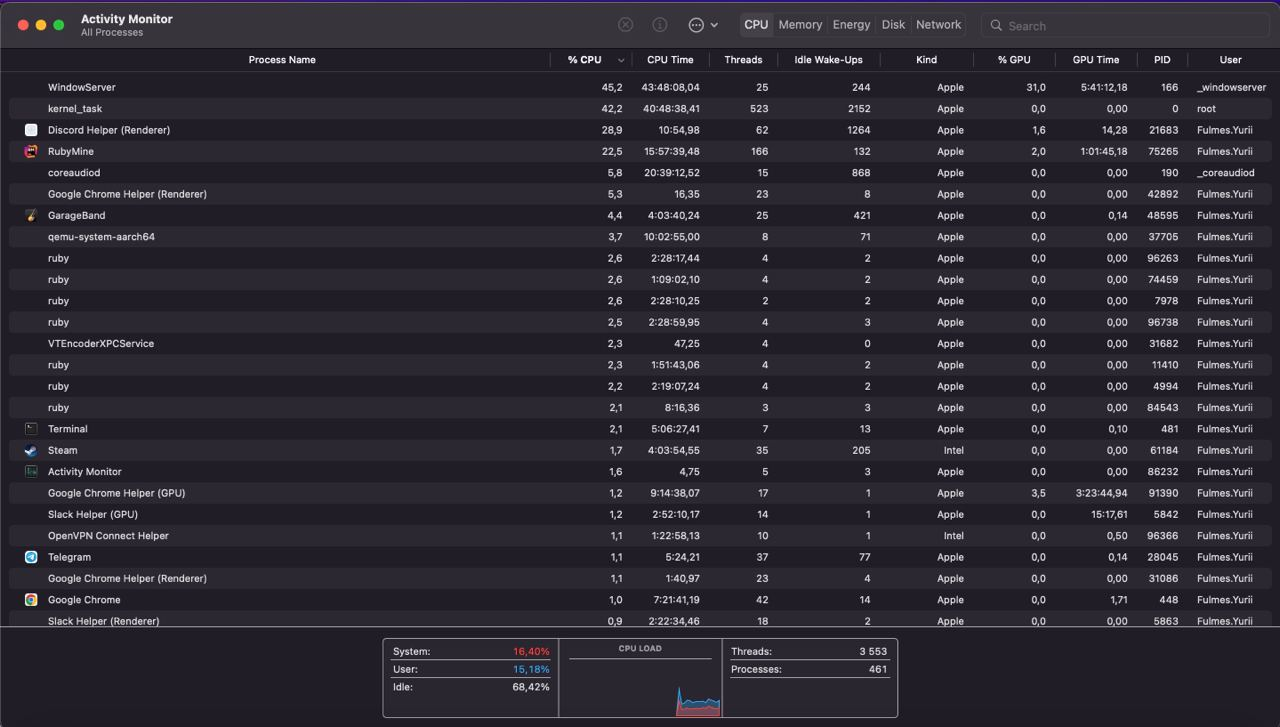


Рис. 23 Повна інформація про процеси за допомогою Activity Monitor в MacOS

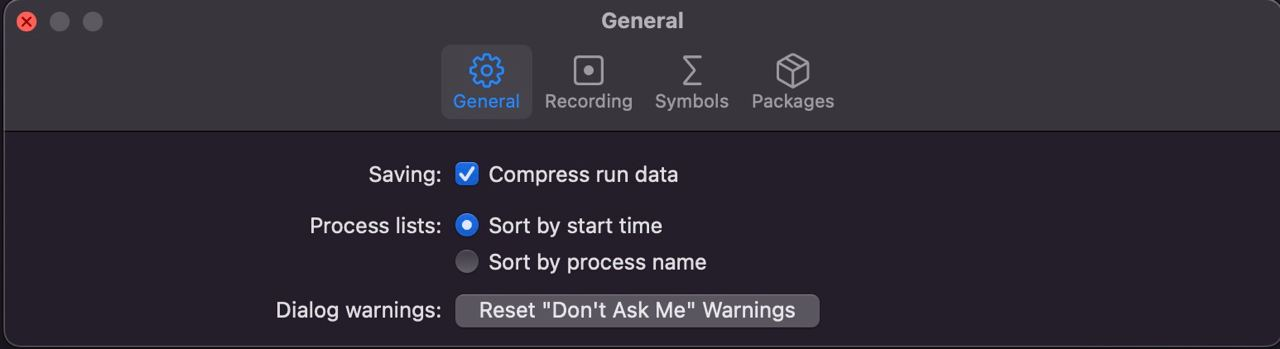


Рис. 24 Xcode панель налаштувань, де немає пункту CPUs, адже MacOS на процеорі M1, де неможливо змінити кількість ядер.

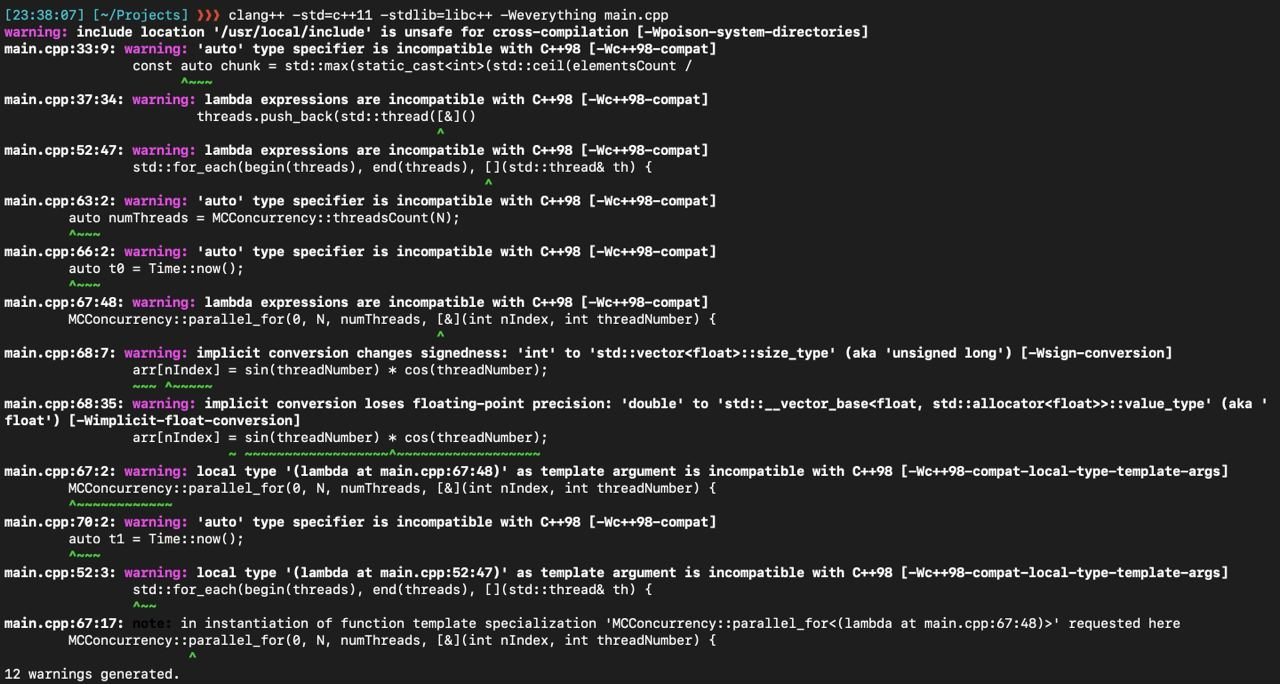


Рис. 25 Компіляція файлу у консолі за допомогою clang



Рис. 26 Запуск файлу на виконання і виведення результату.

**Висновок**

На цій лабораторній роботі я навчилась працювати з такими операційними системами як Linux i MacOs, дізналась як працювати з процесами на цих операційних системах та порівняла їхній функціонал – і на Linux і на MacOS є можливість дізнатись повну інформацію про процеси , змінювати параметри процесу, але на MacOS на процесорі M1, а не на процесорі Intel, що присутній у новіших версіях ноутбуків/комп’ютерів, немає можливості змінити кількість ядер процесу, відповідно немає можливості розрахувати A,p,S.