МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з навчальної дисципліни
«Операційні системи»
для студентів денної та заочної форми навчання

спеціальності 122 «Комп'ютерні науки»

Затверджено
на засіданні групи забезпечення спеціальності
Протокол № __6__ від «_6_» _02_2024р.

Голова групи _____ Кузніченко С.Д.

Затверджено
на засіданні кафедри _____
Протокол № __7__ від «_15_» _02__2024р.

Завідувач кафедри _____ Казакова Н.Ф.

Одеса 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з навчальної дисципліни
«Операційні системи»

для студентів денної та заочної форми навчання

спеціальності 122 «Комп'ютерні науки»

Затверджено
на засіданні ГЗС
Протокол № 6
від «6» 02 2024р.

Одеса – 2024

Методичні вказівки до лабораторних робіт з дисципліни „***Операційні системи***”, для студентів І року навчання денної та заочної форми за спеціальністю 122 «Комп’ютерні науки», рівень вищої освіти бакалавр /Терещенко Т.М., Гадяцький І.А. – Одеса, ОДЕКУ, 2024.

ЗМІСТ

ВСТУП

Метою дисципліни є підготовка фахівців з комп'ютерних наук в галузі сучасних методів, технологій та засобів обробки даних заснованих на використанні системного програмного забезпечення.

Учбовий курс присвячений вивченню понять та прийомів роботи в командному рядку та програмних оболонках операційної системи сімейства Windows та операційних систем Linux, вивченню основ написання.

Основні поняття, що входять до програми дисципліни зосереджені на структурі файлової системи, методах та інструментах роботи з пам'яттю, утилітах управління процесами та потоками.

У результаті вивчення дисципліни студенти повинні надбати:

знання:

- Про реалізацію сторінкової організації пам'яті.
- Про способи і алгоритми реалізації сегментації.
- Про віртуальну пам'ять процесорів Pentium та UltraSPARC.
- Про віртуальні команди вводу-виводу та способи їхньої реалізації.
- Про віртуальні команди для паралельної обробки.
- Про віртуальну пам'ять UNIX і Windows.
- Про віртуальний ввід-вивід у системах UNIX і Windows.
- Про керування процесами в системах UNIX і Windows.

уміння:

- Керувати розподілом оперативної пам'яті в операційних системах UNIX і Windows.
- Синхронізувати потоки та здійснювати обмін інформацією між ними.
- Працювати з файловими системами та здійснювати захист операційних систем.

компетентність:

- Здатність застосовувати знання у практичних ситуаціях.
- Здатність оцінювати та забезпечувати якість виконуваних робіт.
- Здатність забезпечити організацію обчислювальних процесів в інформаційних системах різного призначення з урахуванням архітектури, конфігурування, показників результативності функціонування операційних систем і системного програмного забезпечення.

Практична частина спрямована на вивчення питань створення скриптів та отримання навичок створення та запуску bash-скриптів для роботи з

операційними системами.

Для успішного виконання лабораторних робіт необхідно встановити та налаштувати програмне забезпечення, що зазначено в розділі "Встановлення та налаштування програмного забезпечення". Цей процес забезпечує правильне функціонування програм та допомагає у виконанні завдань згідно з вимогами курсу.

Дані методичні вказівки до виконання лабораторних робот містять теоретичні відомості та методику виконання 5 лабораторних робіт з дисципліни, які входять до практичного модулю П1:

Лабораторна робота №1 – Основи роботи в командному рядку операційної системи сімейства Windows.

Лабораторна робота №2 – Основи роботи в терміналі операційної системи сімейства Linux.

Лабораторна робота №3 – Створення та запуск bash-скриптів.

Лабораторна робота №4 – Управління процесами та потоками в операційних системах Windows і Linux.

Лабораторна робота №5 – Управління пам’яттю в операційних системах Windows і Linux.

Після вивчення ЗМ–П1 студент повинен вміти: використовувати сучасні методи, технології та засоби роботи з операційними системами UNIX і Windows для вирішення практичних задач.

Контролюючим заходом передбаченим для цього змістовного модуля є усне опитування. По кожній лабораторній роботі студент повинен скласти звіт, якій містить в собі:

- назву роботи,
- мету роботи,
- загальне та індивідуальне завдання згідно варіанта,
- послідовний алгоритм розв’язання задачі, який обов’язково ілюструється екранними формами з поясненнями що до виконаних дій,
- текст основних програмних модулів,
- відповіді на контрольні питання.

Варіант індивідуального завдання узгоджується з викладачем. Оформлений звіт захищається студентом усно. Студент повинен чітко і правильно відповідати на контрольні питання, які оголошені наприкінці кожної лабораторної роботи. Виконана та захищена лабораторна робота оцінюється згідно з умовами, які викладені в силабусі дисципліни.

ПРАВИЛА ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНА ПРАЦІ

Лабораторні роботи з дисципліни проводяться у лабораторіях кафедри інформаційних технологій або кафедри АСМНСІ, які оснащені комп'ютерною технікою з відповідним програмним забезпеченням. Студенти зобов'язані дотримуватися правил техніки безпеки та правил користування обчислювальною технікою в лабораторіях кафедр.

Згідно з «Правилами техніки безпеки в лабораторіях» студентам забороняється:

- з`являтися та знаходитись приміщенні в нетверезому стані;
- ставити поруч з клавіатурою ємності з рідиною;
- перебувати в приміщенні у верхньому одязі та завалювати ним робочі столи та стільці;
- працювати в лабораторії більше 6–ти годин на день (для вагітних жінок більше 4–х годин);
- за власною ініціативою змінювати закріплені за ними робочі місця та знаходитись в приміщенні під час роботи іншої учбової групи;
- самостійно виконувати вмикання електроживлення лабораторії та заміну складових частин ПК, що вийшли із ладу.

У випадку виявлення несправностей обчислювальної техніки студент повинен сповістити про це викладача чи будь–кого з навчально–допоміжного персоналу лабораторії.

ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Встановлення дистрибутиву Ubuntu через компоненти WSL

Компонент WSL за замовчуванням вимкнено у Windows. Сучасні дистрибутиви Windows 10 і Windows 11 для встановлення середовища WSL достатньо виконати команду:

wsl --install

Так команда автоматично включить всі необхідні компоненти Windows, виконає необхідні роботи WSL, встановить оновлення ядра Linux для WSL2, завантажить дистрибутив Ubuntu (за замовчуванням) і встановить їх у WSL.

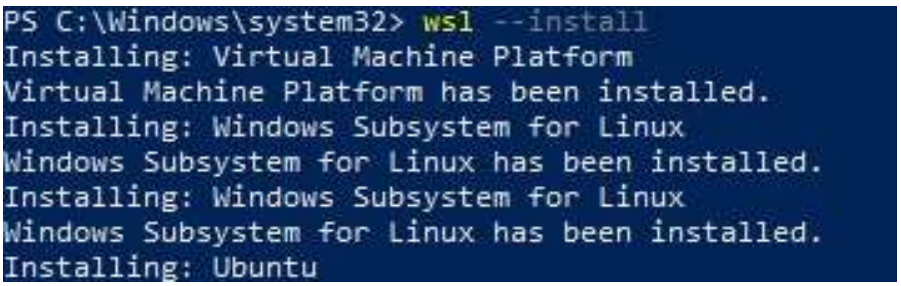


Рисунок 1 – Процес роботи команди wsl --istall

Наступним етапом необхідно перезавантажити комп'ютер та встановити дистрибутив Linux. Виведіть список доступних дистрибутивів командою:

wsl --list --online

Вкажіть ім'я дистрибутива Linux, який можна встановити у WSL, наприклад:

wsl --install -d Ubuntu-22.04

```
PS C:\Windows\system32> wsl -l -o
The following is a list of valid distributions that can be installed.
Install using 'wsl --install -d <Distro>'.

NAME                                   FRIENDLY NAME
Ubuntu                                 Ubuntu
Debian                                 Debian GNU/Linux
kali-linux                             Kali Linux Rolling
Ubuntu-18.04                           Ubuntu 18.04 LTS
Ubuntu-20.04                           Ubuntu 20.04 LTS
Ubuntu-22.04                           Ubuntu 22.04 LTS
OracleLinux_7_9                        Oracle Linux 7.9
OracleLinux_8_7                        Oracle Linux 8.7
OracleLinux_9_1                        Oracle Linux 9.1
openSUSE-Leap-15.5                     openSUSE Leap 15.5
SUSE-Linux-Enterprise-Server-15-SP4    SUSE Linux Enterprise Server 15 SP4
SUSE-Linux-Enterprise-15-SP5           SUSE Linux Enterprise 15 SP5
openSUSE-Tumbleweed                    openSUSE Tumbleweed
```

Рисунок 2 – Список доступних дистрибутивів Linux для WSL

Встановлення дистрибутиву Ubuntu на VirtualBox 7.0

1. Встановлення програмного забезпечення

Перейдіть за посиланням: https://www.virtualbox.org/ та натисніть кнопку «Download VirtualBox 7.0».

Рисунок 3 – Головна сторінка www.virtualbox.org

Необхідно обрати актуальний пакет з урахуванням наявної операційної системи та дочекатися завантаження файлу установки.

Рисунок 4 – Сторінка вибору пакету для завантаження

Далі необхідно запустити файл від імені адміністратора та встановити програмне забезпечення.

Рисунок 5 – Встановлення програмного забезпечення VirtualBox 7.0

2. Завантаження образу Ubuntu

Для завантаження образу необхідно перейти за посиланням: https://ubuntu.com/download. У верхній частині сторінки в пункті «Download» у списку, що випадає, необхідно обрати розділ «Ubuntu Desktop» та в ньому останню стабільну версію (зеленого кольору), натиснути та дочекатися завершення завантаження.

Рисунок 6 – Встановлення дистрибутиву Ubuntu

3. Створення віртуальної машини

Після встановлення VirtualBox 7.0 необхідно додати ОС (кнопка «Додати»). Далі вказати ім’я системи, наприклад, Ubuntu, обрати посилання, де буде зберігатись віртуальна машина, обрати образ, який завантажили раніше та натиснути «Next».

На цьому етапі необхідно створити ім’я користувача та пароль для входу в систему, після чого натиснути «Next». Задати об’єм оперативної пам’яті та кількість процесорів, бажано використовувати від 2Гб оперативної пам’яті та 1 ЦП. Задати об’єм пам’яті, враховуючи, що для коректної роботи системи необхідно від 32 Гб.

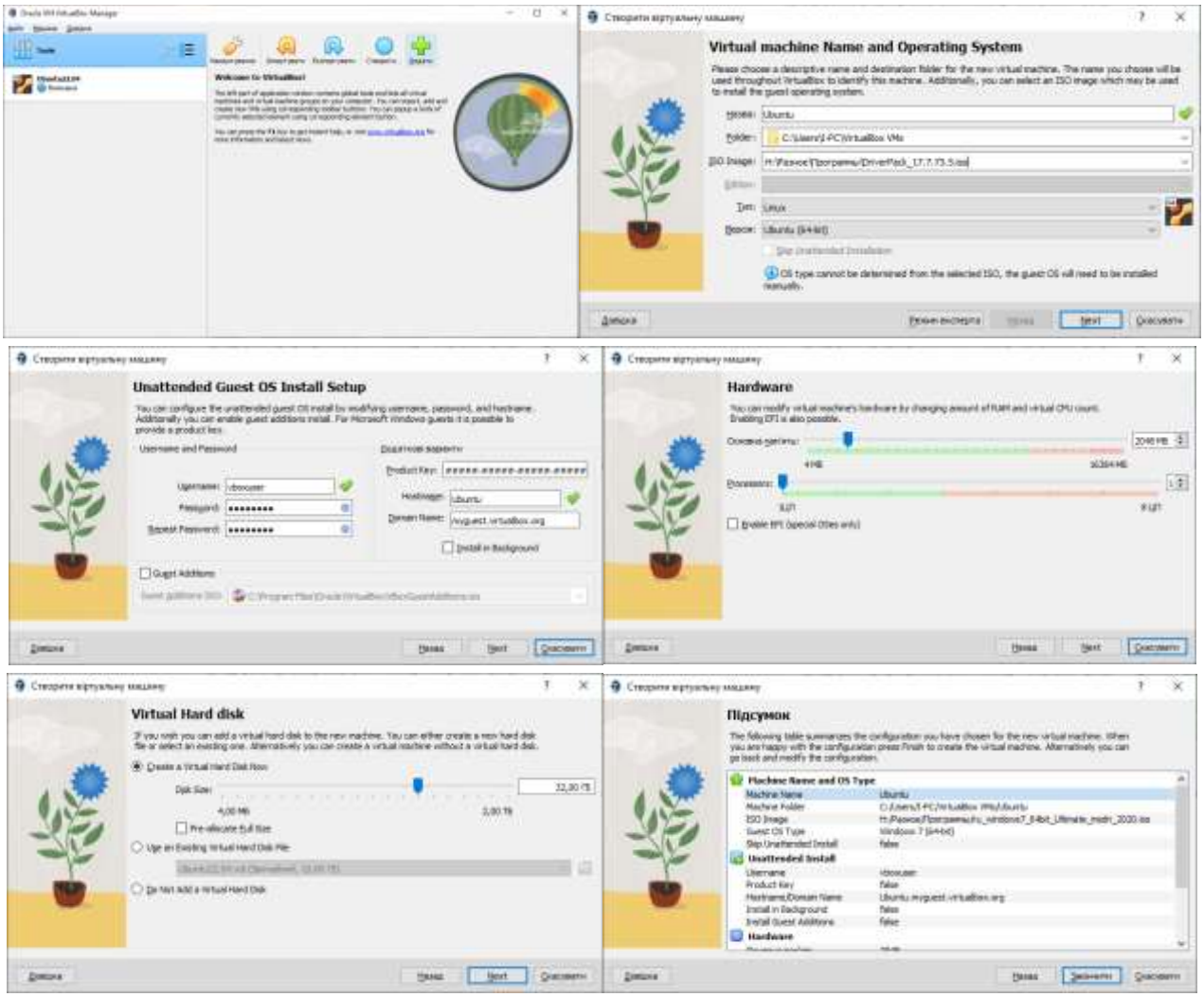


Рисунок 7 – Додавання нової ОС до VirtualBox 7.0

На цьому етапі маємо можливість передивитися всі налаштування системи, при необхідності повернутися на минулі етапи та змінити

налаштування. Для завершення процесу установки натиснути кнопку «Закінчити».

Після успішного створення на головному екрані програми з’явиться віртуальна машина та запуститься система.

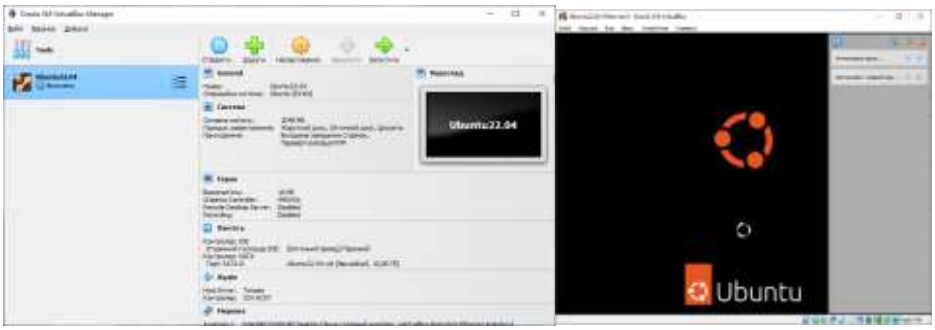


Рисунок 8 – Перший запуск Ubuntu

4. Налаштування віртуальної машини

Для доступу до параметрів налаштування в головному меню необхідно обрати створену віртуальну машину та натиснути кнопку «Налаштування». Розділ «Система» необхідний для зміни об’єму оперативної пам’яті, зміни порядку завантаження віртуальних пристроїв, увімкнення/вимкнення IO-APIC (потрібна для використання декількох ядер процесора). Розділ «Процесор» необхідний для зміни кількості процесорів та ядер, а також увімкнення/вимкнення PAE/NX (дозволяє 32-бітним системам використовувати до 64 Гб оперативної пам’яті). Розділ «Мережа» необхідний для додавання віртуального мережевого хосту. У пункті «Під’єднаний до» слід обрати «Лише головний адаптер», «Назва» за замовчуванням та у розділі «Додатково» нічого не змінювати.

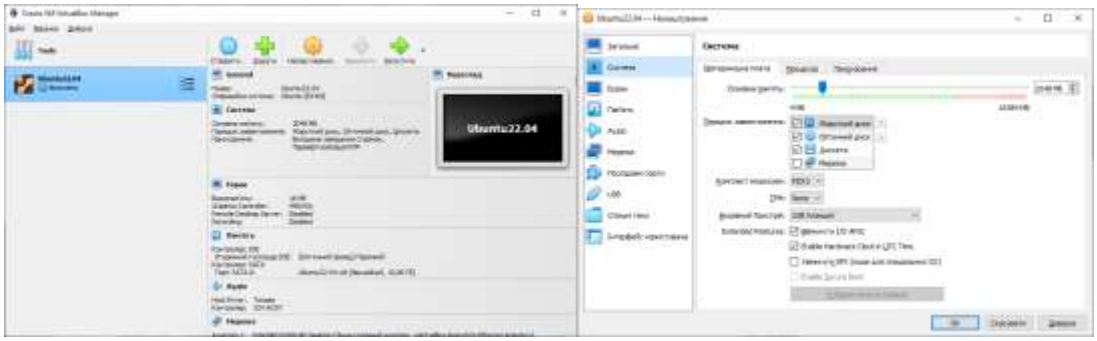


Рисунок 9 – Налаштування Ubuntu (розділ система)

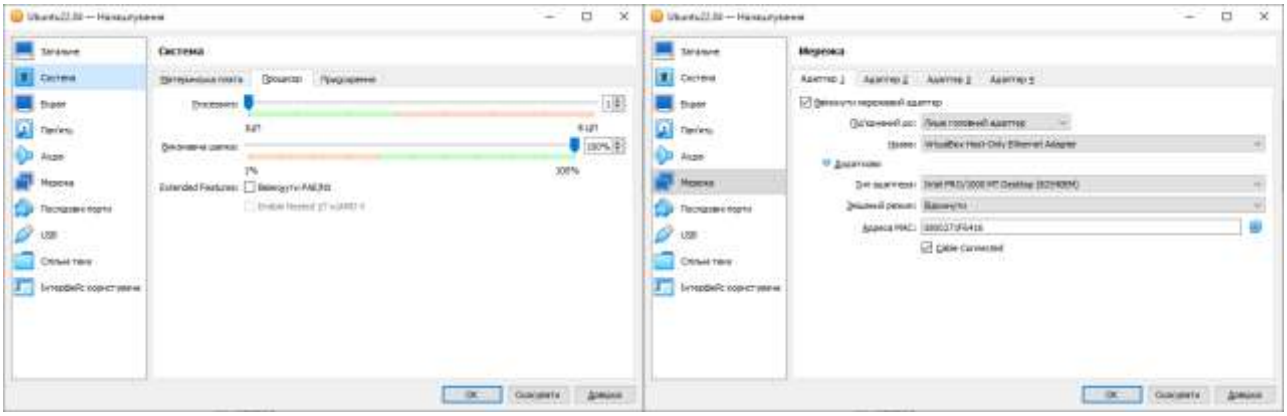


Рисунок 10 – Налаштування Ubuntu (розділи процесор та мережа)

5. Перший запуск віртуальної машини

Для запуску віртуальної машини у головному меню слід натиснути кнопку «Запустити» та очікувати завантаження. В процесі запуску проводиться вставлення необхідних компонентів системи. Якщо вставлення необхідних компонентів системи виконано успішно, то система перезавантажиться та з’явиться екран вибору користувачів. Логін та пароль були створені в 3 пункті (стор. 11). (login: vboxuser pass: changeme)

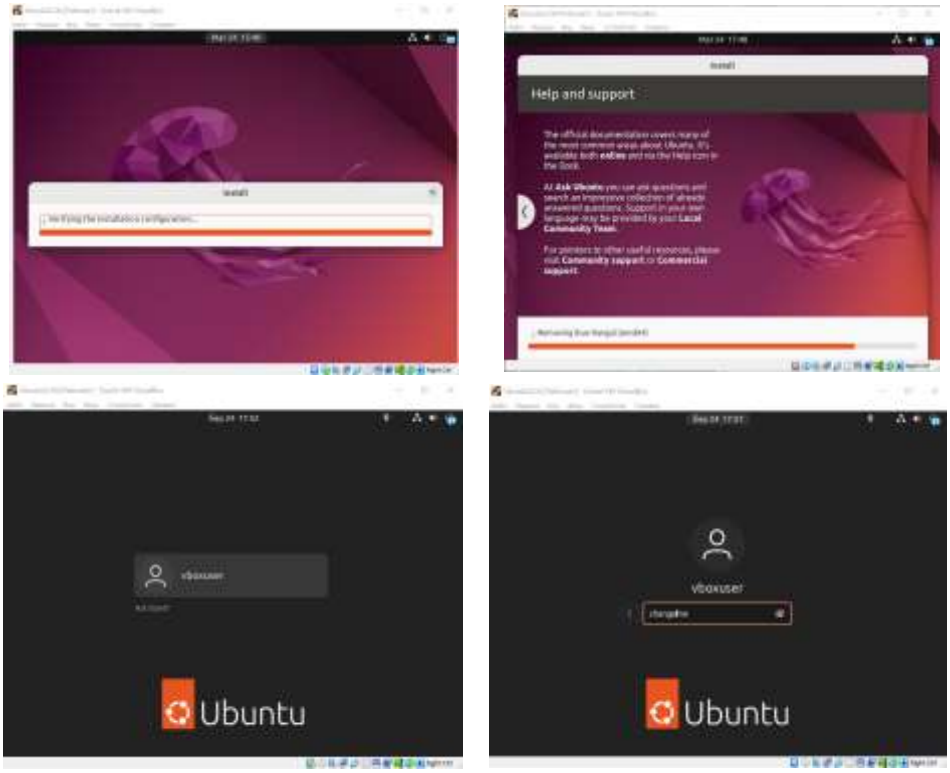


Рисунок 11 – Фінальне налаштування системи Ubuntu

ЛАБОРАТОРНА РОБОТА №1

Основи роботи в командному рядку операційної системи сімейства Windows

Мета роботи: ознайомлення і вивчення елементарних понять та прийомів роботи з файлами та каталогами в командному рядку операційної системи сімейства Windows.

Постановка завдання: створити дерево каталогу засобами командного рядка, створити текстові файли із заданим текстом, перейменувати створені файли та перемістити їх в задану папку.

Теоретичні відомості

Командний рядок Windows – це окреме ПЗ, яке входить до складу операційної системи (ОС) та забезпечує взаємозв'язок між користувачем та ОС. З його допомогою можна виконувати команди MS-DOS та інші комп'ютерні команди. Основна перевага командного рядка полягає в тому, що він дозволяє вводити всі команди без участі графічного інтерфейсу, що є набагато швидшим і має масу додаткових можливостей, які не можуть бути здійснені в графічному інтерфейсі.

Командний рядок запускається у своїй оболонці та призначений для більш досвідчених користувачів. Він допомагає у таких складних ситуаціях, коли інші команди вже не працюють. Наприклад, через командний рядок вводять команди у разі зараження вірусами або "поломки" системних файлів, а також відновлення Windows.

Методи запуску командного рядка:

1) Пуск - Все программы - Стандартные - Командная строка

2) Пуск - Выполнить - вводим cmd.exe

3) Win + R - вводим cmd.

4) Запустити файл cmd.exe, який знаходиться в системній папці.

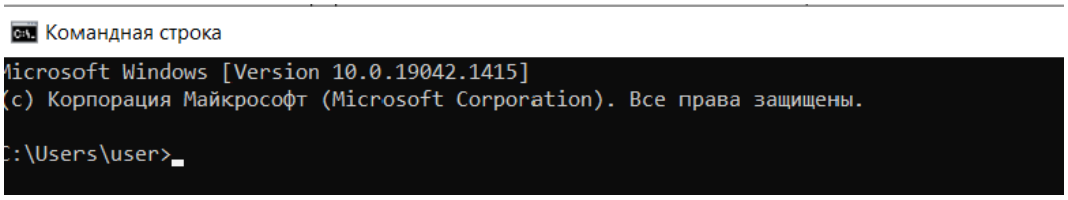


Рис. 1.1 – Вікно інтерпретатора Windows

Для відкриття вікна установки властивостей консолі командного рядка слід клацнути правою клавішею миші в верхньому полі вікна, відкриється додаткове вікно (рис. 1.2), в якому слід встановити потрібні налаштування.

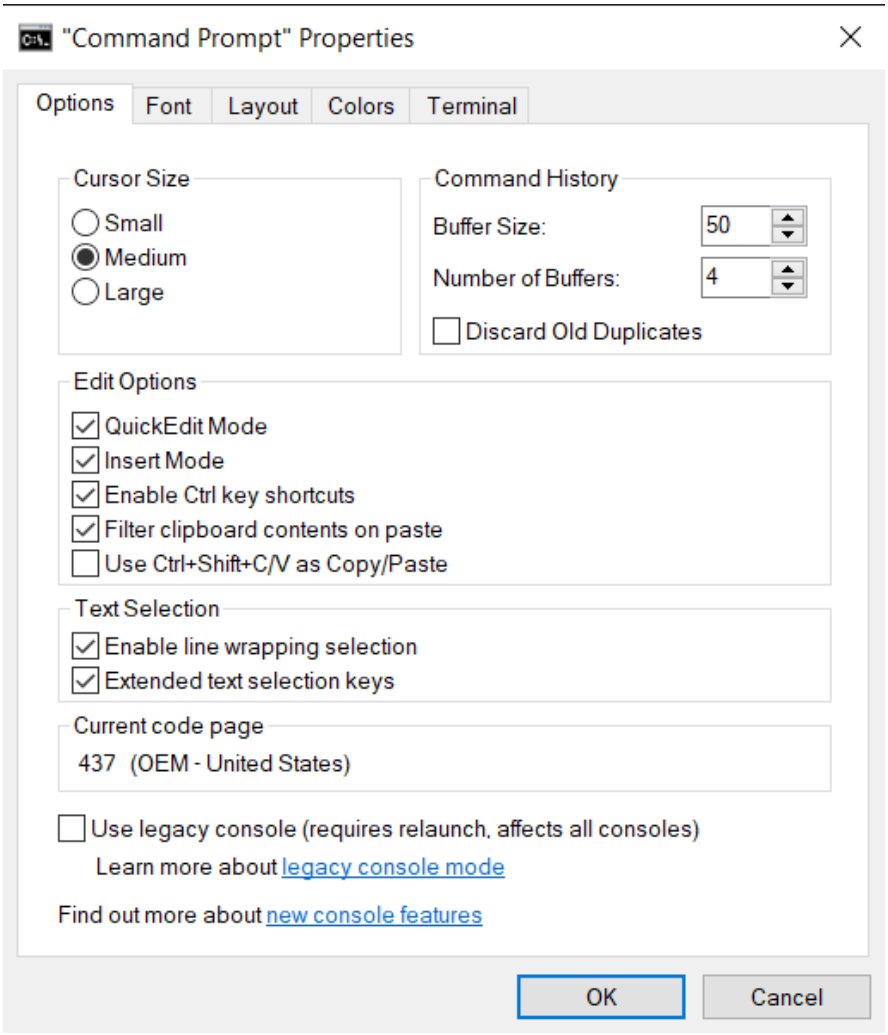


Рис. 1.2 – Вікно «Властивості» командного рядка

Командний рядок має приблизно 100 команд, їх умовно можна поділити на групи: мережеві, системні та для виклику системних утиліт. Розглянемо деякі з них.

1. Мережеві команди.

ipconfig/all – відображення повної інформації всіх адаптерів мережі.

getmac – отримати MAC-адресу мережевих карт.

arp -a – відобрадення arp таблиці.

ping [кінцевий_вузол] [ключ] – продзвонити мережеву адресу.

Якщо в командному рядку набрати команду ping/?, то з’явиться перелік всіх параметрів цієї команди з поясненнями (рис.1.3).

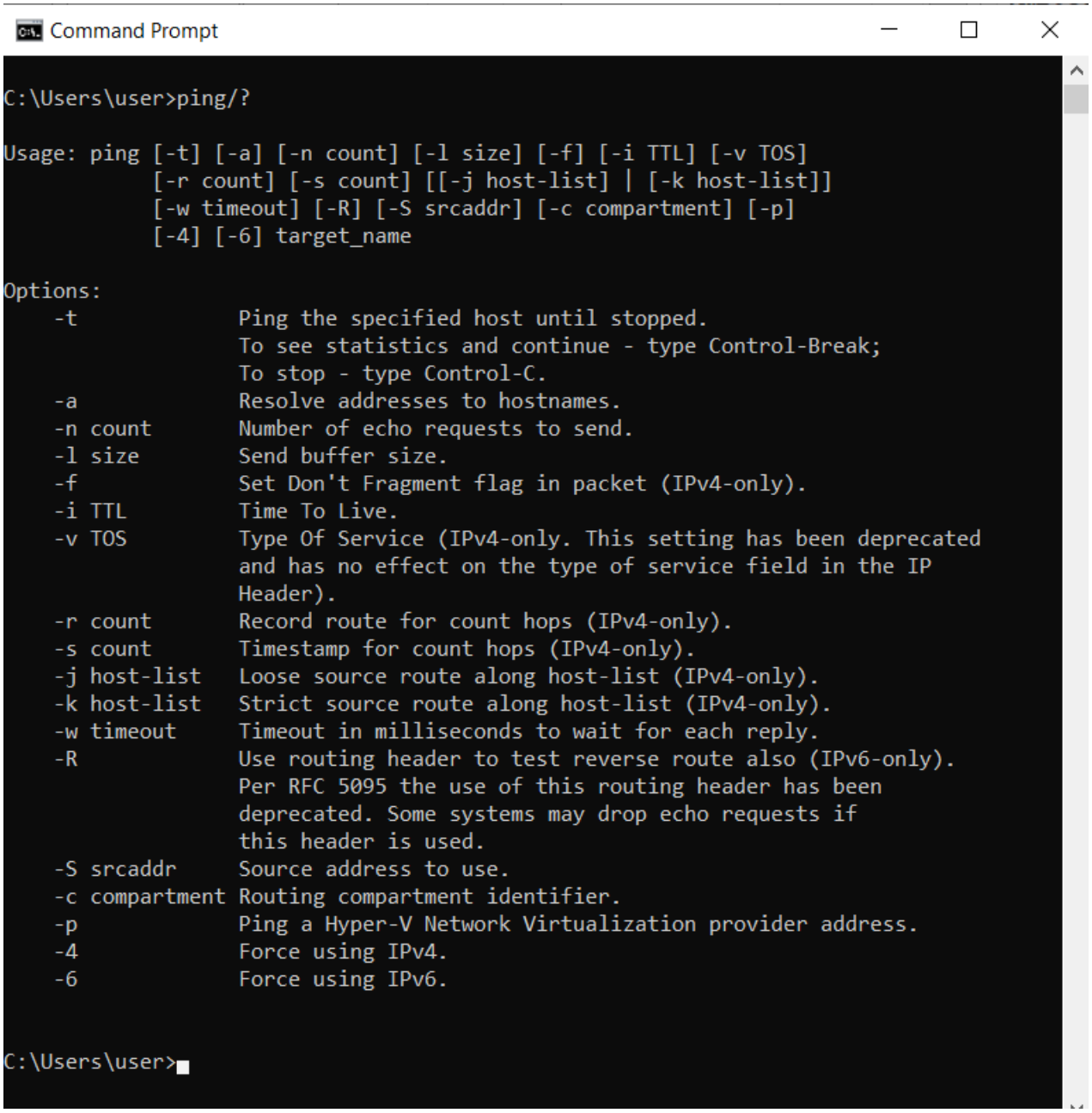


Рис. 1.3 – Результат виконання команди ping/?

tracert [кінцевий_вузол] – трасування маршруту..

pathping [кінцевий_вузол] – обєднує в собі команди pathping та tracert.

netstat [ключ] – відображення статистики поточних мережевих підключень TCP/IP.

Аналогічно команді ping, якщо в командному рядку набрати команду netstat/?, то з’явиться перелік всіх параметрів цієї команди з поясненнями (рис. 1.4).

Рис. 1.4 – Результат виконання команди netstat/?

netsh interface ip show address – відображення поточної конфігурації ip, маски та шлюзу.

netsh interface ip set address name="[і'мя_мережевого_інтерфейсу]" static [ip адреса] [маска] [шлюз] – встановлення ip, маски та шлюзу.

netsh interface ip show dnsservers – відображення поточної конфігурації dns-серверов.

netsh interface ip set dnsserver "[і'мя_мережевого_інтерфейсу]" static [встановлений_dns-сервер] – встановлення dns-сервера.

netsh interface ip add dnsserver "[і'мя_мережевого_інтерфейсу]" [альтернативний_dns-сервер] index=2 – установка альтернативного dns-серверу.

route -p add [адреса_мережі] mask [маска] [шлюз] – додавання статичного маршруту.

route delete [адреса_мережі] – видалення маршруту.

route print – відображення таблиці маршрутів.

nslookup – DNS клієнт.

ftp [адреса_сервера] – ftp клієнт.

2. Системні команди.
shutdown/r – перезавантаження комп’ютера.
shutdown/s – вимкнення комп'ютера.
qprocess * – список всіх процесів.
chcp – поточне кодування.
chcp [кодування] – зміна кодування (866 – dos, 1251 – windows1251, 65001 – UTF-8).
sfc/scannow – перевірка та відновлення системних файлів.
wuauclt – управління оновленнями Windows.
[команда] > c:\file.txt – перенаправлення виводу в файл.
[команда] & [команда] – послідовне виконання команд.
[команда]/? – короткий опис команди, перелік всіх параметрів.
help – список основних команд (рис. 1.5).

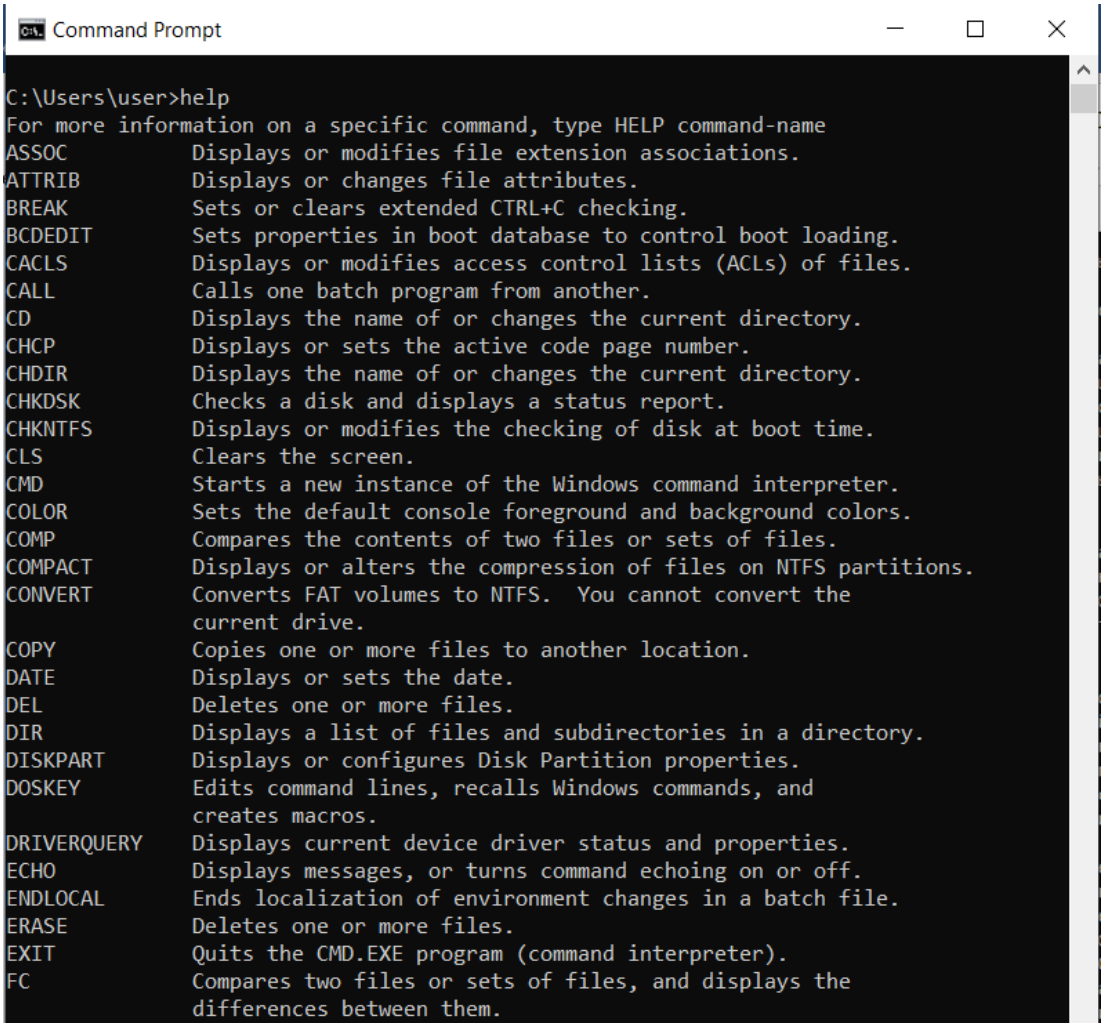


Рис. 1.5 – Результат виконання команди help

3. Команди виклику системних утиліт.
cttune – налаштування згладжування шрифтів.
compmgmt.msc – управління комп’ютером.
calc – виклик калькулятора.
charmap – таблиця символів.
chkdsk – утиліта перевірки дисків.
control – запуск панелі управління.
control color – властивості екрану – оформлення.
control desktop – властивості екрану.
control folders – властивості папки.
devmgmt.msc – диспетчер пристроїв.
diskmgmt.msc – управління дисками.
dxdiag – засіб діагностування direcx.
dfrg.msc – дефрагментація дисків.
eventvwr.msc – перегляд подій.
eudcedit – редактор особистих символів.
explorer – запуск Explorer.
fsmgmt.msc – спільні папки.
gpedit.msc – групова політика.
iexplore – запуск Internet Explrorer.
lusrmgr.msc – локальні користувачі.
mmc – виклик консолі.
mstsc – підключенні до віддаленого робочого столу.
msconfig – конфігурація системи.
notepad – запуск Блокнот.
osk – запуск додатку екранної клавіатури.
perfmon.msc – системний монітор.
powercfg – налаштування електроживлення ПКю
regedit – редактор реєстру.
services.msc – служби windows.
shrpubw – мастер створення спільної папки.
taskmgr – запуск диспетчера задач.
wmimgmt.msc – управління WMI.

Робота з файловою системою Windows.

Практично вся інформація на комп'ютерах представлена у вигляді файлів. Файл є основною одиницею зберігання даних та програм, що обробляють ці

дані. Файл – це названа (має ім'я) область зовнішньої пам'яті. Зазвичай файли тимчасово або постійно зберігаються у зовнішній пам'яті комп'ютера – на дисках, USB-флеш-носіях тощо. Крім імені файли характеризуються цілою низкою атрибутів, таких як розмір, час створення ін.

Операційна система та прикладні програми (додатки) отримують доступ до файлу за допомогою його імені. Максимальна довжина імені файлу або каталогу в Windows 256 символів, включаючи розширення. Ім'я та розширення розділяються точкою. Розширення вказує на вид інформації або на програму, якою може бути відкритий цей файл, наприклад, myfile.txt – текстовий файл, myfile.doc – документ MS Word та ін.

Файли зберігаються у системі вкладених каталогів (директорій) і організуються у файлову систему. Таким чином, файловою системою називається сукупність файлів та каталогів, організованих у деревоподібну структуру (рис. 1.6).

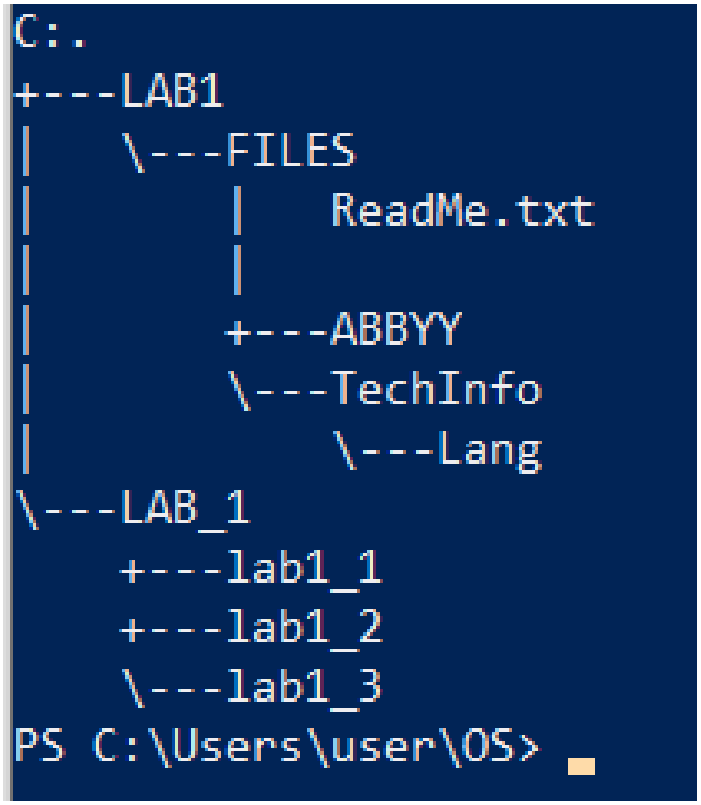


Рис. 1.6 – Дерево каталогів папки OS створене засобами командного рядка

Для створення структури каталогів за заданим деревом використовується команда **md [диск:]путь** (рис. 1.7).

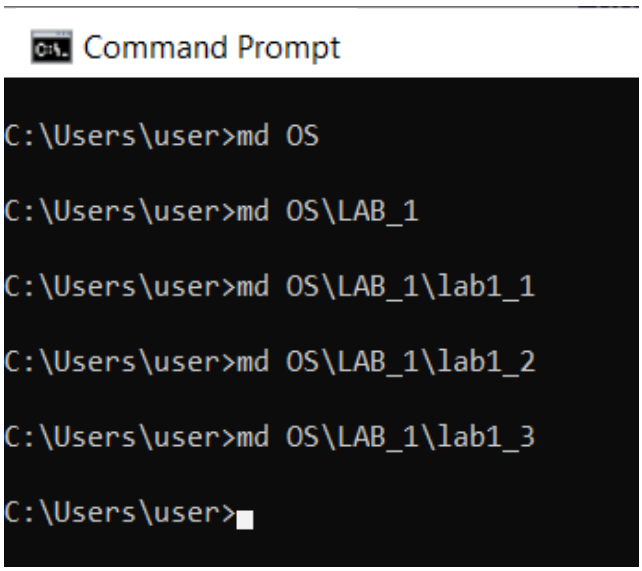


Рис. 1.7 – Створення папок на диску командою md

Після виконання вказаних команд отримає наступну структуру каталогів в папці користувача (рис. 1.8).

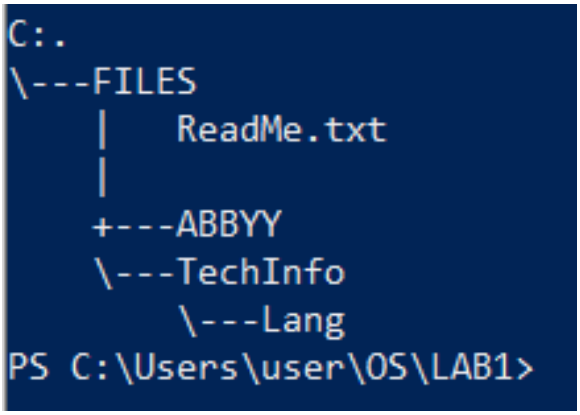


Рис. 1.8 – Дерево каталогів папки OS після виконання команд md

Послідовно командою md створюємо всі папки, які входять до структури наведеної на рис. 1.6.

Для створення текстових файлів використовується команда copy con [ім’я файла]. Після натискання <Enter> треба ввести послідовність символів, які будуть збережені в файлі. Для завершення вводу текстової інформації в файл та закриття файлу використовують послідовності F6 <Enter> або Ctrl+Z. Команда copy con копіює з консолі набір символів в файл.

В командному рядку можливо використання інших ‘гарячих’ клавіш для прискорення та полегшення роботи. Наприклад, <TAB> для автодоповнення

команди, <↑> і <↓> для навігації по командах, які вже вводилися в командному рядку.

Створимо файл lab1.txt та внесемо в нього прізвище, ім’я та номер групи так, яка показано на рис. 1.9.

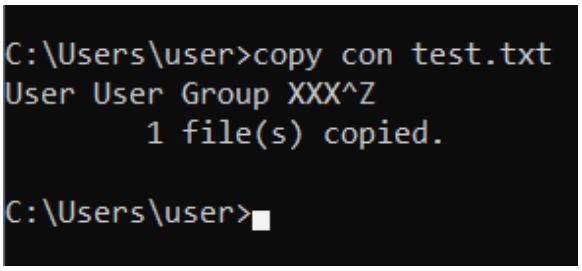


Рис. 1.9 – Створення текстового файлу

Після виконання зазначеної команди в папці C:\Users\user\ буде створено файл test.txt, який містить задану послідовність символів (рис. 1.10).

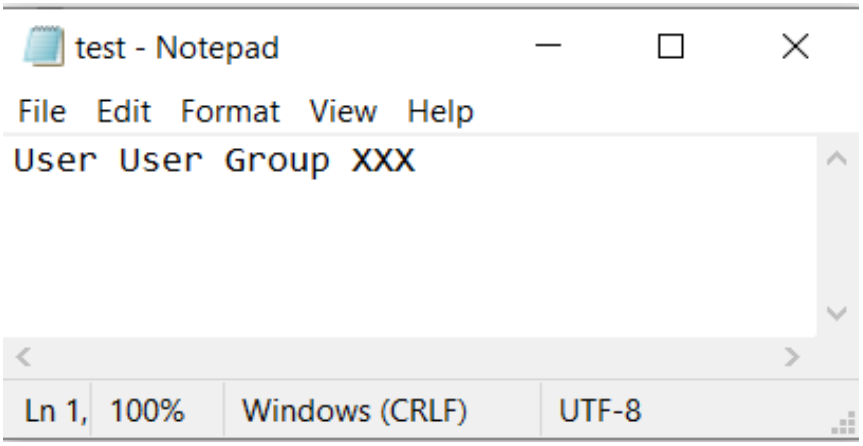


Рис. 1.10 – Вміст текстового файлу lab1.txt

Для відображення вмісту файлу в консоль використовуємо команду type ім’я файла. Копіювати файл можна командою:

copy і’мя_файла_який_копіюється і’мя_файла_куди_копіюється.

Аналогічно створюємо всі файли, які вказані в дереві каталогу на рис. 1.6. Для перейменування файлів використовується команда ren (і’мя_файла) (нове і’мя_файла). Наприклад, перейменуємо файл lab1.txt в файл lab1_copy.txt: ren lab1.txt lab1_copy.txt.

Для відображення створених каталогів використовується команда dir. Вона виводить в консоль список всіх папок для заданого каталогу, в нашому

прикладі це папка C:\Users\user\OS (рис. 1.11).

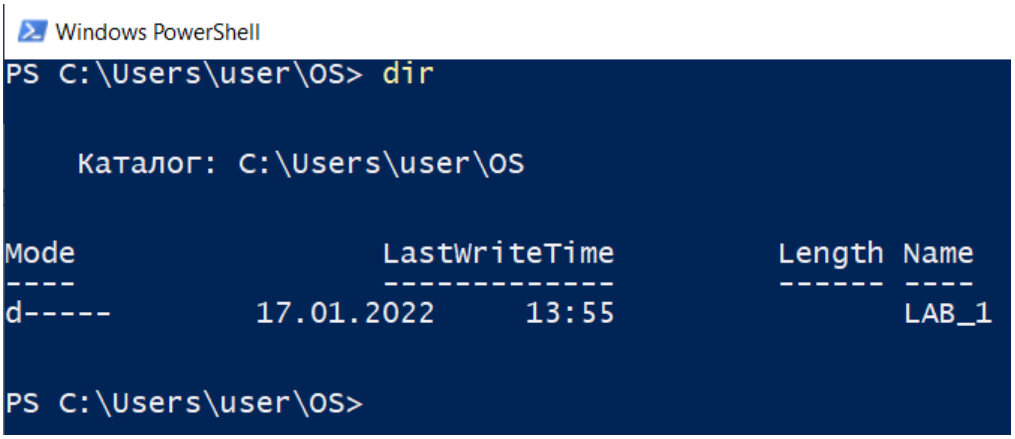


Рис. 1.11 – Результат виконання команди dir

Для копіювання файлів і каталогів із збереженням їхньої структури використовується команда xcopy з обов'язковою вказівкою параметрів копіювання.

Для видалення каталогу використовується команда rd ім'я, для видалення файлу – del ім'я файла. Для переходу в інший каталог використовується команда cd .. Для відображення дерева каталогів використовується команда tree.

Завдання для самостійного виконання:

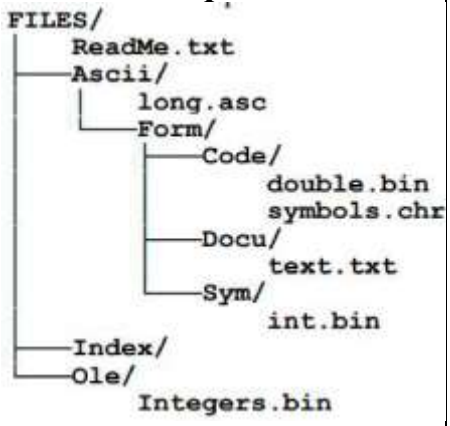
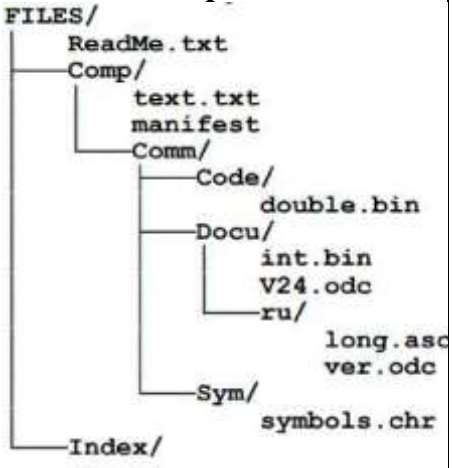
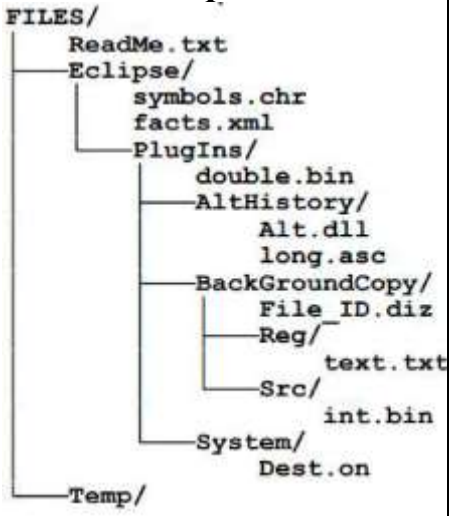
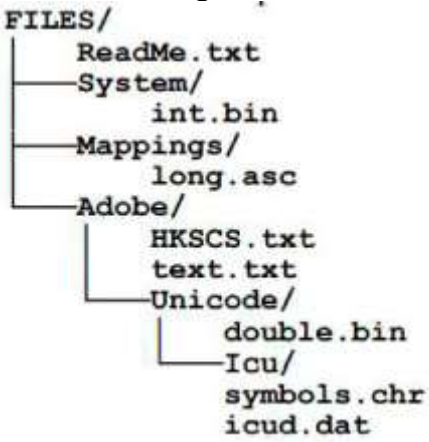
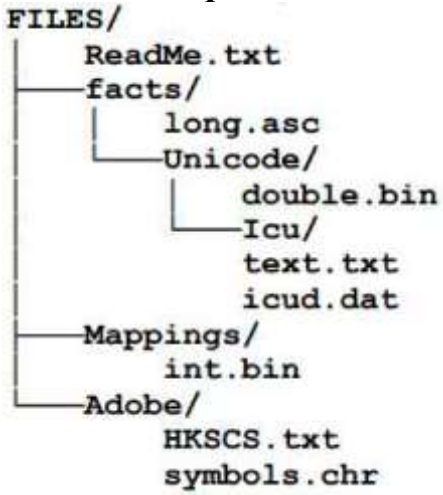
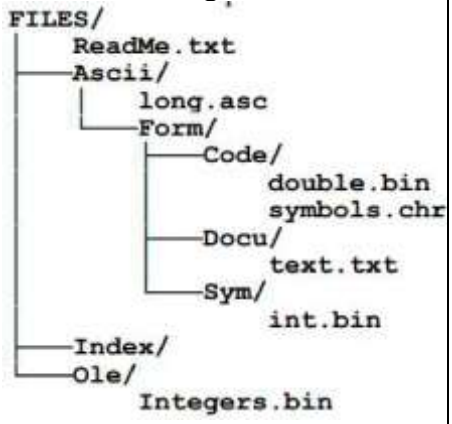
1. В поточному каталозі засобами командної строки створити дерево каталогів відповідно до номера варіанта (табл. 1.1).
2. Створити файл ReadMe.txt та записати в нього номер лабораторної роботи, тему роботи, групу, прізвище, ім'я та по батькові автора роботи.
3. Скопіювати створений файл в text.txt у відповідну варіанту директорію.
4. У випадковому каталозі створити файл lab1.txt та записати в нього поточну дату та дату свого народження.
5. Вивести в консоль дерево каталогів командою tree.
6. Оформити звіт з лабораторної роботи. У звіті для кожного завдання навести скрін консолі з командами та відповіді на контрольні запитання.

Таблиця 1.1

Варіанти індивідуальних завдань

1 варіант FILES/ ReadMe.txt Overheads/ Ses00 int.bin Demo/ Extra.bin text.txt PTR01/ double.bin symbols.chr CHAR/ file.c long.asc	***2 варіант*** FILES/ ReadMe.txt System/ int.bin Mappings/ long.asc Adobe/ HKSCS.txt text.txt Unicode/ double.bin Icu/ symbols.chr icud.dat	***3 варіант*** FILES/ ReadMe.txt facts/ long.asc Unicode/ double.bin Icu/ text.txt icud.dat Mappings/ int.bin Adobe/ HKSCS.txt symbols.chr
4 варіант FILES/ ReadMe.txt PlugIns/ File.diz BY/ Fine/ long.asc Ablib.dll Lingvo/ double.bin Reg/ text.txt int.bin symbols.chr Src/ Src.rar	***5 варіант*** FILES/ ReadMe.txt TechInfo/ double.bin Lang/ af.txt text.txt ABBYY/ Reader/ int.bin Zlib.dll Support/ long.asc symbols.chr	***6 варіант*** FILES/ ReadMe.txt double.bin Panel/ long.asc Lingvo/ int.bin Abbrev.lsd Dic/ text.txt Index/ Support Adobe/ symbols.chr Index.dat
7 варіант FILES/ ReadMe.txt Acronis/ Sandal/ symbols.chr text.txt Common/ int.bin link.dll TrueHome/ license.txt long.asc Common/ double.bin Image/	***8 варіант*** FILES/ ReadMe.txt CaliPo/ libre.exe Calibre/ recomp.exe long.asc plugins/ imagefor/ text.txt resour/ symbols.chr content/ int.bin box.png read/ fonts/ double.bin	***9 варіант*** FILES/ ReadMe.txt BVRDE/ symbols.chr Lex/ int.bin Solutions/ Sounds/ long.asc Templates/ text.txt Files/ double.bin Wizard/ makefile Sym/

Продовження таблиці 1.1

10 варіант 	***11 варіант*** 	***12 варіант*** 
13 варіант 	***14 варіант*** 	***15 варіант*** 

Контрольні питання:

1. Навіщо потрібен командний рядок Windows? Як його викликати, а потім закрити?
2. Як здійснюється введення команд, очищення екрана?
3. Які групи команд використовуються в командному рядку? Наведіть приклади команд із кожної групи.
4. Які команди використовуються для створення, видалення та відображення каталогу?
5. Як створити текстовий файл та записати в нього символи?

ЛАБОРАТОРНА РОБОТА №2
Основи роботи в терміналі операційної системи сімейства Linux

Мета роботи: ознайомлення і вивчення елементарних понять та прийомів роботи з файлами та каталогами в командному рядку операційної системи сімейства Linux.

Постановка завдання: створити дерево каталогу засобами командного рядка, створити текстові файли із заданим текстом, перейменувати створені файли та перемістити їх в задану папку.

Теоретичні відомості

В операційних системах сімейства Linux передбачено використання терміналу з графічним інтерфейсом. Термінал функціонує за схожими принципами як і командний рядок Windows. Вся представлена в лабораторній роботі інформація стосується дистрибутива Ubuntu 20.04 LTS.

Якщо для виконання роботи використовується інша операційна система сімейства Linux слід враховувати, що деякі команди можуть відрізнятися, тому в такому випадку, необхідно звертатися до інструкцій з тієї ОС, яка використовується. Але загальні принципи роботи незмінні.

Для систем Windows 10 та їм подібним створена підсистема Windows для Linux (WSL). Вона реалізована як функція операційної системи Windows, яка дозволяє запускати файлову систему Linux, а також програми командного рядка Linux і програми GUI з графічним інтерфейсом користувача безпосередньо на Windows.

Після встановлення дистрибутива Ubuntu 20.04 LTS на ПК з Windows та запуску цього додатку відкривається термінал з графічним інтерфейсом (рис. 2.1). При використанні оболонки PowerShell запустити термінал встановленого дистрибутива Linux можна командою wsl (рис. 2.2.).

Рис. 2.1 – Термінал Ubuntu з графічним інтерфейсом

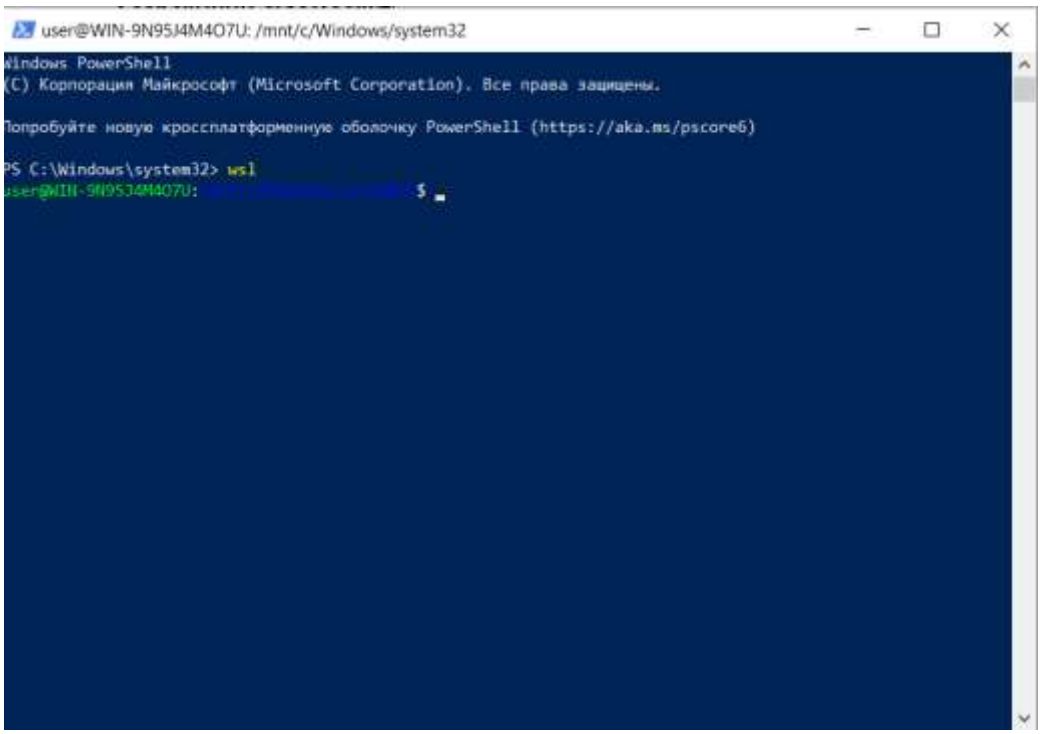


Рис. 2.2 – Запуск термінала Ubuntu через PowerShell

Команди роботи з файлами та каталогами Linux використовуються для редагування конфігураційних файлів, зборку програм, адміністрування та ін. Робота в терміналі Linux дозволяє відображати вміст папок, перехід між папками, створювати та видаляти файли. Для роботи з файлами та каталогами використовуються наступні команди:

- ls – список файлів в директорії;
- cd – перехід між директоріями;
- rm – видалити файл;
- rmdir – видалити папку;
- mv – переміщення файлу;
- cp – копіювання файлу;
- mkdir – створити папку;
- ln – створити посилання;
- chmod – змінити права файла;
- touch створити порожній файл.

Відображення вмісту папки та створення нового каталогу

Команда ls дозволяє вивести список файлів визначеної папки, за замовчуванням, буде виводитися список файлів поточної папки. Для виводу списку файлів із всіх підкаталогів використовується опція -R: ls -R. Щоб

вивести список файлів потрібної папки, необхідно передати її адресу утиліти наступним чином:ls /home. Щоб отримати більше інформації та вивести всі імена файлів у вигляді списку, використовується опція -l: ls -l /home/. Результати виконання вище зазначеної команди представлені на рис. 2.3.

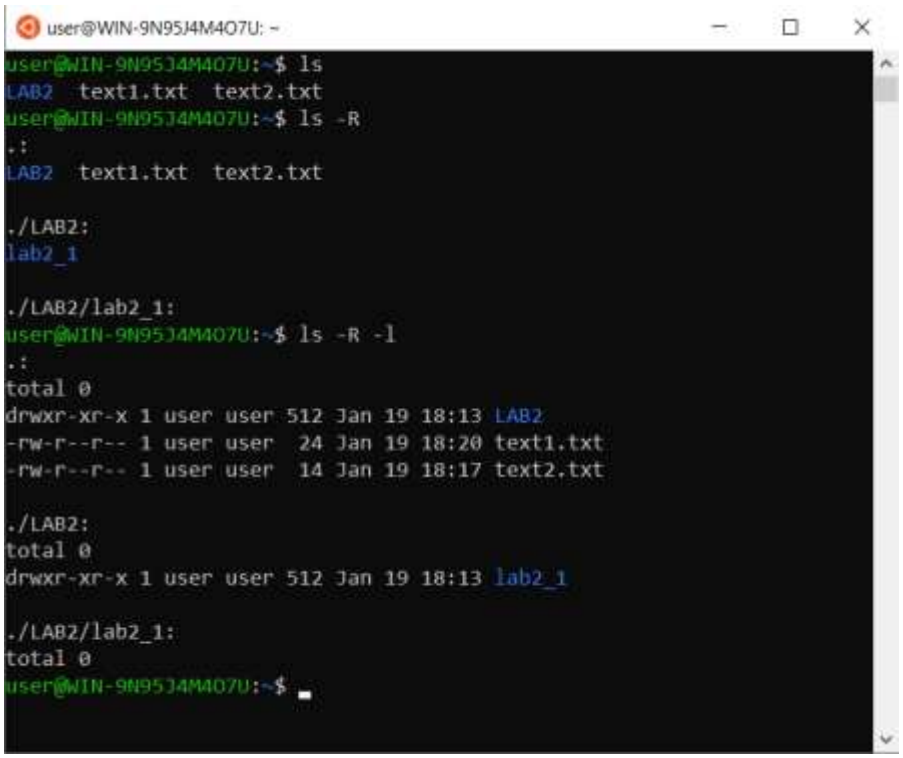


Рис. 2.3 – Результати виконання команди ls

Для створення нової папки використовується команда mkdir. Якщо треба створити нову папку в заданому каталозі, то необхідно вказати повний шлях, наприклад, створюємо папку test командою mkdir /home/user/test. Для очищення терміналу використовується команда clear або клавіші Ctrl+L. В терміналі можливо використання інших ‘гарячих’ клавіш для прискорення та полегшення роботи. Наприклад, <TAB> для автодоповнення команди, <↑> і <↓> для навігації по командах, які вже вводилися в командному рядку.

Зміна робочого каталогу

Команда cd дозволяє змінити поточну папку на іншу. За замовчуванням, поточною вважається домашня папка, наприклад, cd Desktop змінює папку на робочий стіл, якщо виконати її з домашнього каталогу.

Для переходу в root-каталог використовується наступна команда: cd /. Можливо вказати повний шлях до папки: cd /usr/share/. Команда cd .. переходить до папки, яка знаходиться вище на одну у файловій системі. Для повернення до

попередньої робочої папки використовується команда: cd --.

Результати виконання вище зазначених команд представлені на рис. 2.4.

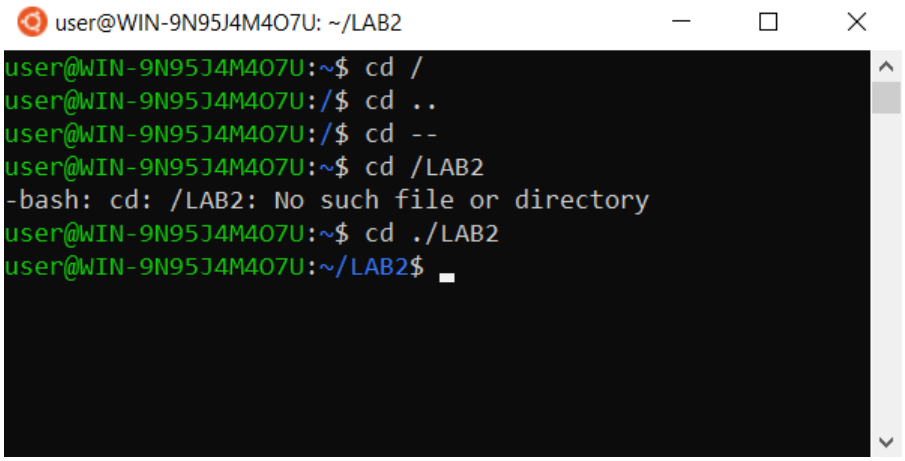


Рис. 2.4 – Результати виконання команди cd

Зверніть увагу на правильність написання адреси розташування папки (шлях до каталогу). Необхідно вказувати повний шлях до папки ./LAB2.

Видалення, переміщення, перейменування, копіювання файлів та каталогів

Команда rm дозволяє видалити файл, вона видаляє файл без підтвердження. Наприклад, команда rm file видалить файл з ім'ям file, який знаходиться у поточній папці. Можливо вказати повний шлях файлу, наприклад: rm /usr/share/file. Для видалення папки необхідно використовувати опцію -r. Вона включає рекурсивне видалення всіх файлів та папок на всіх рівнях вкладеності, наприклад, rm -r /home/user/photo/. Ця команда видаляє файли безповоротно. Для видалення пустої папки використовують команду rmdir.

Для переміщення файлів у нове місце використовують команду mv. Вона також може бути використана для перейменування файлів. Наприклад, mv file newfile перейменує файл file на newfile. Щоб перемістити файл в іншу папку потрібно вказати шлях до неї, наприклад, перемістимо файл file в папку /home/user/tmp/ командою mv file /home/user/tmp/.

Для відображення дерева каталогів використовується команда tree. Якщо цей пакет не встановлено слід набрати команду sudo apt install tree. Аналогічно встановлюються будь-які пакети дистрибутива.

Команди cp та mv схожі команди для роботи з файлами. Вони працюють

аналогічно, тільки вихідний файл для команди mv залишається на своєму місці. Для рекурсивного копіювання папки використовують опцію -r. Команда cp -r скопіює всю папку разом з усіма файлами та вкладеними папками в нове місце.

Скопіюємо папку lab в папку TEST. Для цього використовуємо команду cp з опцією -r. Дерево каталогів до копіювання та після копіювання показано на рис. 2.5 і 2.6.

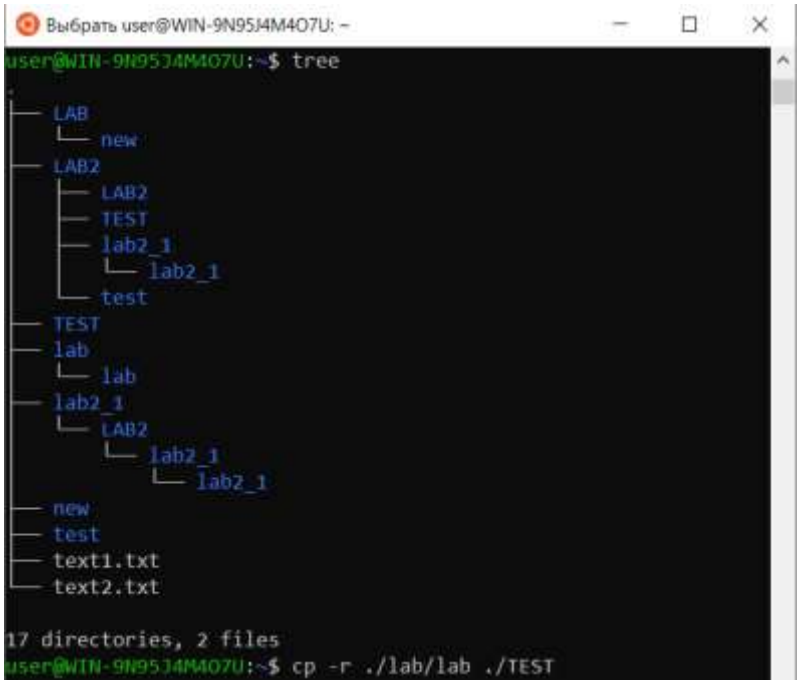


Рис. 2.5 – Результати виконання команди tree до копіювання

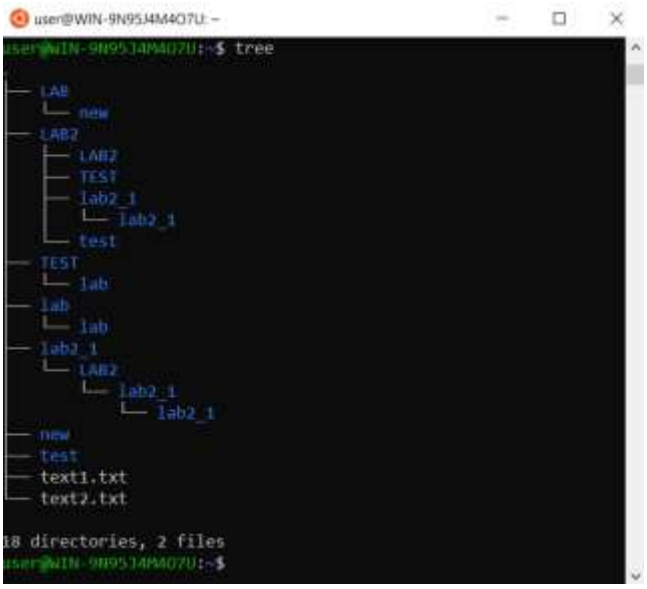


Рис. 2.6 – Результати виконання команди tree після копіювання

Створення файлів та запис даних в файл

Створити файл в Linux можливо декількома способами. Розгляне один з них. Для створення файлу в поточному каталозі або в заданій папці використовується команда touch. Створимо файл newfile1.txt в поточному каталозі і файл newfile2.txt в папці new. Результати виконання обох команд відображені в дереві каталогів (рис. 2 7).

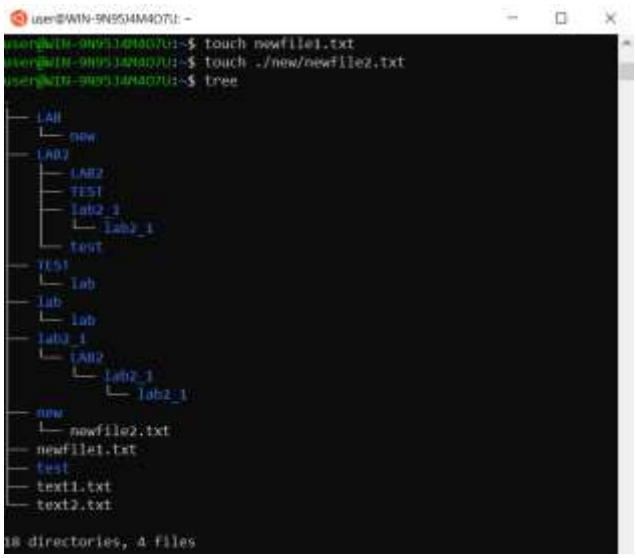


Рис. 2.7 – Створення файлів командою touch

Для запису тексту в файл використовується команда echo “text” > ім’я файлу, наприклад, команда echo "Ivanov Ivan K-21">newfile1.txt вносить текст в створений раніше файл. Відображення вмісту файлу можливо за допомогою команди cat. На рис. 2.8 представлені результати запису тексту в створений файл.

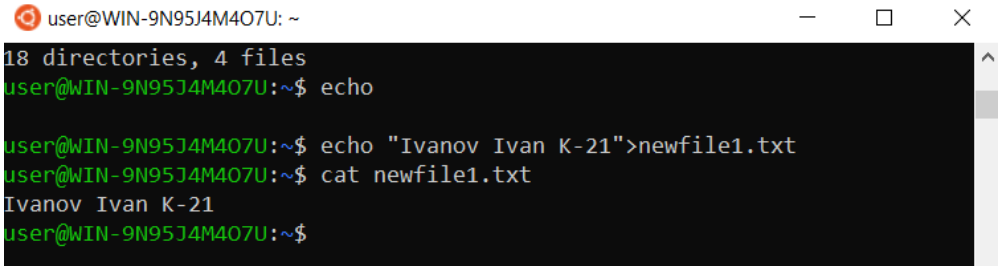


Рис. 2.8 – Створення файлів командою touch

Завдання для самостійного виконання:

1. В поточному каталозі засобами командної строки створити дерево

каталогів відповідно до номера варіанта (табл. 2.1).

2. Створити файл ReadMe.txt та записати в нього номер лабораторної роботи, тему роботи, групу, прізвище, ім'я та по батькові автора роботи.

3. Скопіювати створений файл в text.txt в каталог відповідно варіанту.

4. В випадковому каталозі створити файл lab1.txt та записати в нього поточну дату та дату свого народження.

5. Вивести в консоль дерево каталогів командою tree.

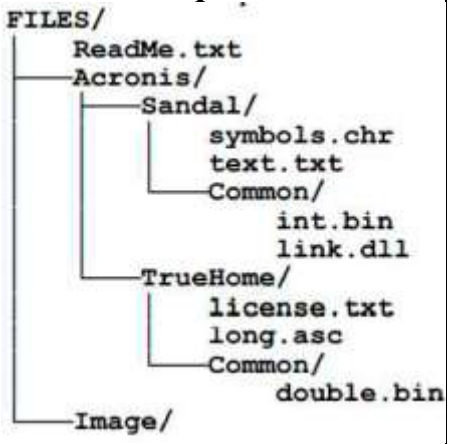
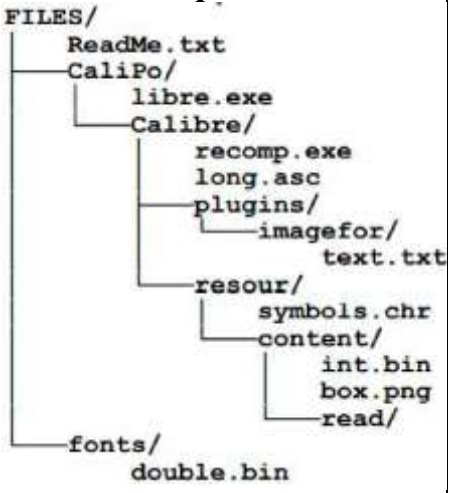
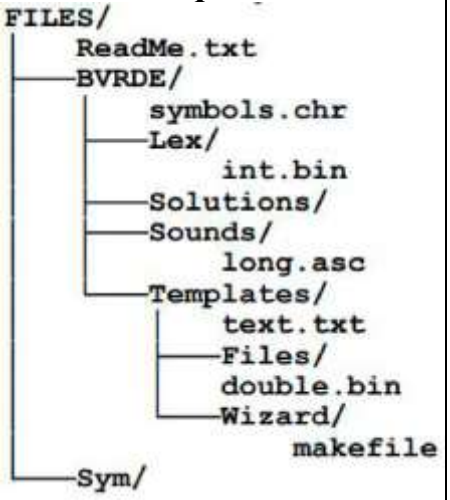
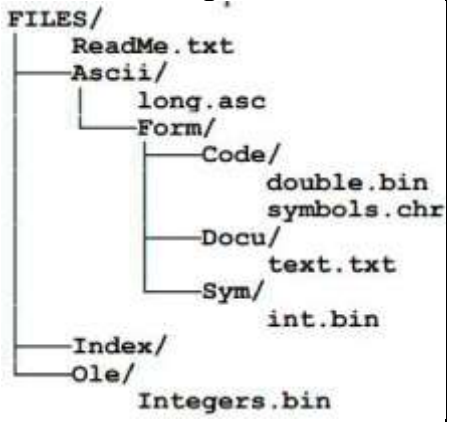
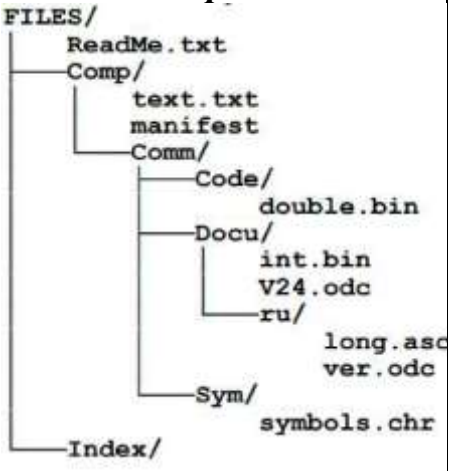
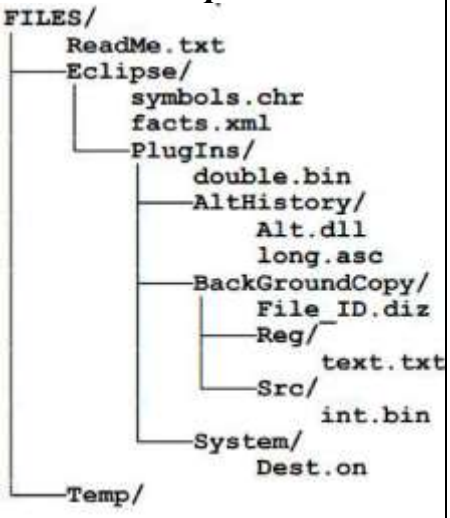
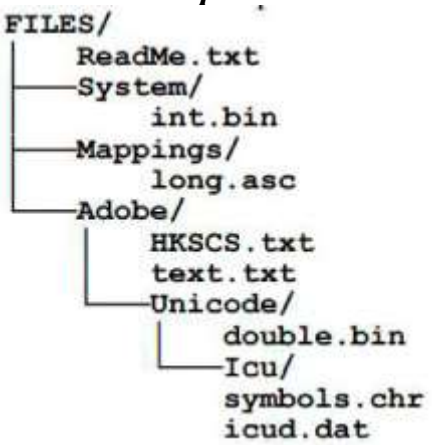
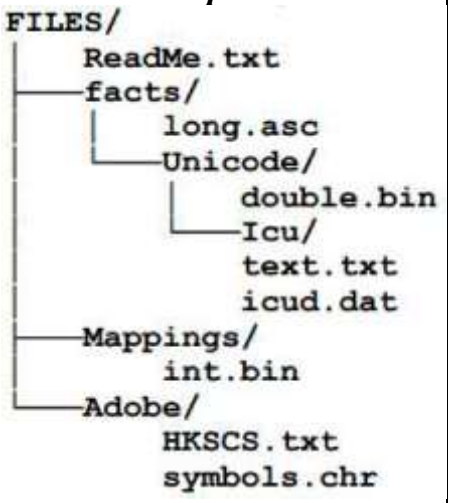
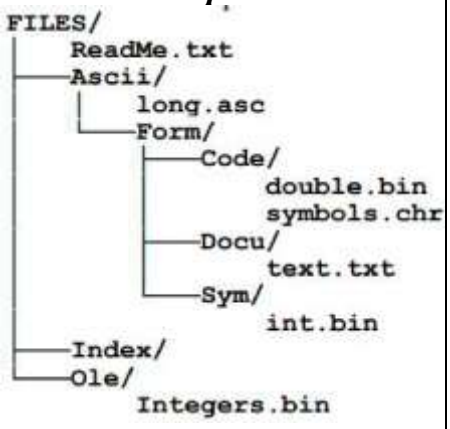
6. Оформити звіт з лабораторної роботи. У звіті для кожного завдання навести скрін терміналу з командами та відповіді на контрольні запитання.

Таблиця 2.1

Варіанти індивідуальних завдань

1 варіант FILES/ ReadMe.txt Overheads/ Ses00 int.bin Demo/ Extra.bin text.txt PTR01/ double.bin symbols.chr CHAR/ file.c long.asc	***2 варіант*** FILES/ ReadMe.txt System/ int.bin Mappings/ long.asc Adobe/ HKSCS.txt text.txt Unicode/ double.bin Icu/ symbols.chr icud.dat	***3 варіант*** FILES/ ReadMe.txt facts/ long.asc Unicode/ double.bin Icu/ text.txt icud.dat Mappings/ int.bin Adobe/ HKSCS.txt symbols.chr
4 варіант FILES/ ReadMe.txt PlugIns/ File.diz BY/ Fine/ long.asc Ablib.dll Lingvo/ double.bin Reg/ text.txt int.bin symbols.chr Src/ Src.rar	***5 варіант*** FILES/ ReadMe.txt TechInfo/ double.bin Lang/ af.txt text.txt ABBYY/ Reader/ int.bin Zlib.dll Support/ long.asc symbols.chr	***6 варіант*** FILES/ ReadMe.txt double.bin Panel/ long.asc Lingvo/ int.bin Abbrev.lsd Dic/ text.txt Index/ Support Adobe/ symbols.chr Index.dat

Продовження таблиці 2.1

7 варіант	***8 варіант***	***9 варіант***
		
10 варіант	***11 варіант***	***12 варіант***
		
13 варіант	***14 варіант***	***15 варіант***
		

Контрольні питання:

1. Навіщо потрібен термінал Ubuntu? Як його викликати, а потім закрити?
2. Як здійснюється введення команд, очищення екрана?
3. Якою командою викликати довідку для певної команди?
4. Як становити потрібний пакет у випадку його відсутності?
5. Які команди використовуються для створення, видалення та відображення каталогу?
6. Які команди використовуються для створення, перейменування та копіювання файлів?
7. Як створити текстовий файл та записати в нього заданий набір символів?
8. Якою командою можливо переглянути вміст файлу?

ЛАБОРАТОРНА РОБОТА №3

Створення та запуск bash-скриптів

Мета роботи: ознайомлення і вивчення елементарних понять та прийомів створення та запуску bash-скриптів.

Постановка завдання: створити прості bash-скрипти для виконання операцій роботи з файлами та каталогами.

Теоретичні відомості

Скрипт (сценарій) – це послідовність команд, які по черзі зчитує та виконує програма-інтерпретатор. Для скриптів роботи з Unix-подібними операційними системами найчастіше використовують оболонку bash.

Скрипт – це текстовий файл, в якому перераховані команди, які можна вводити вручну, а також зазначена програма, яка їх виконуватиме. Завантажувач, який виконує скрипт не вміє працювати зі змінними оточення, тому йому потрібно передати точний шлях до програми, яку потрібно запустити. Далі він передає скрипт цій програмі та розпочинається виконання.

Найпростіший приклад скрипта для командної оболонки Bash:

```
#!/bin/bash
echo "Hello world"
```

Цей скрипт виводить у вікно термінала фразу «Hello world». Для створення цього скрипту використовуємо команду nano, яка запускає редактор файлів (рис. 3.1).

Рис. 3.1 – Вікно редактора файлів nano

Натискаємо Ctrl+X та вводимо ім’я файлу, вказуємо розширення ch (означає, що це bash-скрипт), файл створено. Далі для того, щоб запустити цей файл набираємо лише повний шлях до нього. Але файл повинен бути виконуваним (мати флаг X).

В операційній системі Linux для керування флагами файлів використовується утиліта chmod. Синтаксис виклику утиліти:

$ chmod категорія дія прапор адреса_файлу

Категорія – флаги можуть встановлюватися для трьох категорій: власника файлу, групи файлу та решти користувачів. У команді вони вказуються символами u (user), g (group), o (other) відповідно.

Дія – може бути + (плюс), що означає установити прапор або – (мінус) зняти флаг.

Флаг – один із доступних прапорів – r (читання), w (запис), x (виконання).

Для того, щоб зробити створений вище файл виконуваним набираємо команду chmod u+x lab3_1.sh. Далі запускаємо цей файл, у вікні терміналу виведеться фраза із файлу, оскільки команда echo виводить в поточне вікно все, що записано в « » (рис. 3.2).

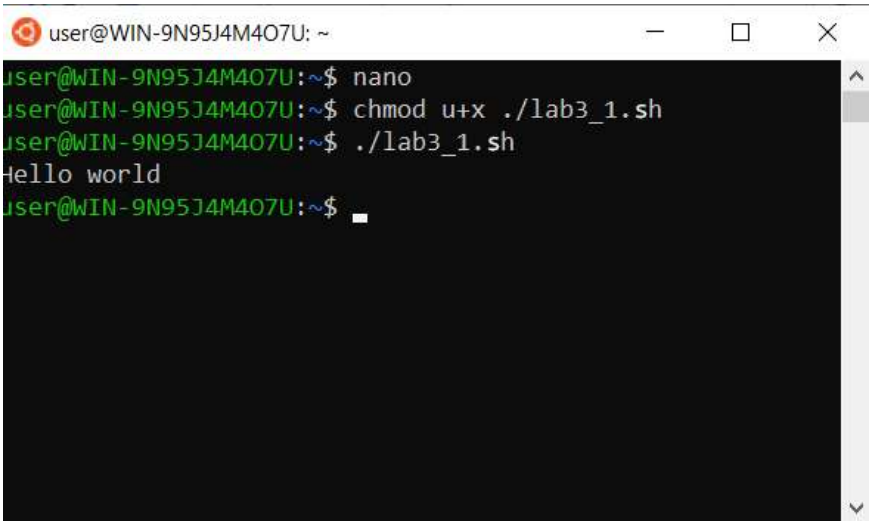


Рис. 3.2 – Результати виконання скрипту lab3_1.sh

Для використання змінних в bash-скриптах слід знати, що оболонка не розрізняє типи, всі змінні будуть типу string. Але bash дозволяю результати виконання утиліт записувати як значення змінних. Для виводу значення змінної

на екран використовується знак $, наприклад, є змінна string, вона має значення string=”Hello”. Для того щоб вивести строку Hello на екран використовується команда: echo $string. А для виводу поточної дати використовуємо наступну команду:

echo $(date)

В результаті додавання її в файл lab3_1.sh після виконання на екрані з’явиться поточна дата в стандартному форматі (рис. 3.3).

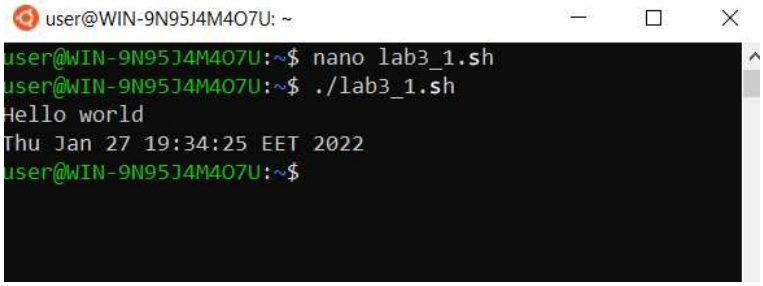


Рис. 3.3 – Результати виконання скрипту lab3_1.sh

Для передачі параметрів в скрипт використовують змінні, наприклад, необхідно передати в скрипт ім’я користувача та його вік, тобто необхідно передати 2 параметри. Для цього в самому скрипті використовують $1, $2, $3… (рис. 3.4). Коли запускають скрипт, то після імені через пробіл (« ») вказують значення, які необхідно передати в скрипт (Kate – перший параметр $1, $2 – другий параметр $2) (рис. 3.5).

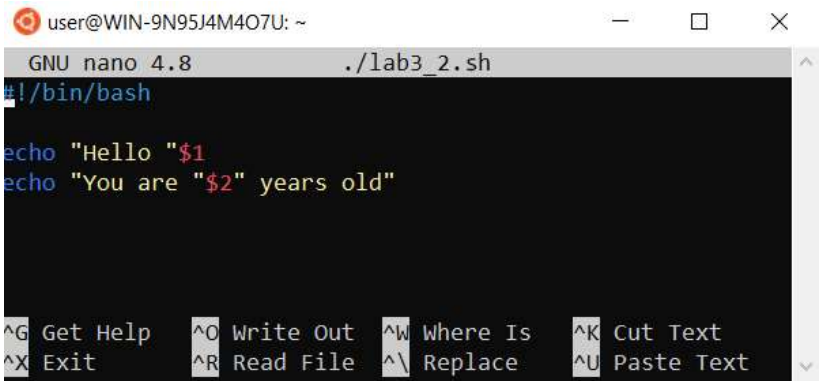


Рис. 3.4 –Вміст файлу lab3_2.sh

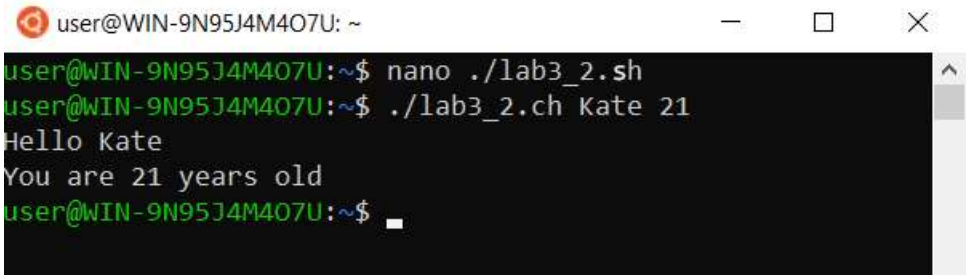


Рис. 3.5 – Результати виконання скрипту lab3_2.sh

В попередньому прикладі розглядалася ситуація, коли параметри передаються командою запуску скрипта. Існує можливість вводу даних користувачем, для цього призначена команда read, наприклад, ввести ім’я користувача по запиту можна командою:

read -p “What is your name” name.

При цьому, p – флаг запиту, name – змінна для збереження введеного значення.

При написанні bash-скриптів виникають ситуації, коли необхідно перевіряти параметри або умови. Для цього використовують команди умовного розгалуження коду, синтаксис наступний:

```
if [ <умова> ]; then
   <команда>
else
   <команда>
Fi
```

Наприклад, код перевірки наявності файлу в поточній директорії:

```
if [ -f "file.txt" ]; then
   echo "Файл існує"
else
   echo "Файл не знайдено"
fi
```

Таблиця 3.1

Основні параметри, які використовуються при записі умови в операторах розгалуження

Параметр	Умова, яка перевіряється
-z	Перевіряє, чи є рядок порожнім
-n	Перевіряє, чи не є рядок порожнім
-f	Перевіряє, чи існує файл та є це звичайний файл
-d	Перевіряє, чи існує файл та є це каталог
-e	Перевіряє, чи існує файл
-r	Перевіряє, чи є файл доступним для читання
-w	Перевіряє, чи є файл доступним для запису
-x	Перевіряє, чи є файл виконуваним
-s	Перевіряє, чи є розмір файлу більшим за 0
-eq	Перевіряє, чи є два числа рівними
-ne	Перевіряє, чи є два числа нерівними
-lt	Перевіряє, чи є перше число меншим за друге
-le	Перевіряє, чи є перше число меншим або рівним другому
-gt	Перевіряє, чи є перше число більшим за друге
-ge	Перевіряє, чи є перше число більшим або рівним другому

Для організації циклічного виконання частини коду використовують команди циклу for (цикл з параметрами), синтаксис оператору наступний:

```
for <змінна> in <список>
do
   <команда>
Done
```

Наприклад, наступний код виводить потоком числа від 1 до 10:

```
for i in {1..10}; do
   echo "Цикл for: $i"
done
```

Команда for послідовно привласнює змінній значення зі списку та виконує команди, які розташовані між do і done.

Для організації циклу з передумовою використовують оператор while, синтаксис якого наступний:

```
while <умова>
do
   <команда>
done
```

Приклад коду, який виводить потоком числа від 1 до 10:

```
i=1
while [ $i -le 10 ]; do
   echo "Цикл while: $i"
   i=$((i+1))
done
```

Існує команда циклу, яка працює доки певна умова не буде виконана, цикл з постумовою, синтаксис наступний:

```
until <умова>
do
   <команда>
done
```

Наступний код виводить потоком числа від 1 до 10, але за допомогою циклу з постумовою ***until***:

```
i=1
until [ $i -gt 10 ]; do
  echo "Цикл until: $i"
  i=$((i+1))
done
```

важливо))) у всі приклади кодів циклів додати коментарі на строку з умовою

Конструкція ***case*** в оболонці Bash використовується для визначення блоку коду, який виконується в залежності від значення змінної. Вона є альтернативним варіантом використання послідовності ***if-else*** операторів, які використовуються для перевірки різних варіантів значень однієї змінної та

виконання відповідних дій на основі цих значень. Синтаксис конструкції ***case***:

```
case <змінна> in
   <значення1>)
      <команди1>
      ;;
   <значення2>)
      <команди2>
      ;;
   *)
      <команди_за_замовчуванням>
      ;;
esac
```

Приклад використання case для визначення дій на основі значення змінної:

```
option="B"
case $option in
   A)
      echo "Обрано опцію A"
      ;;
   B)
      echo "Обрано опцію B"
      ;;
   *)
      echo "Невідома опція"
      ;;
esac
```

У цьому прикладі значення змінної ***option*** перевіряється для визначення, яка опція буде обрана. В залежності від значення ***option***, виводиться відповідне повідомлення. Якщо ***option*** не відповідає жодному з варіантів, виконується блок за замовчуванням з ****)***.

При написанні bash-скриптів виникають ситуації, коли необхідно запустити якусь команду та обробити результат її виконання. Наприклад, необхідно запустити команду ping та перевірити працездатність бездротового

роутера (рис. 3.6).

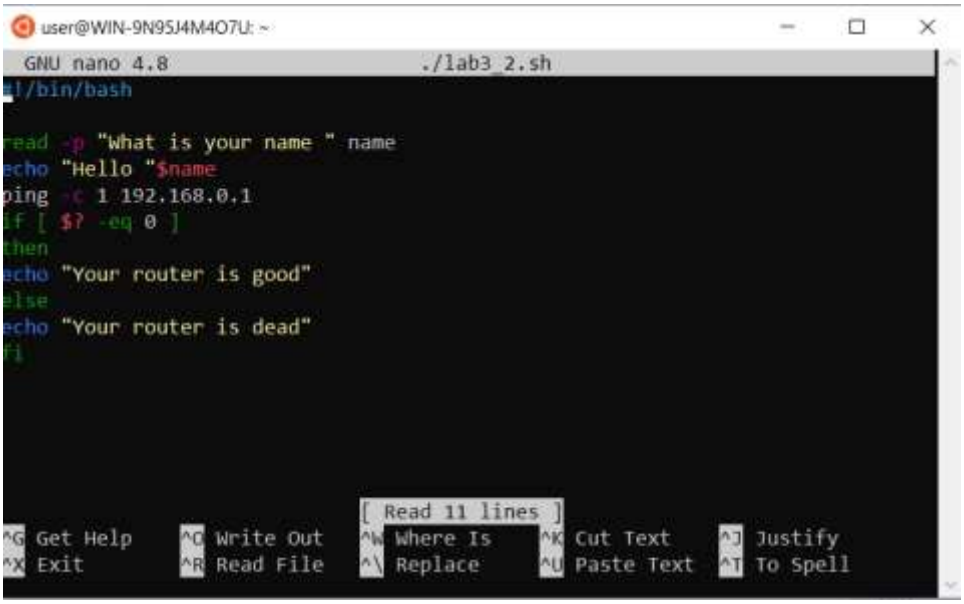


Рис. 3.6 – bash-скрипт перевірки підключення до роутера

Якщо запустити такий скрипт з підключенням до мережі, на екран виведеться результати команди ping та зарезервована фраза. Якщо ж підключення буде відсутнє, то також будуть результати ping та інша зарезервована фраза (рис. 3.7).

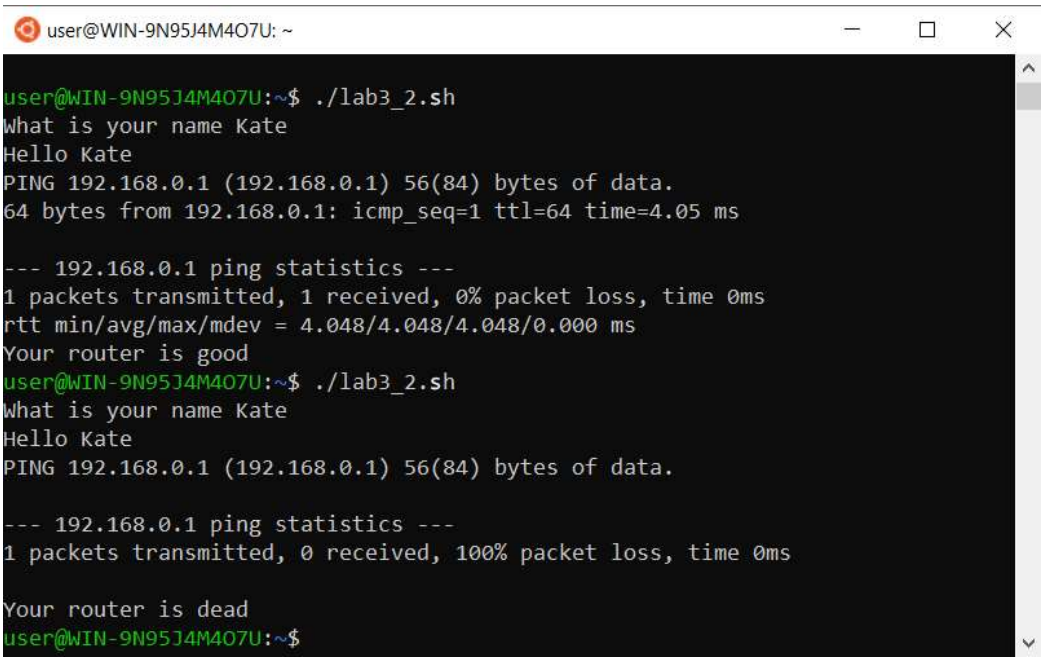


Рис. 3.7 – Результати роботи bash-скрипта перевірки підключення до роутера

В bash-скриптах дозволяється використання логічного оператора && (логічне «AND») для визначення результату виконання команд, синтаксис наступний:

mkdir my_folder && cd my_folder

Наведена у прикладі команда спочатку створить папку ***my_folder***, а потім, якщо це вдасться успішно, перейде до цієї папки.

Конвеєр (***pipe***) – це механізм у системах Unix та Unix-подібних операційних системах, який дозволяє з'єднувати вихідний потік даних однієї програми з вхідним потоком іншої програми. Це дуже потужний інструмент, який дозволяє поєднувати декілька команд разом, створюючи складніші операції обробки даних. У bash конвеєр використовується з символом вертикальної риски |.

Приклад передачі вихідних даних однієї команди до іншої:

ls -l | grep ".txt"

У цьому прикладі команда ***ls -l*** виводить список файлів у поточному каталозі, а потім результати цієї команди передаються через конвеєр команді ***grep ".txt"***, яка шукає файли з розширенням .txt.

Приклад з використанням декількох команд у конвеєрі:

cat file.txt | grep "pattern" | sort

Цей конвеєр починається з читання вмісту file.txt за допомогою ***cat***, результати передаються через ***grep "pattern"***, яка шукає вказаний шаблон, і через ***sort***, сортує вивід у лексографічному порядку.

Завдання для самостійного виконання:

1. Написати bash-скрипт, який створює каталог (за варіантами в попередній лабораторній роботі, табл. 2.1) та виводить його на екран у вигляді дерева.

2. Створити bash-скрипт, який виконує прості завдання згідно варіанту (табл. 3.2).

3. Оформити звіт з лабораторної роботи. У звіті для кожного завдання навести скрін редактора файлів (nano), скрін з результатами роботи скрипту та

відповіді на контрольні запитання.

Таблиця 3.2

Варіанти індивідуальних завдань

№	Завдання №1	Завдання №2	Завдання №3
1	Ввести в командному рядку два числових аргументи, вивести на екран менший аргумент	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести повідомлення в форматі: «Директорія ‘ім’я директорії’ успішно створено»
2	Ввести в командному рядку три числових аргументи, визначити добуток аргументів та вивести її на екран	Ввести в командному рядку два числових аргументи, вивести на екран менший аргумент	Створити тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, для кожної успішної команди створення файлу вивести повідомлення в форматі: «Файл ‘ім’я файлу’ успішно створено»
3	Ввести в командному рядку два числових аргументи, вивести на екран менший аргумент	Ввести в командному рядку два числових аргументи, вивести на екран більший аргумент	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести
4	Ввести в командному рядку три числових аргументи, визначити добуток аргументів та вивести її на екран	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Ввести в командному рядку два числових аргументи, вивести на екран більший аргумент
5	Ввести в командному рядку два числових аргументи, вивести на екран більший аргумент	Ввести в командному рядку два числових аргументи, вивести на екран повідомлення, якщо введені однакові аргументи	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, вивести на екран вміст директорій DIR1 та DIR2
6	Ввести в командному рядку два числових аргументи, вивести на екран повідомлення, якщо введені однакові аргументи	Ввести в командному рядку три числових аргументи, визначити добуток аргументів та вивести її на екран	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести
7	Ввести в командному рядку два числових аргументи, вивести на екран повідомлення, якщо введені однакові аргументи	Ввести в командному рядку два числових аргументи, вивести повідомлення, якщо користувач вводить більше або менше аргументів	Створити тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST
8	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, вивести на екран вміст директорій DIR1 та DIR2	Ввести в командному рядку два числових аргументи, вивести повідомлення, якщо користувач вводить більше або менше аргументів

Продовження таблиці 3.2

№	Завдання №1	Завдання №2	Завдання №3
9	Ввести в командному рядку два числових аргументи, вивести повідомлення, якщо користувач вводить більше або менше аргументів	Ввести в командному рядку строку символів, створити директорію з таким ім'ям, у разі успішного створення вивести на екран каталог у вигляді дерева	Створити тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, для кожної успішної команди створення файлу вивести повідомлення в форматі: «Файл 'ім'я файлу' успішно створено»
10	Створити тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST	Ввести в командному рядку строку символів, створити директорію з таким ім'ям, у разі успішного створення вивести на екран каталог у вигляді дерева	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу аргументів та вивести її на екран та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести повідомлення в форматі: «Директорія 'ім'я директорії' успішно створено»
11	Ввести в командному рядку строку символів, створити директорію з таким ім'ям, у разі успішного створення вивести на екран каталог у вигляді дерева	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, вивести на екран вміст директорій DIR1 та DIR2	Ввести в командному рядку строку символів, створити текстовий файл з таким ім'ям, записати в файл ім'я та дату створення, у разі успішного створення вивести на екран повідомлення
12	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести повідомлення в форматі: «Директорія 'ім'я директорії' успішно створено»	Ввести в командному рядку строку символів, створити текстовий файл з таким ім'ям, записати в файл ім'я та дату створення, у разі успішного створення вивести на екран повідомлення	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу аргументів та вивести її на екран та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести повідомлення в форматі: «Директорія 'ім'я директорії' успішно створено»
13	Ввести в командному рядку строку символів, створити текстовий файл з таким ім'ям, записати в файл ім'я та дату створення, у разі успішного створення вивести на екран повідомлення	Ввести в командному рядку три числових аргументи, визначити суму аргументів та вивести її на екран	Створити тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, для кожної успішної команди створення файлу вивести повідомлення в форматі: «Файл 'ім'я файлу' успішно створено»
14	Створити тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST	Ввести в командному рядку три числових аргументи, визначити суму аргументів та вивести її на екран	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести повідомлення в форматі: «Директорія 'ім'я директорії' успішно створено»

Продовження таблиці 3.2

№	Завдання №1	Завдання №2	Завдання №3
15	Ввести в командному рядку три числових аргументи, визначити суму аргументів та вивести її на екран	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Створити DIR1 та DIR2, в кожній директорії створити по одному файлу аргументів та вивести її на екран та записати в них поточну дату та час, для кожної успішної команди створення директорії вивести повідомлення в форматі: «Директорія ‘ім’я директорії’ успішно створено»

Контрольні питання:

1. Що таке bash-скрипт? Як його створити та запустити?
2. Для чого використовують bash-скрипти?
3. Які команди використовують для вводу-виводу в bash-скриптах?
4. Які команди використовують для розгалуження коду та для циклічного виконання частини коду?
5. Як запустити bash-скрипт з ОС Windows?

ЛАБОРАТОРНА РОБОТА №4

Управління процесами та потоками в операційних системах Windows і Linux

Мета роботи: ознайомлення і вивчення елементарних понять та прийомів управління процесами та потоками в ОС Windows і ОС Linux

Постановка завдання: створити прості bash-скрипти для виконання операцій управління процесами та потоками в ОС Windows і ОС Linux.

Теоретичні відомості

Windows PowerShell має так звані командлети (cmdlets). Це спеціалізовані класи .NET, які реалізують різноманітну функціональність. Вони мають назви у відповідності «дія – об'єкт» Наприклад, Get-Help буквально означає «Отримати-Допомога» або в контексті PowerShell – «Показати-Довідку». Це аналог команди man в Unix-системах і мануали в PowerShell потрібно запитувати саме так, а не викликати командлети з ключем --help або /?.

В командлетах для визначення дій використовуються наступні ключові слова:

Add – додати;
Clear – очистити;
Enable – включити;
Disable – виключити;
New – створити;
Remove – видалити;
Set – задати;
Start – запустити;
Stop – зупинити;
Export – експортувати;
Import – імпортувати.

В PowerShell реалізовані системні, користувальницькі та опціональні командлети. В результаті виконання вони повертають об'єкт чи масив об'єктів. Вони не чутливі до регістру, тобто. з погляду інтерпретатора команд немає різниці між Get-Help та get-help. Якщо в одному рядку виконується кілька командлетів, то необхідно використовувати символ «;».

Для пошуку об'єкта та дії використовується командлет Get-Command. Показати довідку про нього можна наступним чином:

Get-Help Get-Command

Робота з процесами ОС Windows

Windows 10 має інструменти для отримання списку доступних командлетів управління процесами. А саме команда ***Get-Command -Noun Process*** виводить цей список (рис. 4.1).

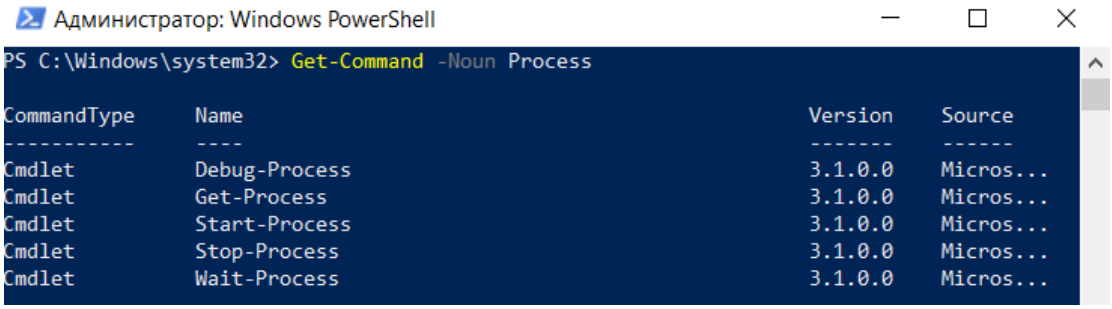


Рис. 4.1 – Перелік командлетів для роботи з процесами Windows 10

Отримати список активних процесів Windows можливо командлетом ***get-process*** в командній оболонці PowerShell (рис. 4.2) або командою tasklist в командному рядку cmd (рис. 4.4).

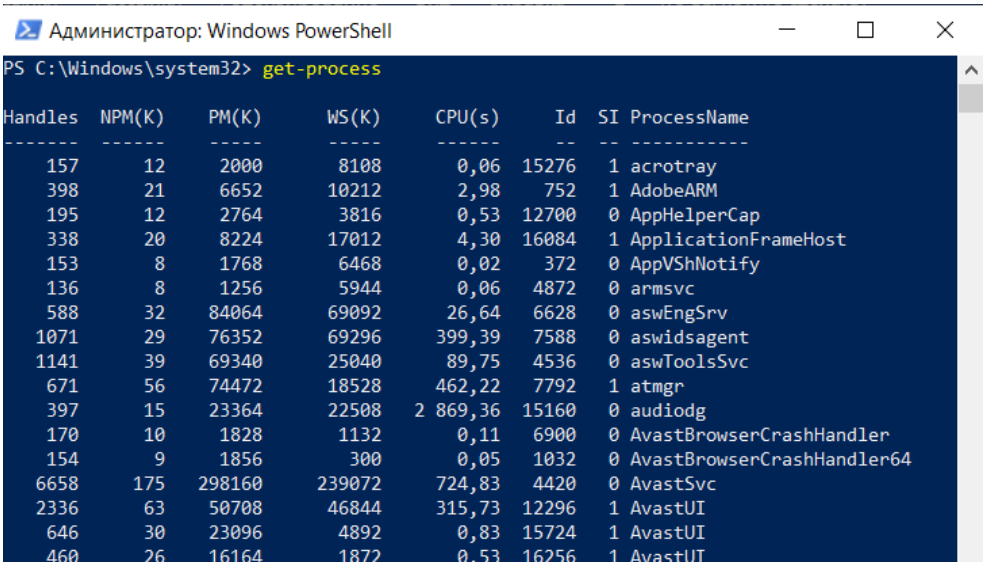


Рис. 4.2 – Результати виконання командлета get-process в PowerShell

Якщо командлет get-process введено без аргументів, то за замовчуванням виводяться наступні властивості запущених процесів:

Handles – кількість дескрипторів вводу–виводу, які відкрив цей процес;

NPM(K) – Non-paged memory (пул, що не вивантажується), розмір даних процесу (в Кб), які ніколи не потрапляють у файл підкачування на диск;

PM(K) – розмір пам'яті процесу, яку можна вивантажити на диск;

WS(K) – розмір фізичної пам'яті у Кб, яка використовується процесом (working set).

CPU(s) – процесорний час, який використовується процесом (враховується час усіх CPU);

ID – ідентифікатор процесу;

SI (Session ID) – ідентифікатор сеансу процесу (0 – запущений для всіх сесій, 1 – для першого користувача, 2 – для другого та ін.);

ProcessName – ім'я процесу.

Для виводу заданих властивостей будь-якого процесу слід використовувати наступний синтаксис командлета (рис. 4.3):

1) ***Get-Process -Name 'notepad' -FileVersionInfo*** – отримати властивості процесу notepad.

2) ***Get-Process -FileVersionInfo -ErrorAction SilentlyContinue*** – отримати властивості всіх активних процесів та ігнорувати помилки, які пов’язані з відсутністю у де-яких процесів властивості «FileVersion».

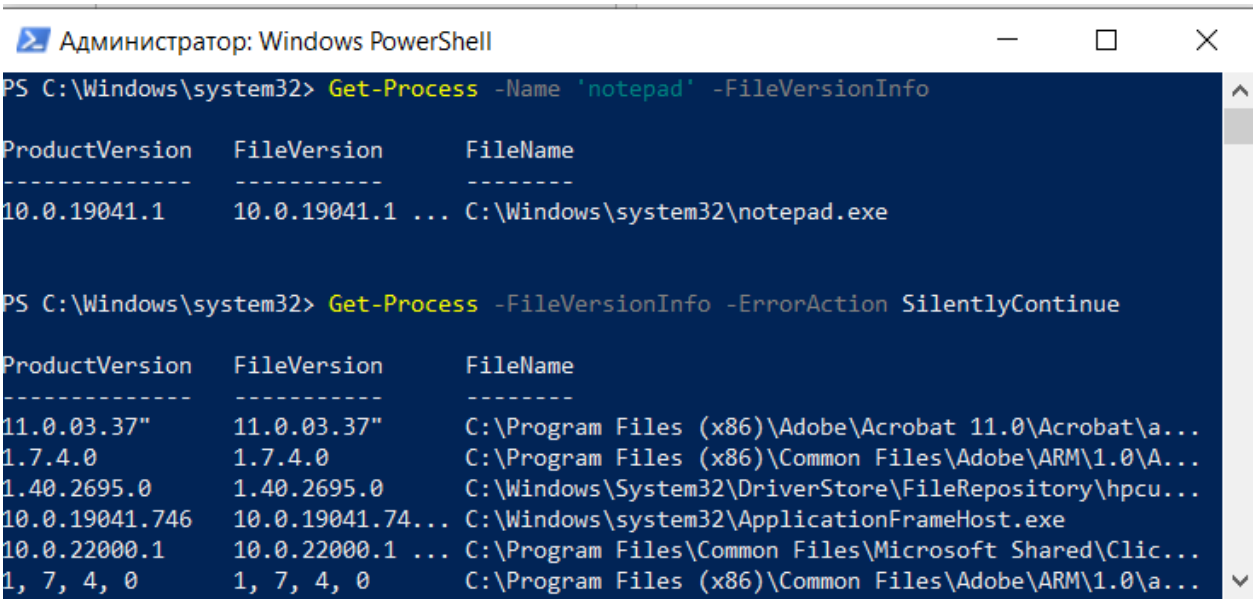


Рис. 4.3 – Результати виконання командлета get-process з заданими аргументами

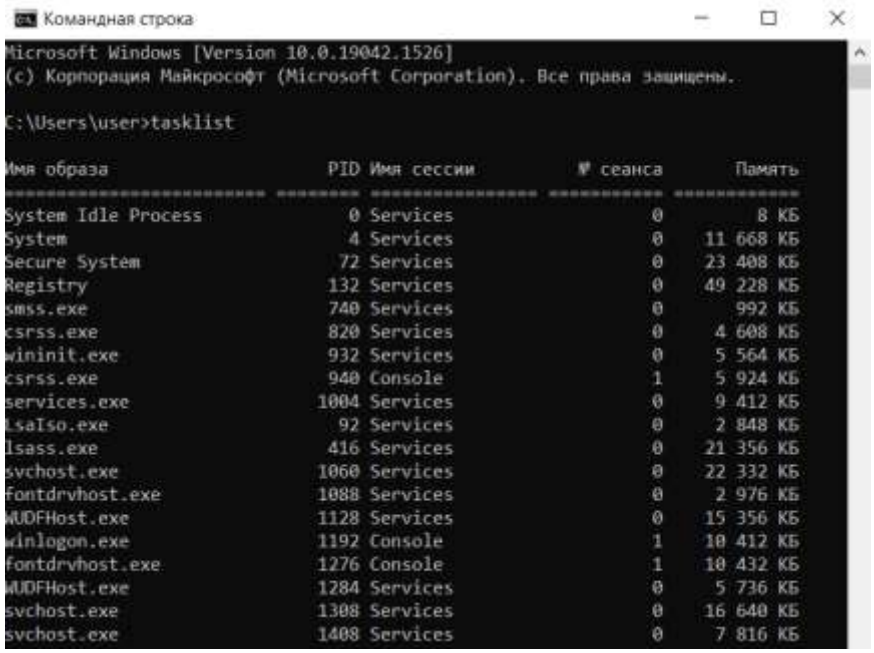


Рис. 4.4 – Результати виконання команди tasklist в командному рядку Windows

В командному рядку існує можливість запуску командної оболонки PowerShell та роботі з нею в консолі cmd. Приклад роботи з командолетом get-process наведено на рис.4.5.

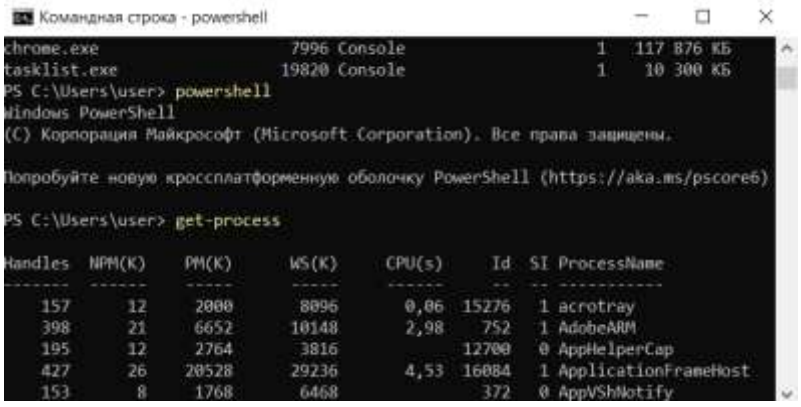


Рис. 4.5 – Результати виконання команди get-process в cmd

В PowerShell реалізована система допомоги, яка дозволяє отримати інформацію по синтаксису та використанню будь-якого командлета. Для цього використовується командлет get-help ‘ім’я командлета’, наприклад, для командлета get-process оболонка виведе всю інформацію (рис. 4.6).

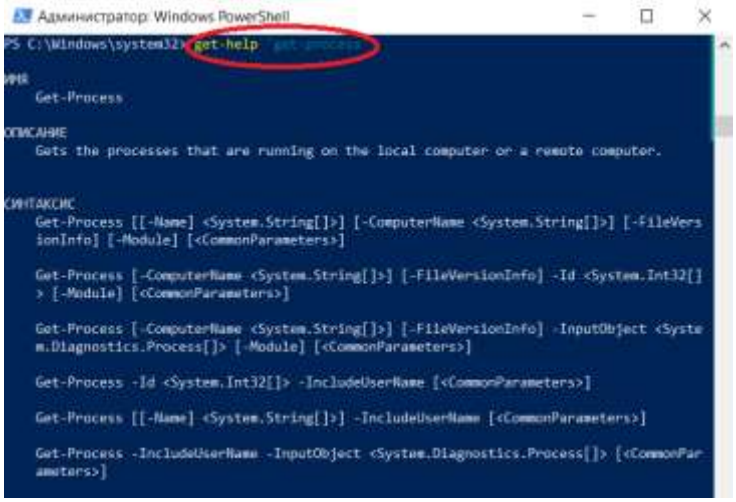


Рис. 4.6 – Результати виконання командлета get-help

Щоб отримати приклади використання командлета, необхідно вказати додатковий аргумент ‘examples’. Приклади використання командлету get-process можна отримати наступним чином: get-help Get-Process -examples.

Запуск та зупинка процесів Windows

Для запуску файлу процесом використовується командлет start-process, синтаксис якої наступний:

Start-Process [-FilePath] *<string>* ***[[-ArgumentList]*** *<string[]>]* ***[-Credential*** *<PSCredential>]* ***[-LoadUserProfile] [-NoNewWindow] [-PassThru] [-RedirectStandardError*** *<string>]* ***[-RedirectStandardInput*** *<string>]* ***[-RedirectStandardOutput*** *<string>]* ***[-UseNewEnvironment] [-Wait] [-WorkingDirectory <string>] [<CommonParameters>]***

Параметри командлету Start-Process:

FilePath – імя (шлях) файлу (обов’язковий параметр);

ArgumentList – параметри або значення параметрів, які використовуються при запуску процесу;

Credential – обліковий запис користувача (за замовчуванням – поточний);

LoadUserProfile – завантажує профіль користувача;

NoNewWindow – вказує на використання поточного вікна для завантаження (за замовчуванням новий процес стартує в новому вікні);

PassThru – повертає об'єкт запущеного (за замовчуванням цей командлет не формує жодних вихідних даних);

Wait **–** чекає завершення процесу, вимикає командний рядок або утримує вікно до завершення процесу;

WorkingDirectory -місцезнаходження файлу (за замовчуванням – поточний каталог);

Приклади використання командлету ***start-process***, які наведені в системі допомоги ***PowerShell***, наведені на рис. 4.7.

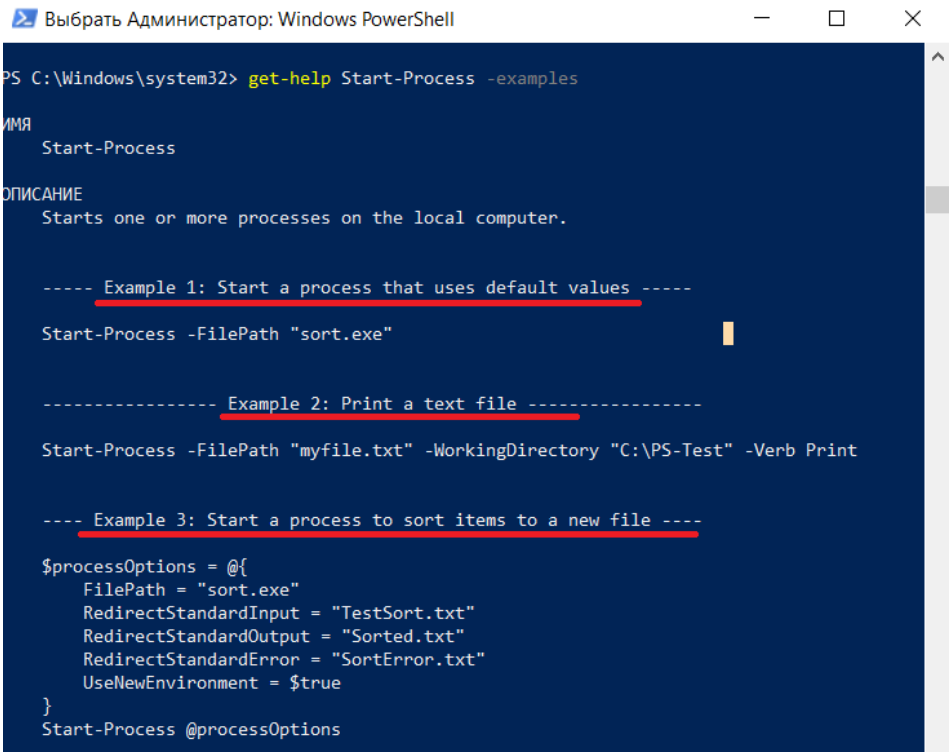


Рис. 4.7 – Результати виконання командлета get-help з прикладами

Завершення будь-якого процесу здійснюється команлетом Stop-Process, синтаксис якого наведено нижче:

Stop-Process [-Id] <Int32[]>/ -InputObject/ -Name [-Force] [-PassThru] [-Confirm] [-WhatIf] [<CommonParameters>]

Командлет ***Stop-Process*** зупиняє один або декілька процесів, що виконуються. Процес можна вказати за допомогою імені, ідентифікатора (***PID***) або об'єкта процесу. Командлет ***Stop-Process*** працює лише з процесами, що виконуються на локальному комп'ютері.

У Windows Vista та пізніших версіях Windows для зупинення процесу,

власником якого не є поточний користувач, необхідно запускати Windows ***PowerShell*** командою "Запуск від імені адміністратора". Крім того, командлет запитує дозвіл, якщо не встановлено параметр ***Force***.

Параметри командлету Stop-Process:

[-Id] <Int32[]>/ -InputObject/ -Name – ідентифікатор процесу (можливо вказати PID процесу, ім’я об’єкту або ім’я файлу)

Force – зупиняє процес без запиту;

PassThru – повертає об'єкт запущеного (за замовчуванням цей командлет не формує жодних вихідних даних);

Confirm – запитує підтвердження перед виконанням команди;

WhatIf – описує, що відбудеться, без фактичного виконання;

CommonParameters – командлет підтримує загальні параметри -Verbose, -Debug, -ErrorAction, -ErrorVariable, -OutBuffer и -OutVariable.

Приклади використання командлету ***stop-process***, які наведені в системі допомоги ***PowerShell***, наведені на рис. 4.8.

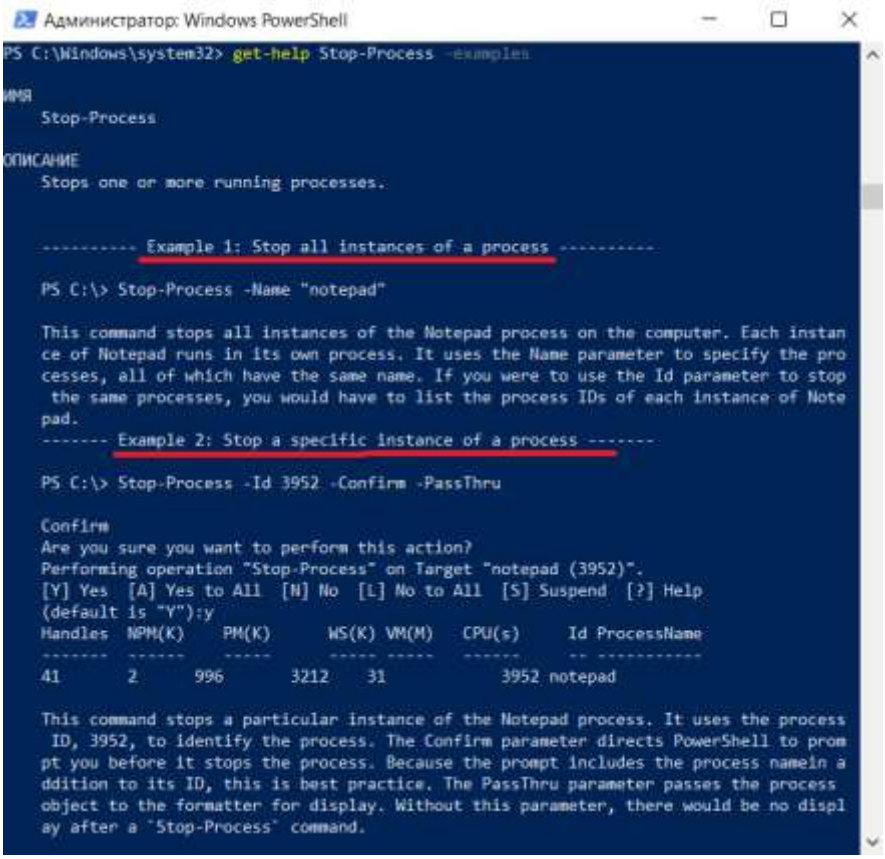


Рис. 4.8 – Результати виконання командлета get-help з прикладами

Робота з процесами ОС Linux

Linux має інструменти для роботи з процесами та потоками. Команда ***ps*** виводить список процесів, якщо вказати її без параметрів, то система виведе тільки ті процеси, які запущені в поточній командній оболонці. Якщо необхідно отримати список всіх запущених процесів слід додати аргументі ***"-eF"***. При цьому система виведе для кожного процесу будуть виведені наступні параметри:

- UID – ім'я користувача, від імені якого працює процес;
- PID – ідентифікатор користувача;
- PPID – ідентифікатор батьківського процесу користувача;
- C – витрати ресурсів процесора у відсотках;
- SZ – розмір процесу;
- RSS – реальний розмір процесу в пам'яті;
- PSR – ядро процесора, на якому виконується процес;
- STIME – час, коли процес було запущено;
- TTY – у випадку, коли процес, прив'язаний до терміналу, виводиться його номер;
- TIME – загальний час виконання процесу (user + system);
- CMD – команда, якою було запущено процес, у випадку, коли програма не може прочитати аргументи процесу, він буде виведено в квадратних скобках.

Для виводу інформації про процеси у вигляді дерева необхідно використовувати комбінацію параметрів "***-efH***". При цьому буде можливо аналізувати ієрархію процесів: батьківські та дочірні процеси.

Відобразити процеси з потоками дозволяє використання аргументу "***L***", при цьому з'явиться два додаткових параметри: ідентифікатор потоку (***LWP***) та кількість потоків процесу (***NLWP***).

Для відображення процесів визначеного користувача слід додати аргумент '***u'*** та ***ім'я користувача***.

Результати роботи вище зазначених команд наведені на рис. 4.9-4.10.

Рис. 4.9 – Результати роботи команди ps

```
user@WIN-9N95J4M4O7U:~$ ps -eF
UID          PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
root           1     0  0  2235   328   0 15:09 ?        00:00:00 /init
root           7     1  0  2235   224   0 15:09 tty1     00:00:00 /init
user           8     7  0  4519  3608   0 15:09 tty1     00:00:00 -bash
user          80     8  0  4666  1896   0 15:10 tty1     00:00:00 ps -eF
user@WIN-9N95J4M4O7U:~$ ps -efH
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 15:09 ?        00:00:00 /init
root           7     1  0 15:09 tty1     00:00:00   /init
user           8     7  0 15:09 tty1     00:00:00     -bash
user          81     8  0 15:11 tty1     00:00:00       ps -efH
user@WIN-9N95J4M4O7U:~$ ps -efL
UID          PID  PPID   LWP  C NLWP STIME TTY          TIME CMD
root           1     0     1  0    2 15:09 ?        00:00:00 /init
root           1     0     6  0    2 15:09 ?        00:00:00 /init
root           7     1     7  0    1 15:09 tty1     00:00:00 /init
user           8     7     8  0    1 15:09 tty1     00:00:00 -bash
user          82     8    82  0    1 15:12 tty1     00:00:00 ps -efL
user@WIN-9N95J4M4O7U:~$ ps -fu user
UID          PID  PPID  C STIME TTY          TIME CMD
user           8     7  0 15:09 tty1     00:00:00 -bash
user          83     8  0 15:13 tty1     00:00:00 ps -fu user
```

Рис. 4.10 – Результати роботи команди ps з різними аргументами

Команда ***top*** відображає інформацію про запущені процеси в режимі реального часу (рис. 4.11).

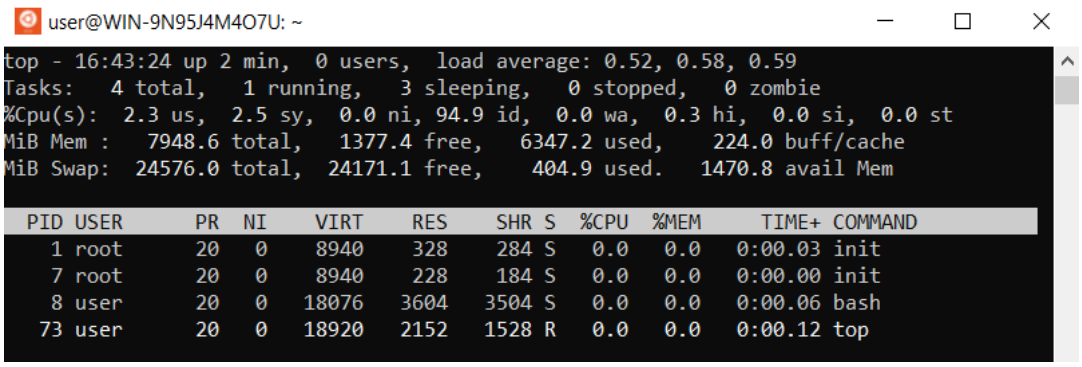


Рис. 4.11 – Результати роботи команди top

При цьому система виведе для кожного процесу наступні параметри:

PID – ідентифікатор процесу.

USER – користувач, якому належить процес.

PR – Пріоритет процесу на рівні ядра.

NI – пріоритет виконання процесу від -20 до 19.

VIRT – загальний обсяг (у кілобайтах) віртуальної пам'яті (фізична пам'ять самого процесу; завантажені з диска файли бібліотек; пам'ять, що

спільно використовується з іншими процесами тощо), що використовується в даний момент.

RES – поточний обсяг (у кілобайтах) фізичної пам'яті процесу.

SHR – обсяг спільної з іншими процесами пам'яті.

S (скор. від "STATUS") – стан процесу:

S (Sleeping) – очікування, що переривається. Процес чекає настання події.

I (Idle) – процес не діє.

R (Running) – процес виконується (або поставлений у чергу на виконання).

Z (Zombie) – зомбі-процес.

% CPU – відсоток використовуваних ресурсів процесора.

%MEM – відсоток пам'яті, що використовується.

TIME+ – кількість процесорного часу, витраченого виконання процесу.

COMMAND – ім'я процесу (команди).

Процеси об'єднані у сесії. Процеси, що належать до однієї сесії, визначаються загальним ідентифікатором сесії – ідентифікатором процесу, який створив цю сесію. Лідер сесії – це процес, ідентифікатор сесії якого збігається з його ідентифікаторами процесу та групи процесів.

Для досвідчених користувачів існує команда ***glances***, яка виводить інформацію про процесі з більш розширеним функціоналом (рис. 4.12).

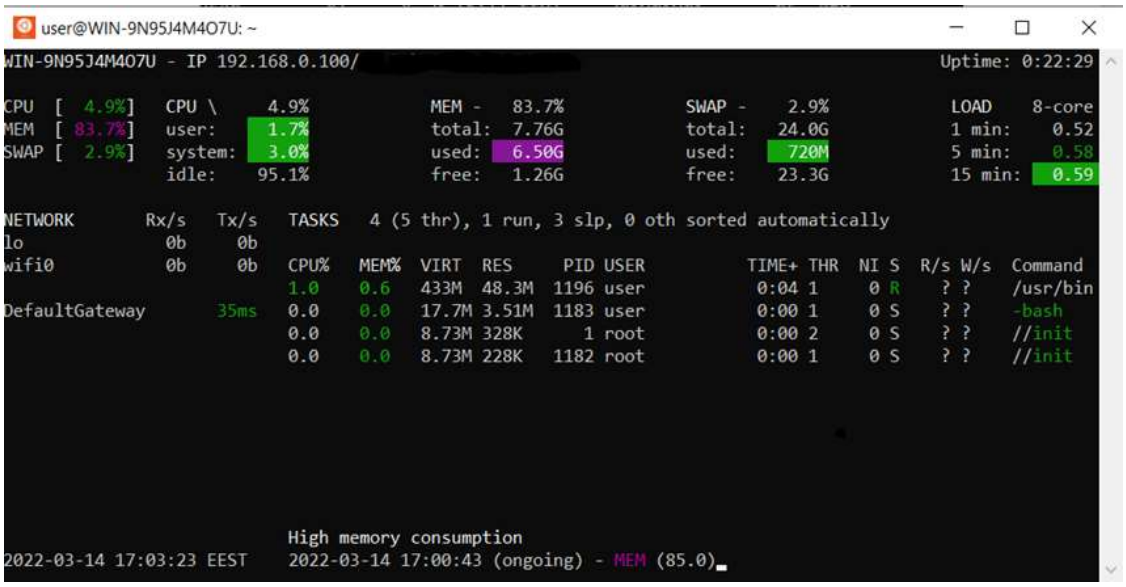


Рис. 4.12 – Результати роботи команди glances

Для управління процесами в Linux використовують наступні команди:

kill – посилає процесу сигнал завершення роботи, опція -9 вимикає безжально процес;

pkill – завершує процес (необхідно вказати ім'я процесу);

pgrep – шукає процес по його імені (і, опціонально, по імені користувача, що його запустив);

killall – завершує всі активні процеси.

Результати роботи деяких з вище зазначених команд наведені на рис. 4.13.

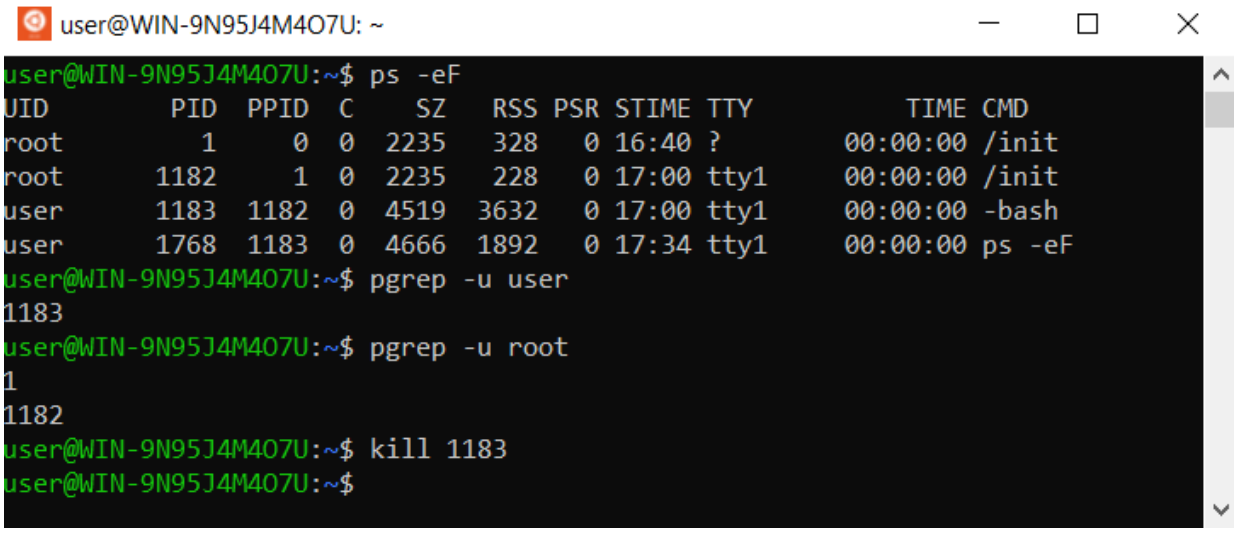


Рис. 4.13 – Результати роботи команд управління процесами

Команда pstree використовується для виведення ієрархії процесів у вигляді дерева. Це дозволяє отримати зручний та зрозумілий огляд процесів та їхніх залежностей.

Приклад використання команди pstree:

pstree

Команда виведе ієрархію всіх процесів від кореневого процесу. Кожен процес представлений виглядом дерева з його sub-процесами. Також можна використовувати опції для деталізації виведення (табл. 4.1).

З цими опціями команда ***pstree*** може бути використана для більш

деталізованого аналізу структури процесів в системі.

Таблиця 4.1

Параметри команди ***pstree***

Параметр	Опис параметру
-a	Показати командну строку для кожного процесу
-c	Сортувати процеси за назвою команди
-h	Відобразити лейбли для кожного процесу
-l	Показати повну командну строку для кожного процесу
-n	Сортувати процеси за PID (ідентифікатор процесу)
-p	Вивести PID (ідентифікатор процесу) для кожного процесу
-u	Відобразити інформацію про користувача для кожного процесу
-v	Вивести версію програми pstree

Команди ***grep*** та ***pgrep*** використовуються для пошуку текстових рядків в текстових файлах та для пошуку процесів за їхніми іменами або атрибутами в системі, відповідно. ***grep в***икористовується для пошуку тексту у вмісті файлів або виведення інших команд, приклад використання:

grep "текст_для_пошуку" test.txt

Команда виведе всі рядки у файлі test.txt, які містять "***текст_для_пошуку***". Команда ***pgrep*** використовується для пошуку ідентифікатора процесу (PID) за іменем процесу або іншими атрибутами, приклад використання:

pgrep "ім'я_процесу"

Команда виведе PID всіх процесів, чиє ім'я відповідає "ім'я_процесу".

Різниця між ***grep*** та ***pgrep*** полягає в тому, що ***grep*** призначений для роботи з текстовими файлами та потоками даних, тоді як ***pgrep*** призначений для пошуку процесів у системі за їхніми іменами або іншими атрибутами. ***pgrep*** не використовується для пошуку вмісту файлів, як ***grep***.

Необхідно відзначити, що обидві команди можуть бути використані в поєднанні з іншими командами та опціями для більш специфічного та потужного пошуку, приклади команд наведені в табл. 4.2 та табл. 4.3.

Таблиця 4.2

Параметри та приклади використання у команді grep

Параметр	Опис	Приклад
-i	Пошук без урахування регістру	grep -i "текст" файл.txt
-r	Рекурсивний пошук у піддиректоріях	grep -r "текст" /шлях/до/директорії
-n	Виведення номерів рядків	grep -n "текст" файл.txt
-v	Виведення рядків, які не містять текст для пошуку	grep -v "текст" файл.txt

Таблиця 4.3

Параметри та приклади використання у команді pgrep

Параметр	Опис	Приклад
-u	Пошук за ідентифікатором визначеного користувача	pgrep -u username
-l	Виведення імен процесів разом із PID	pgrep -l "ім'я_процесу"
-o	Виведення PID найдавнішого процесу	pgrep -o "ім'я_процесу"
-f	Пошук за повною командною стрічкою процесу	pgrep -f "командна_стрічка"

Команда ***awk*** – це потужний інтерпретатор скриптів для обробки та аналізу текстових даних у вигляді таблиць. ***awk*** зазвичай використовується для вибору та обробки конкретних рядків або полів з текстових файлів.

awk 'програма' файл

В команді ***awk*** параметр ***'програма'*** визначає набір правил та дій, які виконуються для кожного рядка у файлі. Для пояснення прикладу використання команди ***awk*** припустимо, що існує файл ***data.txt*** із вмістом:

Ім'я Вік Зарплата
Alice 25 5000
Bob 30 6000
Charlie 35 7000

Команда awk для виводу імен людей та їхніх зарплат виглядає наступним чином:

awk '{print $1, $3}' data.txt

Результат роботи наведеної команди для файлу ***data.txt***:

Ім'я Зарплата
Alice 5000
Bob 6000
Charlie 7000

Визраз ***'{print $1, $3}'*** вказує вивести перше та третє поле (слово) для кожного рядка. ***$1*** та ***$3*** – це змінні awk, які відповідають першому та третьому полю в кожному рядку відповідно.

Команда ***xargs*** використовується для прийняття введених даних з потоку стандартного вводу (***stdin***) та передачі їх як аргументів для інших команд. Вона корисна тоді, коли потрібно виконати якусь операцію на кожному елементі введених даних.

команда | xargs [опції] команда

Розглянемо приклад використання ***xarg***. Примусимо, що існує файли у поточній директорії і необхідно видалити їх за певною умовою. Для цього слід використати команду ***find*** для знаходження файлів та ***xargs*** для передачі їх до команди видалення, конвеєр записується у такому вигляді:

find . -name "*.txt" | xargs rm

Команда ***find*** знаходить всі файли з розширенням .txt у поточній директорії та її піддиректоріях. Команда ***xargs*** передає кожен знайдений файл у якості аргументу для команди ***rm*** (видалення). Цей конвеєр видалить всі файли з розширенням .txt у поточній директорії та її піддиректоріях.

Таблиця 4.4

Параметри та приклади використання у команді xargs

Параметр	Опис	Приклад
-n N	Задає максимальну кількість елементів, які передаються до команди одночасно	echo "file1 file2 file3" | xargs -n 2 echo
-I {}	Задає заповнювач (placeholder) для вставки аргументів	echo "file1 file2 file3" | xargs -I {} cp {} /destination/directory
-d DELIMITER	Задає роздільник для визначення меж аргументів	echo "file1,file2,file3" | xargs -d "," echo

Команда ***sleep*** використовується для затримки виконання процесу на вказаний час. Це корисно, коли потрібно встановити певний інтервал часу між командами або в скриптах для тестування, автоматизації чи планування завдань:

sleep N

де N – це кількість секунд (можна також вказувати в хвилинах чи годинах), на яку потрібно відкласти виконання команди чи скрипта.

Приклад використання ***sleep***:

```
echo "Start"
sleep 5
echo "End"
```

У цьому прикладі між "***Start" і "End***" буде встановлено затримку у 5 секунд. Програма виведе "***Start***", зачекає 5 секунд, а потім виведе "***End***".

Таблиця 4.5

Параметри та приклади використання у команді sleep

Параметр	Опис	Приклад
N	Кількість секунд затримки	sleep 5
Nm	Затримка в хвилинах	sleep 2m
Nh	Затримка в годинах	sleep 1h

Команда ***tr*** використовується для перетворення або вилучення символів з потоку даних. Це корисна утиліта для заміни, вилучення або заміни символів у текстових даних:

tr [опції] 'рядок1' 'рядок2'

де ***рядок1*** та ***рядок2*** визначають маппінг символів з одного набору у інший.

Приклади використання tr для заміни символів у потоці виводу:

echo "Hello, World!" | tr 'o' '0'

У цьому прикладі ***tr*** замінює всі символи 'o' на '0'. Виведення буде Hell0, W0rld!.

Приклади використання tr для вилучення символів з потоку виводу:

echo "Remove all vowels" | tr -d 'aeiou'

У наведеному прикладі команда tr -d вилучає всі голосні ('a', 'e', 'i', 'o', 'u') та виводить наступний потік: Rmv ll vwls.

Приклади використання tr для перетворення регістру:

echo "Convert to UPPERCASE" | tr 'a-z' 'A-Z'

Команда перетворює всі ***літери*** у верхній регістр та виводить наступний потік: CONVERT TO UPPERCASE.

Приклади використання tr для заміни символьних послідовностей:

echo "Replace newlines with spaces" | tr '\n' ' '

У прикладі ***tr*** замінює всі нові рядки на пробіли, таким чином створює однорядкове виведення.

Завдання для самостійного виконання:

1. В оболонці PowerShell відобразити поточні процеси та запустити «Блокнот» (вікно додатку повинно мати максимальний розмір). Запустити командну строку Windows з мінімальним розміром вікна.

2. В терміналі Ubuntu відобразити поточні процеси у вигляді дерева, для визначеного користувача знайти процеси, зупинити знайдені процеси.

3. Написати bash-скрипт для перевірки чи запущений певний процес (задати його ім'я), відобразити його потоки. Якщо кількість потоків більше 3, закрити процес.

4. Написати bash-скрипт, який запускає 10 процесів із затримкою в часі, після запуску останнього закриває всі процеси з парним ID.

5. Оформити звіт з лабораторної роботи. У звіті для кожного завдання навести скрін виконання команд, для завдання 3 і 4 – скрін редактора файлів (nano), скрін з результатами роботи скрипту та відповіді на контрольні запитання.

Контрольні питання:

1. Як отримати список активних процесів Windows?
2. Вкажіть основні аргументи командлета get-process?
3. Для чого використовується командлет get-help?
4. Як отримати приклади використання командлетів?
5. Яким командлетом можливо завершити будь-який процес?
6. Яка команда Linux виводить список процесів?
7. Як отримати інформацію про процеси в Linux в реальному часі?
8. Яке призначення команди pgrep? Наведіть приклади використання.

ЛАБОРАТОРНА РОБОТА №5

Управління пам'яттю в операційних системах Windows і Linux

Мета роботи: ознайомлення і вивчення елементарних понять та прийомів управління пам'яттю в ОС Windows і ОС Linux

Постановка завдання: Закріпити на практиці використання графічних інструментів адміністратора для роботи з пам'яттю, які доступні під ОС Windows та Linux.

Теоретичні відомості

ОС Windows має різні компоненти для реалізації керування пам'яттю. Наприклад, компоненти для відображення сторінок, перекладу, управління віртуальною пам'яттю. Першим вадливим етапом роботи з пам'яттю є отримання інформації про стан, розміри та використання. Інструменти роботи з пам'яттю Windows орієнтовані на адміністраторів, тому вони мають графічний інтерфейс та містять докладну інформацію.

Звичний диспетчер завдань Windows має вбудовані можливості моніторингу системи. Для його запуску натисніть Ctl+Alt+Del або клацніть правою кнопкою миші *Пуск / Запустіть...* і введіть ***taskmgr.*** Далі натисніть *Додаткові відомості*, потім виберіть вкладку *Продуктивність* і натисніть *Пам'ять*. У вікні відображається інформація про загальний обсяг фізичної пам'яті (КБ), який доступний в системі (пам'ять, яка використовується та пам'ять, яка доступна) (рис. 5.1).

Переконайтеся, що працює лише Диспетчер завдань. Потім запустіть браузер і відкрийте кілька вкладок (наприклад, mail.google.com, е-навчання ОДЕКУ, сайт новин або сайт про погоду). Після запуску браузера спостерігаються зміни показників продуктивності фізичної пам'яті.

Вкладка *Процеси* містить інформацію про стан оперативної пам'яті, вона відображає всі *Програми*, які запущені користувачем комп'ютера. Крім цього, відображається інформація про *Фонові процеси*, які самостійно запускає ОС Windows. Вкладка *Програми* відображає список програм, які працюють в даний момент часу. Відсоток використання оперативної пам'яті комп'ютера та абсолютні значення (МБ) використання оперативної пам'яті всіма програмами та фоновими процесами вказано в таблиці (рис. 5.2).

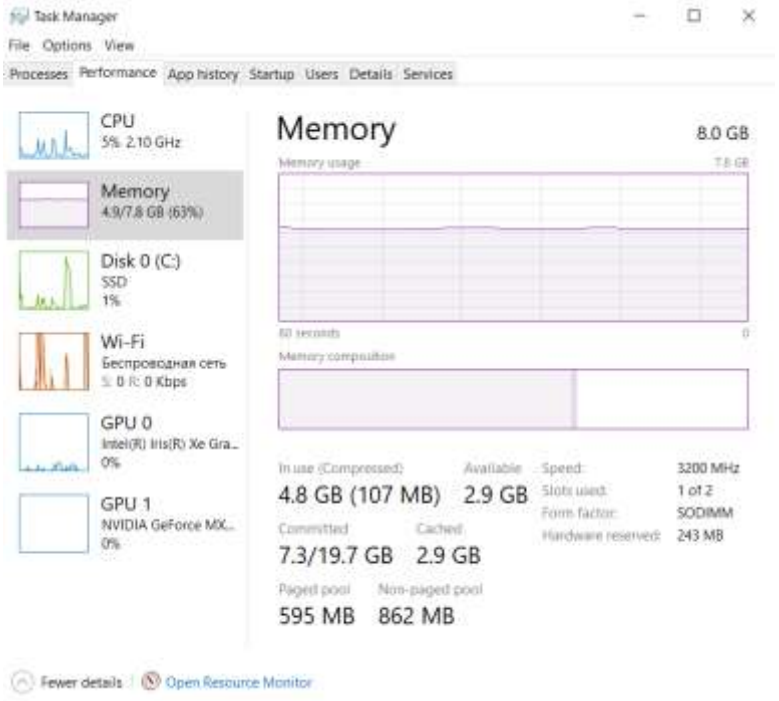


Рис. 5.1 – Відображення інформації про пам'ять в Диспетчері завдань Windows

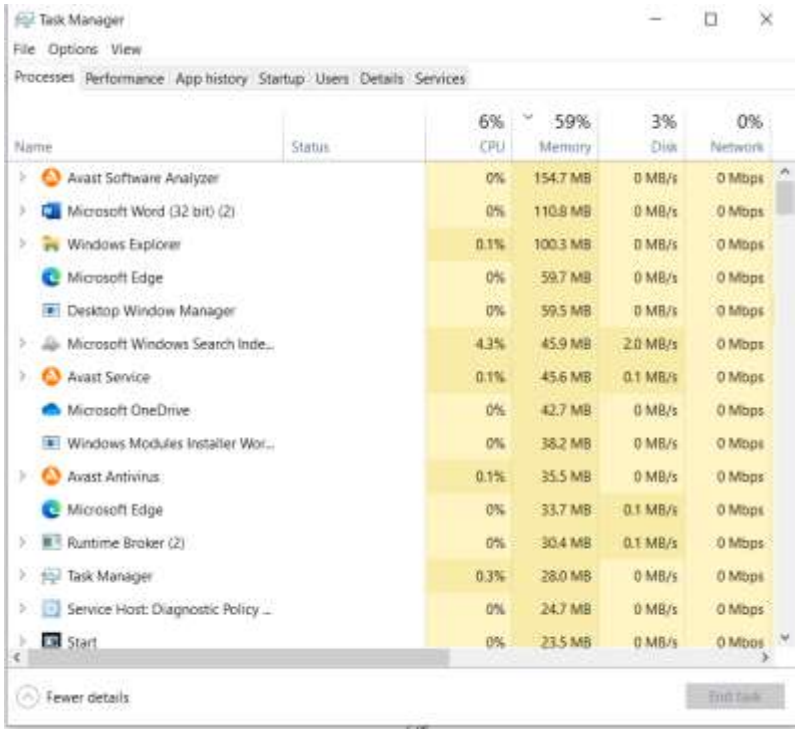


Рис. 5.2 – Відображення інформації про використання пам'яті програмами

Вкладка *Процеси* показує список процесів і пам'ять, яку вони займають, в тому числі фізичну пам'ять, пікове (максимальне) використання пам'яті, віртуальну пам'ять, пули. Слід вказати, що існують деякі обмеження для *Диспетчера завдань*:

1. Список процесів не повний: представлені лише процеси, які зареєстровані у Windows. Зокрема, до цього списку не включаються драйвери пристроїв та деякі системні служби.

2. Вимоги до пам'яті відбивають поточний стан процесу. У списку відображаються обсяги пам'яті, що займаються програмами в даний час, а не їх максимальні значення.

3. Відсутні статистичні дані. Оскільки в *Диспетчері завдань* не виводяться часові характеристики, а лише миттєва картина споживання пам'яті, немає можливості відстежити її зміну.

Оболонка PowerShell має спеціальну утиліту ***TaskList***, яка відображає інформацію про пам'ять, яку використовують програми більш детально (рис. 5.3).

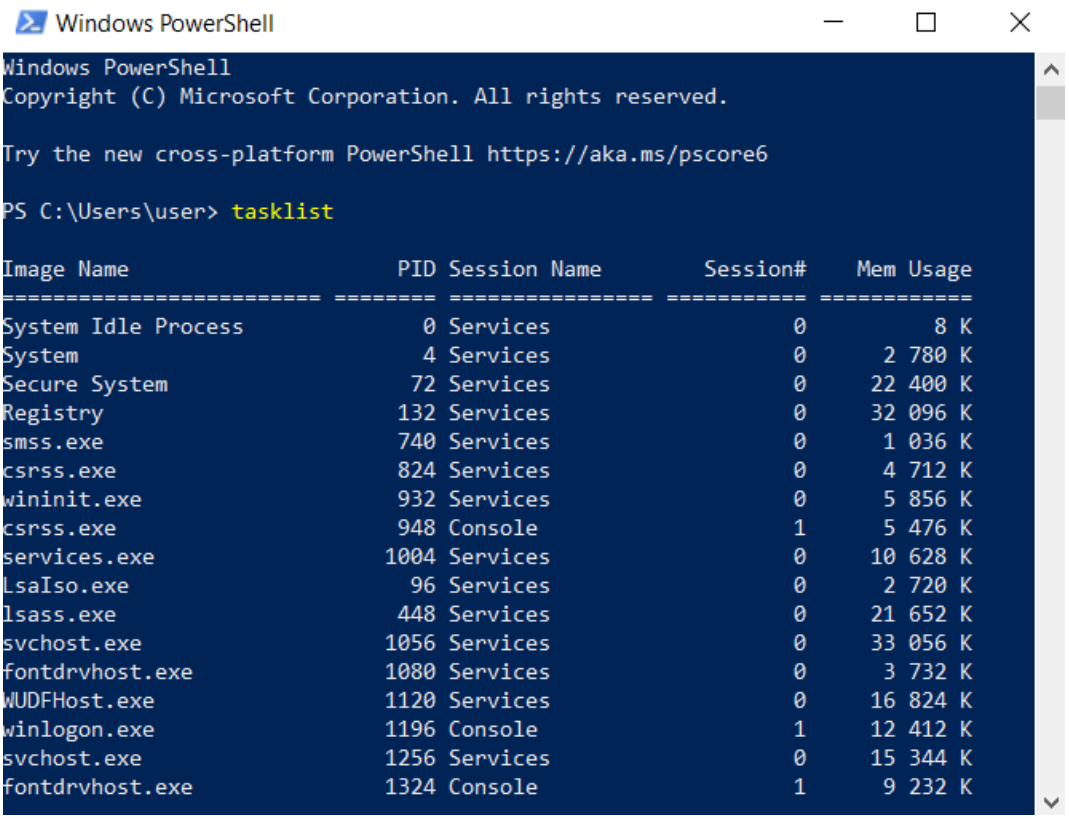


Рис. 5.3 – Результати роботи утиліти ***TaskList*** в PowerShell

Дана утиліта окрім обсягу пам’яті, яку використовує програма, відображає ідентифікатор процесу (*PID*) та ім’я сесії (*Session Name*). Запуск утиліти з параметрами дозволяє отримати додаткову інформацію. Отримати інформацію про параметри утиліти можна вказав ключ ***/?***.

Робота з файлом підкачки

Відомості про основні характеристики організації пам'яті в комп'ютері з ОС Windows можна отримати за допомогою вбудованої службової програми *Відомості про систему*:

1. Повний обсяг встановленої у комп'ютері фізичної пам'яті.
2. Загальний обсяг віртуальної пам'яті та доступної (вільної) в даний момент часу віртуальної пам'яті.
3. Розміщення та обсяг файлу підкачки.

Як і всі сучасні операційні системи загального призначення, Windows використовує віртуальну пам'ять. Кілька інструментів, що входять до складу Windows, надають детальний огляд параметрів системи віртуальної пам'яті. Для отримання інформації про віртуальну пам'ять необхідно виконати наступні дії:

1. Клацніть правою кнопкою миші *Пуск | Запустіть...* і введіть ***msinfo32***, щоб отримати доступ до відомості про систему. В цьому розділі зберігається інформація про обсяги загальної фізичної та віртуальної пам’яті, які доступні у системі (рис. 5.4).

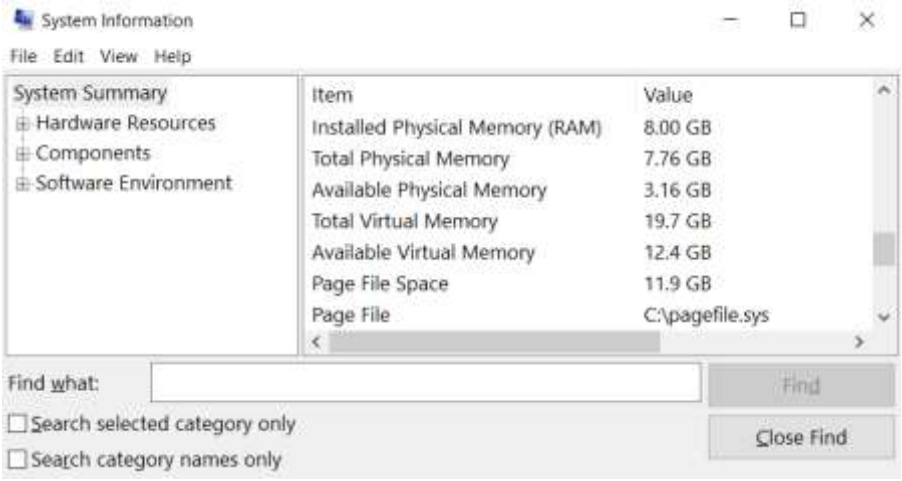


Рис. 5.4 – Результати роботи команди ***msinfo32***

2. Відкрийте *Провідник файлів* , клацніть правою кнопкою миші *Цей комп'ютер* , а потім виберіть *Властивості*. Натисніть *Додаткові параметри системи* , виберіть вкладку *Додатково* , а потім у розділі *Продуктивність* натисніть кнопку *Налаштування*. Виберіть вкладку *Додатково*. У розділі *Віртуальна пам'ять* встановлено розмір файлу підкачки (рис. 5.5).

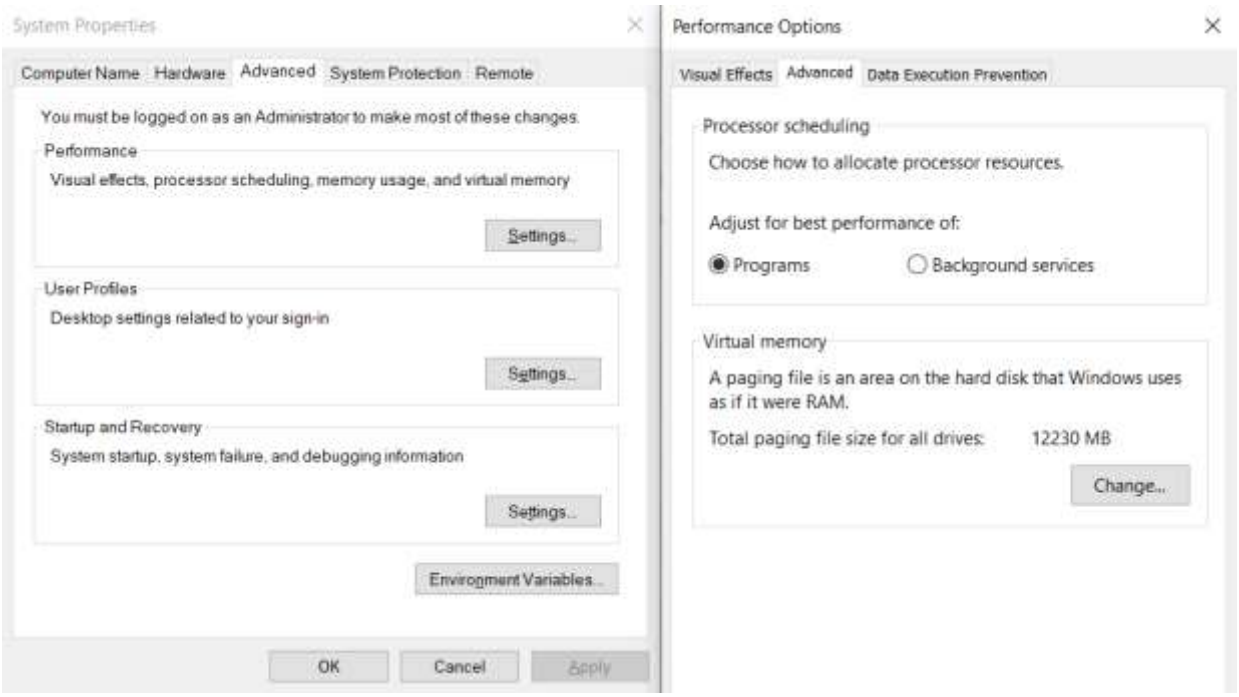


Рис. 5.5 – Налаштування властивостей файлу підкачки

Файл підкачки – це область жорсткого диска, яка використовується ОС Windows для зберігання даних оперативної пам'яті. Він створює ілюзію, що система має більший обсяг оперативної пам'яті, ніж це є насправді.

За замовчуванням Windows видаляє файл підкачки після кожного сеансу роботи і створює його під час завантаження операційної системи. Розмір файлу постійно змінюється в міру виконання програм та контролюється операційною системою. Зазвичай використовується єдиний файл підкачки, що розташований на тому диску, де і операційна система. Проблеми, що виникають у цьому випадку пов'язані з виникненням файлу підкачки великого розміру, що призводить до дефіциту дискового простору і до збільшення непродуктивних витрат на організацію сторінкового обміну, і з фрагментацією файлу підкачки, що призводить до істотного зниження продуктивності внаслідок частого звернення до жорсткого диска.

Ефективність використання файлу підкачки досягається за рахунок:

1. Використанням двох жорстких дисків.

2. Розташуванням його на жорсткому диску у вигляді фрагментів великого обсягу.

3. Періодичним видаленням файлу підкачування для того, щоб уникнути його фрагментації.

4. Встановлення оптимального значення розміру файлу підкачки.

Основне правило визначення розміру файлу підкачки полягає в тому, що при невеликому обсязі оперативної пам'яті файл підкачки повинен бути досить великим, а при великому обсязі оперативної пам'яті (512 Мбайт і більше) файл підкачки можна зменшити. Рекомендується встановити вихідний розмір файлу підкачки, що дорівнює розміру фізичної пам'яті, а максимальний розмір не більше двох розмірів фізичної пам'яті. Для встановлення розміру файлу підкачки необхідно виконати дії в п. 2.

У вікні Параметри швидкодії натиснути кнопку Змінити. Попередньо слід вибрати принцип розподілу часу процесора (для оптимізації роботи програм, якщо це комп'ютер, або служб, що працюють у фоновому режимі, якщо це сервер). Крім того, необхідно встановити режим використання пам'яті. Для комп'ютера – оптимізувати роботу програм, для сервера – системного кеша. Після цього слід натиснути кнопку Задати і переконатися, що нове значення файлу підкачки встановлено. Далі необхідно перезавантаження комп'ютер.

Внаслідок фрагментації жорсткого диска при першому створенні файлу підкачки жорсткий диск зазвичай не готовий до його розміщення. Тому спочатку потрібно виконати дефрагментацію диска і лише потім створити файл підкачки, щоб помістити його в єдину область диска, наступним чином:

1. Якщо комп'ютер має єдиний жорсткий диск, встановити мінімальний розмір файлу підкачки (2 Мбайт).

2. якщо є два жорсткі диски, перемістити файл підкачки на диск з більшим вільним обсягом пам’яті.

3. Провести дефрагментацію диска (у другому випадку – швидкого). Для повної дефрагментації необхідно виконати кілька проходів.

4. Встановити файлу підкачки бажаний розмір.

В результаті робота з файлом підкачки стане максимально швидкою, а процесорна потужність та дисковий простір будуть використовуватися

ефективно.

3. Закрийте вікно налаштувань системи; залиште *Провідник файлів* запущеним. Клацніть правою кнопкою миші *Пуск* | *Запустіть...* і введіть ***resmon***, щоб запустити *Монітор ресурсів*. Виберіть вкладку *Пам'ять*. Наведіть курсор на заголовок стовпця *Commit* (KB) , щоб отримати пояснення які параметри він містить. Натисніть будь-який заголовок, щоб відсортувати його (рис. 5.6).

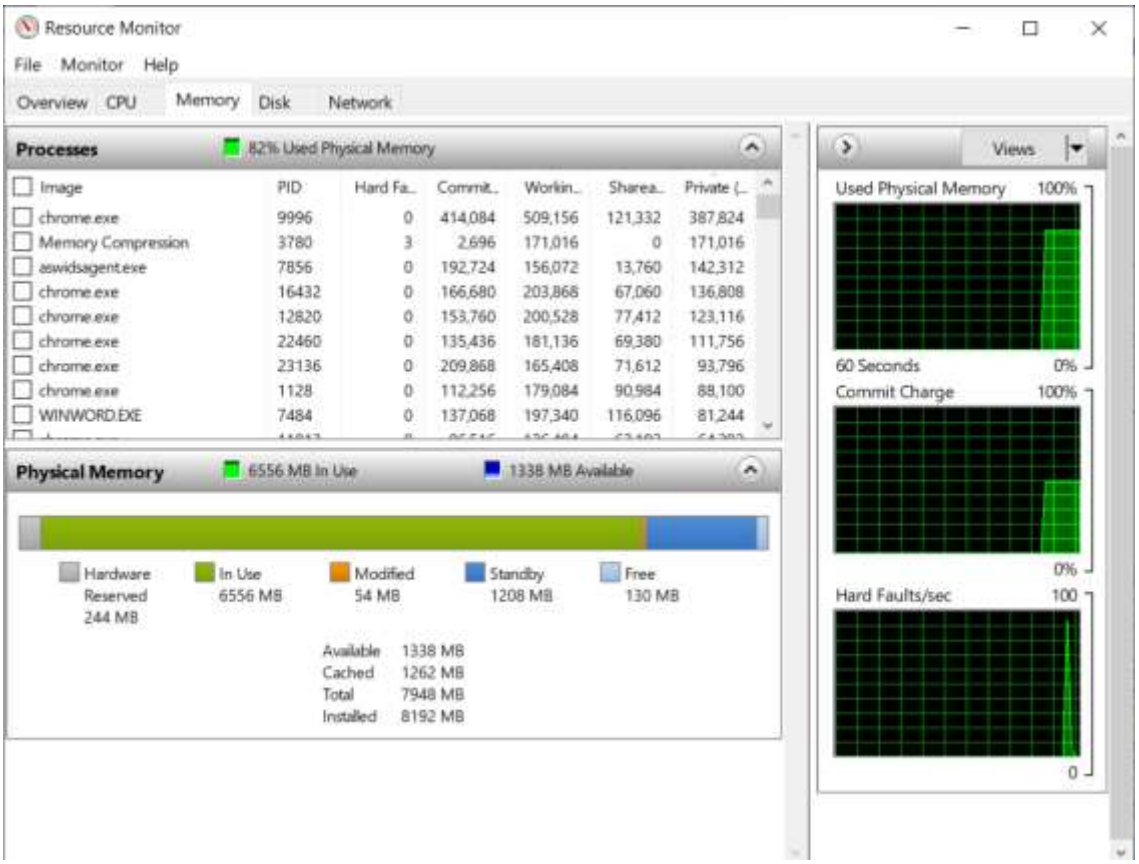


Рис. 5.6 – Монітор ресурсів (вкладка Пам’ять)

Оптимізація роботи віртуальної пам’яті

Підвищення продуктивності роботи віртуальної пам'яті зводиться до наступних дій:

1. Визначення необхідного обсягу фізичної пам'яті.
2. Встановлення раціональної інтенсивності сторінкового обміну.
3. Оптимізація розміру та розміщення файлу підкачки.

Для визначення вимог до пам'яті необхідно до обсягу пам'яті, що

використовується ОС для роботи операційної системи, додати число користувачів, помножене на середній розмір файлів даних, відкритих користувачем (для клієнтського комп'ютера); та число програм, запущених на комп'ютері-сервері, помножене на середній розмір цих програм.

Windows задовольняє вимоги програм до пам'яті шляхом використання вільних (доступних) байтів. Коли обсяг вільної пам'яті знижується до рівня нижче за певне значення, операційна система починає поповнювати його, відбираючи пам'ять у робочих множин або менш активних програм.

Робоча множина – це виділена операційною системою для процесу частина фізичної пам'яті після створення. Якщо пам'яті недостатньо для задоволення вимог усіх активних програм, використовується файл підкачки, що знижує продуктивність.

Для визначення обсягу пам'яті, що використовується програмами, слід спостерігати за певних лічильників на діаграмі *Системного монітора*. Почати можна зі спостереження за лічильником *Процес\Робоча множина*. Значення робочої множини цікавить, коли лічильник *Пам'ять\Доступно байт* опускається нижче певного порога.

Спостереження за ситуаціями, що породжують нестачу пам'яті, рекомендується починати з наступних лічильників:

1. *Пам'ять\Доступно байт*, який показує поточний обсяг пам'яті в байтах, доступний для використання процесами.

2. *Пам'ять\Обмін сторінок в с*, який показує кількість сторінок, отриманих з диска через необхідність звернення до цих сторінок або записаних на диск для звільнення вільної пам'яті в робочій множині.

Низькі значення лічильника *Доступно байт* (4 Мбайт і менше) вказують на загальну нестачу пам'яті на комп'ютері або на те, що будь-яка програма не звільняє пам'ять. Велике значення лічильника *Обмін сторінок в с* (досягає або перевищує 20 с) може вказувати на нестачу пам'яті, або бути результатом роботи програми, яка використовує файл, який відображається в пам'ять. Щоб визначити, чи є причиною остання обставина, потрібно спостерігати за лічильниками «*Доступно байт*», «*Обмін сторінок в с*» та «*Файл підкачки % використання»*.

Детальний аналіз причин виникнення нестачі пам'яті потребує спостереження за лічильниками:

1. Пам'ять\Доступно байт та Пам'ять\Байт виділеної віртуальної пам'яті, щоб відстежити зміни обсягу пам'яті.

2. Процес\Байт виняткового користування, Процес\Робоча множина та Процес\Лічильник дескрипторів процесів, які, як передбачається, викликають нестачу пам'яті.

3. Пам'ять\Байт у невивантажуваному сторінковому пулі, Пам'ять\Розподілів у невивантажуваному сторінковому пулі, Процес (ім'я_процесу)\Байт у невивантажуваному сторінковому пулі, якщо передбачається, що нестача пам'яті викликана процесом ядра.

Оскільки надмірна підкачка тягне завантаження жорсткого диска, в результаті надмірної підкачки сторінок, крім нестачі пам'яті, можливе також виникнення вузького місця в дисковій системі. Тому якщо при визначенні причини надлишкового підкачування сторінок брак пам'яті не простежується явно, разом з лічильниками пам'яті слід спостерігати за такими лічильниками використання диска: Логічний диск % активності диска, Фізичний диск\Середня довжина черги диска.

Дані лічильників *Читання сторінок/с*, *% активності диска* та *Середня довжина черги диска*, що показують поєднання низької активності читання сторінок з високими значеннями активності диска та середньої довжини черги диска, вказують на наявність вузького місця в дисковій системі. Однак, якщо збільшення довжини черги не супроводжується зменшенням частоти читання сторінок, це означає брак пам'яті.

Щоб визначити вплив надмірної підкачки на активність диска, потрібно перемножити значення лічильників «Фізичний диск\Середній час звернення до диска (с)» та «Пам'ять\Обмін сторінок в с». Якщо добуток цих лічильників перевищують 0,1, підкачка займає понад 10% часу доступу до диска. Якщо така ситуація спостерігається тривалий час, слід збільшити обсяг пам'яті.

Доцільно також перевірити залежність надмірного підкачування від запущених програм. Для цього слід зупинити (якщо можливо) роботу програми, коли робоча множина має найбільше значення, і подивитися, як при цьому зміниться частота підкачування сторінок. При виявленні надмірної підкачки потрібно перевірити значення лічильника «Пам’ять\Обмін сторінок в с», що показує кількість сторінок, які мають бути прочитані з диска за відсутності їх у фізичній пам'яті. Цей лічильник відрізняється від лічильника «Помилок сторінки/с», що вказує лише на те, що доступ до даних не отримано негайно, тому що вони були знайдені у заданій робочій множині сторінок пам'яті.

Способи, що дозволяють оптимізувати використання файлу підкачки підвищення продуктивності:

1. Файл підкачки слід розміщувати на окремому логічному диску.

2. За наявності кількох жорстких дисків файл підкачки слід розділити, це підвищує швидкість роботи з ним, оскільки доступ до даних на кількох жорстких дисках здійснюється одночасно.

3. Якщо є два жорсткі диски, з яких один швидше за інший, більш ефективним рішенням буде розміщення файлу підкачки тільки на більш швидкому жорсткому диску.

4. Рекомендується встановити розмір файлу підкачки в 1,5 – 2 рази більше за розмір встановленої оперативної пам'яті. Визначити розмір файлу підкачки можна, дізнавшись у провіднику розмір Pagefile.sys.

5. Якщо на жорсткому диску є вільне місце, можна збільшити розмір файлу підкачки. При запуску кількох програм одночасно, зі збільшенням розміру файлу підкачки їх запуск може прискоритися.

6. Рекомендується збільшити вихідний розмір файлу підкачки, щоб при запуску програм системі не доводилося збільшувати розмір файлу підкачки, фрагментуючи його.

7. Коли розмір файлу підкачки досягає максимального, з'являється повідомлення про можливу зупинку роботи системи. Щоб з'ясувати, чи досягає розмір файлу підкачки максимального значення, потрібно порівняти реальний розмір файлу з його максимальним розміром, який визначається у вікні «*Властивості системи*», що відкривається з *Панелі управління*. Якщо ці значення близькі, слід збільшити вихідний розмір файлу підкачки або одночасно запускати меншу кількість програм.

Іншим способом визначення оптимального значення файлу підкачки є використання лічильників файлу підкачки: "Файл підкачки\% використання" та "Файл підкачки\% використання (пік)". Якщо значення лічильника % використання (пік) досягає максимального розміру файлу підкачки або значення лічильника % використання близько до 100 %, можна спробувати збільшити вихідний розмір файлу підкачки.

Якщо файли підкачки розподілені по кількох дисках, як екземпляри лічильників об'єкта «Файл підкачки» відображатимуться повні імена файлів підкачки. Можна або додати лічильник для кожного файлу підкачки, або вибрати екземпляр "_Total" для спостереження за загальною активністю всіх файлів підкачки.

Завдання для самостійного виконання:

1. За допомогою *Диспетчера завдань* визначте поточні значення всіх статистичних параметрів пам'яті. Запустіть до 10 додатків і визначте вузьке місце в системі (ОЗП або ЦП) шляхом аналізу графіків Хронологія використання пам'яті та Хронологія завантаження ЦП. Напишіть нові значення статистичних параметрів пам'яті. Закрийте відкриті програми та запишіть нові значення статистичних параметрів пам'яті, зробіть висновки. Яке значення параметра Пік? Порівняйте з колишнім його значенням та зробіть висновки.

2. Запустіть програми Блокнот, MS Word, MS Excel. За допомогою Диспетчера завдань визначте обсяги пам'яті, що використовуються процесами: фізичну пам'ять, пікове використання пам'яті, віртуальну пам'ять, пули, що вивантажуються і не вивантажуються. Визначте, як ці параметри змінюються при зміні активності програм.

3. Вивчіть довідкову інформацію про параметри запуску утиліти TaskList. Отримайте за допомогою утиліти інформацію про оперативну пам'ять, яка використовується кожним процесом системи. Запустіть програми MS Word та MS Excel. Отримайте за допомогою утиліти TaskList інформацію про PID їх образів та список усіх модулів, завантажених в оперативну пам'ять та таких що використовуються цими процесами. Визначте працюючі служби.

4. За допомогою інструменту Відомості про систему визначте: повний об'єм фізичної пам'яті на комп'ютері, загальний обсяг віртуальної пам'яті, доступної (вільної) віртуальної пам'яті. Перегляньте відомості про використання фізичної пам'яті апаратними компонентами комп'ютера; визначте діапазон адрес пам'яті, який використовується кожною з них. Запустіть кілька програм і за допомогою програми Відомості про систему визначте обсяг ОП. Те ж саме зробіть для модулів і служб, що вивантажуються.

5. Визначте обсяг оперативної пам'яті комп'ютера та рекомендований обсяг файлу підкачки.

6. Створіть два журнали лічильників (бінарного та текстового форматів) та внесіть у них лічильники, що дозволяють оптимізувати віртуальну пам'ять (пам'ять \ доступно байт, пам'ять \ обмін сторінок у сік, файл підкачки \ % використання) та проведіть спостереження за ситуаціями, що породжують нестачу пам'яті. Запустіть журнали лічильників і поспостерігайте за системою. Результати виведіть у таблицю (в Excel) та на діаграми Системного монітора. Виберіть інші лічильники, зазначені у третьому розділі. Виконайте аналіз отриманих результатів та дайте рекомендації щодо покращення конфігурації ПК.

Контрольні питання:

1. Перелічіть основні статистичні параметри, що характеризують фізичну пам'ять обчислювальної системи. Що означає кожна така характеристика? Які утиліти дозволяють набути значення цих характеристик?

2. Які параметри характеризують використання апаратних компонентів комп'ютера? Що означає кожен такий параметр? Які утиліти дозволяють отримати інформацію про ці параметри?

3. Яку інформацію про використання та організацію пам'яті дозволяє отримати утиліта TaskList?

4. Що таке віртуальна пам'ять? Перерахуйте варіанти організації.

5. Що таке файл підкачування? Навіщо він використовується?

6. Як вибрати оптимальний розмір файлу підкачки?

7. Чому фрагментація файлу завантаження знижує продуктивність обчислювальної системи? Як усунути фрагментацію файлу підкачки?

8. У яких випадках ефективніше розміщувати файл підкачки на одному жорсткому диску, а яких – на кількох?

9. Які лічильники дозволяють провести аналіз нестачі пам'яті?

10. Які лічильники дозволяють виконати аналіз впливу надлишкового підкачування на активність дисків?

ЛІТЕРАТУРА

Основна література:

1. Рольщиков В.Б. Операційні системи: конспект лекцій / Одеса: ОДЕКУ, 2015. 151 с.
2. Шеховцов В.А. Операційні системи / Підручник для студентів вищих навчальних закладів. - К: Видавнича група BHV, 2005. 576 с.

Додаткова література:

1. Evi Nemeth. UNIX and Linux System Administration Handbook, 5th Edition / Evi Nemeth, Garth Snyder, Trent Hein, Ben Whaley, Dan Mackin. – Addison-Wesley Professional, 2017. – 1232 p. ISBN-10: 0134277554, ISBN-13: 978-0134277554.
2. Chris Johnson, Jayant Varma. Pro Bash Programming, Second Edition: Scripting the GNU/ Linux Shell, 2nd Edition. – Apress, 2015. – 279 p. ISBN-10: 1484201221, ISBN-13: 978- 1484201220.
3. Lee Holmes. Windows PowerShell Cookbook: The Complete Guide to Scripting Microsoft's Command Shell, Third edition. – O'Reilly Media, 2013. – 1036 p. ISBN-10: 1449320686, ISBN-13: 978-1449320683.
4. Погребняк Б.І. Операційні системи: навч. посібник / Б.І.Погребняк, М.В.Булаєнко; 13 Харків. нац. ун-т міськ. госп-ва ім. О.М. Бекетова. – Харків: ХНУМГ ім. О.М. Бекетова, 2018. – 104с.
5. Федотова-Півень І.М. Операційні системи: навчальний посібник. [за ред. В.М. Рудницького] / І.М. Федотова-Півень, І.В. Миронець, О.Б. Півень, С.В. Сисоєнко, Т.В. Миронюк; Черкаський державний технологічний університет. – Харків: ТОВ «ДІСА ПЛЮС», 2019. – 216 с.

Інформаційні ресурси:

1. Репозитарій бібліотеки ОДЕКУ URL: http://eprints.library.odeku.edu.ua/.
2. Xshell 4 User Guide Secure Terminal Emualtor – Seoul: NetSarang Computer, Inc., 2011. – 157 p. URL: http://www.netsarang.com/docs/xshell4_manual.pdf
3. A Program for Directing Recompilation GNU make Version 3.82 / Richard M. Stallman, Roland McGrath, Paul D. Smith – Boston: Free Software Foundation, 2010 – 192 p. URL: http://www.gnu.org/software/make/manual/make.pdf