МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра комп'ютерних наук

ОПЕРАЦІЙНІ СИСТЕМИ

Методичні вказівки до виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерні науки» галузі знань 12 Інформаційні технології спеціальності 122 Комп'ютерні науки денної та заочної форм навчання

Частина 1

2025

УДК 004.02(07)
С38

До друку
Голова вченої ради ФКІТ _____І. С. Кондіус
(підпис)

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
Директор бібліотеки _____Н. П. Поліщук
(підпис)

Затверджено вченою радою ФКІТ,
протокол №__ від «__» _____ 2025 року.

Розглянуто та схвалено на засіданні кафедри інженерії програмного забезпечення ЛНТУ,
протокол №__ від «__» _____2025 року.

Завідувач кафедри КН _____В. О. Ліщина

Укладач: _____ Р. А. Хиць, асистент кафедри комп'ютерних наук ЛНТУ
(підпис)

_____ Ю. Й. Тулашвілі, доктор педагогічних наук, професор кафедри
(підпис) комп'ютерних наук ЛНТУ

Рецензент: _____ Н. М. Ліщина, кандидат технічних наук, доцент, доцент кафедри
(підпис) інженерії програмного забезпечення ЛНТУ

Операційні системи: методичні вказівки до виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерні науки» галузі знань 12 Інформаційні технології спеціальності 122 Комп'ютерні науки денної та заочної форм навчання Частина 1/ уклад. Р. А. Хиць, Ю. Й. Тулашвілі. Луцьк: ЛНТУ. 2025. 63 с.

У методичних вказівках наведені лабораторні роботи з дисципліни.
Призначені для студентів спеціальності 122 «Комп'ютерні науки» денної форми навчання.

ЗМІСТ

ВСТУП

Курс «Операційні системи» є однією із навчально-професійних дисциплін спеціальності, яка формує професійні знання майбутніх програмістів, спеціалістів в галузі комп'ютерних технологій. Вивчення курсу забезпечує ознайомлення з особливостями роботи з операційними системами Windows та Linux, розуміння основ взаємодії та файлового середовища систем, способів взаємодії з нею.

Головною метою даної дисципліни є: знайомство з основними поняттями операційної системи, вироблення навиків взаємодії з системами, вивчення особливостей та аналіз відмінностей систем як у взаємодії, так і організації подальшої діяльності у створенні програм для тих чи інших операційних систем. Освоєння створення скриптів як базовий функціонал систем. Розвиток навичок користування командним рядком та графічними оболонками.

Завдання вивчення курсу «Операційні системи» є теоретична та практична підготовка студентів спеціальності підходу до розв'язання технічних проблем, знання загальних принципів та методів взаємодії з операційними системами сімейства Windows та Linux, а також вміти їх використовувати при вдосконаленні та створенні програмних продуктів, і роботи в системах.

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 програмне забезпечення, яке входить до складу операційної системи (ОС) і 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 [1, с.21]. Наприклад, через командний рядок вводять команди у разі зараження вірусами або «поломки» системних файлів, а також для відновлення Windows (рис. 1.1).

Крім того, командний рядок дозволяє:

– Автоматизувати рутинні задачі за допомогою пакетних файлів (batch files), що значно спрощує виконання складних процедур.

– Виконувати адміністраторські команди, які можуть не бути доступні через графічний інтерфейс, такі як управління мережевими налаштуваннями та користувачами.

– Налагоджувати проблеми з мережевими підключеннями, використовуючи різноманітні команди для діагностики та тестування зв'язку.

– Налаштовувати системні параметри, які зазвичай недоступні для зміни через GUI, наприклад, управління службами та драйверами.

Методи запуску командного рядка:

– Пуск → Усі програми → Стандартні → Командний рядок.

– Пуск → Виконати → вводимо cmd.exe.

– Win + R – вводимо cmd.

– Запустіть файл cmd.exe, який знаходиться в системній папці.

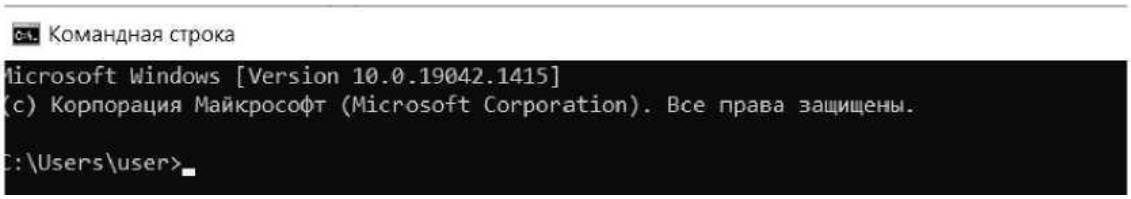


Рисунок 1.1 – Вікно інтерпретатора Windows

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.

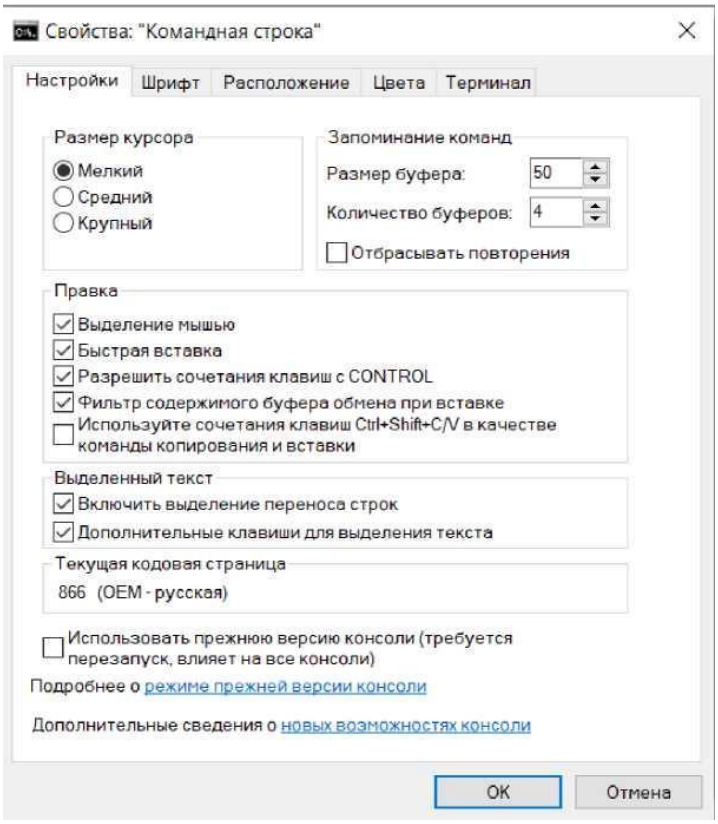


Рисунок 1.2 – Вікно «Властивості» командного рядка

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

– Мережеві команди:

a) ipconfig /all – відображає детальну інформацію про мережеві адаптери. Показує IP-адреси, маски підмережі, шлюзи та DNS-сервери, що допомагає діагностувати мережеві налаштування;

b) getmac – отримує MAC-адреси всіх мережевих карт. Використовується для ідентифікації пристроїв у локальній мережі;

c) arp -a – відображає ARP-таблицю. Містить IP-адреси разом з їхніми відповідними MAC-адресами, що допомагає зрозуміти, які пристрої нещодавно з’єднувались [2, с. 45];

d) ping [кінцевий_вузол] [ключ] – перевіряє доступність мережевої адреси. Якщо в командному рядку набрати команду ping/?, то з'явиться перелік всіх параметрів цієї команди з поясненнями (рис. 1.3);

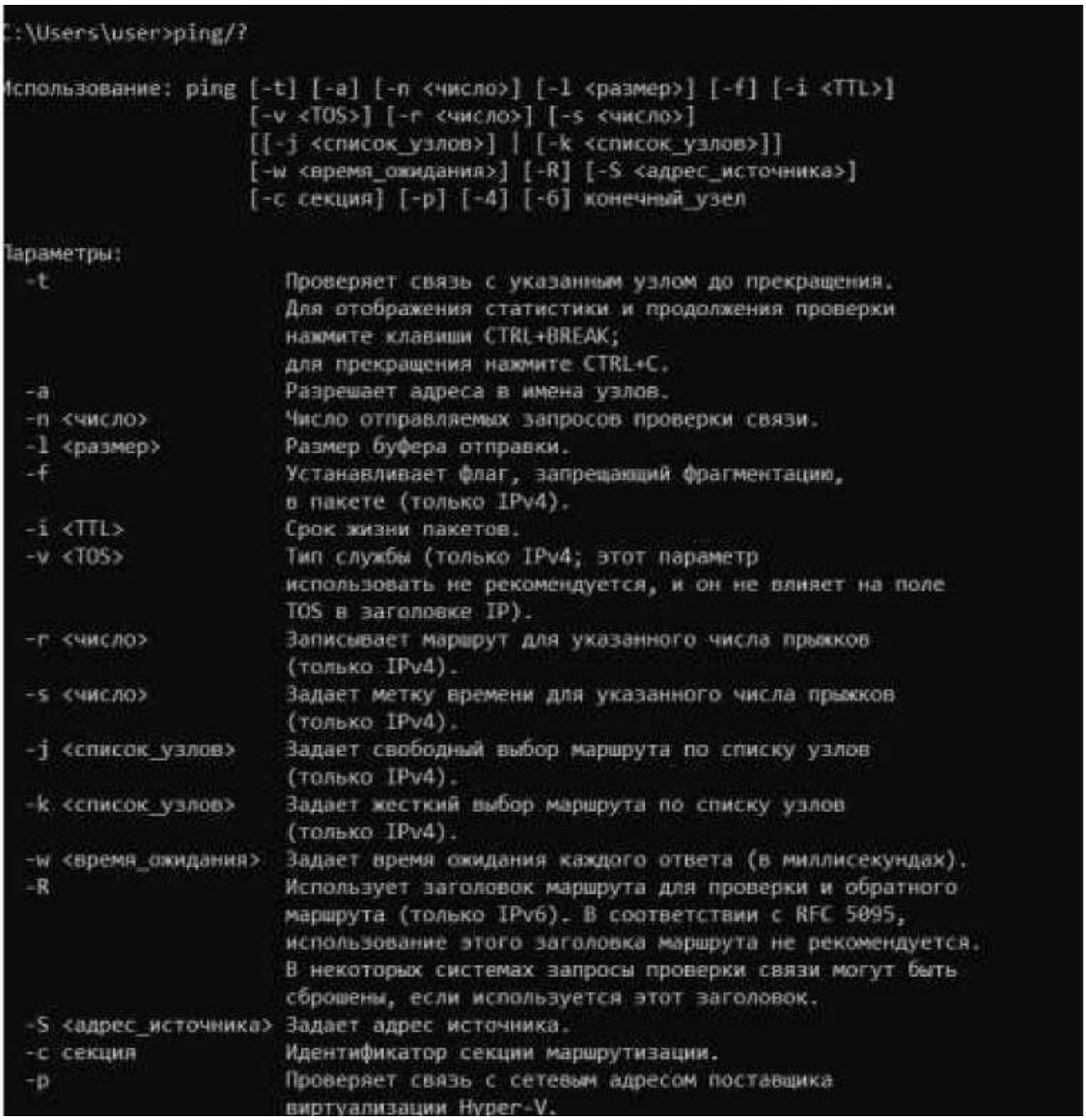


Рисунок 1.3 – Результат виконання команди ping/?

e) tracert [кінцевий_вузол] – трасує маршрут до вказаного мережевого вузла. Визначає, через які маршрутизатори проходить запит, що допомагає виявити проблеми з підключенням;

f) pathping [кінцевий_вузол] – комбінація ping та tracert. Надає детальну інформацію про затримки та втрати пакетів на кожному етапі маршруту [3, с. 87];

g) netstat [ключ] – відображає статистику поточних мережевих підключень. Показує відкриті порти, активні з'єднання, їхній стан та статистику протоколів, що корисно для моніторингу мережі. Аналогічно команді ping, якщо в командному рядку набрати команду netstat/?, то з'явиться перелік всіх параметрів цієї команди з поясненнями (рис. 1.4) ;

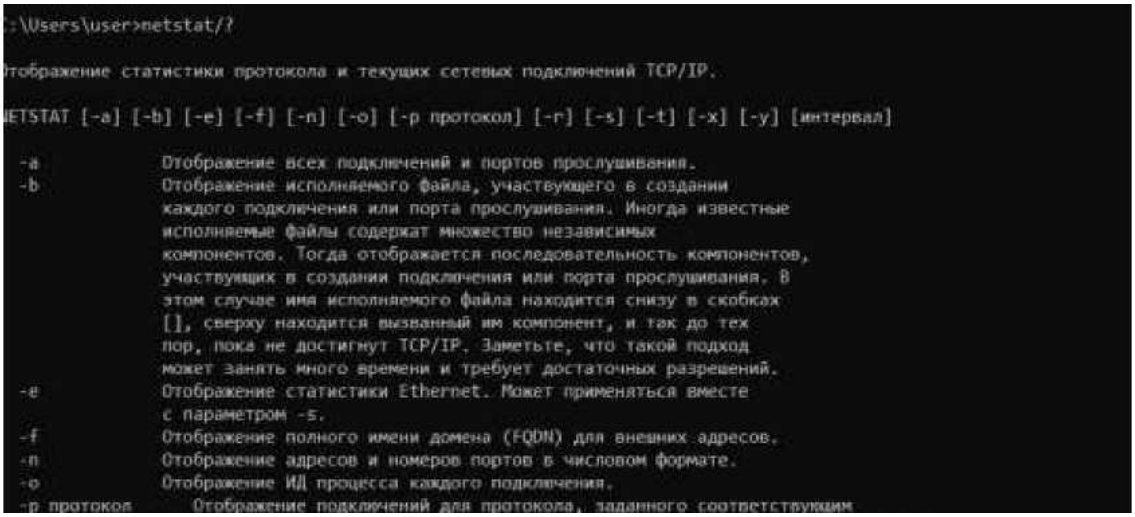


Рисунок 1.4 – Результат виконання команди netstat/?

h) netsh interface ip show address – відображає поточну конфігурацію IP, маски та шлюзу. Використовується для перевірки налаштувань мережевих інтерфейсів і швидкого діагностування проблем;

i) netsh interface ip set address name=«[ім’я мережевого інтерфейсу]» static [IP адреса] [маска] [шлюз] Встановлює статичну IP-адресу, маску та шлюз для зазначеного мережевого інтерфейсу. Корисно для налаштування мережі в умовах, де DHCP не використовується;

j) netsh interface ip show dnsservers – відображає поточну конфігурацію DNS-серверів. Дозволяє перевірити, які DNS-сервери використовуються для резолюції доменних імен;

k) netsh interface ip set dnsserver «[ім’я мережевого інтерфейсу]» static [встановлений dns-сервер] – встановлює статичний DNS-сервер для мережевого інтерфейсу. Допомагає у випадках, коли потрібна конкретна DNS-резолюція;

l) netsh interface ip add dnsserver «[ім’я мережевого інтерфейсу]» [альтернативний dns-сервер] index=2 – додає альтернативний DNS-сервер. Використовується для налаштування резервного DNS-сервера на випадок, якщо основний не працює;

m) route -p add [адреса_мережі] mask [маска] [шлюз] – додає статичний маршрут до таблиці маршрутів. Використовується для управління маршрутами, що особливо корисно в складних мережах.

n) route delete [адреса_мережі] – Видаляє зазначений маршрут. Дозволяє очищати таблицю маршрутів від застарілих або некоректних записів;

o) route print – відображає таблицю маршрутів. Надає детальну інформацію про активні маршрути, що допомагає в діагностиці;

p) nslookup – DNS-клієнт для запитів до DNS-серверів. Використовується для перевірки резолюції доменних імен та діагностики проблем з DNS;

q) ftp [адреса_сервера] – FTP-клієнт для підключення до FTP-серверів. Дозволяє передавати файли між клієнтом і сервером, що корисно для завантаження та завантаження даних.

– Системні команди:

a) shutdown /r – перезавантажує комп'ютер. Використовується для завершення роботи системи з подальшим автоматичним перезапуском, що корисно після установки оновлень;

b) shutdown /s – вимикає комп'ютер. Це команда для безпечного завершення роботи ОС, що дозволяє зберегти всі відкриті файли.

c) qprocess * – відображає список всіх процесів. Допомагає моніторити запущені програми й процеси, включаючи їхній статус та використання ресурсів;

d) chcp – відображає поточне кодування. Показує, яке кодування використовується в командному рядку, що важливо для роботи з текстовими файлами;

e) chcp [кодування] – змінює кодування. Наприклад, chcp 866 – для DOS, chcp 1251 – для Windows-1251, chcp 65001 – для UTF-8. Це корисно при обробці текстів з різними символами;

f) sfc /scannow – перевіряє та відновлює системні файли. Використовується для виявлення та виправлення пошкоджених або відсутніх системних файлів, що може допомогти у відновленні стабільності ОС;

g) wuauclt – управляє оновленнями Windows. Використовується для перевірки наявності оновлень, автоматичного їх завантаження та установки, що важливо для безпеки системи;

h) [команда] > c:\file.txt – перенаправляє вивід команди в файл. Це дозволяє зберігати результати виконання команд для подальшого аналізу або документації;

i) [команда] & [команда] – виконує команди послідовно. Дозволяє виконувати декілька команд в одному рядку, спрощуючи автоматизацію задач;

j) [команда] /? – відображає короткий опис команди. Показує перелік всіх параметрів та їх опис, що допомагає розібратися з використанням конкретної команди;

k) help – відображає список основних команд (рис. 1.5). Дозволяє швидко ознайомитися з доступними командами в командному рядку, що може бути корисно для новачків.

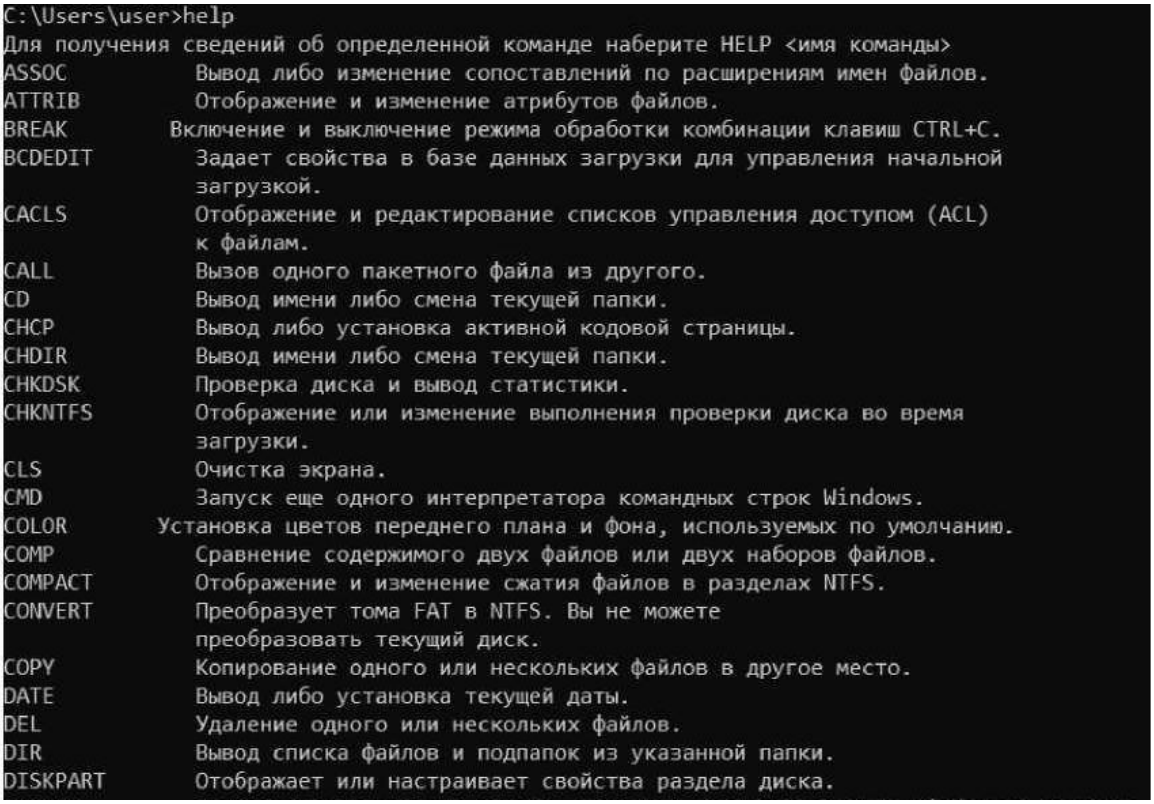


Рисунок 1.5 – Результат виконання команди help

– Команди виклику системних утиліт:

a) cttune – налаштування згладжування шрифтів. Використовується для покращення якості відображення тексту на екрані;

b) compmgmt.msc – управління комп’ютером. Відкриває консоль управління, де можна керувати дисками, подіями та службами;

c) calc – виклик калькулятора. Дозволяє швидко виконувати арифметичні обчислення;

d) charmap – таблиця символів. Надає доступ до всіх символів, які можна використовувати в текстах, включаючи спеціальні символи;

e) chkdsk – утиліта перевірки дисків. Виконує перевірку дисків на наявність помилок та відновлює їх;

f) control – запуск панелі управління. Дозволяє отримати доступ до всіх системних налаштувань Windows;

g) control color – властивості екрана – оформлення. Надає можливість налаштувати колірну палітру вікон і елементів інтерфейсу;

h) control desktop – властивості екрана. Використовується для налаштування фону робочого столу та інших параметрів;

i) control folders – властивості папки. Дозволяє налаштувати параметри відображення та поведінки папок у Windows;

j) devmgmt.msc – диспетчер пристроїв. Використовується для управління встановленими пристроями, їхніми драйверами та вирішення проблем;

k) diskmgmt.msc – управління дисками. Дозволяє розглядати та змінювати розділи дисків, їх форматування та інші параметри;

l) dxdiag – засіб діагностування DirectX. Надає інформацію про компоненти системи, які використовуються для мультимедіа;

m) dfrg.msc – дефрагментація дисків. Використовується для дефрагментації жорстких дисків, що покращує швидкість доступу до даних;

n) eventvwr.msc – перегляд подій. Відкриває журнал подій, що містить інформацію про системні помилки, попередження та інші події;

o) eudcedit – редактор особистих символів. Дозволяє створювати та редагувати власні символи для використання у документах;

p) explorer – запуск Windows Explorer. Дозволяє переглядати файли та папки на комп'ютері;

q) fsmgmt.msc – спільні папки. Відкриває інтерфейс для управління спільними папками на комп'ютері;

r) gpedit.msc – групова політика. Дозволяє налаштовувати параметри групової політики для локальних користувачів і комп'ютерів;

s) iexplore – запуск Internet Explorer. Дозволяє відкривати веб-сторінки у браузері Internet Explorer;

t) lusrmgr.msc – локальні користувачі. Використовується для управління обліковими записами локальних користувачів;

u) mmc – виклик консолі Microsoft Management Console. Дозволяє створювати та управляти різними адміністративними інструментами;

v) mstsc – підключення до віддаленого робочого столу. Дозволяє підключатися до інших комп'ютерів через RDP (Remote Desktop Protocol) ;

w) msconfig – конфігурація системи. Використовується для управління програмами, які запускаються при старті Windows;

x) notepad – запуск Блокнот. Простий текстовий редактор для створення та редагування текстових файлів;

y) perfmon.msc – системний монітор. Відкриває інструменти для моніторингу продуктивності системи;

z) powercfg – налаштування електроживлення ПК. Дозволяє управляти параметрами енергозбереження та живлення;

aa) regedit – редактор реєстру. Дозволяє переглядати та редагувати записи реєстру Windows;

bb) services.msc – служби Windows. Відкриває інтерфейс для управління службами, що працюють у системі;

cc) shrpubw – майстер створення спільної папки. Дозволяє легко створювати спільні папки для доступу інших користувачів;

dd) taskmgr – запуск диспетчера задач. Дозволяє переглядати запущені процеси, активність системи та управління програмами;

ее) wmimgmt.msc – управління WMI (Windows Management Instrumentation). Дозволяє переглядати та налаштовувати параметри WMI для моніторингу системи [5, с. 100].

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-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).

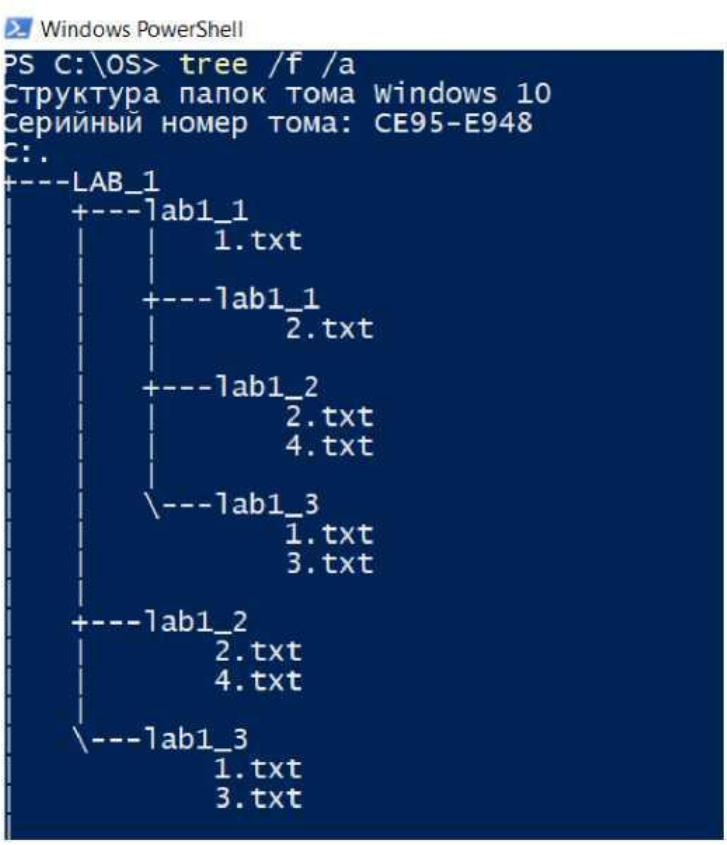


Рисунок 1.6 – Дерево каталогів папки OS створене засобами командного рядка

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

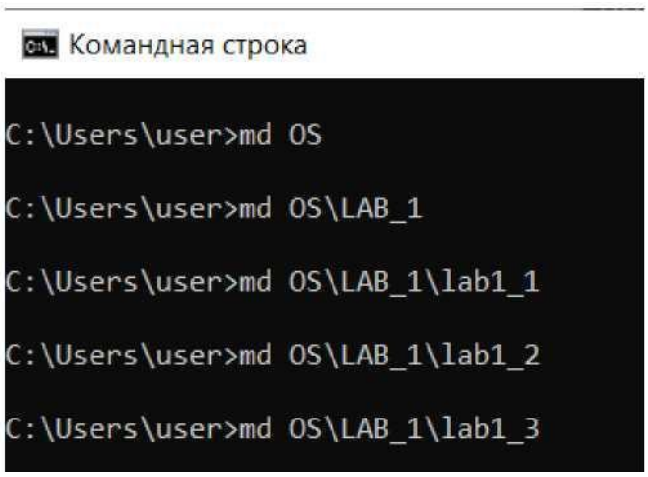


Рисунок 1.7 – Створення папок на диску командою md

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

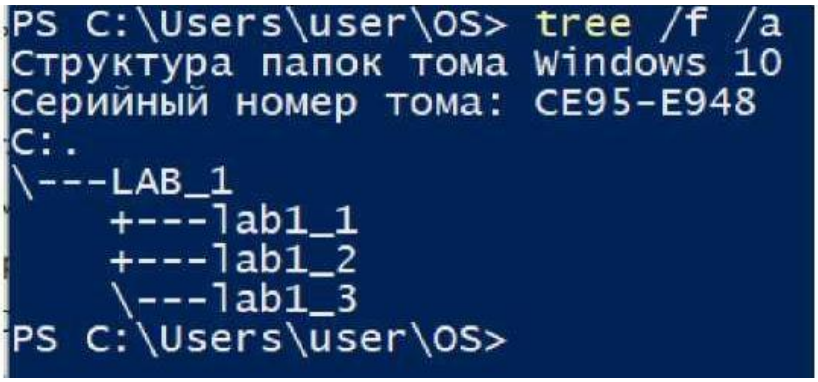


Рисунок 1.8 – Дерево каталогів папки OS після виконання команд md

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

Для створення текстових файлів використовується команда copy con [ім’я файлу]. 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 [6, с. 155].

Створимо файл lab1.txt та внесемо в нього прізвище, ім’я та номер групи так, які показані на рисунку 1.9.

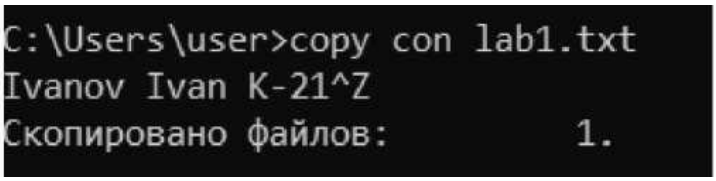


Рисунок 1.9 – Створення текстового файлу

Після виконання зазначеної команди в папці C:\Users\user\ буде створено файл lab1.txt, який містить задану послідовність символів (рис. 1.10).

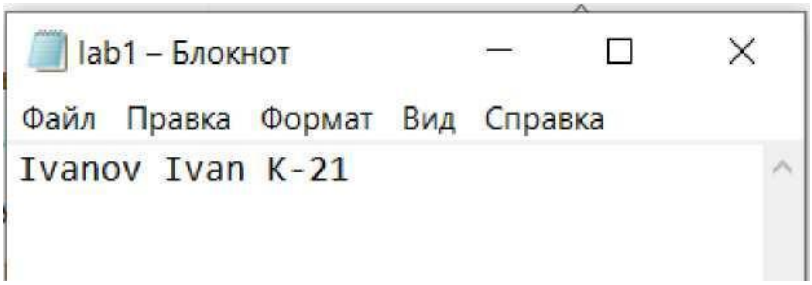


Рисунок 1.10 – Вміст текстового файлу lab1.txt

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

copy [і’мя_файлу_який_копіюється] [і’мя_файлу_куди_копіюється]. Аналогічно створюємо всі файли, які вказані в дереві каталогу на рисунку 1.6.

Для перейменування файлів використовується команда ren [і’мя_файлу] [нове і’мя_файлу]. 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).

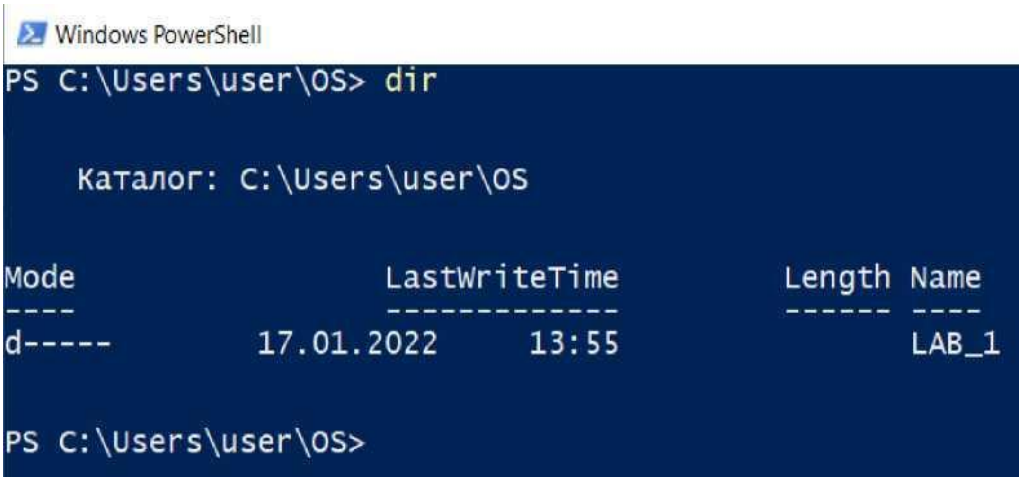


Рисунок 1.11 – Результат виконання команди dir

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

Для видалення каталогу використовується команда rd [ім'я файлу], для видалення файлу – del [ім'я файлу].

Для переходу в інший каталог використовується команда – "cd .."

Для відображення дерева каталогів використовується команда – tree.

Завдання для самостійного виконання

– В поточному каталозі засобами командного рядка створити дерево каталогів відповідно до номера варіанта (табл. 1.1).

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

– Скопіювати створений файл в text.txt у відповідну варіанту директорію.

– У випадковому каталозі створити файл lab1.txt та записати в нього поточну дату та дату свого народження.

– Вивести в консоль дерево каталогів командою tree.

– Оформити звіт з лабораторної роботи. У звіті для кожного завдання навести скриншот консолі з командами та відповіді на контрольні запитання.

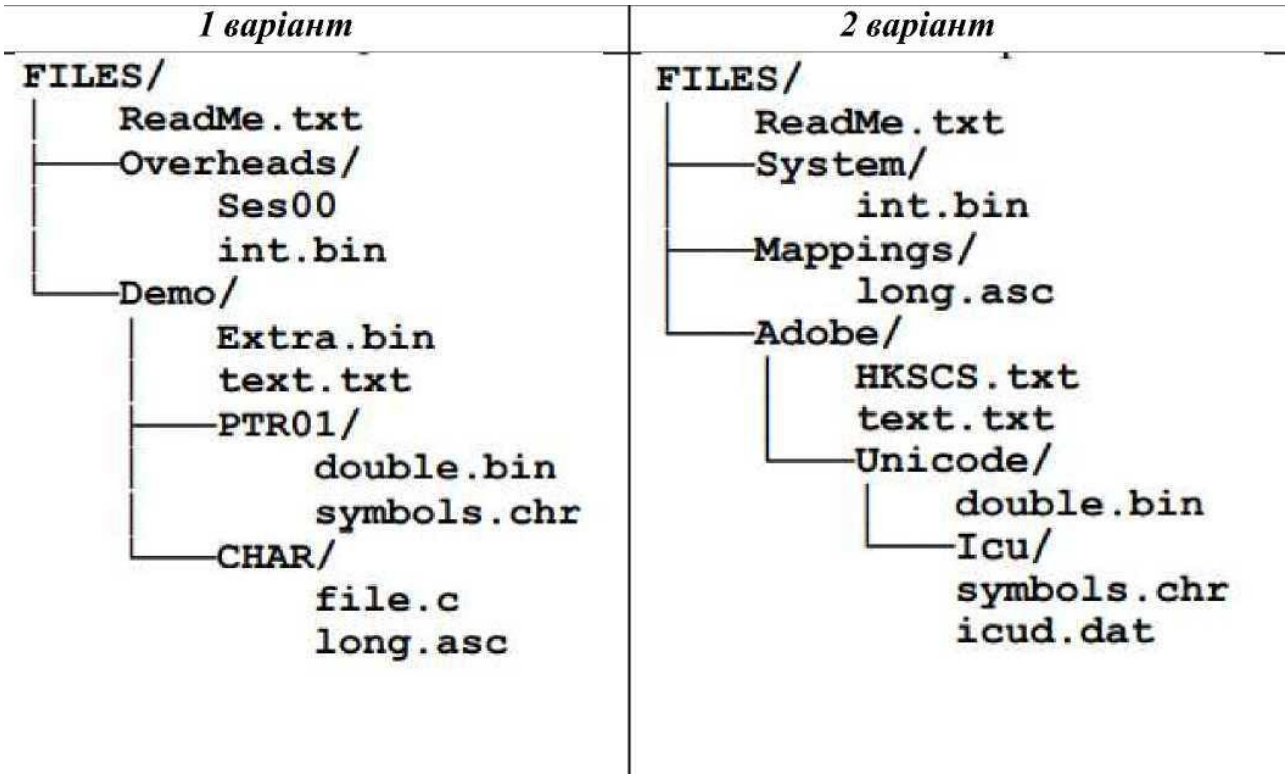
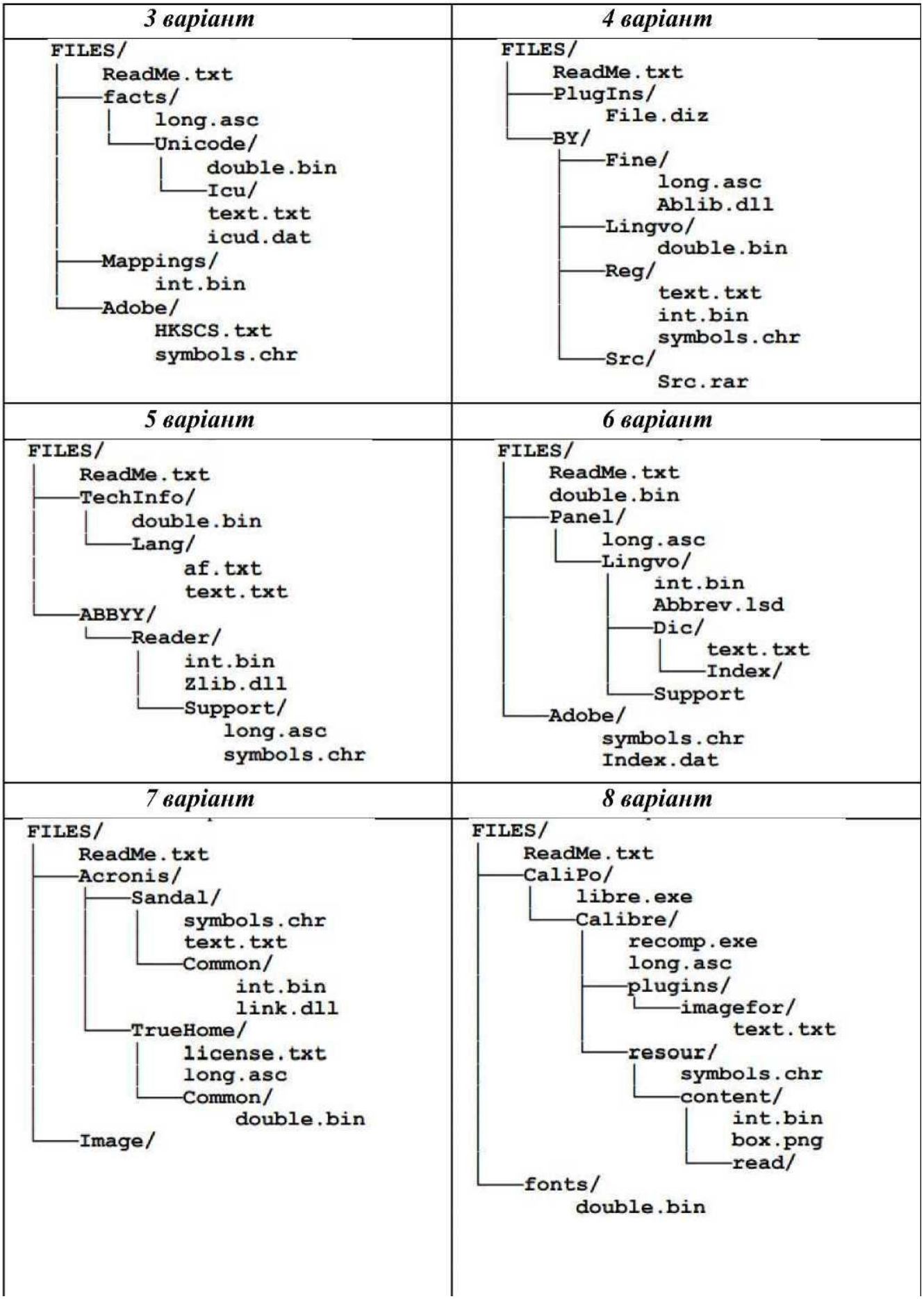
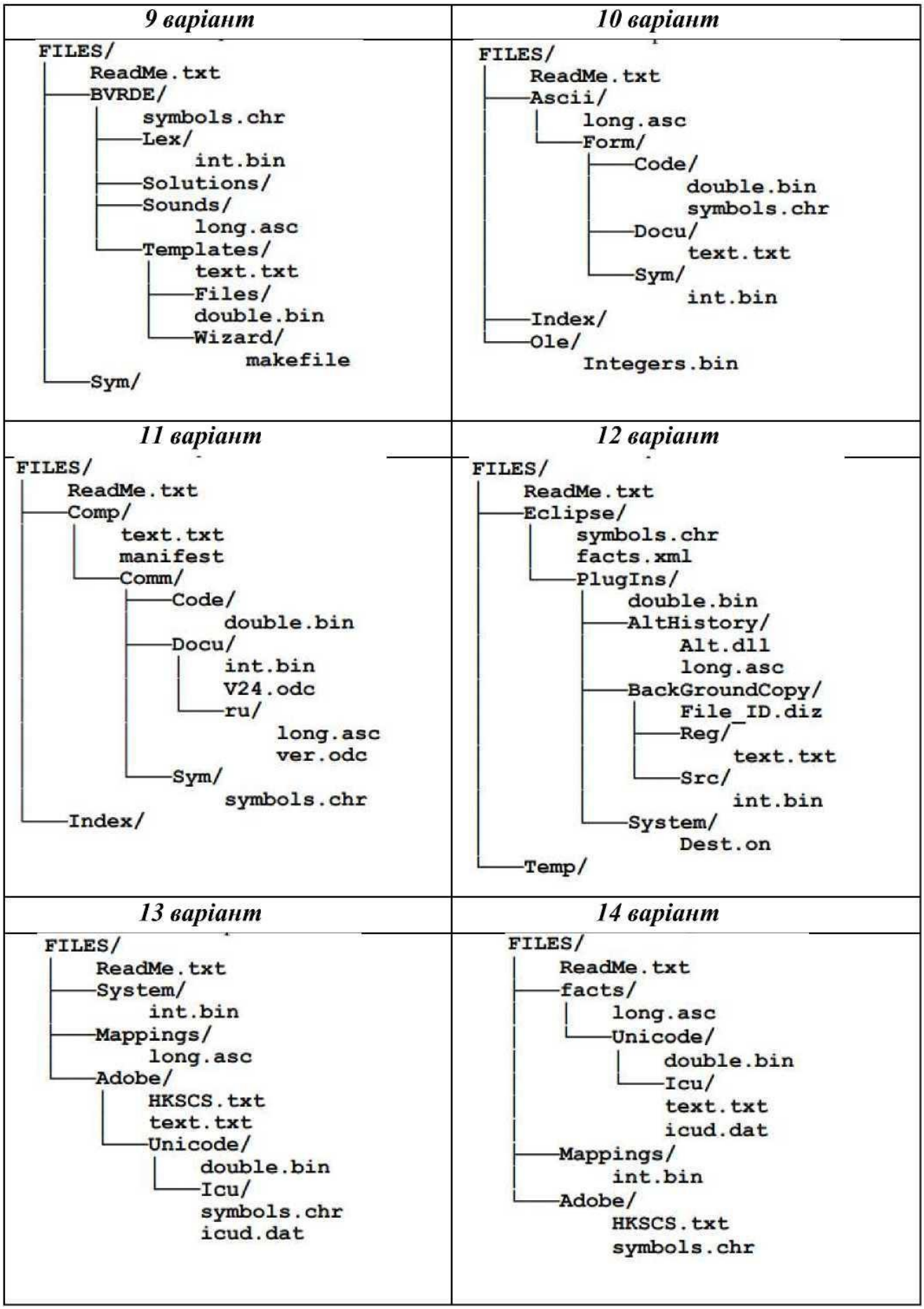


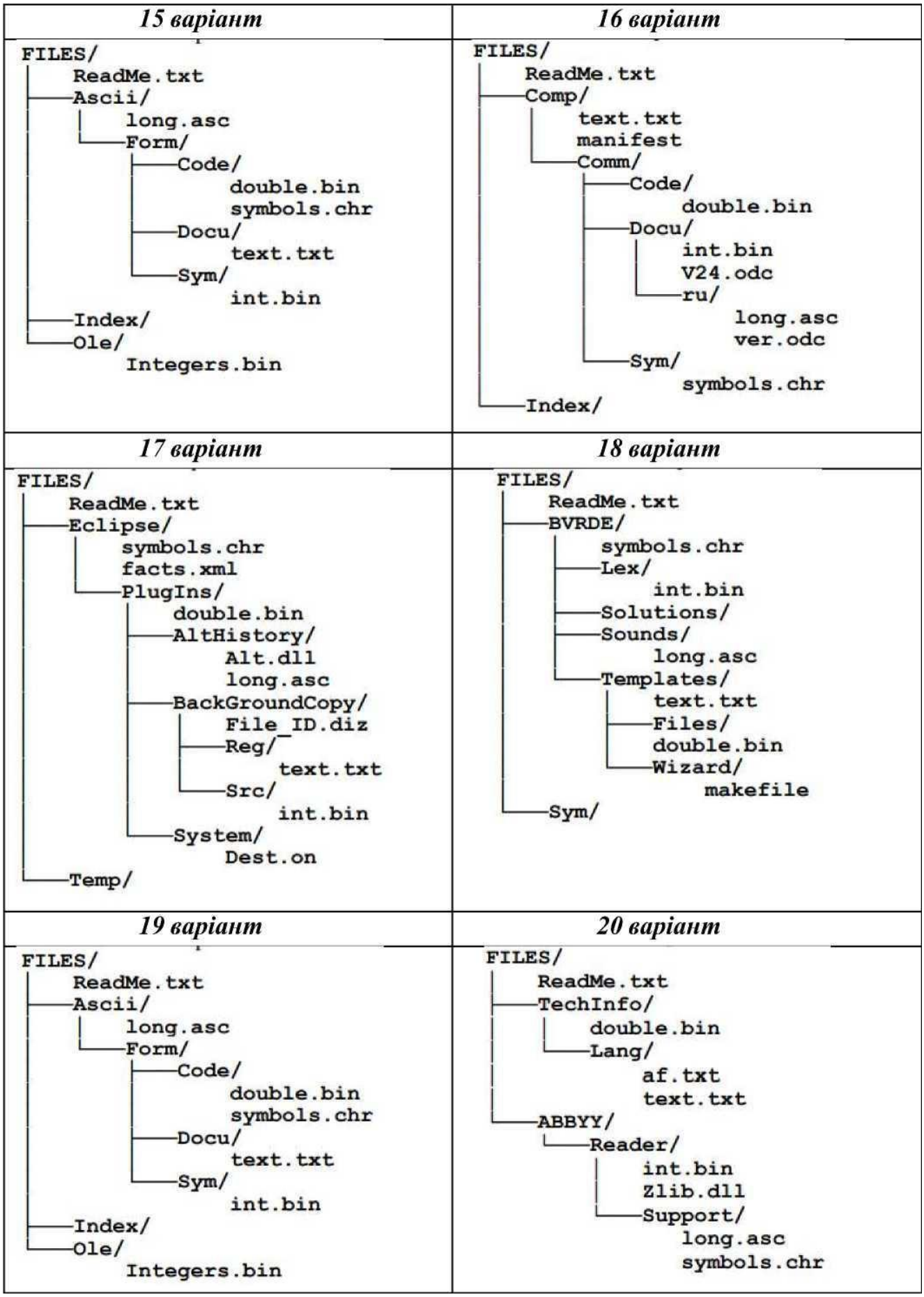
Рисунок 1.12 – Варіанти індивідуальних робіт



Продовження рисунку 1.12 – Варіанти індивідуальних робіт



Продовження рисунку 1.12 – Варіанти індивідуальних робіт



Продовження рисунку 1.12 – Варіанти індивідуальних робіт

Контрольні питання

1. Навіщо потрібен командний рядок Windows? Як її 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 термінала з графічним інтерфейсом. Термінал функціонує за схожими принципами як і командний рядок Windows. Вся представлена в лабораторній роботі інформація стосується дистрибутиву 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.

WSL підтримує різні дистрибутиви Linux, такі як Ubuntu, Debian та Fedora, що дозволяє користувачам обирати серед знайомих їм середовищ. Завдяки WSL, розробники можуть працювати у звичному середовищі Linux, використовуючи інструменти та бібліотеки, які можуть бути недоступні у Windows. Це також полегшує розробку кросплатформних застосунків.

Починаючи з WSL 2, підсистема отримала значні поліпшення, включаючи покращену продуктивність та повну реалізацію ядра Linux. Це дозволяє запускати навіть складні програми та сервіси, які потребують високої продуктивності [4, с. 52].

Після встановлення дистрибутиву 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 дистрибутиву Linux можна командою wsl (рис. 2.2.).

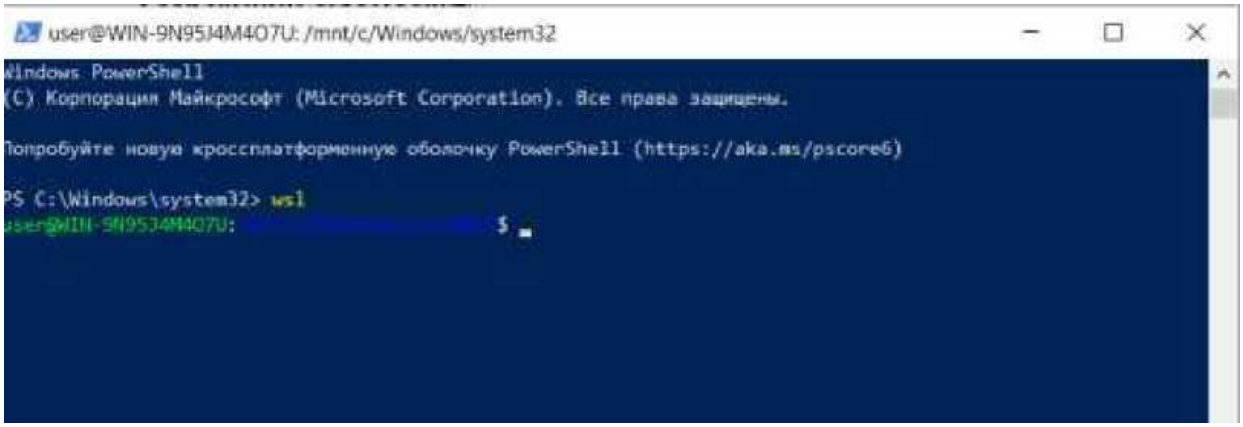


Рисунок 2.1 – Термінал Ubuntu з графічним інтерфейсом

Рисунок 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 файлу;
- touch створити порожній файл [7, с. 33].

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 рисунку 2.3.

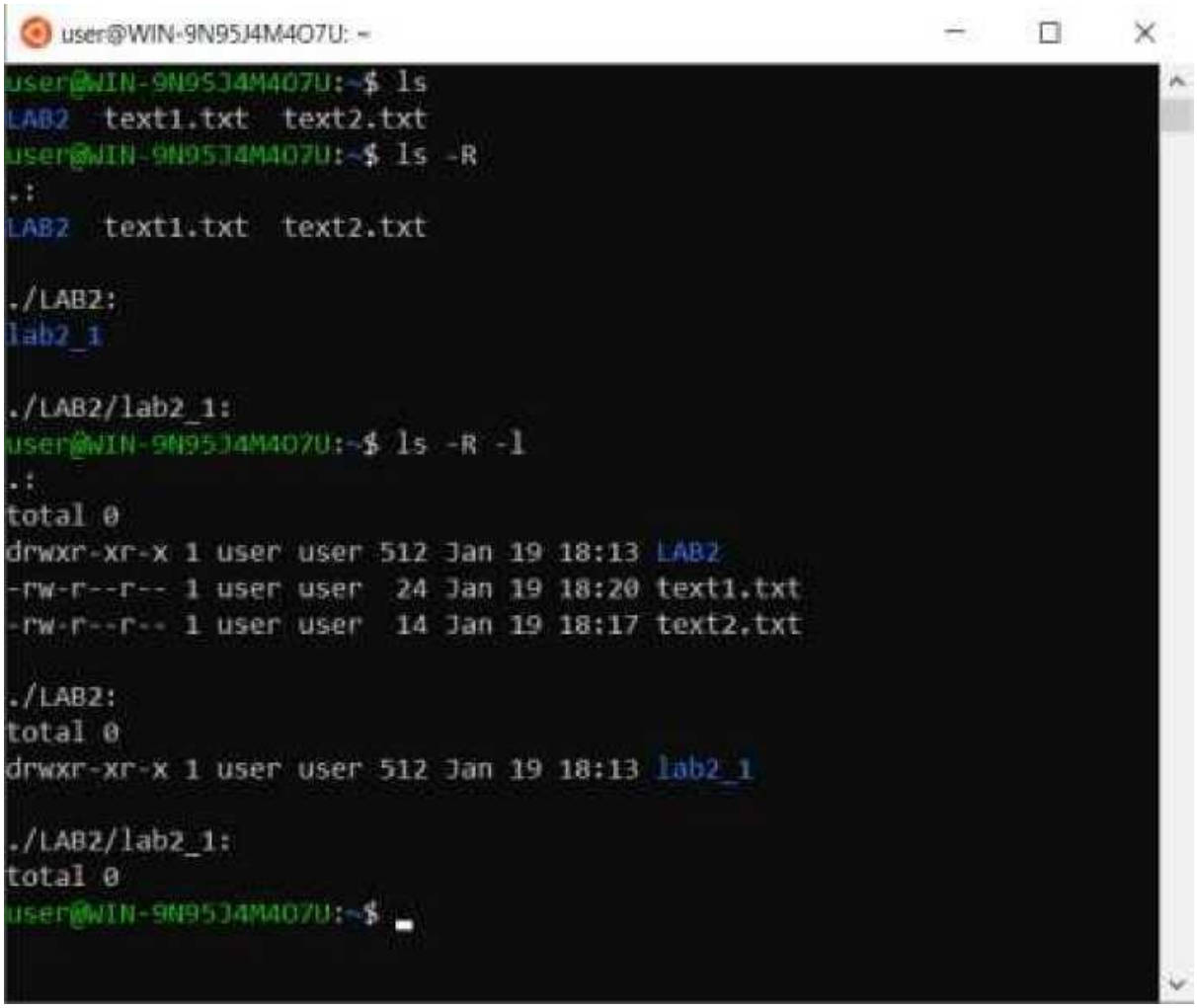


Рисунок 2.3 – Результати виконання команди ls

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 термінала 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 рисунку 2.4.

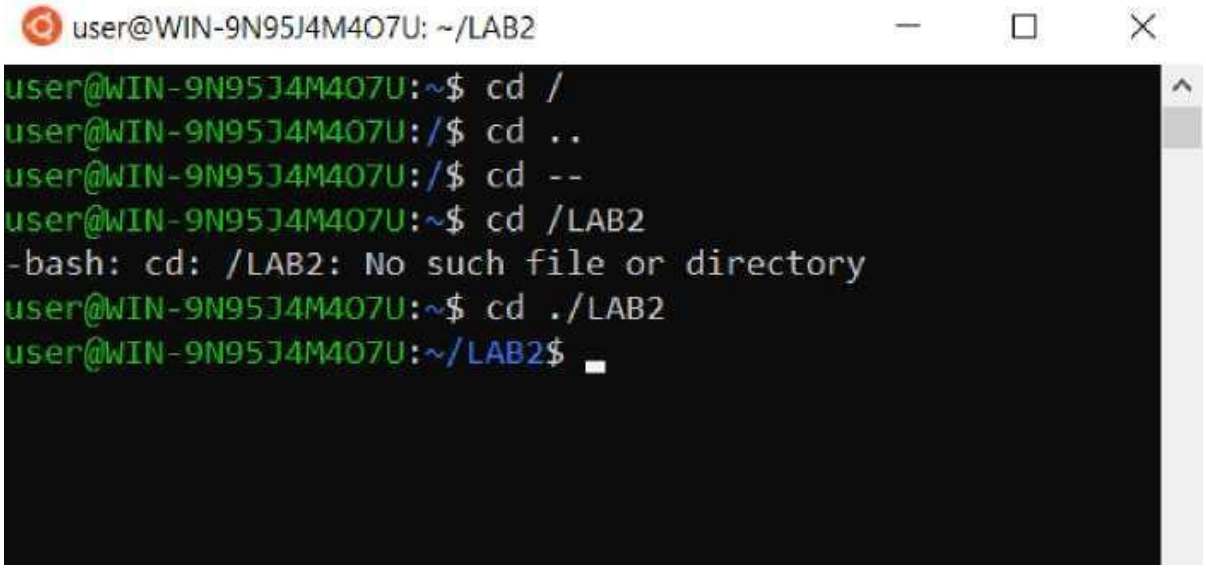


Рисунок 2.4 – Результати виконання команди cd

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 дистрибутиву [8, с.73].

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 рисунку 2.5 і 2.6.

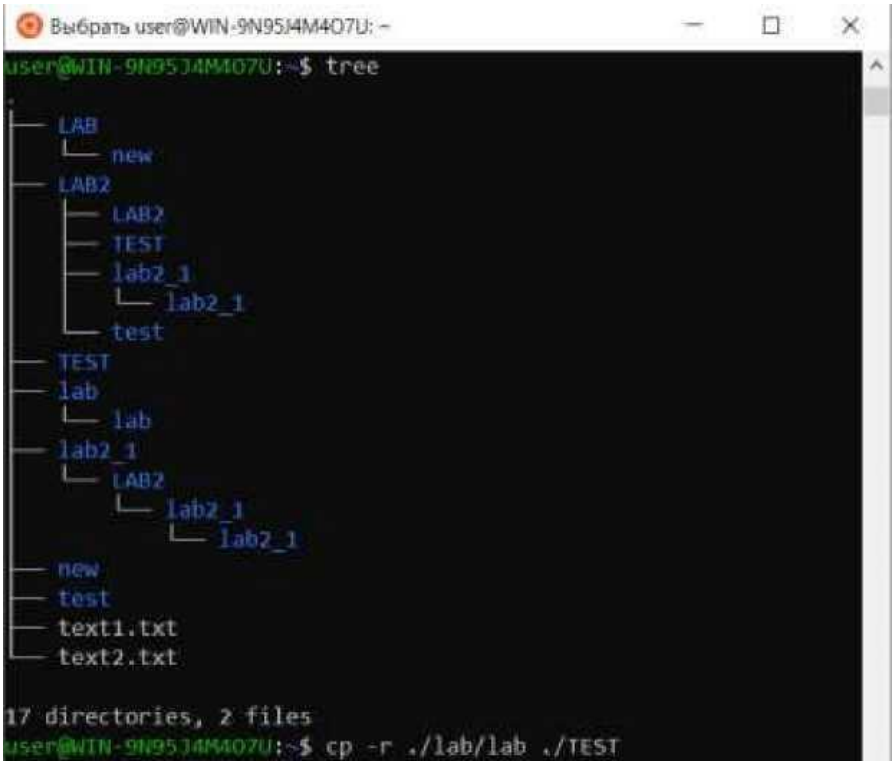


Рисунок 2.5 – Результати виконання команди tree до копіювання

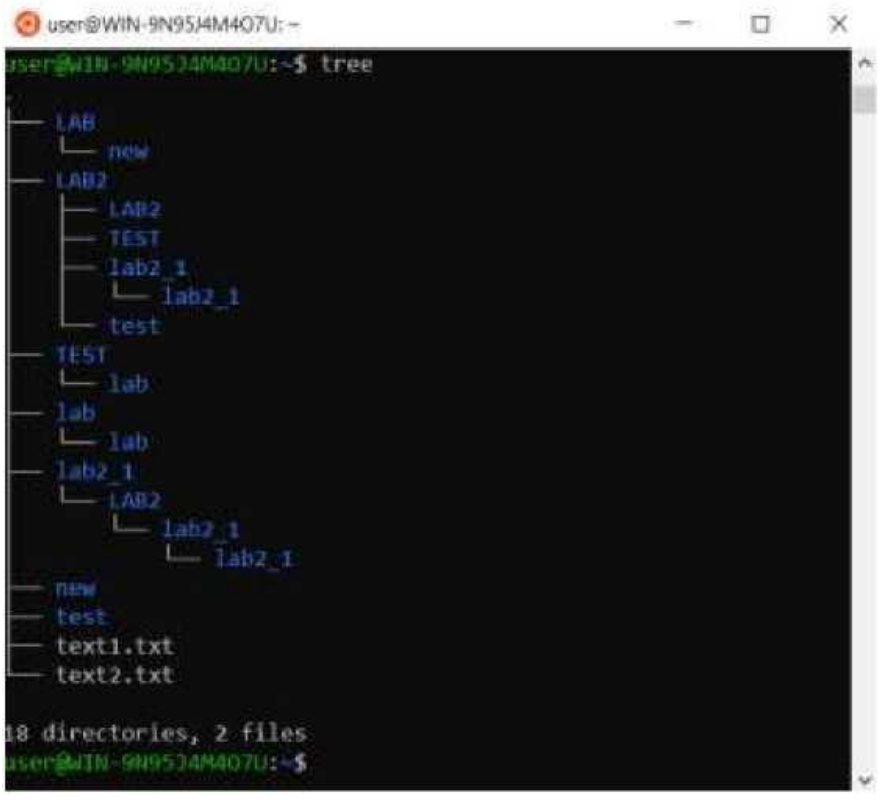


Рисунок 2.6 – Результати виконання команди tree після копіювання

Створення файлів та запис даних у 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 файл newfile2.txt в папці new. Результати виконання обох команд відображені в дереві каталогів [9, с.66] (рис. 2 7).

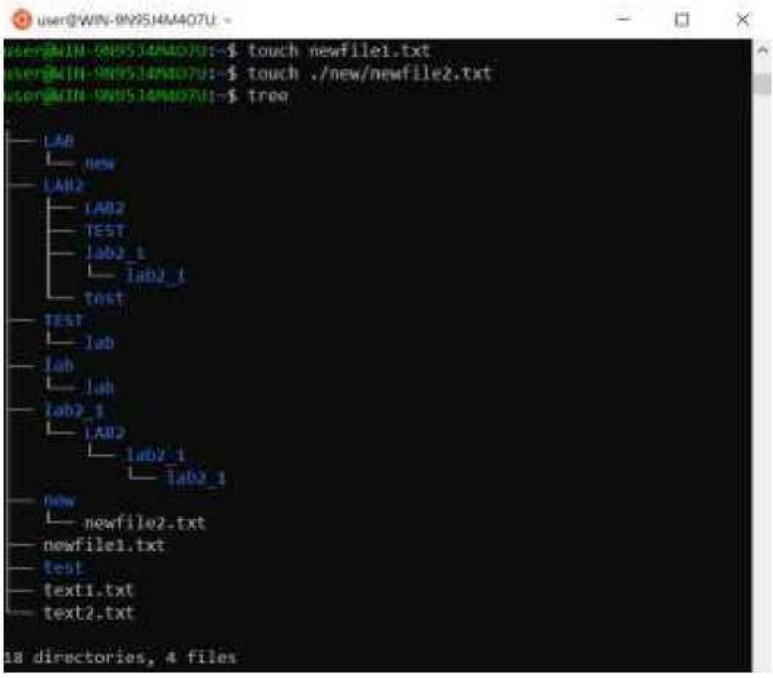


Рисунок 2.7 – Створення файлів командою touch

Для запису тексту в файл використовується команда echo «text»>[ім'я_файлу], розглянемо її на наприкладі, команда echo «Ivanov Ivan K-21»>newfile1.txt вносить текст в створений раніше файл. Відображення вмісту файлу можливо за допомогою команди cat. На рисунку 2.8 представлені результати запису тексту в створений файл.

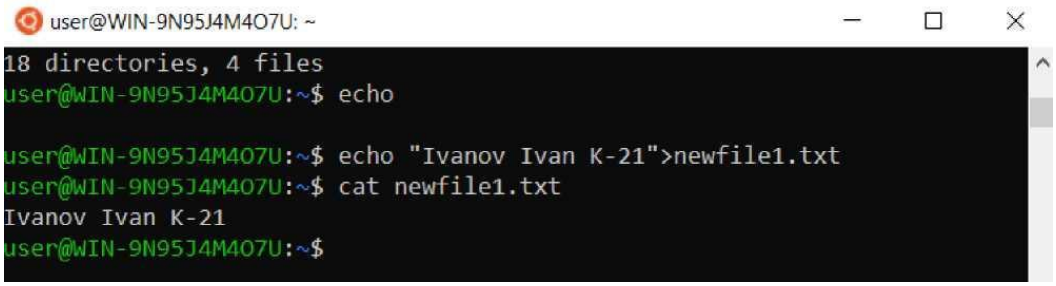


Рисунок 2.8 – Створення файлів командою touch

Завдання для самостійного виконання:

1. В поточному каталозі засобами командного рядка 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 скриншот термінала з командами та відповіді на контрольні запитання.

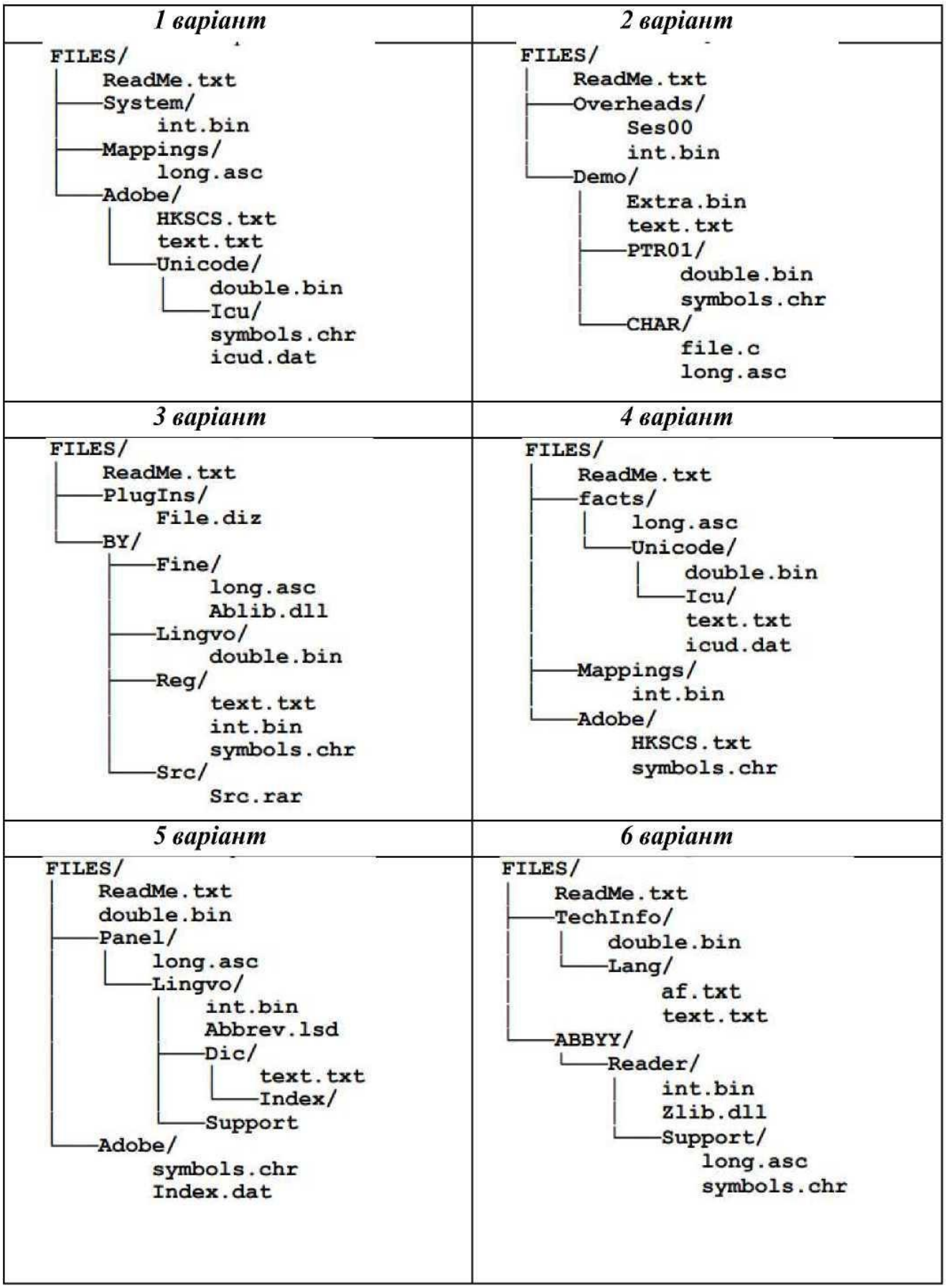
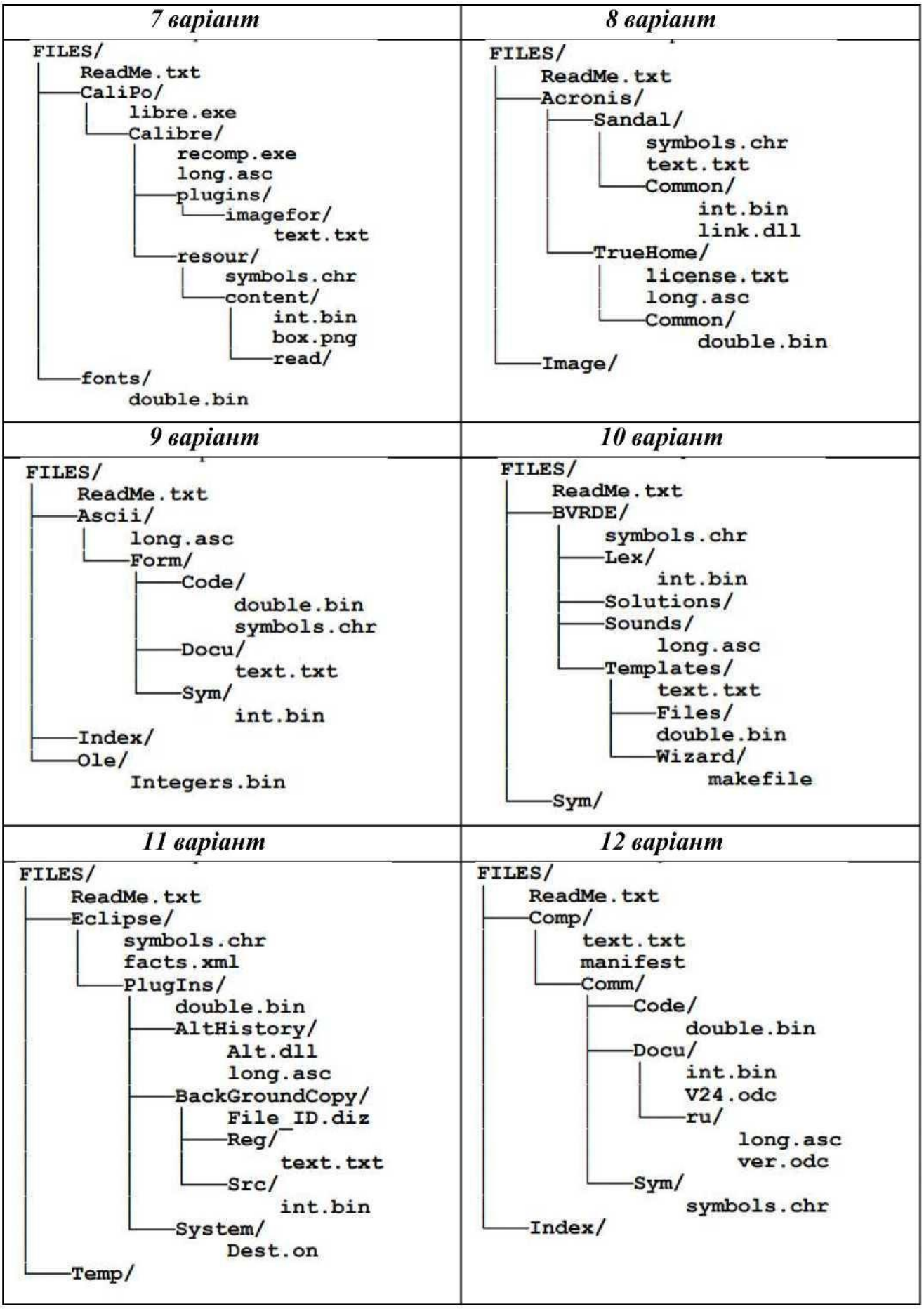
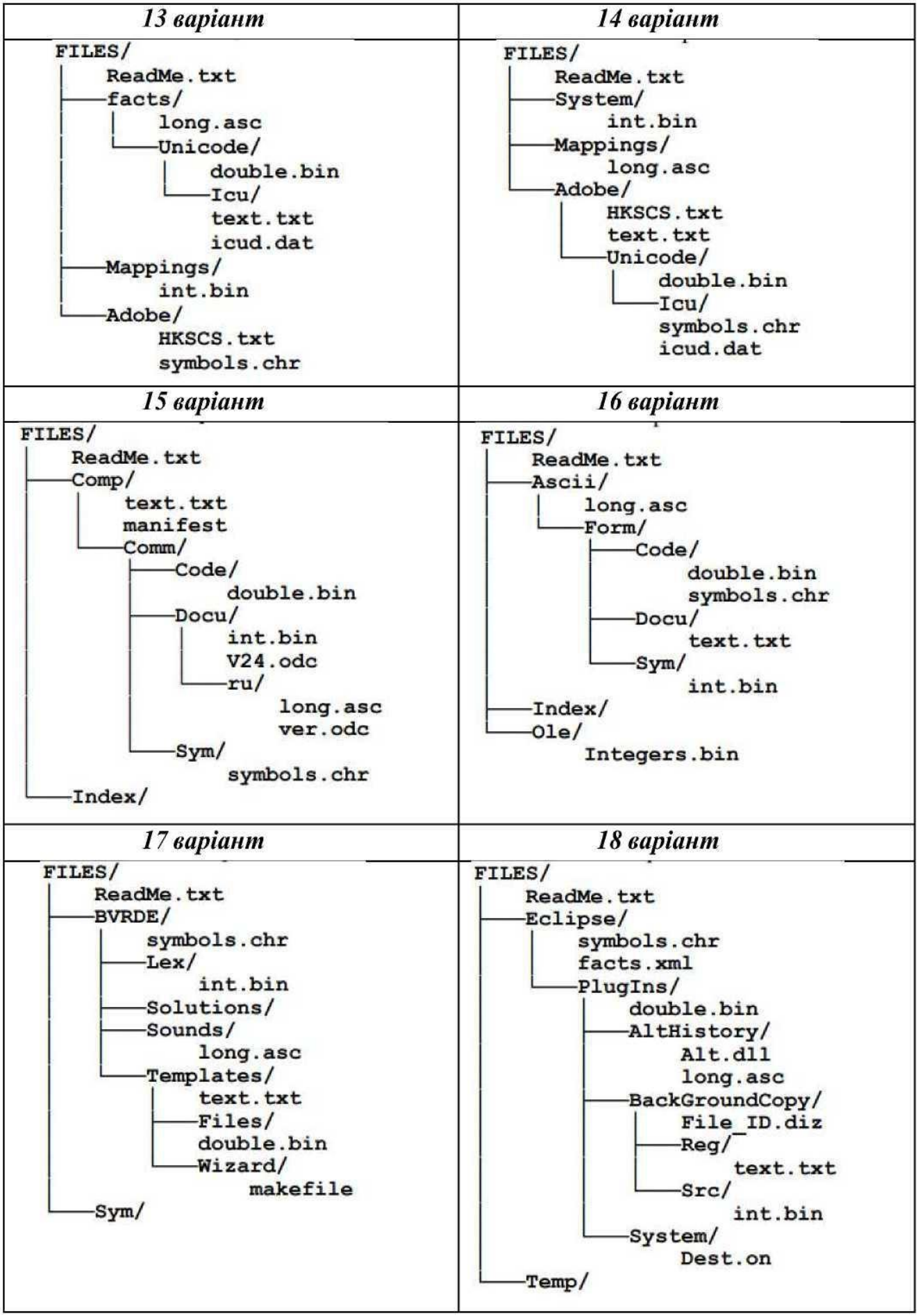


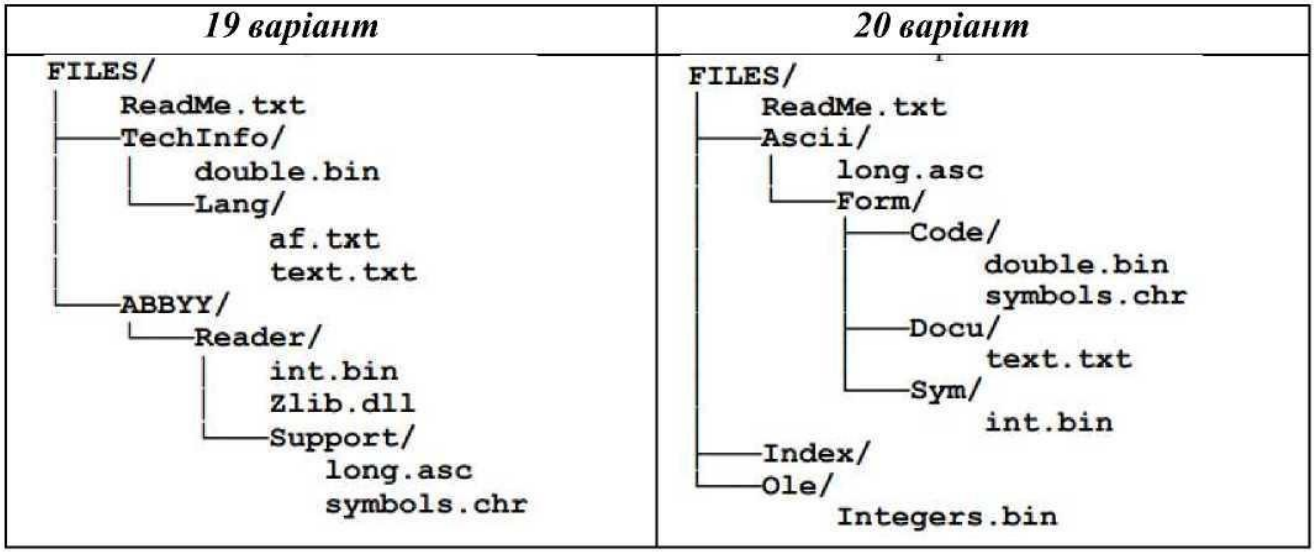
Рисунок 2.9 - Варіанти індивідуальних завдань



Продовження рисунку 2.9 - Варіанти індивідуальних завдань



Продовження рисунку 2.9 - Варіанти індивідуальних завдань



Продовження рисунку 2.9 - Варіанти індивідуальних завдань

Контрольні питання:

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

Лабораторна робота №3
Створення та запуск bash-скриптів

Мета роботи: ознайомлення і вивчення елементарних понять та способів 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 скрипту 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).

Рисунок 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) [10, с. 452].

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

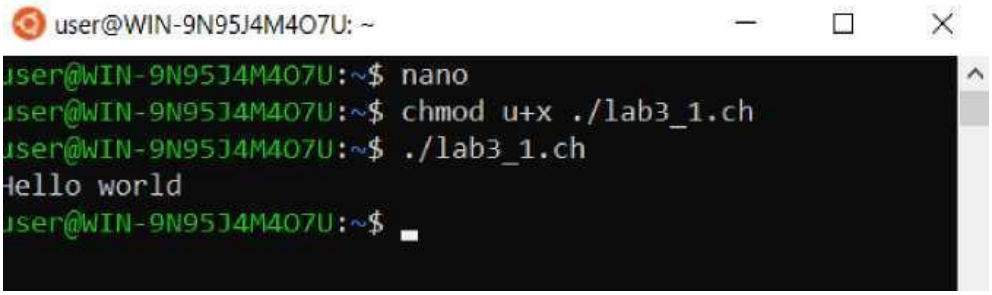


Рисунок 3.2 – Результати виконання скрипту lab3_1.ch

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.ch після виконання на екрані з’явиться поточна дата в стандартному форматі (рис. 3.3).

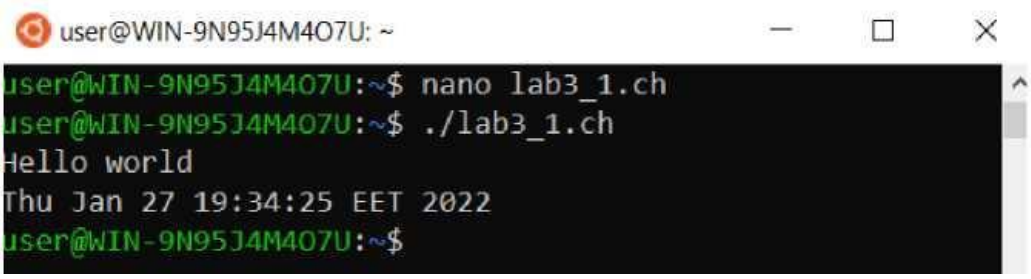


Рисунок 3.3 – Результати виконання скрипту lab3_1.ch

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).

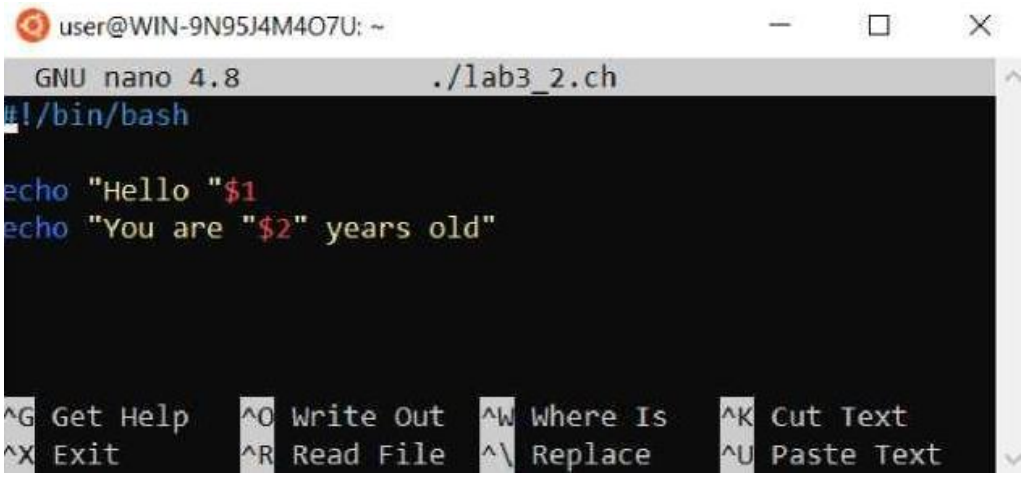


Рисунок 3.4 – Вміст файлу lab3_2.ch

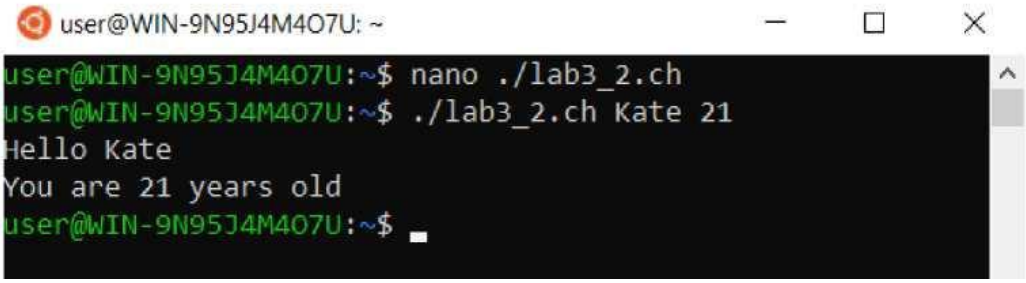


Рисунок 3.5 – Результати виконання скрипту lab3_2.ch

В попередньому прикладі розглядалася ситуація, коли параметри передаються командою запуску скрипту. 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_виконання then

команда

else
команда
fi

Для організації циклічного виконання частини коду використовують команди циклу for (цикл за параметром), синтаксис наступний:

for змінна in список do
команда
done

Команда for послідовно присвоюємо змінній значення зі списку та виконує команди, які розташовані між do і done [11, с. 95].

Існує команда циклу, яка працює з передумовою, це while (цикл з передумовою), синтаксис наступний:

while умова_виконання_циклу do
команда
done

Цикл while виконує команди, які розташовані між do і done, поки умова вірна, повертається значення 1.

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).

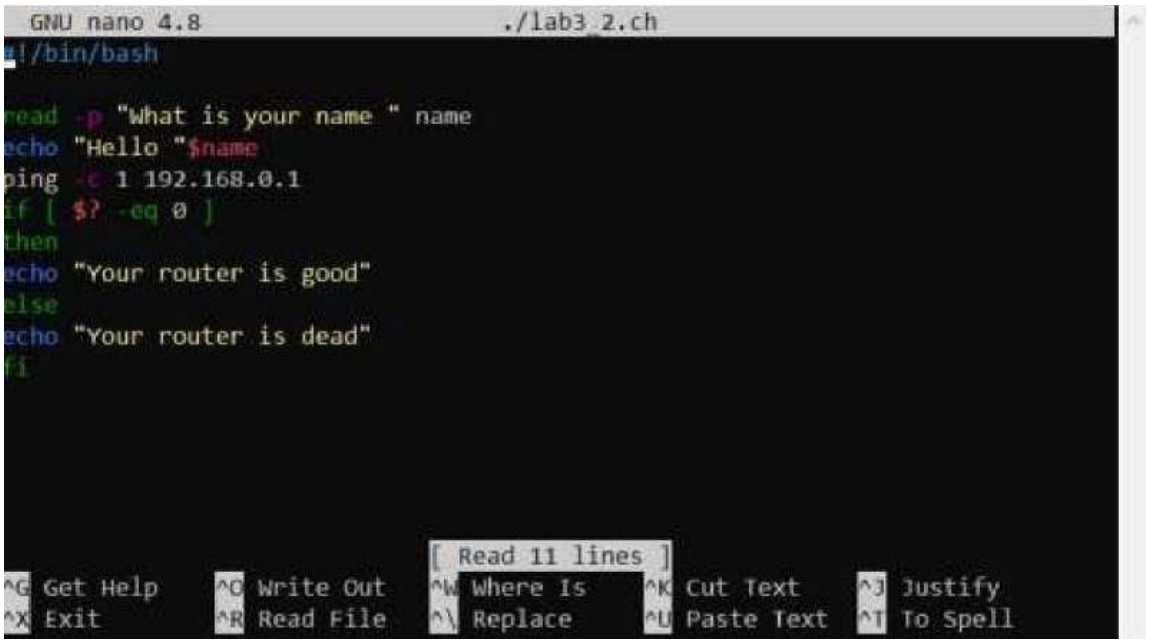


Рисунок 3.6 – bash-скрипт перевірки підключення до роутера

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

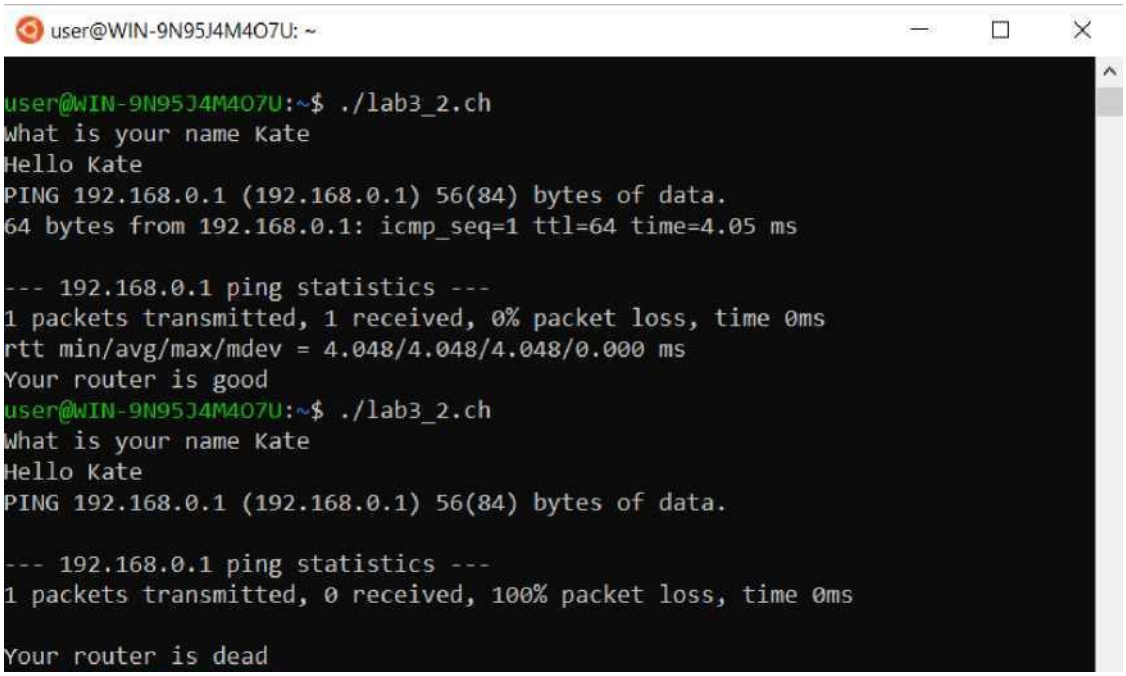


Рисунок 3.7 – Результати роботи bash-скрип та перевірки підключення до роутера

Для запуску bash-скриптів у Windows слід використовувати оболонку PowerShell та системну команду wsl (перехід до командного рядка Linux). Наприклад, для запуску скрипту lab3_2.ch запускаємо PowerShell від імені адміністратора, далі wsl переходить в командний рядок Linux. Для переходу в потрібну директорію user використовується команда cd. Далі запускаємо скрипт lab3_2.ch, результати роботи якого наведені на рисунку 3.8.

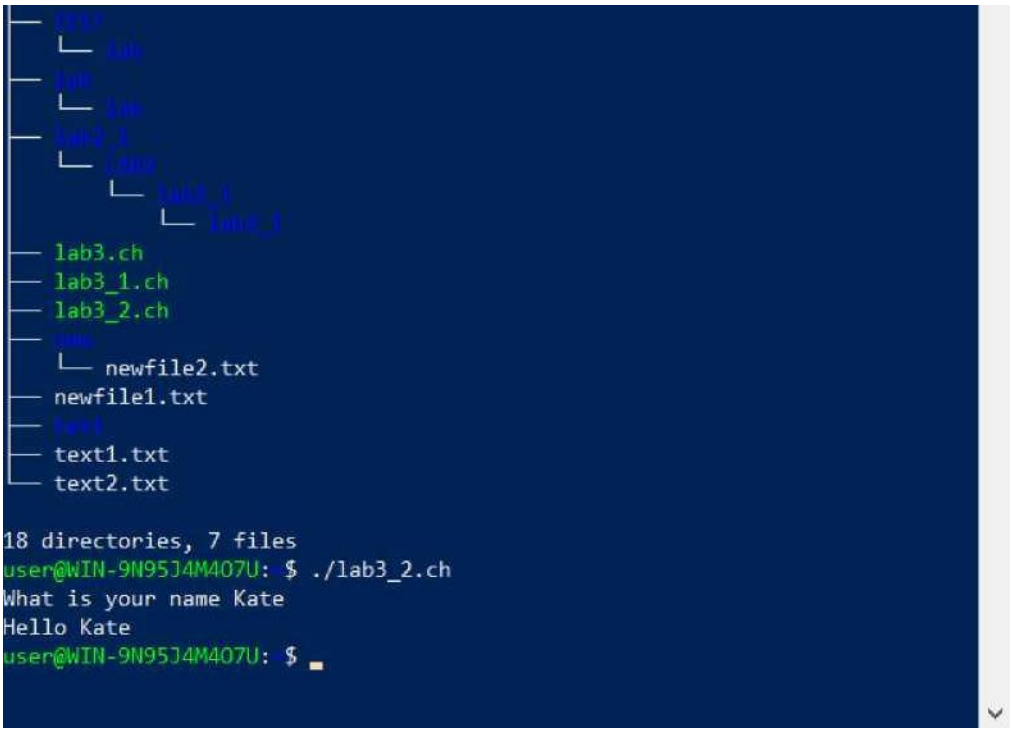


Рисунок 3.8 – Результати виконання скрипту lab3_2.ch в PowerShell

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 з варіантом (табл. 3.1).

3. Для одного зі скриптів показати результати роботи в Windows (використовувати PowerShell).

4. Оформити звіт з лабораторної роботи. У звіті для кожного завдання навести скриншот редактора файлів (nano), скриншот з результатами роботи скрипту та відповіді на контрольні запитання.

Таблиця 3.1 – Варіанти індивідуальних завдань

№	**Завдання 1**	**Завдання 2**
1	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
2	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 на екран більший аргумент
3	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Ввести в командному рядку два числових аргументи, вивести повідомлення, якщо користувач вводить більше або менше аргументів
4	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Ввести в командному рядку два числових аргументи, вивести на екран повідомлення, якщо введені однакові аргументи
5	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»	Ввести в командному рядку строку символів, створити директорію з таким ім’ям, у разі успішного створення вивести на екран каталог у вигляді дерева
6	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
7	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 тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST

Продовження таблиці 3.1 – Варіанти індивідуальних завдань

№	Завдання 1	Завдання 2
8	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»
9	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 тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST
10	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»
11	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
12	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 на екран більший аргумент
13	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Ввести в командному рядку два числових аргументи, вивести повідомлення, якщо користувач вводить більше або менше аргументів
14	Командою ping перевірити доступ до заданого вузла, вивести на екран результати перевірки	Ввести в командному рядку два числових аргументи, вивести на екран повідомлення, якщо введені однакові аргументи
15	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»	Ввести в командному рядку строку символів, створити директорію з таким ім’ям, у разі успішного створення вивести на екран каталог у вигляді дерева
16	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

Продовження таблиці 3.1 – Варіанти індивідуальних завдань

№	Завдання 1	Завдання 2
17	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 тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST
18	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»
19	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 тестову директорію (DIRTEST), створити три текстових файли, записати в них теми 1,2,3 лабораторних робіт, вивести на екран вміст директорії DIRTEST
20	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»

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 способів 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).

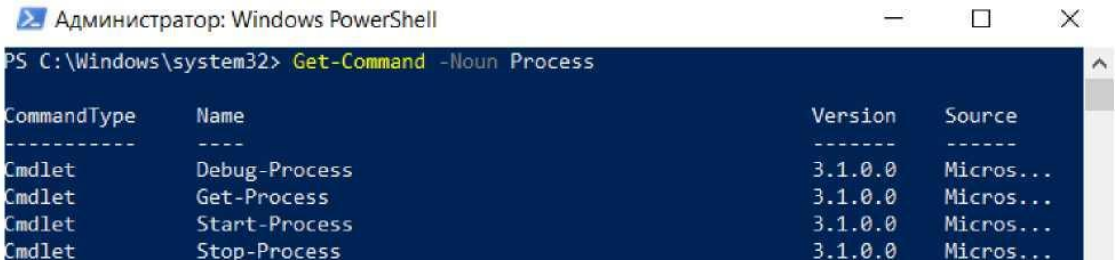


Рисунок 4.1 – Перелік командлетів для роботи з процесами Windows 10

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).

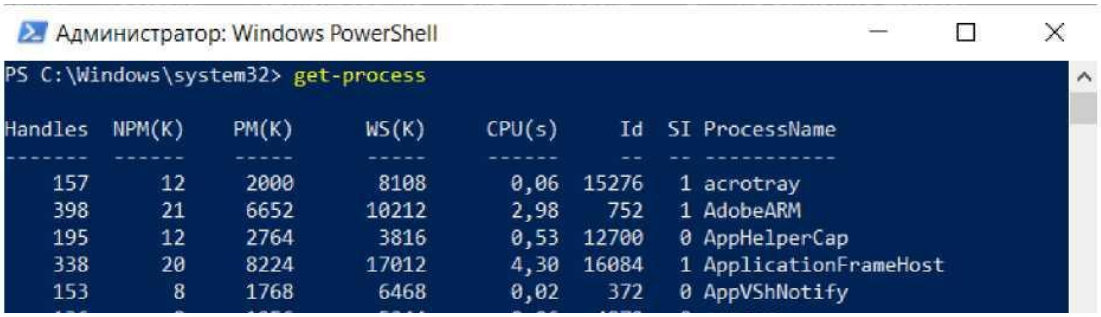


Рисунок 4.2 – Результати виконання командлета get-process в PowerShell

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):

– Get-Process -Name ‘notepad’ -FileVersionInfo – отримати властивості процесу notepad.

– Get-Process -FileVersionInfo -ErrorAction SilentlyContinue – отримати властивості всіх активних процесів та ігнорувати помилки, які пов’язані з відсутністю у деяких процесів властивості «FileVersion» [12, с. 140].

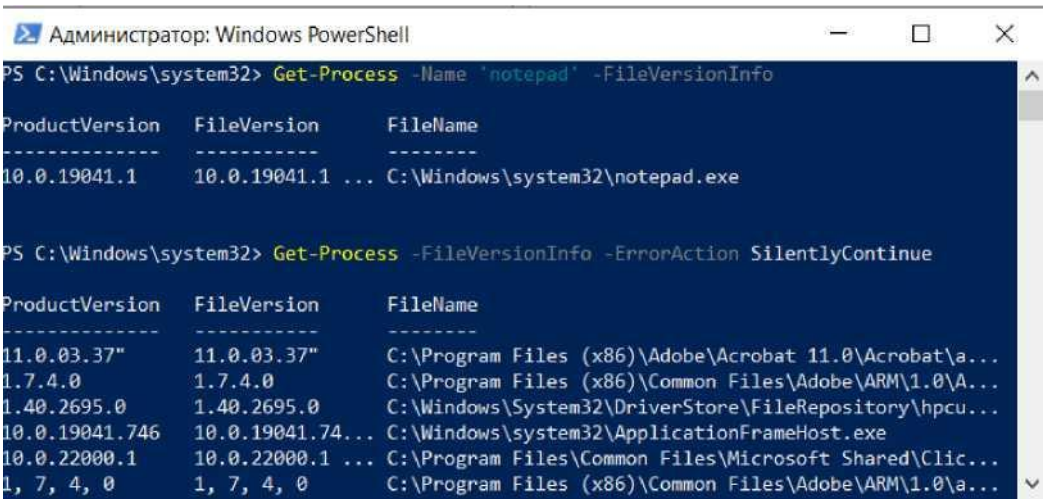


Рисунок 4.3 – Результати виконання командлета get-process з заданими аргументами

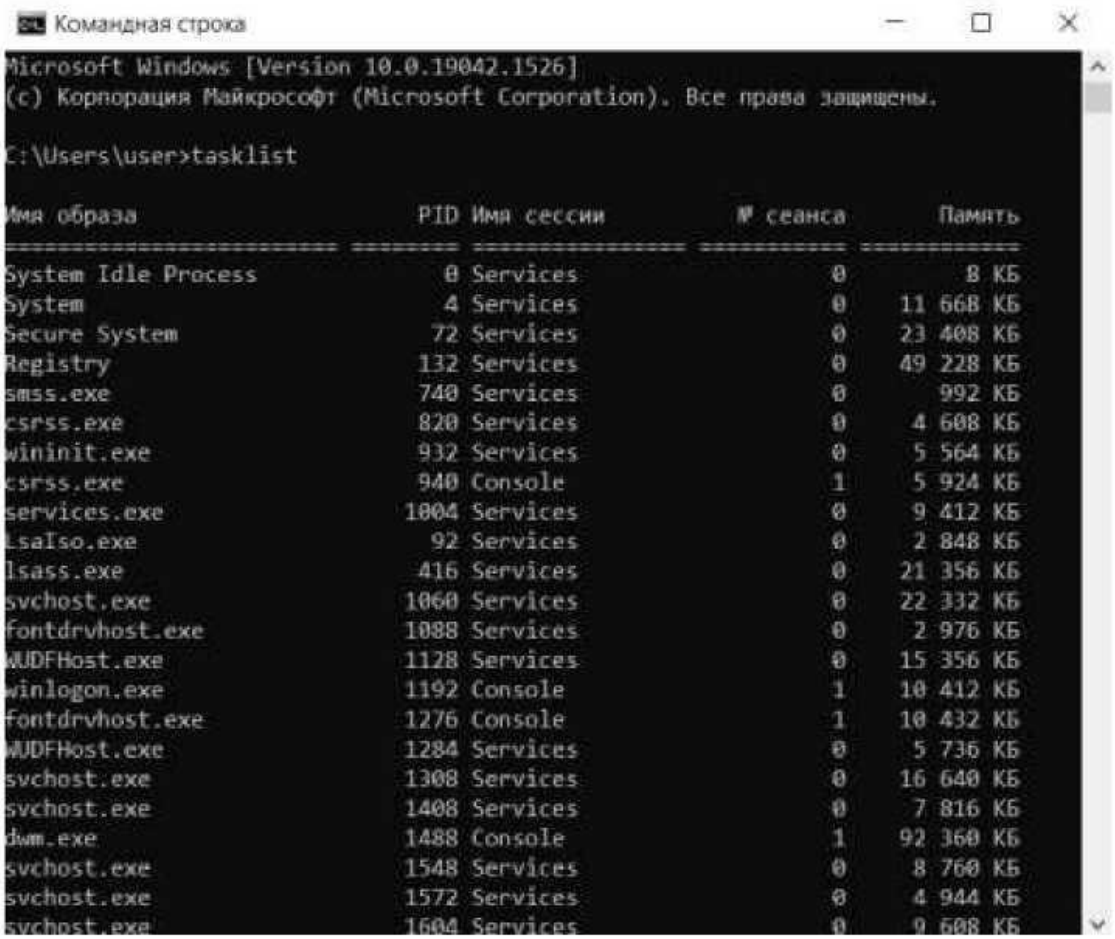


Рисунок 4.4 – Результати виконання команди tasklist в командному рядку Windows

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

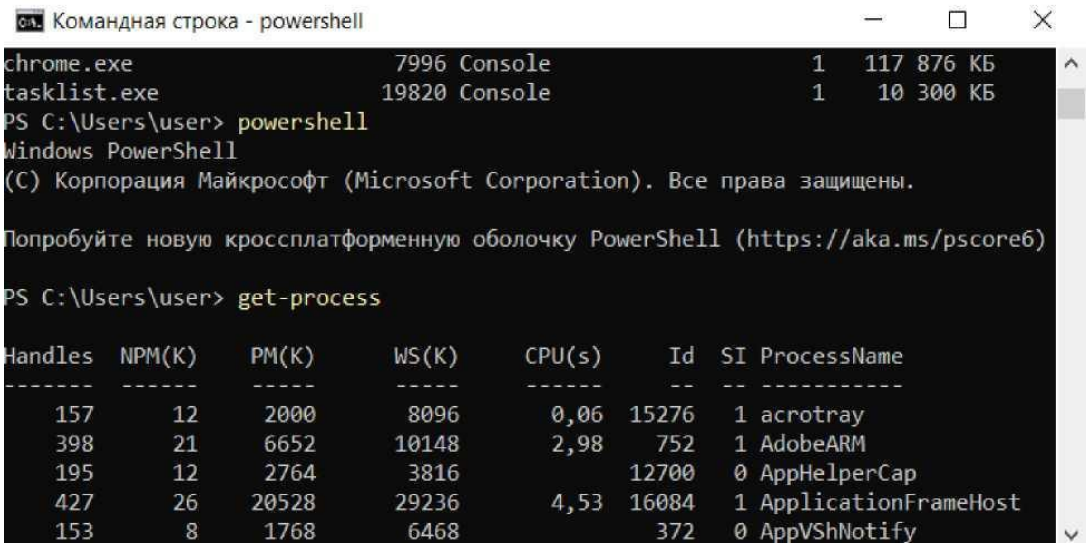


Рисунок 4.5 – Результати виконання команди get-process в cmd

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).

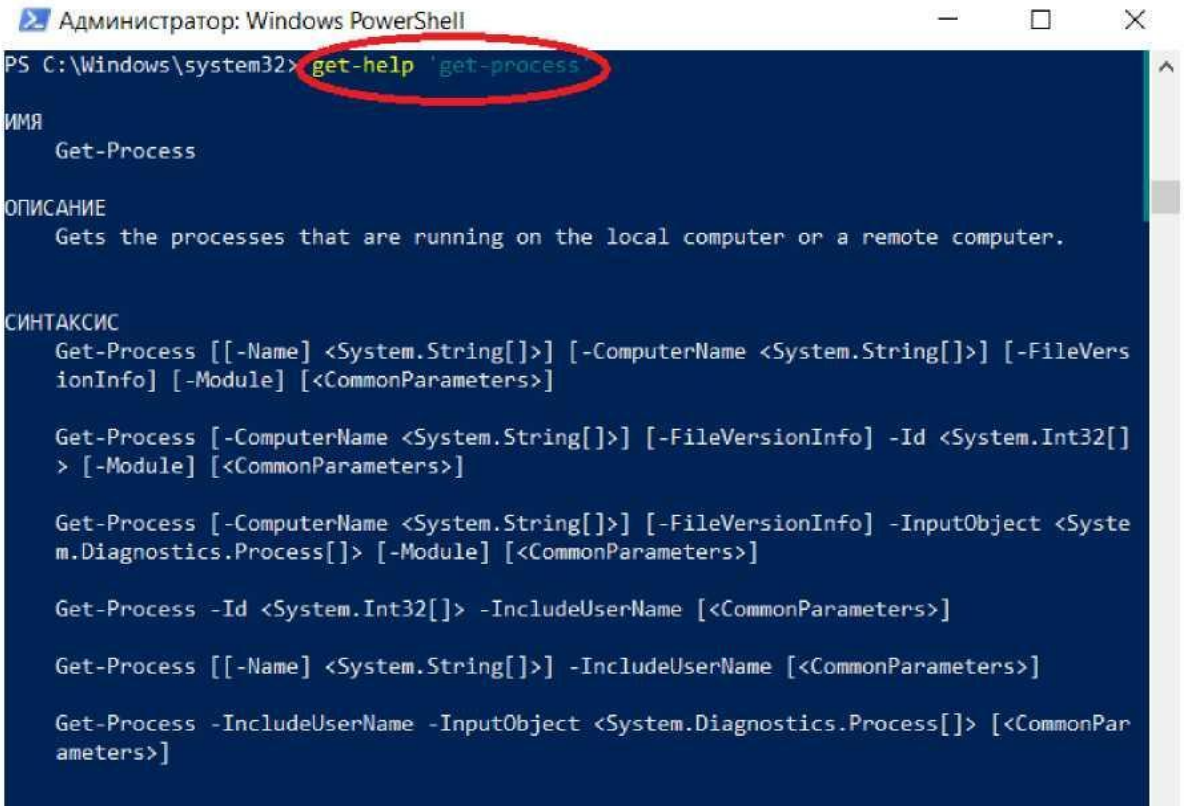


Рисунок 4.6 – Результати виконання командлета get-help

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 – ім’я (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 – місце де знаходиться файл (за замовчуванням – поточний каталог);

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

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.

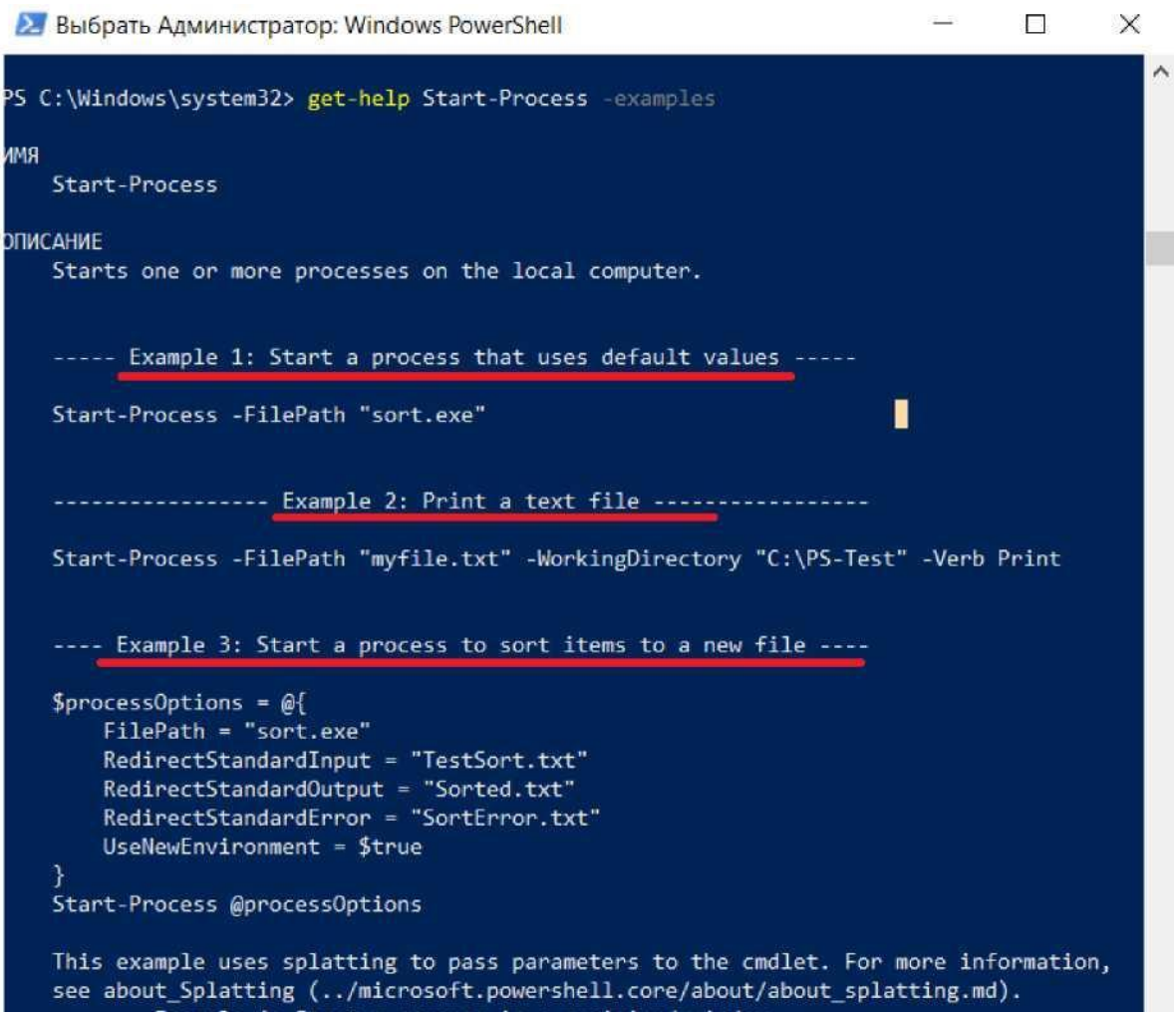


Рисунок 4.7 – Результати виконання командлета get-help з прикладами

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’єкта 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, -OutVariable [13].

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

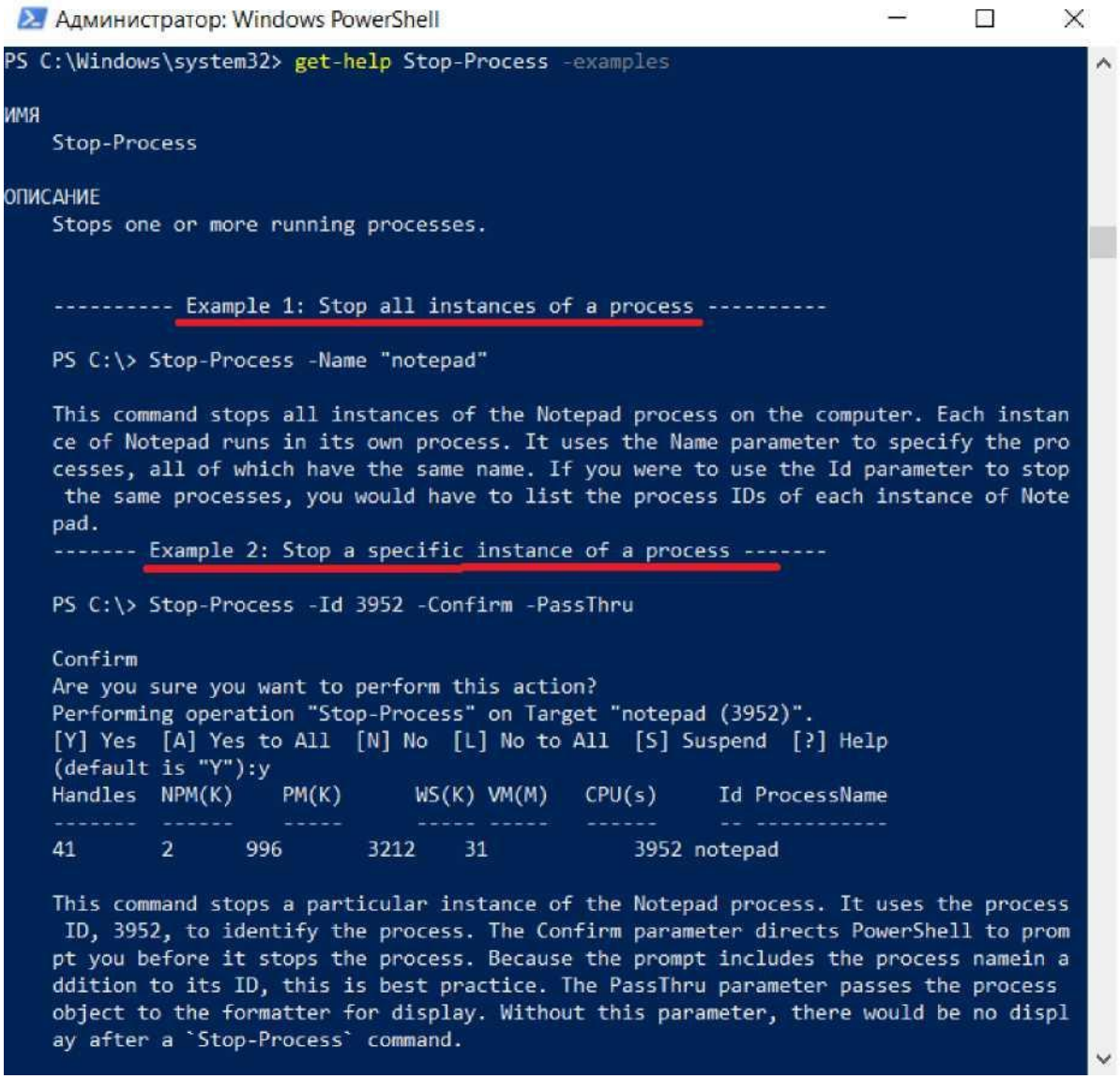


Рисунок 4.8 – Результати виконання командлета get-help з прикладами

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 аргумент “-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 термінала, 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 [14, с. 329].

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 рисунку 4.9.

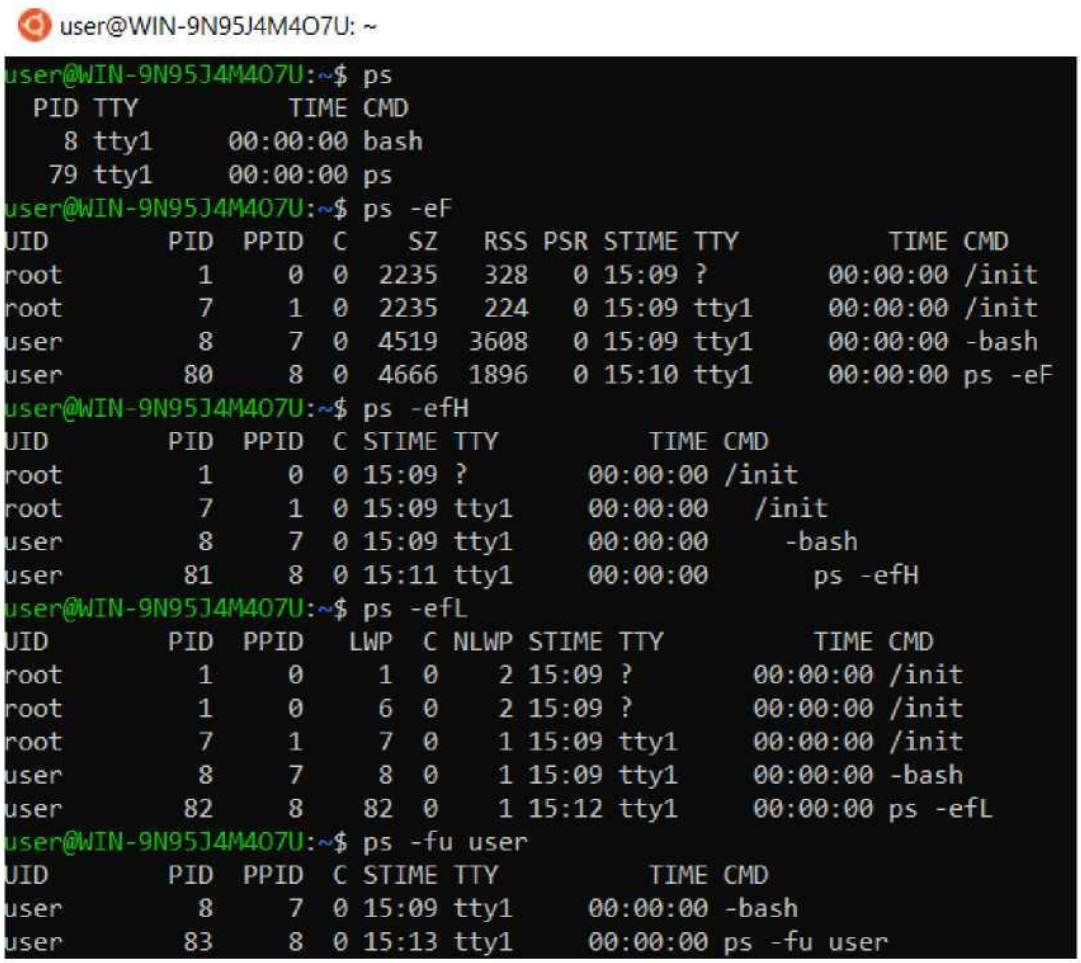


Рисунок 4.9 – Результати роботи команди ps (з різними аргументами)

Команда top відображає інформацію про запущені процеси в режимі реального часу (рис. 4.10).

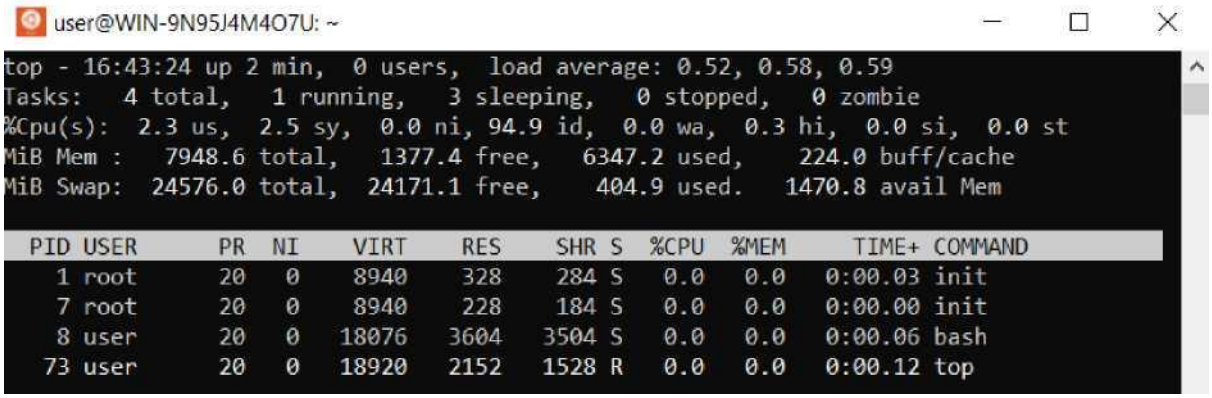


Рисунок 4.10 – Результати роботи команди top

При цьому система виведе для кожного процесу будуть виведені 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 цей 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 процес з більш розширеним функціоналом (рис. 4.11).

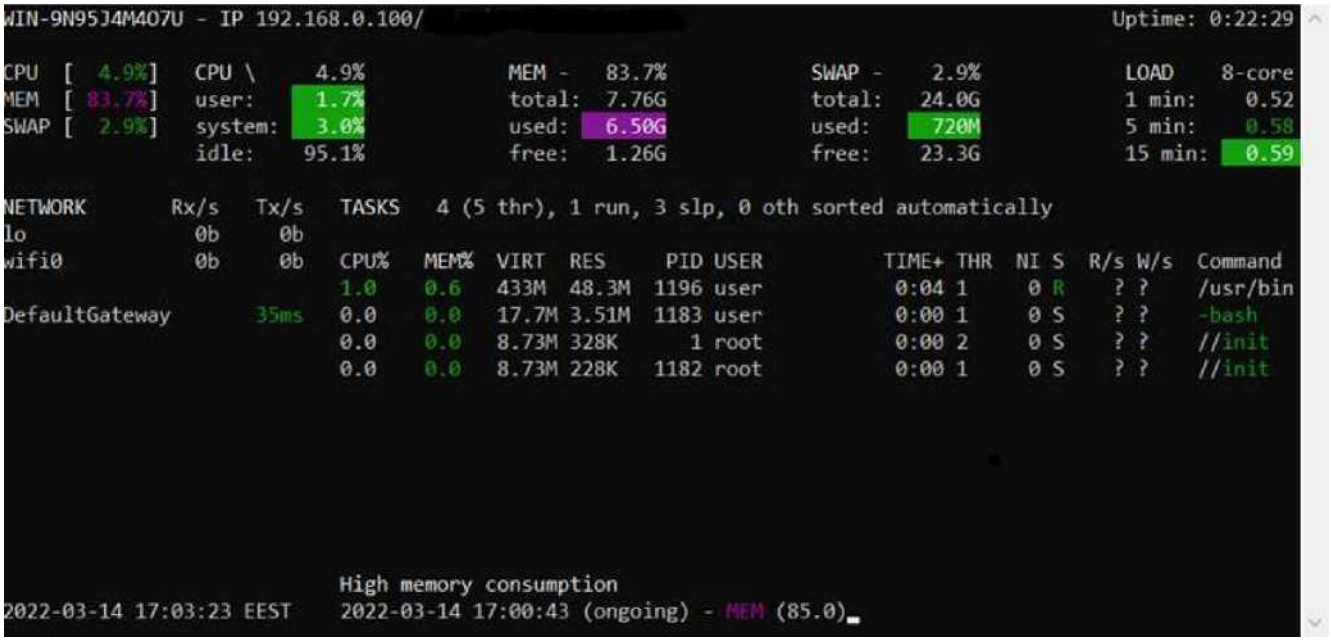


Рисунок 4.11 – Результати роботи команди glances

Для управління процесами в Linux використовують наступні команди:

- kill – посилає процесу сигнал завершення роботи;
- pkill – завершує процес (необхідно вказати ім’я процесу);
- pgrep – шукає процес по його імені (і, опціонально, за іменем користувача, що його запустив);
- killall – завершує всі активні процеси.

Результати роботи деяких з вище зазначених команд наведені на рисунку 4.12.

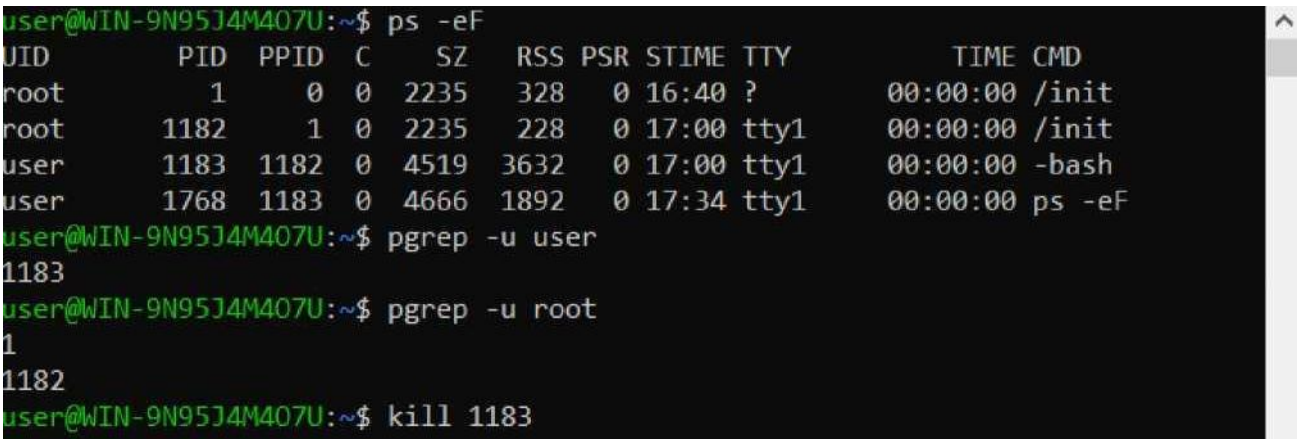


Рисунок 4.12 – Результати роботи команд управління процесами

Завдання для самостійного виконання:

1. В оболонці PowerShell відобразити поточні процеси та запустити «Блокнот» (вікно додатка повинно мати максимальний розмір). Запустити командний рядок 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 ніж 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 скриншот виконання команд, для завдання 3 і 4 – скриншот редактора файлів (nano), скриншот 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 способів 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, або натиснути правою кнопкою миші по панелі завдань Windows та натиснути диспетчер завдань. 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. Ви побачите зростання навантаження на процесор, можливо зміну частоти процесора, якщо не застосовано режим продуктивності, а також можете спостерігати збільшення використання оперативної пам’яті комп’ютера та зростання кількості передавання пакетів мережевою картою чи адаптером.

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 цей 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 [15, с. 223] (рис. 5.2).

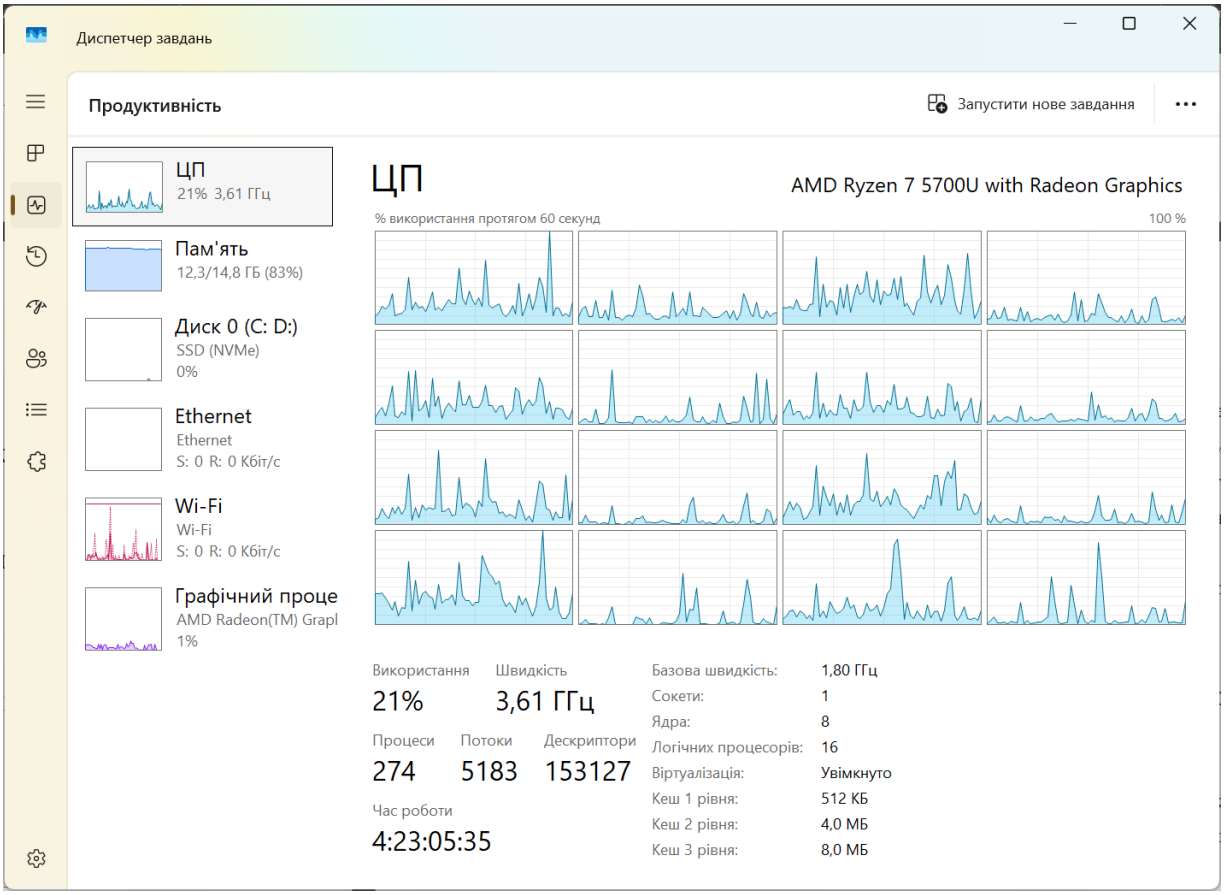


Рисунок 5.1 – Відображення інформації про пам’ять в Диспетчері завдань Windows

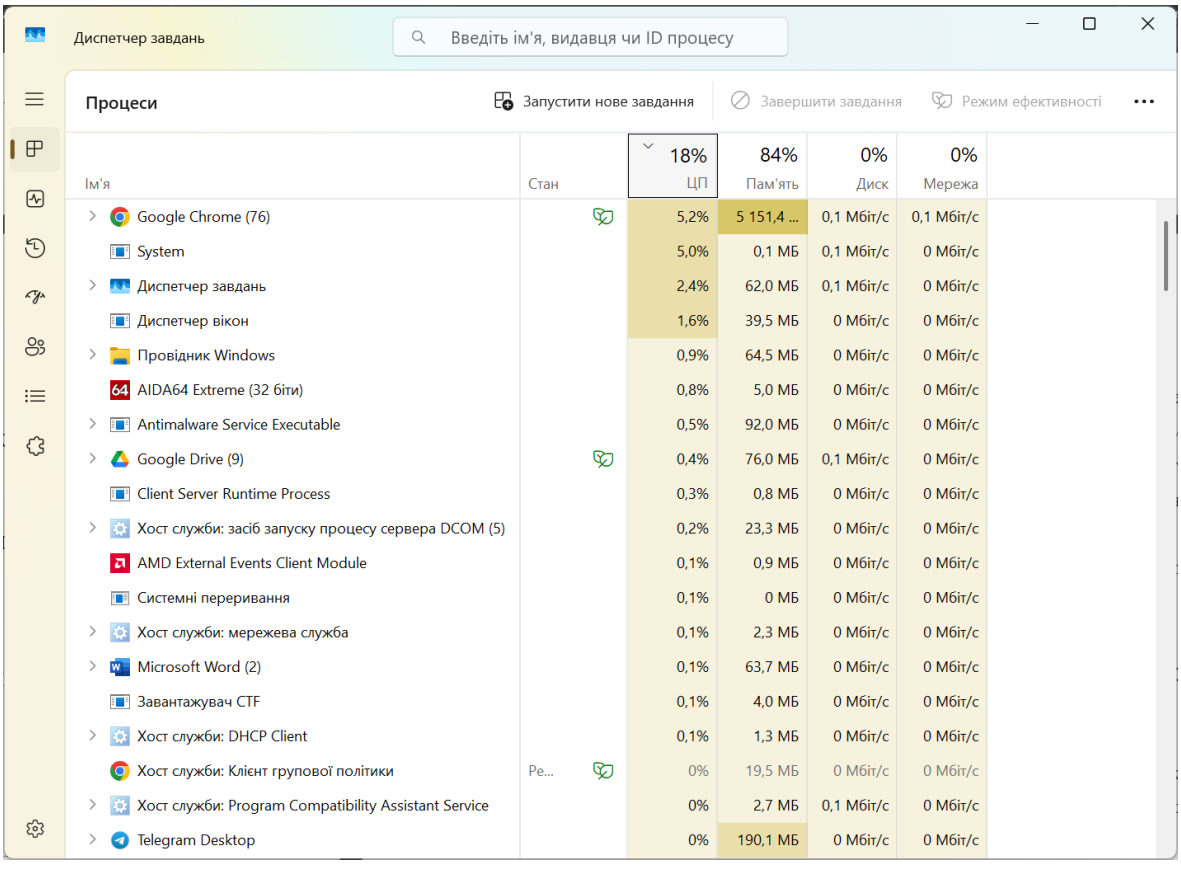


Рисунок 5.2 – Відображення інформації про використання пам’яті програмами

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:

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

– Вимоги до пам’яті відбивають поточний стан процесу. У списку відображаються обсяги пам’яті, що займають програми в цей момент часі, а не їх максимальні значення.

– 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).

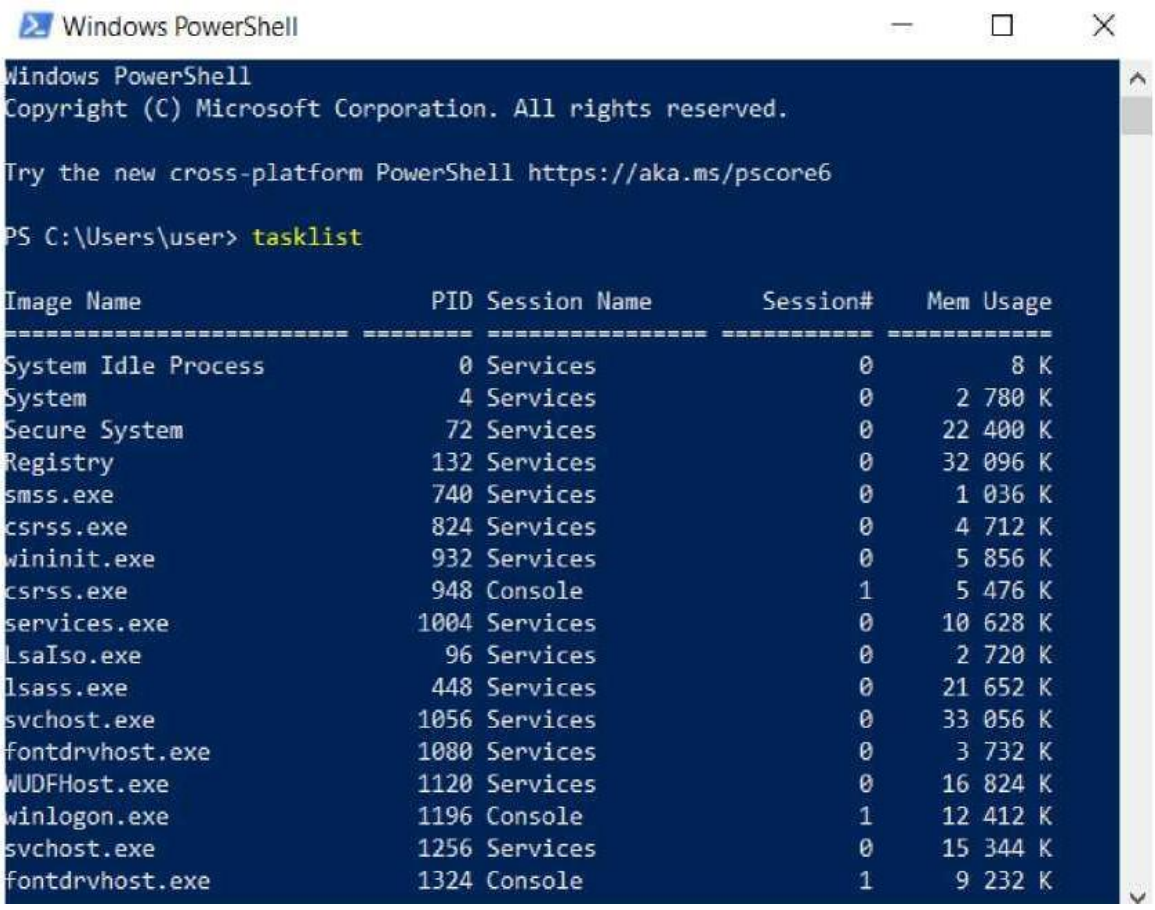


Рисунок 5.3 – Результати роботи утиліти TaskList в PowerShell

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 вказати 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:

– Повний обсяг встановленої у комп’ютері фізичної пам’яті.

– Загальний обсяг віртуальної пам’яті та доступної (вільної) в цей момент часу віртуальної пам’яті.

– 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).

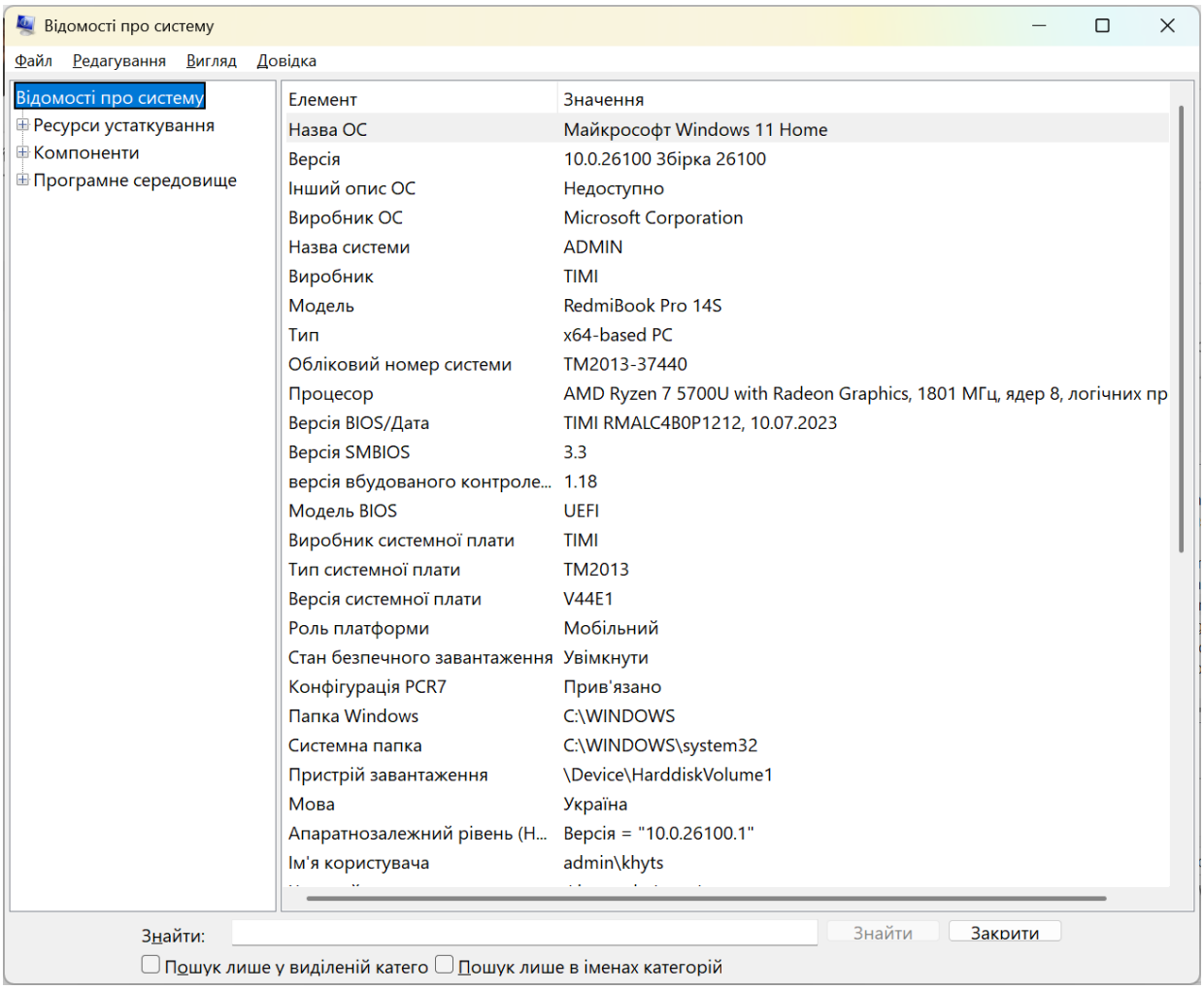


Рисунок 5.4 – Результати роботи команди msinfo32

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).

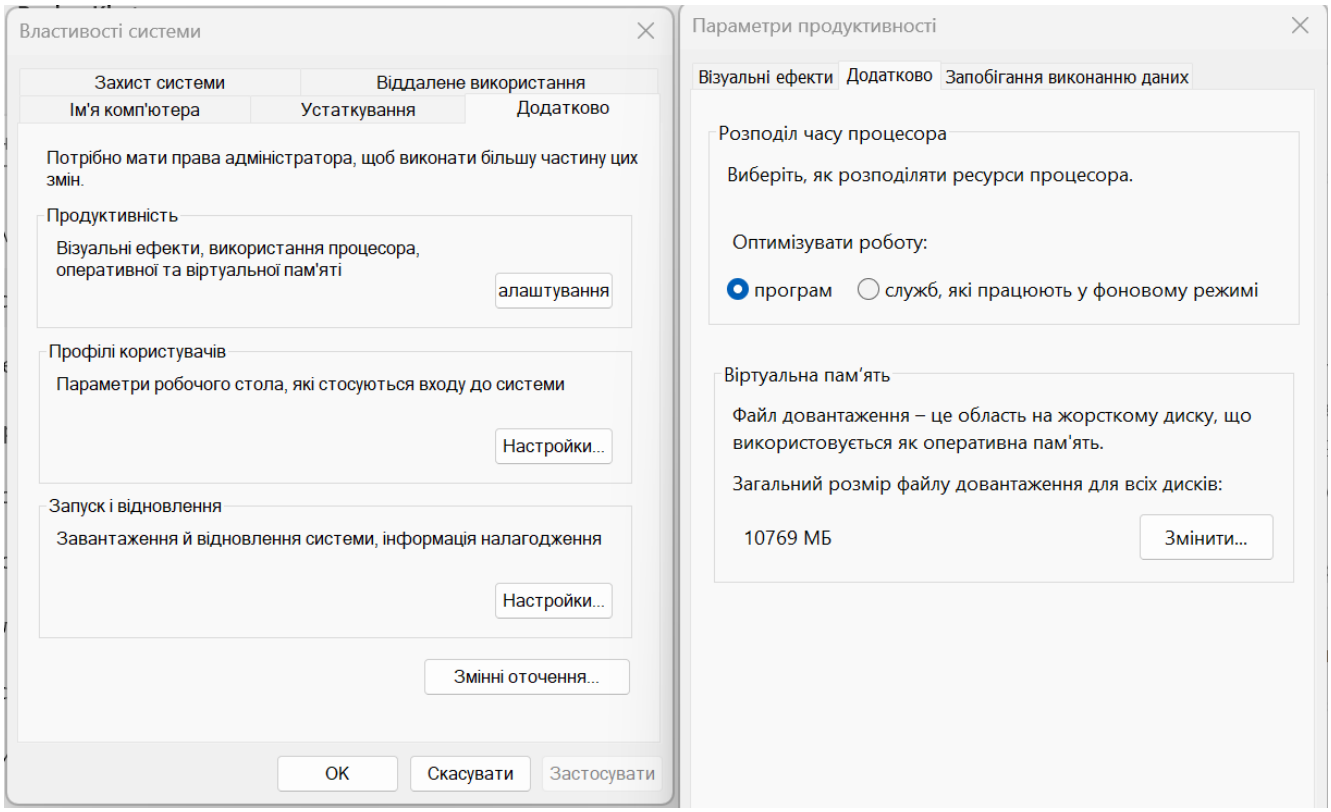


Рисунок 5.5 – Налаштування властивостей файлу підкачки

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 та 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 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 шляхом:

– Використанням двох жорстких дисків.

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

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

– 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 кешу. Після цього слід натиснути кнопку Задати й 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:

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

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

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

– Встановити файлу підкачки бажаний розмір. В результаті робота з файлом підкачки стане максимально швидкою, а процесорна потужність та дисковий простір будуть використовуватися ефективно.

– 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).

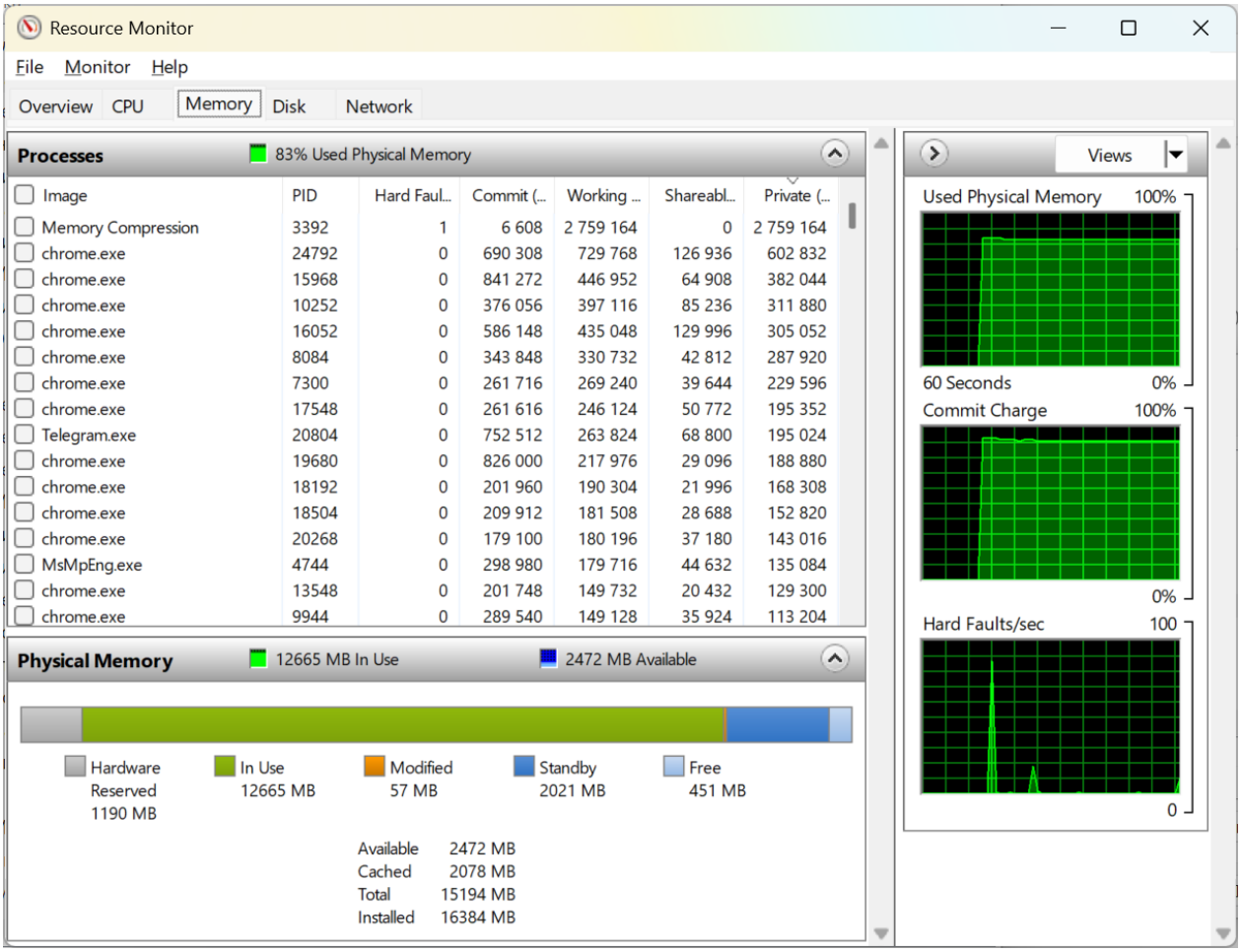


Рисунок 5.6 – Монітор ресурсів (вкладка Пам’ять)

Оптимізація роботи віртуальної пам’яті. Підвищення продуктивності роботи віртуальної пам’яті зводиться до наступних дій:

– Визначення необхідного обсягу фізичної пам’яті.
– Встановлення раціональної інтенсивності сторінкового обміну.
– 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:

– Пам’ять\Доступно байт, який показує поточний обсяг пам’яті в байтах, доступний для використання процесами.

– 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:

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

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

– Пам’ять\Байт у не вивантажуваному сторінковому пулі, Пам’ять\Розподілів у не вивантажуваному сторінковому пулі, Процес (ім’я процесу)\Байт у не вивантажуваному 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:

– Файл підкачки слід розміщувати на окремому логічному диску.

– 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.

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

– 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.

– 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.

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

– 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 обсяг 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?

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Андреєв А. П. Основи операційних систем: структура та функціональність. Львів : Вид-во Львів. політехніки, 2020. 275 с.

2. Вольман Д. Основи операційних систем: Linux, Windows та інші: навчальний посібник. Харків : ХНУРЕ, 2020. 540 с.

3. Горбунов Ю. М. Операційні системи: навчально-методичні рекомендації. Одеса : ОНПУ, 2022. 178 с.

4. Іванов Ю. П. Основи адміністрування Linux. Одеса : Од. нац. ун-т, 2021. 312 с.

5. Коваленко В. Л. Операційні системи: лабораторний курс. Київ : НАУ, 2023. 220 с.

6. Лавренюк В. С. Конспект лекцій з операційних систем (Windows, Linux). Київ : КПІ ім. Ігоря Сікорського, 2021. 215 с.

7. Мельник О. В., Левченко Р. П. Операційні системи: лабораторний практикум. Дніпро : ДНУ, 2023. 312 с.

8. Сілбершац А., Гальвін П., Гейдж Г. Операційні системи: концепції та механізми. Львів : ЛНУ, 2022. 960 с.

9. Столяренко О. О., Василенко О. А. Операційні системи: базові принципи та архітектура. Харків : ХНУ ім. В. Н. Каразіна, 2022. 328 с.

10. Таненбаум Е., Бос Г. Сучасні операційні системи: підручник. Київ : Наук. думка, 2021. 1168 с.

11. Титов С. Г., Демиденко І. В. Операційні системи: методологія, принципи, практика. Донецьк : ДонНТУ, 2021. 290 с.

12. Черненко О. І. Практикум з операційних систем Linux та Windows. Тернопіль : ТНТУ, 2022. 196 с.

13. Bovet D. P., Cesati M. Understanding the linux kernel. O’Reilly Media, 2021 p.

14. Love R. Linux kernel development. Addison-Wesley, 2021. 456 p.

15. Mauro J., McDougall R. Solaris internals: solaris 10 and opensolaris kernel architecture. Pearson, 2020. 1176 p.

Операційні системи: конспект лекцій з навчальної дисципліни для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерні науки» галузі знань 12 Інформаційні технології спеціальності 122 Комп'ютерні науки денної та заочної форм навчання Частина 1/ уклад. Р.А. Хиць, Ю.Й. Тулашвілі. Луцьк: ЛНТУ. 2025. 63 с.

Комп'ютерний набір і верстка: Р.А. Хиць

Підписано до друку 2025 р.
Формат 60х 84/16. Гарнітура Times New Roman.
Папір офсетний 80 г/м2. Друк офсетний.
Ум. друк. арк. 7,5. Обл.-вид. арк. 75.
Наклад 50 прим. Зам. №

Інформаційно-видавничий відділ
Луцького національного технічного університету
43018, Луцьк-18, вул. Львівська,75.
Друк – ІВВ ЛНТУ