


МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ  
до лабораторних робіт з навчальної дисципліни  
**«Операційні системи»**  
**Частина 1**  
для студентів денної та заочної форми навчання  
спеціальності 122 «Комп'ютерні науки»

Затверджено  
на засіданні групи забезпечення спеціальності  
Протокол №   3   від «  21  »  03  2022р.

Голова групи  Кузніченко С.Д.

Затверджено  
на засіданні кафедри \_\_\_\_\_  
Протокол №   6   від «  28  »  01  2022р.

Завідувач кафедри  Казакова Н.Ф.

Одеса 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ  
до лабораторних робіт з навчальної дисципліни  
*«Операційні системи»*  
*Частина 1*

для студентів денної та заочної форми навчання  
спеціальності 122 «Комп'ютерні науки»

Затверджено  
на засіданні групи  
забезпечення спеціальності  
Протокол № 6  
від « 28 » 01 2022р.

Одеса – 2022

Методичні вказівки до лабораторних робіт з дисципліни „*Операційні системи*”, частина 1 для студентів I року навчання денної та заочної форми за спеціальністю 122 «Комп’ютерні науки», рівень вищої освіти бакалавр /Терещенко Т.М. – Одеса, ОДЕКУ, 2022.

## ЗМІСТ

|   |           |
|---|-----------|
| <b>ВСТУП.....</b>   | <b>5</b>  |
| <b>ЛАБОРАТОРНА РОБОТА №1.....</b>   | <b>8</b>  |
| <i>Основи роботи в командному рядку операційної системи сімейства Windows .....</i> | <i>8</i>  |
| <b>ЛАБОРАТОРНА РОБОТА №2.....</b>   | <b>23</b> |
| <i>Основи роботи в терміналі операційної системи сімейства Linux .....</i>          | <i>23</i> |
| <b>ЛАБОРАТОРНА РОБОТА №3.....</b>   | <b>34</b> |
| <i>Створення та запуск bash-скриптів.....</i>                                       | <i>34</i> |
| <b>ЛАБОРАТОРНА РОБОТА №4.....</b>   | <b>45</b> |
| <i>Управління процесами та потоками в операційних системах Windows і Linux...</i>   | <i>45</i> |
| <b>ЛІТЕРАТУРА.....</b>  | <b>61</b> |

## 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.

Дані методичні вказівки до виконання лабораторних робіт містять теоретичні відомості та методику виконання 4 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.

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.

# 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.



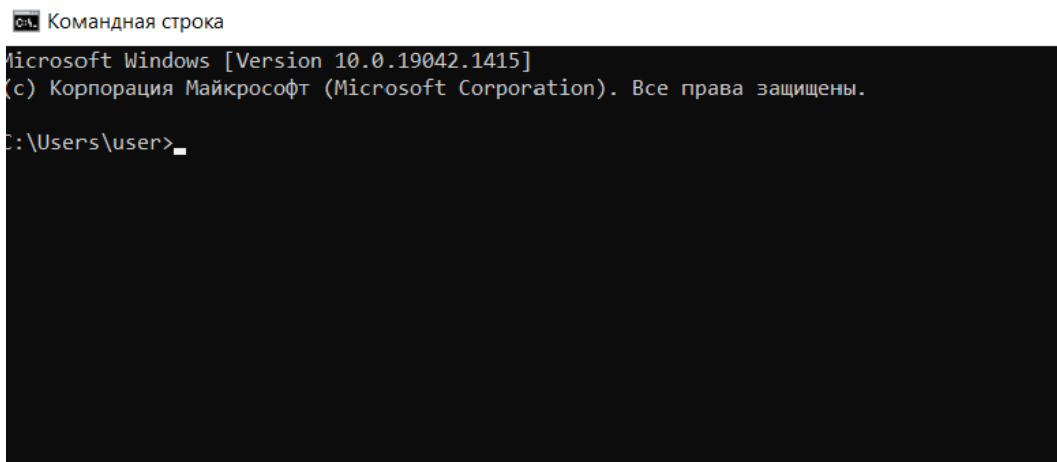


Рис. 1.1 – Вікно інтерпретатора Windows

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.

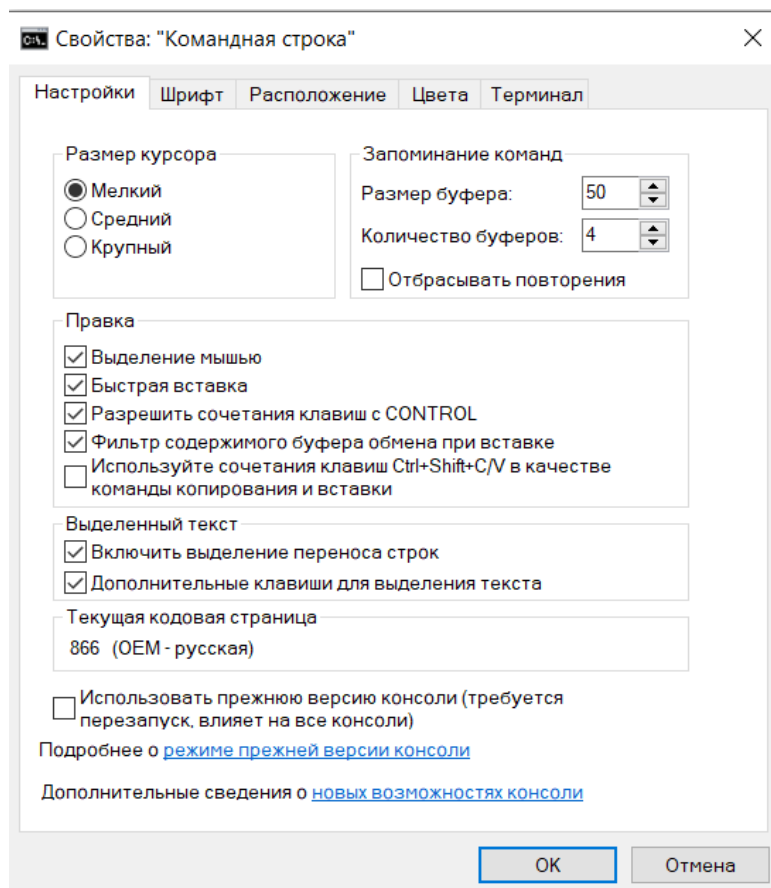


Рис. 1.2 – Вікно «Властивості» командного рядка

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.

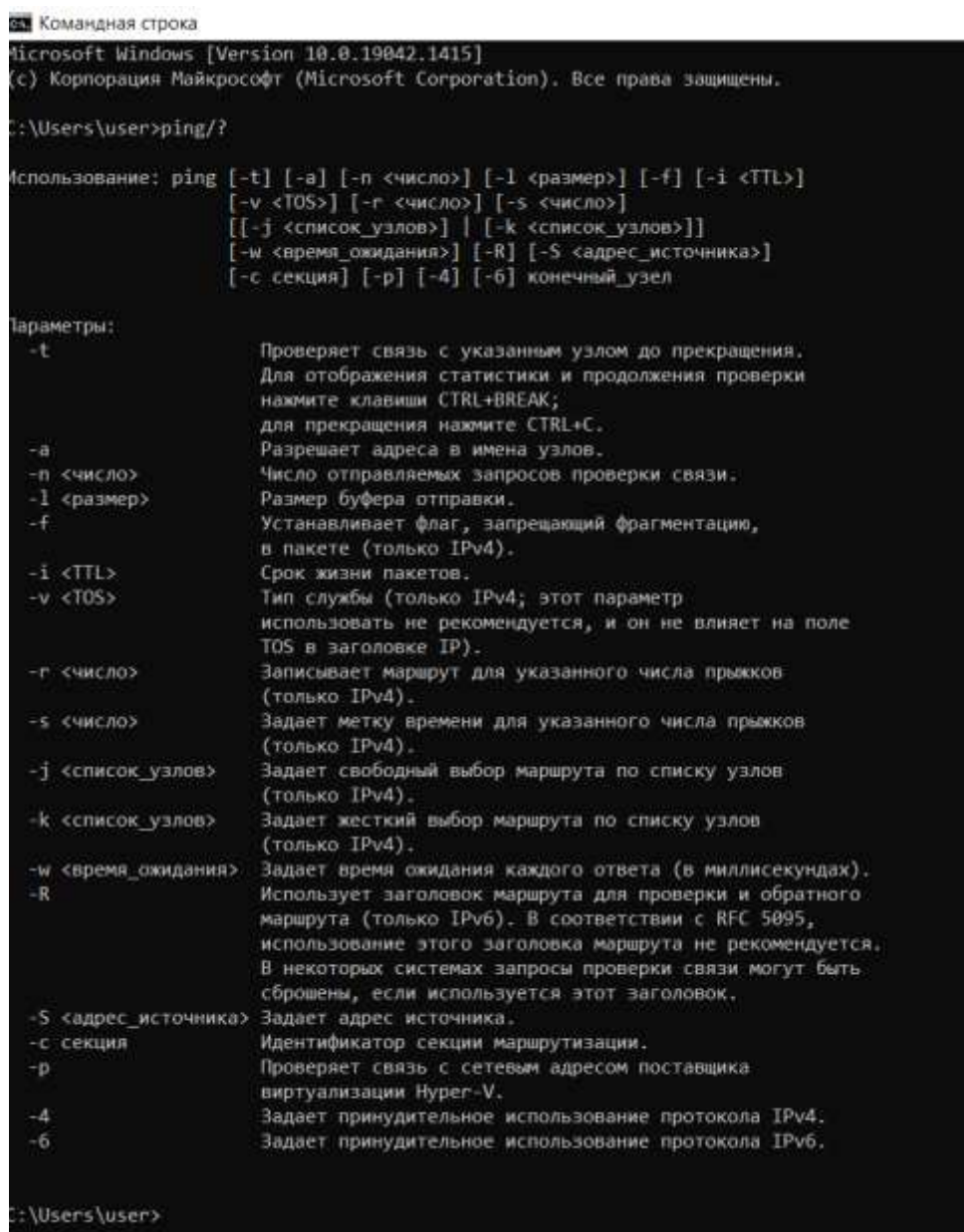
ipconfig/all – відображення повної інформації всіх адаптерів мережі.

getmac – отримати MAC-адресу мережевих карт.

arp -a – відображення 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).



```
Командная строка
Microsoft Windows [Version 10.0.19042.1415]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\user>ping/?

Использование: ping [-t] [-a] [-n <число>] [-l <размер>] [-f] [-i <TTL>]
[-v <TOS>] [-r <число>] [-s <число>]
[[ -j <список_узлов> ] | [ -k <список_узлов> ]]
[-w <время_ожидания>] [-R] [-S <адрес_источника>]
[-c секция] [-p] [-4] [-6] конечный_узел

Параметры:
-t                Проверяет связь с указанным узлом до прекращения.
                  Для отображения статистики и продолжения проверки
                  нажмите клавиши CTRL+BREAK;
                  для прекращения нажмите CTRL+C.
-a                Разрешает адреса в имена узлов.
-n <число>        Число отправляемых запросов проверки связи.
-l <размер>       Размер буфера отправки.
-f                Устанавливает флаг, запрещающий фрагментацию,
                  в пакете (только IPv4).
-i <TTL>          Срок жизни пакетов.
-v <TOS>          Тип службы (только IPv4; этот параметр
                  использовать не рекомендуется, и он не влияет на поле
                  TOS в заголовке IP).
-r <число>        Записывает маршрут для указанного числа прыжков
                  (только IPv4).
-s <число>        Задаёт метку времени для указанного числа прыжков
                  (только IPv4).
-j <список_узлов> Задаёт свободный выбор маршрута по списку узлов
                  (только IPv4).
-k <список_узлов> Задаёт жёсткий выбор маршрута по списку узлов
                  (только IPv4).
-w <время_ожидания> Задаёт время ожидания каждого ответа (в миллисекундах).
-R               Использует заголовок маршрута для проверки и обратного
                  маршрута (только IPv6). В соответствии с RFC 5095,
                  использование этого заголовка маршрута не рекомендуется.
                  В некоторых системах запросы проверки связи могут быть
                  сброшены, если используется этот заголовок.
-S <адрес_источника> Задаёт адрес источника.
-c секция        Идентификатор секции маршрутизации.
-p              Проверяет связь с сетевым адресом поставщика
                  виртуализации Hyper-V.
-4              Задаёт принудительное использование протокола IPv4.
-6              Задаёт принудительное использование протокола IPv6.

C:\Users\user>
```

Рис. 1.3 – Результат выполнения команды ping/?

tracert [кінцевий\_вузол] – трасування маршруту..

pathping [кінцевий\_вузол] – об'єднує 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).



```
Командная строка
-6          Задает принудительное использование протокола IPv6.

C:\Users\user>netstat/?

Отображение статистики протокола и текущих сетевых подключений TCP/IP.

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p протокол] [-r] [-s] [-t] [-x] [-y] [интервал]

-a          Отображение всех подключений и портов прослушивания.
-b          Отображение исполняемого файла, участвующего в создании
каждого подключения или порта прослушивания. Иногда известные
исполняемые файлы содержат множество независимых
компонентов. Тогда отображается последовательность компонентов,
участвующих в создании подключения или порта прослушивания. В
этом случае имя исполняемого файла находится снизу в скобках
[], сверху находится вызванный им компонент, и так до тех
пор, пока не достигнут TCP/IP. Заметьте, что такой подход
может занять много времени и требует достаточных разрешений.
-e          Отображение статистики Ethernet. Может применяться вместе
с параметром -s.
-f          Отображение полного имени домена (FQDN) для внешних адресов.
-n          Отображение адресов и номеров портов в числовом формате.
-o          Отображение ID процесса каждого подключения.
-p протокол Отображение подключений для протокола, заданного соответствующим
параметром. Допустимые значения для протокола: TCP, UDP, TCPv6 или UDPv6.
Если используется вместе с параметром -s для отображения
статистики по протоколам, допустимы следующие значения:
IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP или UDPv6.
-q          Отображение всех подключений, портов прослушивания и ограниченных
непрослушиваемых TCP-портов. Ограниченные непрослушиваемые порты могут быть или не быть
связанными с активными подключениями.
-r          Отображение таблицы маршрутов.
-s          Отображение статистики по протоколам. По умолчанию статистика
отображается для протоколов IP, IPv6, ICMP, ICMPv6, TCP, TCPv6,
UDP и UDPv6. Параметр -p позволяет указать подмножество данных по умолчанию.
-t          Отображение состояния разгрузки для текущего подключения.
-x          Отображение подключений, прослушивателей и общих конечных точек
NetworkDirect.
-y          Отображение шаблона подключений TCP для всех подключений.
Не может использоваться вместе с другими параметрами.
interval  Повторное отображение выбранной статистики с паузой
между отображениями, заданной интервалом в секундах. Чтобы прекратить повторное отображение
статистики, нажмите клавиши CTRL+C.
Если этот параметр опущен, netstat напечатает текущую информацию о конфигурации один раз.

C:\Users\user>
```

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="[імя\_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.

wuauctl – 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).

```

C:\Users\user>help
Для получения сведений об определенной команде наберите HELP <имя команды>
ASSOC          Вывод либо изменение сопоставлений по расширениям имен файлов.
ATTRIB        Отображение и изменение атрибутов файлов.
BREAK         Включение и выключение режима обработки комбинации клавиш CTRL+C.
BCDEDIT       Задаёт свойства в базе данных загрузки для управления начальной
              загрузкой.
CACLS         Отображение и редактирование списков управления доступом (ACL)
              к файлам.
CALL          Вызов одного пакетного файла из другого.
CD            Вывод имени либо смена текущей папки.
CHCP         Вывод либо установка активной кодовой страницы.
CHDIR        Вывод имени либо смена текущей папки.
CHKDSK       Проверка диска и вывод статистики.
CHKNTFS      Отображение или изменение выполнения проверки диска во время
              загрузки.
CLS          Очистка экрана.
CMD          Запуск еще одного интерпретатора командных строк Windows.
COLOR        Установка цветов переднего плана и фона, используемых по умолчанию.
COMP         Сравнение содержимого двух файлов или двух наборов файлов.
COMPACT      Отображение и изменение сжатия файлов в разделах NTFS.
CONVERT      Преобразует тома FAT в NTFS. Вы не можете
              преобразовать текущий диск.
COPY         Копирование одного или нескольких файлов в другое место.
DATE        Вывод либо установка текущей даты.
DEL          Удаление одного или нескольких файлов.
DIR          Вывод списка файлов и подпапок из указанной папки.
DISKPART     Отображает или настраивает свойства раздела диска.
DOSKEY       Редактирует командные строки, повторно вызывает команды Windows и создает
              макросы.
DRIVERQUERY Отображает текущее состояние и свойства драйвера устройства.
ECHO        Отображает сообщения и переключает режим отображения команд на экране.
ENDLOCAL    Завершает локализацию изменений среды для пакетного файла.
ERASE       Удаляет один или несколько файлов.
EXIT        Завершает работу программы CMD.EXE (интерпретатора командных строк).
FC          Сравнивает два файла или два набора файлов и
              отображает различия между ними.

```

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 Explorer.  
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.  
shrpwb – мастер створення спільної папки.  
taskmgr – запуск диспетчера задач.  
wimgmt.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).

```
Windows PowerShell
PS C:\OS> tree /f /a
Структура папок тома windows 10
Серийный номер тома: CE95-E948
C: .
+---LAB_1
|   +---lab1_1
|   |   1.txt
|   |   +---lab1_1
|   |   |   2.txt
|   |   +---lab1_2
|   |   |   2.txt
|   |   |   4.txt
|   |   \---lab1_3
|   |       1.txt
|   |       3.txt
|   +---lab1_2
|   |   2.txt
|   |   4.txt
|   \---lab1_3
|       1.txt
|       3.txt
```

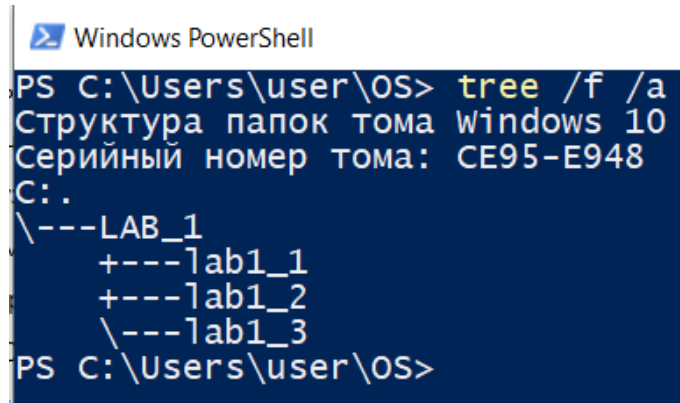
Рис. 1.6 – Дерево каталогів папки OS створене засобами командного рядка

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

```
Командная строка
C:\Users\user>md OS
C:\Users\user>md OS\LAB_1
C:\Users\user>md OS\LAB_1\lab1_1
C:\Users\user>md OS\LAB_1\lab1_2
C:\Users\user>md OS\LAB_1\lab1_3
C:\Users\user>
```

Рис. 1.7 – Створення папок на диску командою md

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



```
Windows PowerShell
PS C:\Users\user\OS> tree /f /a
Структура папок тома Windows 10
Серийный номер тома: CE95-E948
C: .
\---LAB_1
    +---lab1_1
    +---lab1_2
    \---lab1_3
PS C:\Users\user\OS>
```

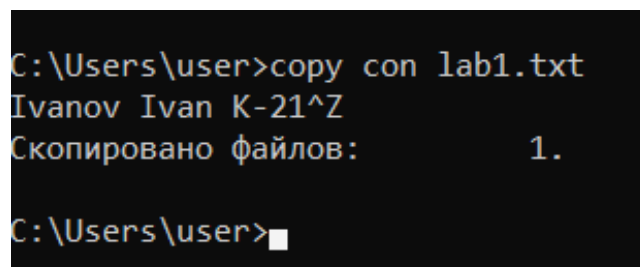
Рис. 1.8 – Дерево каталогів папки OS після виконання команд md

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.



```
C:\Users\user>copy con lab1.txt
Ivanov Ivan K-21^Z
Скопировано файлов:      1.
C:\Users\user>█
```

Рис. 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).

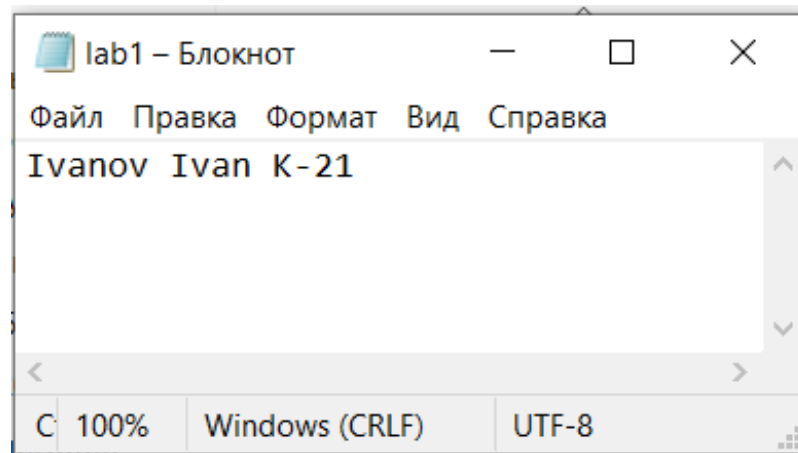


Рис. 1.10 – Вміст текстового файлу lab1.txt

Для відображення вмісту файлу в консоль використовуємо команду type ім'я\_файла. Копіювати файл можна командою:

сору 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).

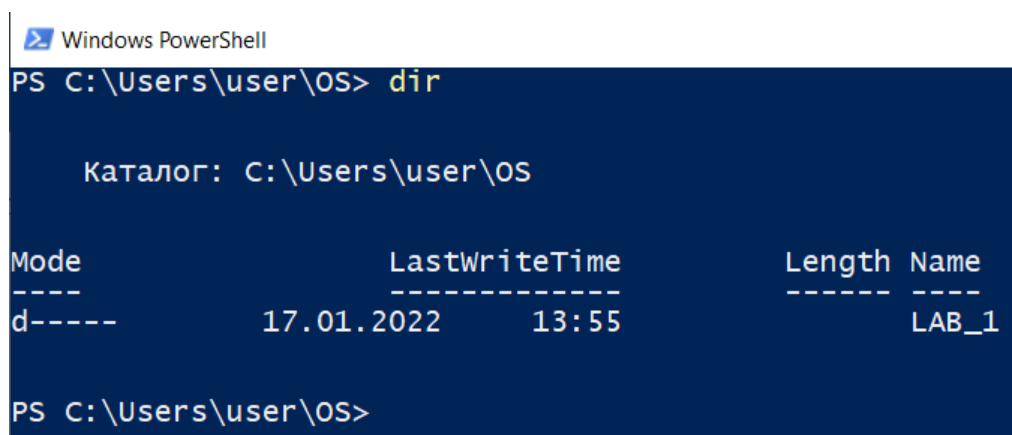


Рис. 1.11 – Результат виконання команди dir

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**

| <i>1 варіант</i>   | <i>2 варіант</i>  |
|--|---|
| <pre> FILES/ ├── ReadMe.txt ├── Overheads/ │   ├── Ses00 │   └── int.bin └── Demo/     ├── Extra.bin     ├── text.txt     ├── PTR01/     │   ├── double.bin     │   └── symbols.chr     └── CHAR/         ├── file.c         └── long.asc         </pre> | <pre> FILES/ ├── ReadMe.txt ├── System/ │   └── int.bin ├── Mappings/ │   └── long.asc └── Adobe/     ├── HKSCS.txt     ├── text.txt     └── Unicode/         ├── double.bin         └── Icu/             ├── symbols.chr             └── icud.dat         </pre> |

| 3 вариант  | 4 вариант  |
|--|--|
| <pre> FILES/ ├── ReadMe.txt ├── facts/ │   ├── long.asc │   └── Unicode/ │       ├── double.bin │       └── Icu/ │           ├── text.txt │           └── icud.dat ├── Mappings/ │   └── int.bin └── Adobe/     ├── HKSCS.txt     └── symbols.chr </pre>   | <pre> 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 </pre>  |
| 5 вариант  | 6 вариант  |
| <pre> FILES/ ├── ReadMe.txt ├── TechInfo/ │   ├── double.bin │   └── Lang/ │       ├── af.txt │       └── text.txt ├── ABBYY/ │   └── Reader/ │       ├── int.bin │       ├── Zlib.dll │       └── Support/ │           ├── long.asc │           └── symbols.chr </pre>                                    | <pre> FILES/ ├── ReadMe.txt ├── double.bin ├── Panel/ │   ├── long.asc │   └── Lingvo/ │       ├── int.bin │       ├── Abbrev.lsd │       └── Dic/ │           ├── text.txt │           └── Index/ │               └── Support └── Adobe/     ├── symbols.chr     └── Index.dat </pre>   |
| 7 вариант  | 8 вариант  |
| <pre> FILES/ ├── ReadMe.txt ├── Acronis/ │   ├── Sandal/ │   │   ├── symbols.chr │   │   ├── text.txt │   │   └── Common/ │   │       ├── int.bin │   │       └── link.dll │   └── TrueHome/ │       ├── license.txt │       ├── long.asc │       └── Common/ │           └── double.bin └── Image/ </pre> | <pre> 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 </pre> |

| 9 варіант  | 10 варіант  |
|--|---|
| <pre> FILES/   README.txt   BVRDE/     symbols.chr     Lex/       int.bin     Solutions/     Sounds/       long.asc     Templates/       text.txt       Files/         double.bin       Wizard/         makefile   Sym/ </pre> | <pre> FILES/   README.txt   Ascii/     long.asc   Form/     Code/       double.bin       symbols.chr     Docu/       text.txt     Sym/       int.bin   Index/   Ole/     Integers.bin </pre>  |
| 11 варіант   | 12 варіант  |
| <pre> FILES/   README.txt   Comp/     text.txt     manifest   Comm/     Code/       double.bin     Docu/       int.bin       V24.odc       ru/     Sym/       long.asc       ver.odc       symbols.chr   Index/ </pre>         | <pre> FILES/   README.txt   Eclipse/     symbols.chr     facts.xml   PlugIns/     double.bin     AltHistory/       Alt.dll       long.asc     BackGroundCopy/       File_ID.diz       Reg/         text.txt       Src/         int.bin     System/       Dest.on   Temp/ </pre> |
| 13 варіант   | 14 варіант  |
| <pre> FILES/   README.txt   System/     int.bin   Mappings/     long.asc   Adobe/     HKSCS.txt     text.txt     Unicode/       double.bin       Icu/         symbols.chr         icud.dat </pre>                              | <pre> FILES/   README.txt   facts/     long.asc     Unicode/       double.bin       Icu/         text.txt         icud.dat   Mappings/     int.bin   Adobe/     HKSCS.txt     symbols.chr </pre>  |

| 15 вариант  | 16 вариант   |
|---|--|
| <pre> FILES/   ReadMe.txt   Ascii/     long.asc     Form/       Code/         double.bin         symbols.chr       Docu/         text.txt       Sym/         int.bin   Index/   Ole/     Integers.bin </pre>  | <pre> FILES/   ReadMe.txt   Comp/     text.txt     manifest   Comm/     Code/       double.bin     Docu/       int.bin       V24.odc       ru/         long.asc         ver.odc   Sym/     symbols.chr   Index/ </pre>         |
| 17 вариант  | 18 вариант   |
| <pre> FILES/   ReadMe.txt   Eclipse/     symbols.chr     facts.xml   PlugIns/     double.bin     AltHistory/       Alt.dll       long.asc     BackGroundCopy/       File ID.diz       Reg/         text.txt       Src/         int.bin     System/       Dest.on   Temp/ </pre> | <pre> FILES/   ReadMe.txt   BVRDE/     symbols.chr     Lex/       int.bin     Solutions/     Sounds/       long.asc     Templates/       text.txt       Files/         double.bin       Wizard/         makefile   Sym/ </pre> |
| 19 вариант  | 20 вариант   |
| <pre> FILES/   ReadMe.txt   Ascii/     long.asc     Form/       Code/         double.bin         symbols.chr       Docu/         text.txt       Sym/         int.bin   Index/   Ole/     Integers.bin </pre>  | <pre> FILES/   ReadMe.txt   TechInfo/     double.bin     Lang/       af.txt       text.txt   ABBYY/     Reader/       int.bin       Zlib.dll     Support/       long.asc       symbols.chr </pre>                              |

***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 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.).

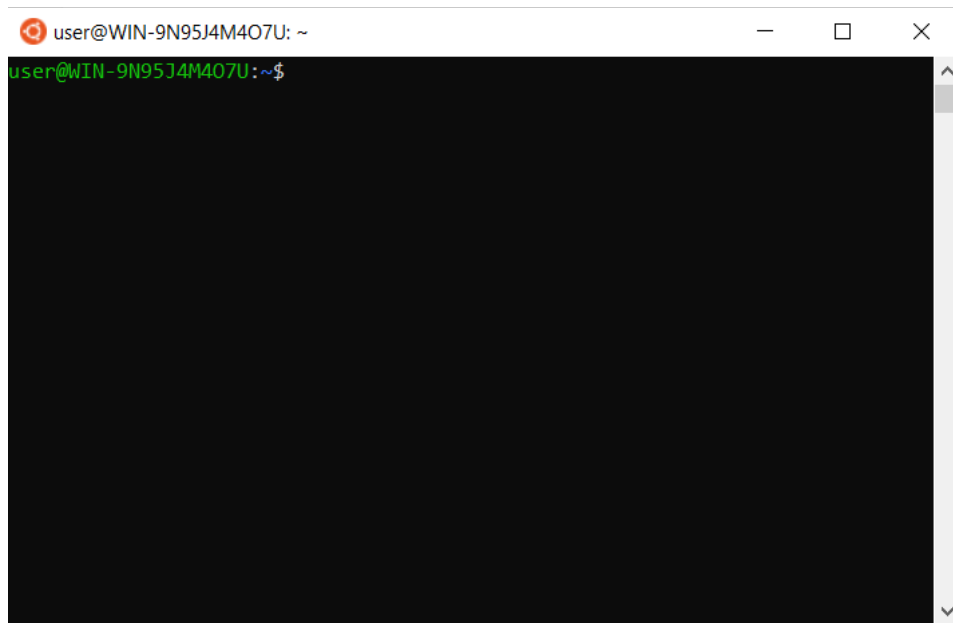


Рис. 2.1 – Термінал Ubuntu з графічним інтерфейсом

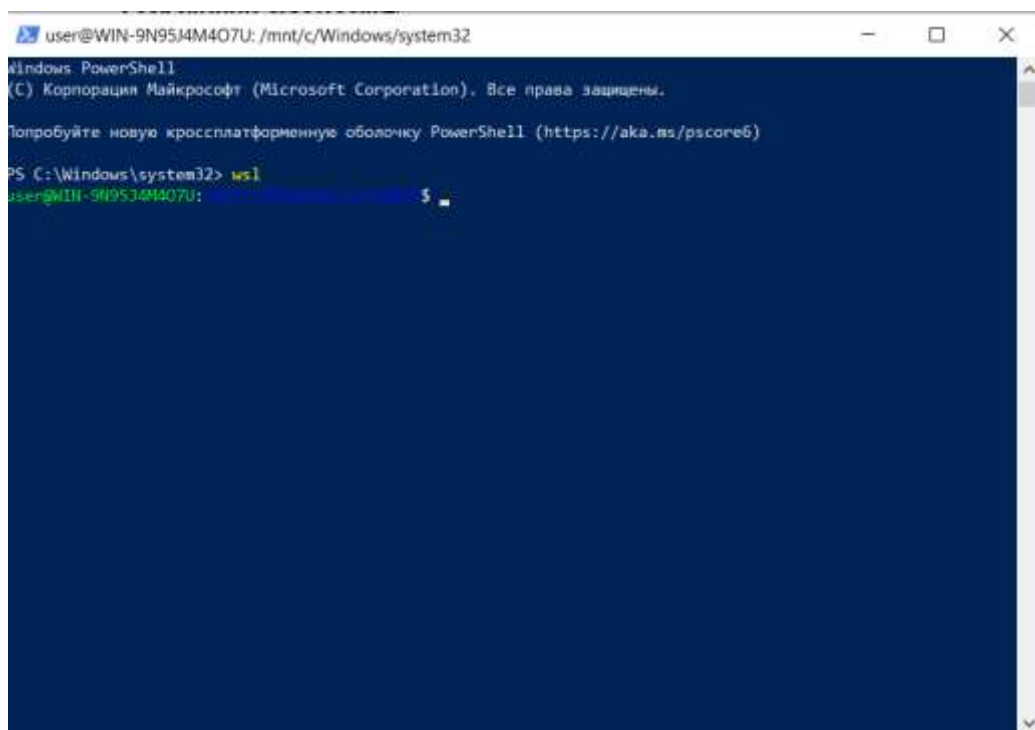


Рис. 2.2 – Запуск терміналу Ubuntu через PowerShell

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.

```

user@WIN-9N95J4M407U: ~
user@WIN-9N95J4M407U:~$ ls
LAB2  text1.txt  text2.txt
user@WIN-9N95J4M407U:~$ ls -R
.:
LAB2  text1.txt  text2.txt

./LAB2:
lab2_1

./LAB2/lab2_1:
user@WIN-9N95J4M407U:~$ ls -R -l
.:
total 0
drwxr-xr-x 1 user user 512 Jan 19 18:13 LAB2
-rw-r--r-- 1 user user  24 Jan 19 18:20 text1.txt
-rw-r--r-- 1 user user  14 Jan 19 18:17 text2.txt

./LAB2:
total 0
drwxr-xr-x 1 user user 512 Jan 19 18:13 lab2_1

./LAB2/lab2_1:
total 0
user@WIN-9N95J4M407U:~$

```

Рис. 2.3 – Результати виконання команди ls

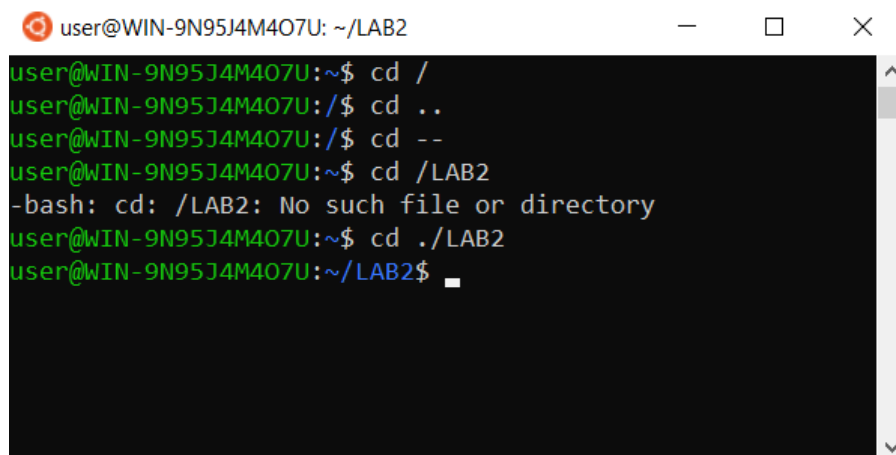
Для створення нової папки використовується команда `mkdir`. Якщо треба створити нову папку в заданому каталозі, то необхідно вказати повний шлях, наприклад, створюємо папку `test` командою `mkdir /home/user/test`. Для очищення терміналу використовується команда `clear` або клавіші `Ctrl+L`. В терміналі 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.



```
user@WIN-9N95J4M407U: ~/LAB2
user@WIN-9N95J4M407U:~$ cd /
user@WIN-9N95J4M407U:/$ cd ..
user@WIN-9N95J4M407U:/$ cd --
user@WIN-9N95J4M407U:~$ cd /LAB2
-bash: cd: /LAB2: No such file or directory
user@WIN-9N95J4M407U:~$ cd ./LAB2
user@WIN-9N95J4M407U:~/LAB2$
```

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

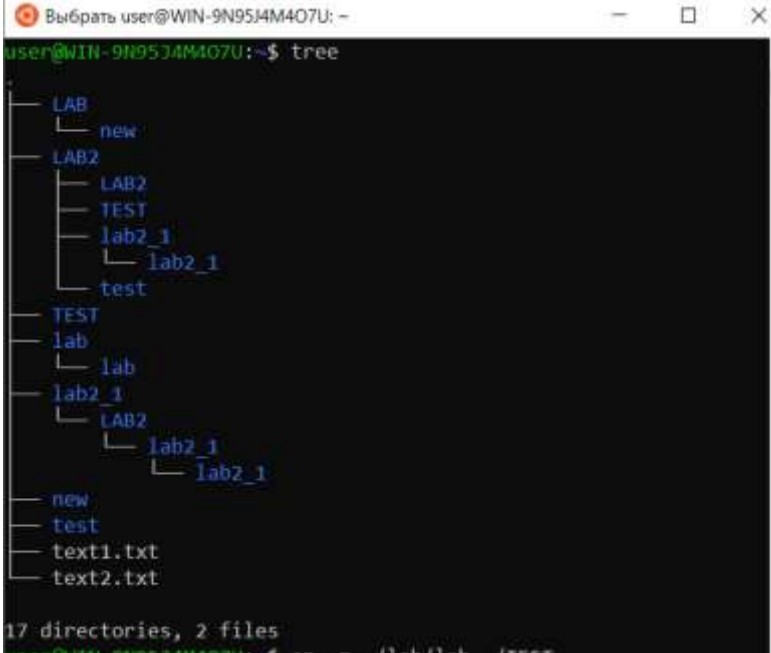
знаходиться у поточній папці. Можливо вказати повний шлях файлу, наприклад: `rm /usr/share/file`. Для видалення папки необхідно використовувати опцію `-r`. Вона включає рекурсивне видалення всіх файлів та папок на всіх рівнях вкладеності, наприклад, `rm -r /home/user/photo/`. Ця команда видаляє файли 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.



```
user@WIN-9N95J4M407U:~$ tree
├── LAB
│   └── new
├── LAB2
│   ├── LAB2
│   ├── TEST
│   ├── lab2_1
│   │   └── lab2_1
│   └── test
├── TEST
│   ├── lab
│   │   ├── lab
│   │   └── lab2_1
│   │       └── LAB2
│   │           ├── lab2_1
│   │           └── lab2_1
│   ├── new
│   ├── test
│   ├── text1.txt
│   └── text2.txt
└── 17 directories, 2 files
user@WIN-9N95J4M407U:~$ cp -r ./lab/lab ./TEST
```

Рис. 2.5 – Результати виконання команди `tree` до копіювання

```
user@WIN-9N95J4M407U: ~$ tree
.
├── LAB
│   └── new
├── LAB2
│   ├── LAB2
│   ├── TEST
│   ├── lab2_1
│   │   └── lab2_1
│   └── test
├── TEST
│   └── lab
├── lab
│   └── lab
├── lab2_1
│   └── LAB2
│       └── lab2_1
│           └── lab2_1
├── new
├── test
├── text1.txt
└── text2.txt

18 directories, 2 files
user@WIN-9N95J4M407U: ~$
```

Рис. 2.6 – Результати виконання команди tree після копіювання

### ***Створення файлів та запис даних в файл***

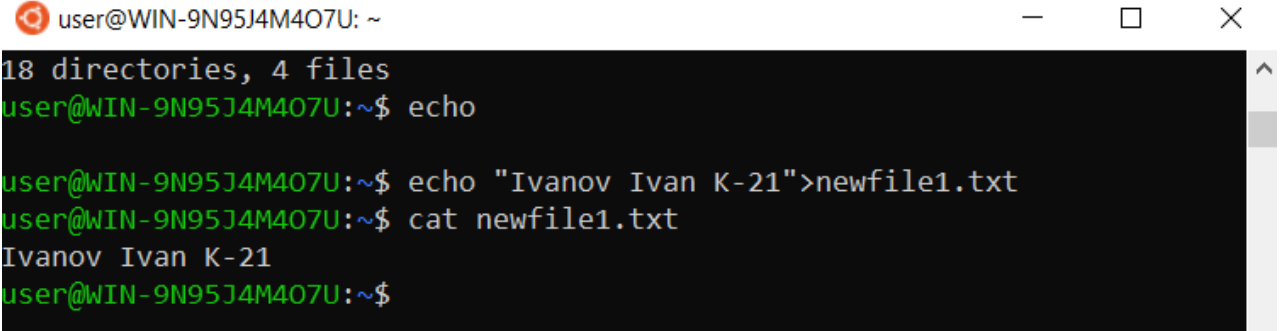
Створити файл в Linux можливо декількома способами. Розгляне один з них. Для створення файлу в поточному каталозі або в заданій папці використовується команда touch. Створимо файл newfile1.txt в поточному каталозі і файл newfile2.txt в папці new. Результати виконання обох команд відображені в дереві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

```
user@WIN-9N95J4M407U: ~$ touch newfile1.txt
user@WIN-9N95J4M407U: ~$ touch ./new/newfile2.txt
user@WIN-9N95J4M407U: ~$ tree
.
├── LAB
│   └── new
├── LAB2
│   ├── LAB2
│   ├── TEST
│   ├── lab2_1
│   │   └── lab2_1
│   └── test
├── TEST
│   └── lab
├── lab
│   └── lab
├── lab2_1
│   └── LAB2
│       └── lab2_1
│           └── lab2_1
├── new
│   └── newfile2.txt
├── newfile1.txt
├── test
├── text1.txt
└── text2.txt

18 directories, 4 files
```

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.



```
user@WIN-9N95J4M407U: ~  
18 directories, 4 files  
user@WIN-9N95J4M407U:~$ echo  
user@WIN-9N95J4M407U:~$ echo "Ivanov Ivan K-21">newfile1.txt  
user@WIN-9N95J4M407U:~$ cat newfile1.txt  
Ivanov Ivan K-21  
user@WIN-9N95J4M407U:~$
```

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.

## Варіанти індивідуальних завдань

| <i>1 варіант</i>  | <i>2 варіант</i>  |
|---|---|
| <pre> FILES/ ├── ReadMe.txt ├── System/ │   └── int.bin ├── Mappings/ │   └── long.asc └── Adobe/     ├── HKSCS.txt     ├── text.txt     └── Unicode/         ├── double.bin         └── Icu/             ├── symbols.chr             └── icud.dat </pre>                               | <pre> FILES/ ├── ReadMe.txt ├── Overheads/ │   ├── Ses00 │   └── int.bin └── Demo/     ├── Extra.bin     ├── text.txt     └── PTR01/         ├── double.bin         └── symbols.chr CHAR/ ├── file.c └── long.asc </pre>  |
| <i>3 варіант</i>  | <i>4 варіант</i>  |
| <pre> 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 </pre> | <pre> FILES/ ├── ReadMe.txt ├── facts/ │   ├── long.asc │   └── Unicode/ │       ├── double.bin │       └── Icu/ │           ├── text.txt │           └── icud.dat ├── Mappings/ │   └── int.bin └── Adobe/     ├── HKSCS.txt     └── symbols.chr </pre>                |
| <i>5 варіант</i>  | <i>6 варіант</i>  |
| <pre> FILES/ ├── ReadMe.txt ├── double.bin ├── Panel/ │   ├── long.asc │   └── Lingvo/ │       ├── int.bin │       ├── Abbrev.lsd │       └── Dic/ │           ├── text.txt │           └── Index/ │               └── Support └── Adobe/     ├── symbols.chr     └── Index.dat </pre>  | <pre> FILES/ ├── ReadMe.txt ├── TechInfo/ │   ├── double.bin │   └── Lang/ │       ├── af.txt │       └── text.txt └── ABBYY/     └── Reader/         ├── int.bin         ├── Zlib.dll         └── Support/             ├── long.asc             └── symbols.chr </pre> |

| 7 вариант   | 8 вариант  |
|---|--|
| <pre> FILES/   ReadMe.txt   CaliPo/     libre.exe     Calibre/       recomp.exe       long.asc       plugins/         imagefor/           text.txt       resour/         symbols.chr         content/           int.bin           box.png           read/             double.bin       fonts/         double.bin </pre> | <pre> FILES/   ReadMe.txt   Acronis/     Sandal/       symbols.chr       text.txt     Common/       int.bin       link.dll     TrueHome/       license.txt       long.asc     Common/       double.bin   Image/ </pre>         |
| 9 вариант   | 10 вариант   |
| <pre> FILES/   ReadMe.txt   Ascii/     long.asc   Form/     Code/       double.bin       symbols.chr     Docu/       text.txt     Sym/       int.bin   Index/   Ole/     Integers.bin </pre>  | <pre> FILES/   ReadMe.txt   BVRDE/     symbols.chr     Lex/       int.bin     Solutions/     Sounds/       long.asc     Templates/       text.txt       Files/         double.bin       Wizard/         makefile   Sym/ </pre> |
| 11 вариант  | 12 вариант   |
| <pre> FILES/   ReadMe.txt   Eclipse/     symbols.chr     facts.xml   PlugIns/     double.bin     AltHistory/       Alt.dll       long.asc     BackGroundCopy/       File_ID.diz       Reg/         text.txt       Src/         int.bin     System/       Dest.on   Temp/ </pre>   | <pre> FILES/   ReadMe.txt   Comp/     text.txt     manifest   Comm/     Code/       double.bin     Docu/       int.bin       V24.odc       ru/         long.asc         ver.odc     Sym/       symbols.chr   Index/ </pre>     |

| <i>13 вариант</i>  | <i>14 вариант</i>   |
|--|---|
| <pre> FILES/   README.txt   facts/     long.asc     Unicode/       double.bin       Icu/         text.txt         icud.dat   Mappings/     int.bin   Adobe/     HKSCS.txt     symbols.chr </pre>               | <pre> FILES/   README.txt   System/     int.bin   Mappings/     long.asc   Adobe/     HKSCS.txt     text.txt     Unicode/       double.bin       Icu/         symbols.chr         icud.dat </pre>   |
| <i>15 вариант</i>  | <i>16 вариант</i>   |
| <pre> FILES/   README.txt   Comp/     text.txt     manifest   Comm/     Code/       double.bin     Docu/       int.bin       V24.odc       ru/   Sym/     long.asc     ver.odc     symbols.chr   Index/ </pre> | <pre> FILES/   README.txt   Ascii/     long.asc   Form/     Code/       double.bin       symbols.chr     Docu/       text.txt     Sym/       int.bin   Index/   Ole/     Integers.bin </pre>  |
| <i>17 вариант</i>  | <i>18 вариант</i>   |
| <pre> FILES/   README.txt   BVRDE/     symbols.chr   Lex/     int.bin   Solutions/   Sounds/     long.asc   Templates/     text.txt   Files/     double.bin   Wizard/     makefile   Sym/ </pre>               | <pre> FILES/   README.txt   Eclipse/     symbols.chr     facts.xml   PlugIns/     double.bin   AltHistory/     Alt.dll     long.asc   BackgroundCopy/     File_ID.diz   Reg/     text.txt   Src/     int.bin   System/     Dest.on   Temp/ </pre> |



| <i>19 варіант</i>   | <i>20 варіант</i>  |
|---|--|
| <pre> FILES/ ├── README.txt ├── TechInfo/ │   ├── double.bin │   └── Lang/ │       ├── af.txt │       └── text.txt ├── ABBYY/ │   └── Reader/ │       ├── int.bin │       ├── zlib.dll │       └── Support/ │           ├── long.asc │           └── symbols.chr </pre> | <pre> FILES/ ├── README.txt ├── Ascii/ │   ├── long.asc │   └── Form/ │       ├── Code/ │           ├── double.bin │           └── symbols.chr │       ├── Docu/ │           └── text.txt │       └── Sym/ │           └── int.bin ├── Index/ └── Ole/     └── Integers.bin </pre> |

**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 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).

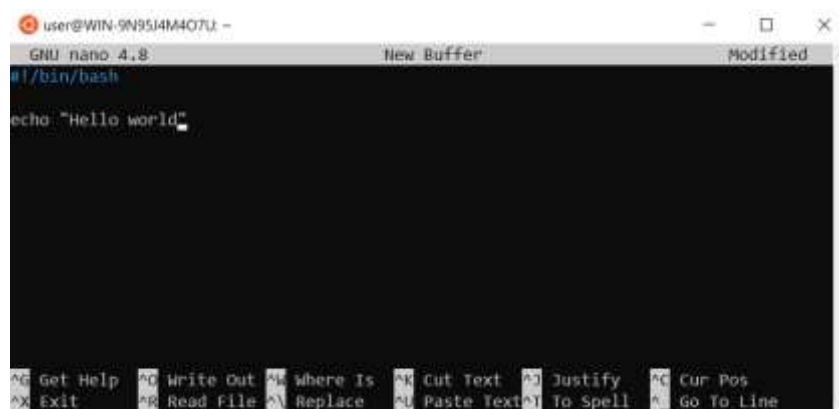


Рис. 3.1 – Вікно редактора файлів *nano*

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:

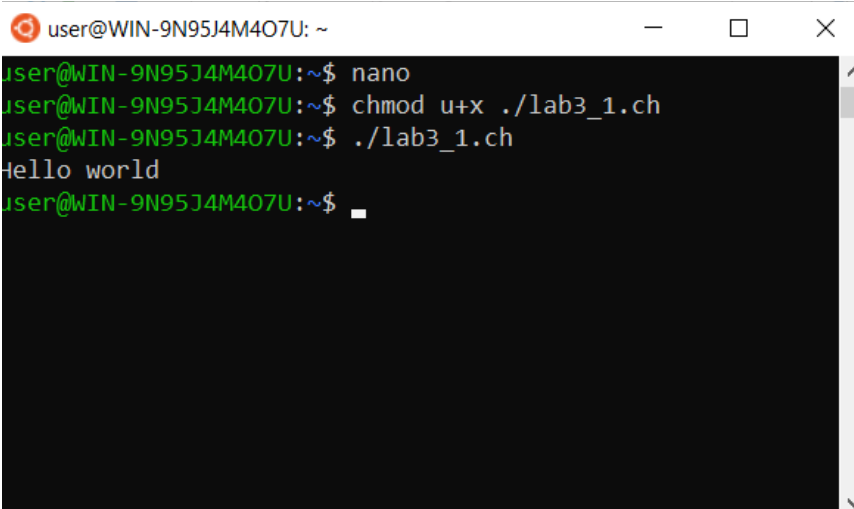
### *\$ chmod категорія дія прапор адреса\_файлу*

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 прапор.

Прапор – 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.ch. 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).

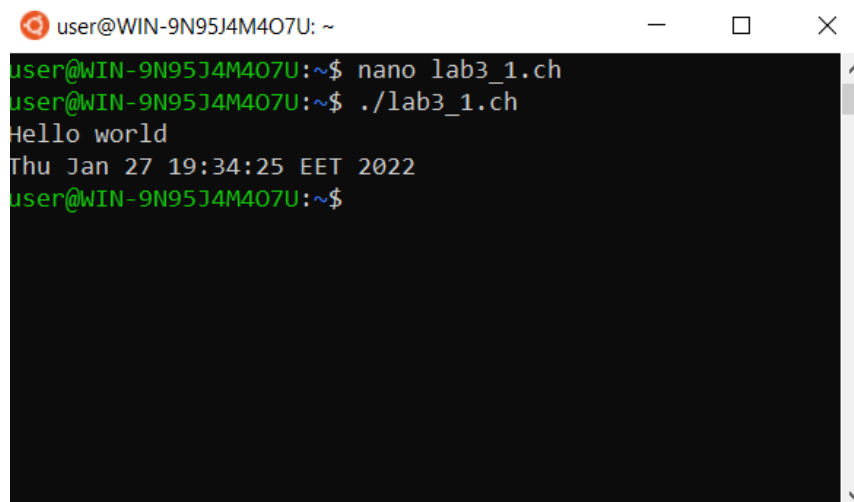


```
user@WIN-9N95J4M407U: ~  
user@WIN-9N95J4M407U:~$ nano  
user@WIN-9N95J4M407U:~$ chmod u+x ./lab3_1.ch  
user@WIN-9N95J4M407U:~$ ./lab3_1.ch  
Hello world  
user@WIN-9N95J4M407U:~$
```

Рис. 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 дозволяють результати

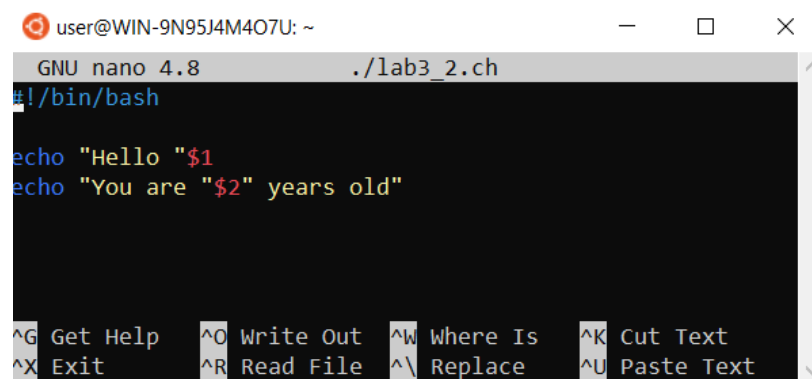
виконання утиліт записувати як значення змінних. Для виводу значення змінної на екран використовується знак \$, наприклад, є змінна string, вона має значення string="Hello". Для того щоб вивести строку Hello на екран використовується команда: echo \$string. А для виводу поточної дати використовуємо наступну 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).



```
user@WIN-9N95J4M4O7U: ~  
user@WIN-9N95J4M4O7U:~$ nano lab3_1.ch  
user@WIN-9N95J4M4O7U:~$ ./lab3_1.ch  
Hello world  
Thu Jan 27 19:34:25 EET 2022  
user@WIN-9N95J4M4O7U:~$
```

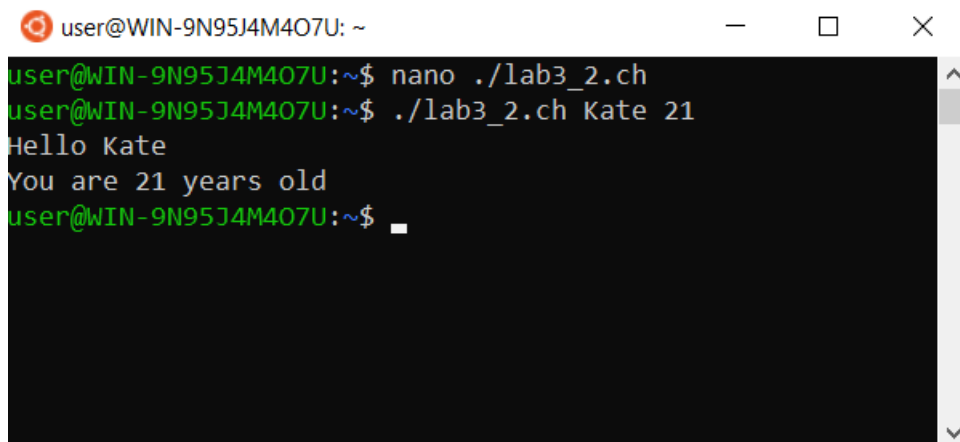
Рис. 3.3 – Результати виконання скрипту lab3\_1.ch

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).



```
GNU nano 4.8 ./lab3_2.ch  
#!/bin/bash  
echo "Hello \"$1"  
echo "You are \"$2\" years old"  
  
^G Get Help    ^O Write Out  ^W Where Is   ^K Cut Text  
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text
```

Рис. 3.4 – Вміст файлу lab3\_2.ch



```
user@WIN-9N95J4M4O7U: ~  
user@WIN-9N95J4M4O7U:~$ nano ./lab3_2.ch  
user@WIN-9N95J4M4O7U:~$ ./lab3_2.ch Kate 21  
Hello Kate  
You are 21 years old  
user@WIN-9N95J4M4O7U:~$
```

Рис. 3.5 – Результати виконання скрипту lab3\_2.ch

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 запиті можна командою:

***read -p "What is your name" name.***

При цьому, `p` – флаг запиті, `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*

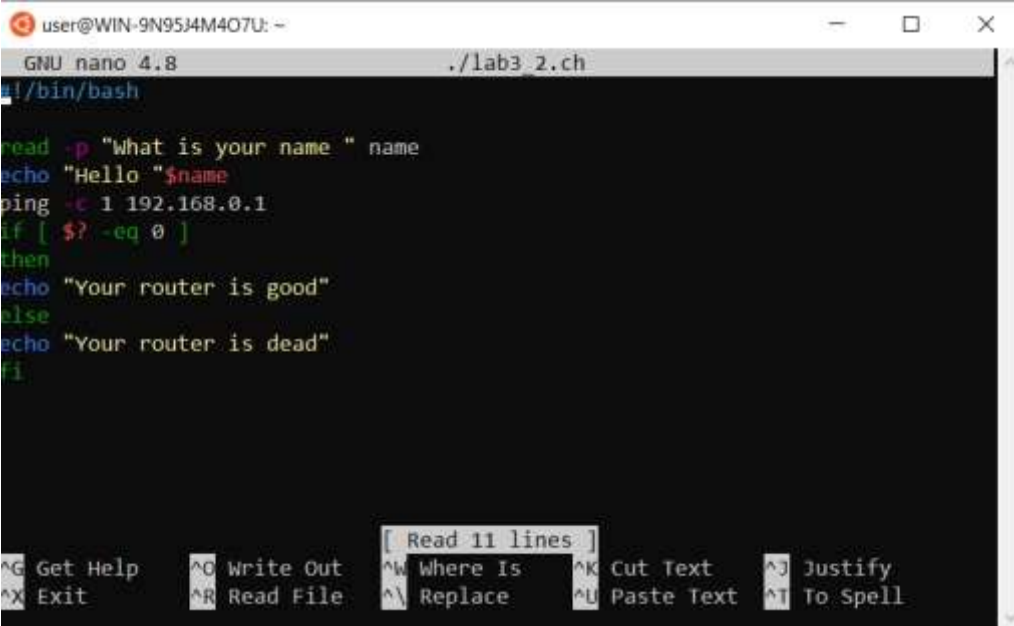
Команда `for` послідовно привласнює змінній значення зі списку та виконує команди, які розташовані між `do` і `done`.

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.

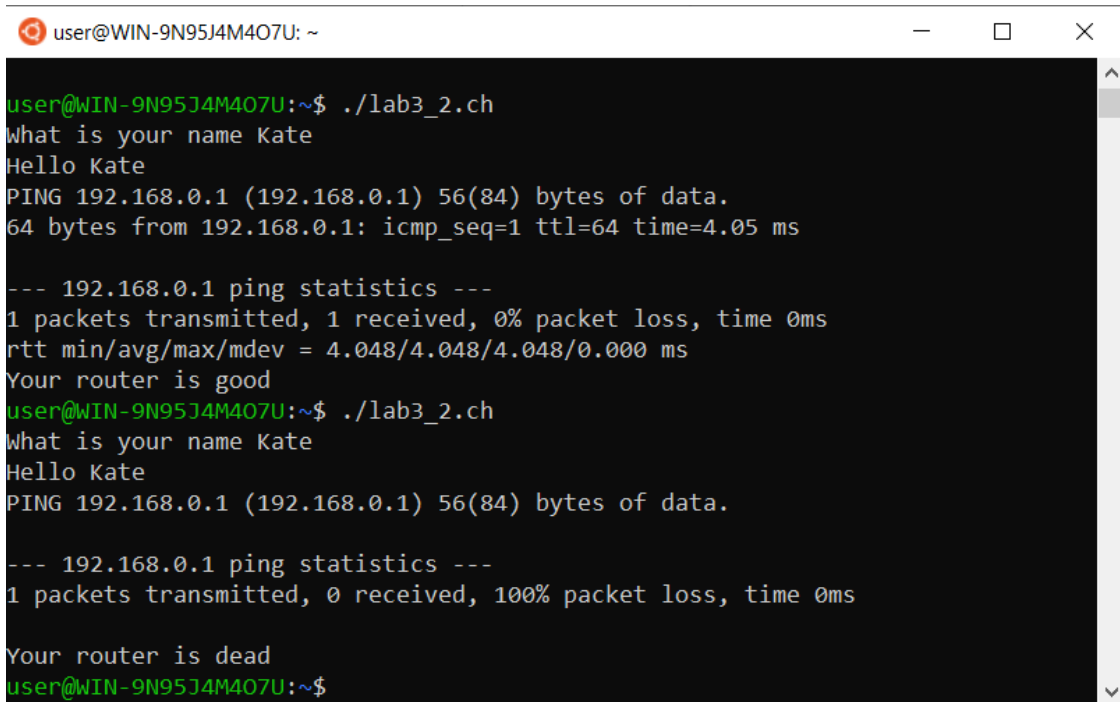
При написанні `bash`-скриптів виникають ситуації, коли необхідно запустити якусь команду та обробити результат її виконання. Наприклад, необхідно запустити команду `ping` та перевірити працездатність бездротового роутера (рис. 3.6).



```
user@WIN-9N95J4M4O7U: ~
GNU nano 4.8 ./lab3_2.ch
#!/bin/bash
read -p "what is your name " name
echo "Hello "$name
ping -c 1 192.168.0.1
if [ $? -eq 0 ]
then
echo "Your router is good"
else
echo "Your router is dead"
fi
^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text      ^J Justify
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^T To Spell
```

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).



```
user@WIN-9N95J4M407U: ~  
user@WIN-9N95J4M407U:~$ ./lab3_2.ch  
What is your name Kate  
Hello Kate  
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.  
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=4.05 ms  
  
--- 192.168.0.1 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 0ms  
rtt min/avg/max/mdev = 4.048/4.048/4.048/0.000 ms  
Your router is good  
user@WIN-9N95J4M407U:~$ ./lab3_2.ch  
What is your name Kate  
Hello Kate  
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.  
  
--- 192.168.0.1 ping statistics ---  
1 packets transmitted, 0 received, 100% packet loss, time 0ms  
Your router is dead  
user@WIN-9N95J4M407U:~$
```

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

Для запуску bash-скриптів в Windows слід використовувати оболонку PowerShell та системну команду wsl (перехід до командної строки 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 рис. 3.8.

```
user@WIN-9N95J4M407U: ~
PS C:\Windows\system32> wsl
user@WIN-9N95J4M407U:/mnt/c/Windows/system32$ cd
user@WIN-9N95J4M407U:~$ tree
.
├── LAB
│   └── new
├── LAB2
│   ├── LAB2
│   ├── TEST
│   ├── lab2_1
│   │   └── lab2_1
│   └── test
├── TEST
│   └── lab
├── lab
│   └── lab
├── lab2_1
│   └── LAB2
│       └── lab2_1
│           └── lab2_1
├── lab3.ch
├── lab3_1.ch
├── lab3_2.ch
├── new
│   └── newfile2.txt
├── newfile1.txt
├── test
├── text1.txt
└── text2.txt

18 directories, 7 files
user@WIN-9N95J4M407U:~$ ./lab3_2.ch
What is your name Kate
Hello Kate
user@WIN-9N95J4M407U:~$
```

Рис. 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 варіанту (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 скрін редактора файлів (nano), скрін 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                       |

|    |  |   |
|----|--|---|
| 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   | Ввести в командному рядку строку символів, створити директорію з таким ім'ям, у разі успішного  |

|    |   |   |
|----|---|---|
|    | лабораторних робіт, для кожної успішної команди створення файлу вивести повідомлення в форматі: «Файл 'ім'я файлу' успішно створено»  | 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  |
| 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 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

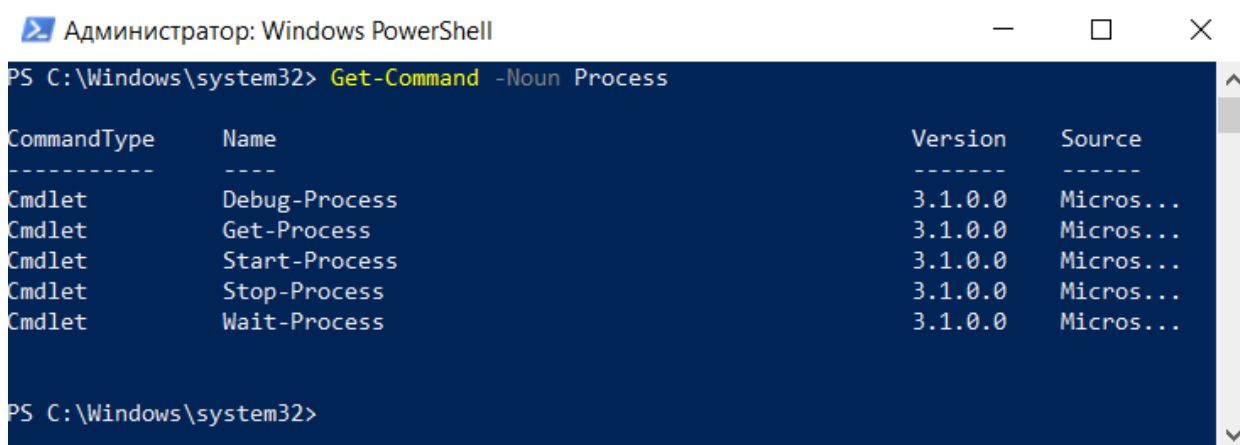
командлетів, то необхідно використовувати символ «;».

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).



```
Администратор: Windows PowerShell
PS C:\Windows\system32> Get-Command -Noun Process

CommandType      Name                Version            Source
-----
Cmdlet            Debug-Process      3.1.0.0           Micros...
Cmdlet            Get-Process        3.1.0.0           Micros...
Cmdlet            Start-Process      3.1.0.0           Micros...
Cmdlet            Stop-Process       3.1.0.0           Micros...
Cmdlet            Wait-Process       3.1.0.0           Micros...

PS C:\Windows\system32>
```

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).

```

Адміністратор: Windows PowerShell
PS C:\Windows\system32> get-process

Handles  NPM(К)  PM(К)   WS(К)   CPU(s)   Id  SI ProcessName
-----  -
157      12      2000    8108    0,06     15276  1 acrotray
398      21      6652    10212   2,98     752    1 AdobeARM
195      12      2764    3816    0,53     12700  0 AppHelperCap
338      20      8224    17012   4,30     16084  1 ApplicationFrameHost
153      8       1768    6468    0,02     372    0 AppVShNotify
136      8       1256    5944    0,06     4872  0 armsvc
588      32      84064   69092   26,64    6628  0 aswEngSrv
1071     29      76352   69296   399,39   7588  0 aswidsagent
1141     39      69340   25040   89,75    4536  0 aswToolsSvc
671      56      74472   18528   462,22   7792  1 atmgr
397      15      23364   22508   2 869,36 15160  0 audiodg
170      10      1828    1132    0,11     6900  0 AvastBrowserCrashHandler
154      9       1856    300     0,05     1032  0 AvastBrowserCrashHandler64
6658     175     298160  239072  724,83   4420  0 AvastSvc
2336     63      50708   46844   315,73   12296  1 AvastUI
646      30      23096   4892    0,83     15724  1 AvastUI
460      26      16164   1872    0,53     16256  1 AvastUI
517      29      18408   7500    3,75     16268  1 AvastUI
107      8       6504    1116    0,44     6272  1 conhost
159      11      6912    10292   12,73    8404  0 conhost
273      15      7324    18976   1,83     23908  1 conhost
828      27      2348    4564    5,06     820    0 csrss
690      22      2984    5716   143,75    940    1 csrss
975      19      5816   22924   16,06    2704  1 ctfmon
207      12      2852    6204    0,56     6936  0 DiagsCap

```

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

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

WS(К) – 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.);

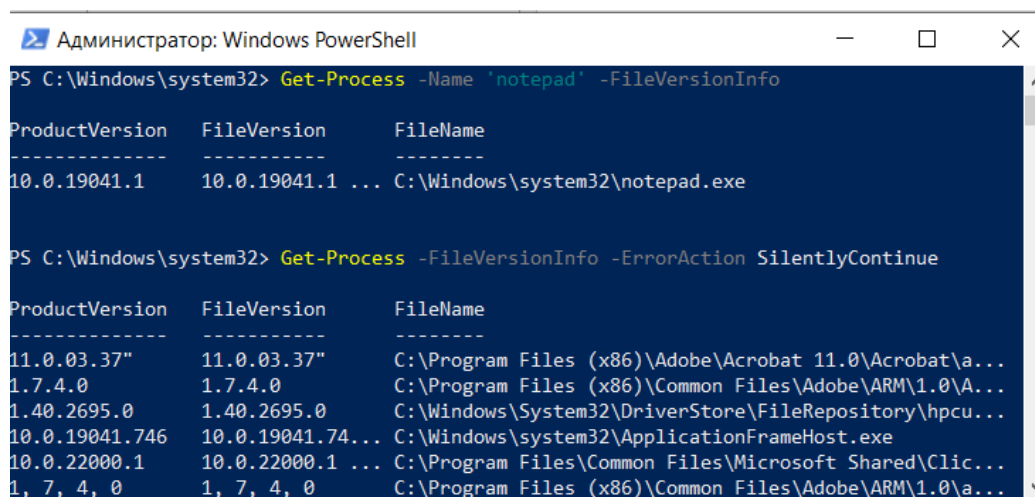
ProcessName – ім'я процесу.

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».



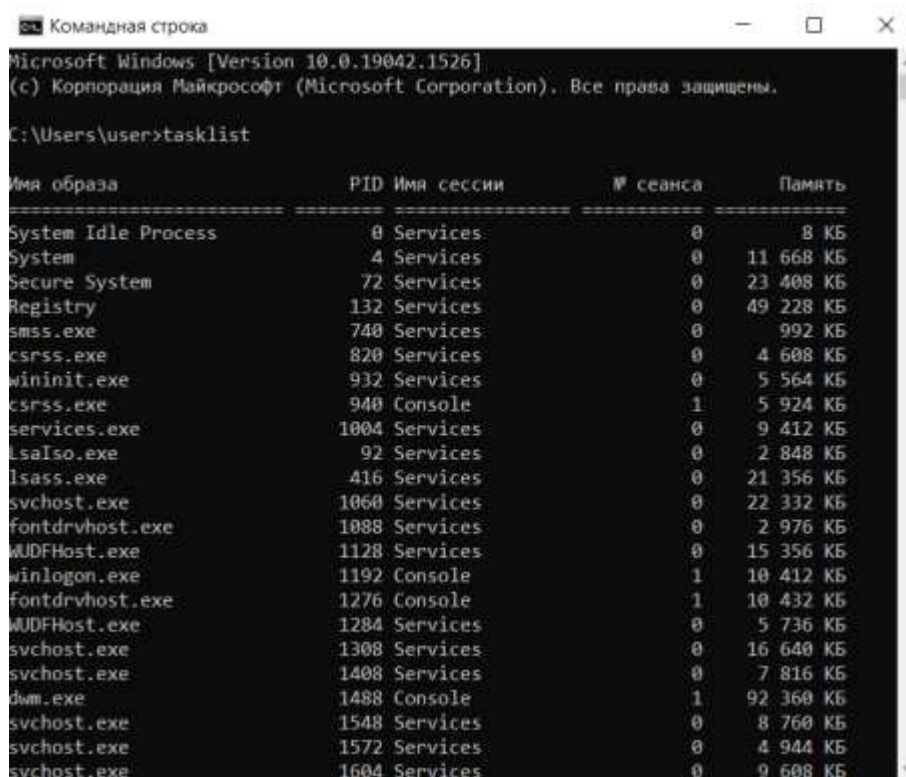
```
Администратор: Windows PowerShell
PS C:\Windows\system32> Get-Process -Name 'notepad' -FileVersionInfo

ProductVersion  FileVersion      FileName
-----
10.0.19041.1    10.0.19041.1 ... C:\Windows\system32\notepad.exe

PS C:\Windows\system32> Get-Process -FileVersionInfo -ErrorAction SilentlyContinue

ProductVersion  FileVersion      FileName
-----
11.0.03.37"     11.0.03.37"     C:\Program Files (x86)\Adobe\Acrobat 11.0\Acrobat\a...
1.7.4.0         1.7.4.0         C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\A...
1.40.2695.0     1.40.2695.0     C:\Windows\System32\DriverStore\FileRepository\hpcu...
10.0.19041.746  10.0.19041.74... C:\Windows\system32\ApplicationFrameHost.exe
10.0.22000.1    10.0.22000.1 ... C:\Program Files\Common Files\Microsoft Shared\Clic...
1, 7, 4, 0     1, 7, 4, 0     C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\A...
```

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



```
Командная строка
Microsoft Windows [Version 10.0.19042.1526]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

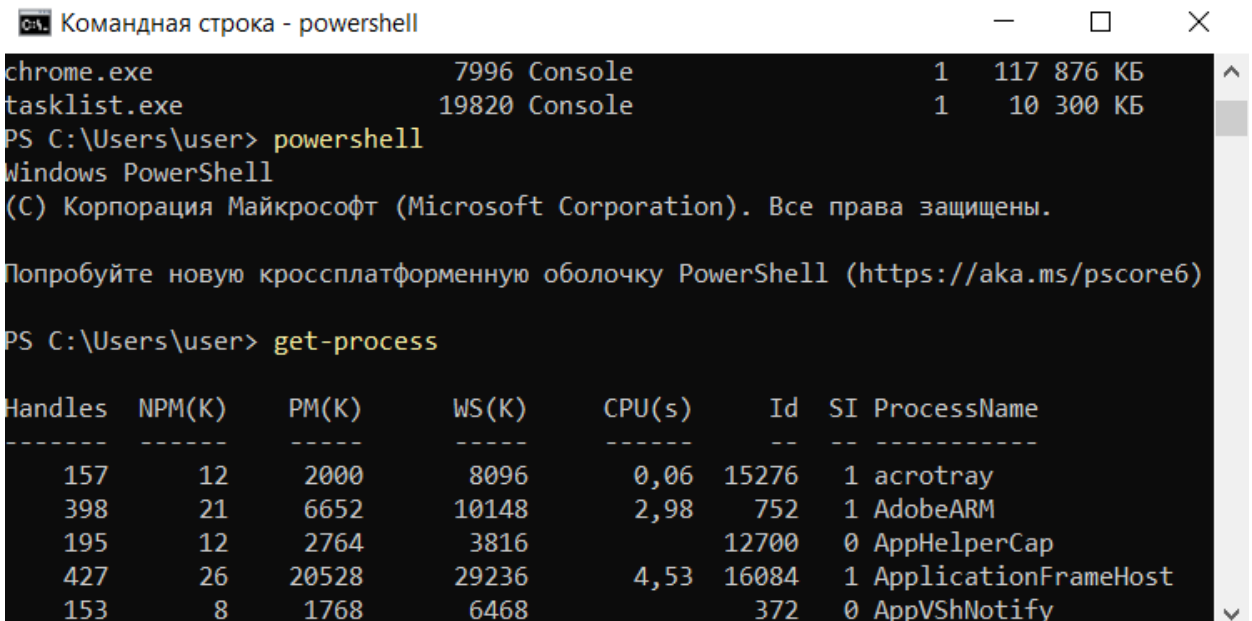
C:\Users\user>tasklist

Имя образа                PID Имя сессии                № сеанса                Память
-----
System Idle Process        0 Services                    0                        8 КБ
System                     4 Services                    0                        11 668 КБ
Secure System              72 Services                    0                        23 408 КБ
Registry                   132 Services                    0                        49 228 КБ
smss.exe                   740 Services                    0                        992 КБ
csrss.exe                  820 Services                    0                        4 608 КБ
wininit.exe                932 Services                    0                        5 564 КБ
csrss.exe                  940 Console                       1                        5 924 КБ
services.exe               1004 Services                    0                        9 412 КБ
lsass.exe                  92 Services                    0                        2 848 КБ
lsass.exe                  416 Services                    0                        21 356 КБ
svchost.exe                1060 Services                    0                        22 332 КБ
fontdrvhost.exe           1088 Services                    0                        2 976 КБ
WUDFHost.exe              1128 Services                    0                        15 356 КБ
winlogon.exe              1192 Console                       1                        10 412 КБ
fontdrvhost.exe           1276 Console                       1                        10 432 КБ
WUDFHost.exe              1284 Services                    0                        5 736 КБ
svchost.exe                1308 Services                    0                        16 640 КБ
svchost.exe                1408 Services                    0                        7 816 КБ
dwm.exe                    1488 Console                       1                        92 360 КБ
svchost.exe                1548 Services                    0                        8 760 КБ
svchost.exe                1572 Services                    0                        4 944 КБ
svchost.exe                1604 Services                    0                        9 608 КБ
```



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

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



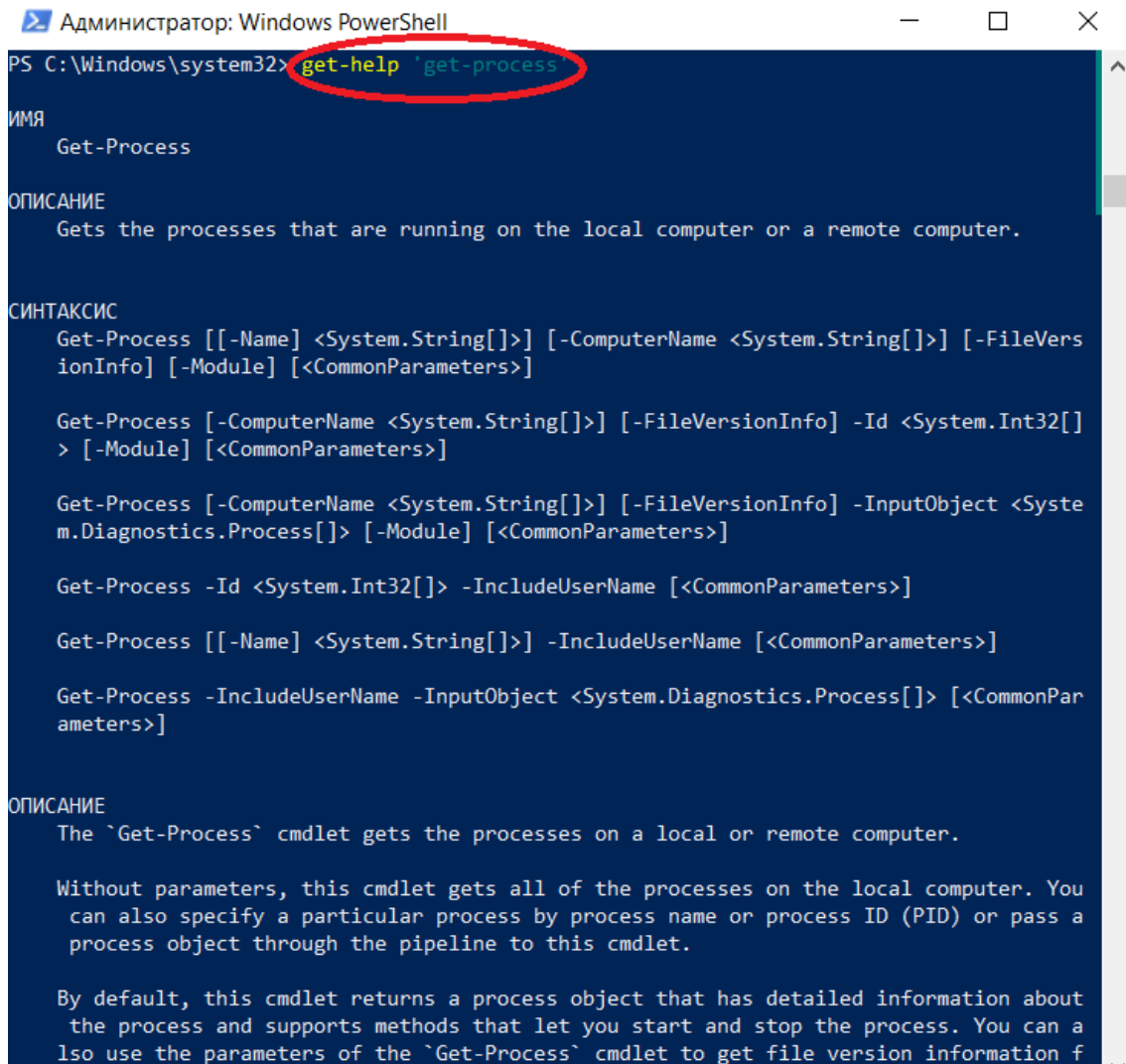
```
Командная строка - powershell
chrome.exe          7996 Console          1  117 876 КБ
tasklist.exe       19820 Console          1   10 300 КБ
PS C:\Users\user> powershell
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/powershell)
PS C:\Users\user> get-process

Handles  NPM(K)  PM(K)  WS(K)  CPU(s)  Id  SI ProcessName
-----  -
157      12     2000   8096   0,06    15276  1 acrotray
398      21     6652  10148   2,98     752  1 AdobeARM
195      12     2764   3816   0,00    12700  0 AppHelperCap
427      26    20528  29236   4,53    16084  1 ApplicationFrameHost
153       8     1768   6468   0,00     372  0 AppVShNotify
```

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).



```
Администратор: Windows PowerShell
PS C:\Windows\system32> get-help 'get-process'

ИМЯ
    Get-Process

ОПИСАНИЕ
    Gets the processes that are running on the local computer or a remote computer.

СИНТАКСИС
    Get-Process [[-Name] <System.String[]>] [-ComputerName <System.String[]>] [-FileVersionInfo] [-Module] [<CommonParameters>]

    Get-Process [-ComputerName <System.String[]>] [-FileVersionInfo] -Id <System.Int32[]> [-Module] [<CommonParameters>]

    Get-Process [-ComputerName <System.String[]>] [-FileVersionInfo] -InputObject <System.Diagnostics.Process[]> [-Module] [<CommonParameters>]

    Get-Process -Id <System.Int32[]> -IncludeUserName [<CommonParameters>]

    Get-Process [[-Name] <System.String[]>] -IncludeUserName [<CommonParameters>]

    Get-Process -IncludeUserName -InputObject <System.Diagnostics.Process[]> [<CommonParameters>]

ОПИСАНИЕ
    The `Get-Process` cmdlet gets the processes on a local or remote computer.

    Without parameters, this cmdlet gets all of the processes on the local computer. You can also specify a particular process by process name or process ID (PID) or pass a process object through the pipeline to this cmdlet.

    By default, this cmdlet returns a process object that has detailed information about the process and supports methods that let you start and stop the process. You can also use the parameters of the `Get-Process` cmdlet to get file version information f
```

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

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

```
Выбрать Администратор: Windows PowerShell
PS C:\Windows\system32> get-help Start-Process -examples
ИМЯ
    Start-Process
ОПИСАНИЕ
    Starts one or more processes on the local computer.

----- Example 1: Start a process that uses default values -----
Start-Process -FilePath "sort.exe"

----- Example 2: Print a text file -----
Start-Process -FilePath "myfile.txt" -WorkingDirectory "C:\PS-Test" -Verb Print

---- Example 3: Start a process to sort items to a new file ----
$processOptions = @{
    FilePath = "sort.exe"
    RedirectStandardInput = "TestSort.txt"
    RedirectStandardOutput = "Sorted.txt"
    RedirectStandardError = "SortError.txt"
    UseNewEnvironment = $true
}
Start-Process @processOptions

This example uses splatting to pass parameters to the cmdlet. For more information,
see about_Splatting (../microsoft.powershell.core/about/about_splatting.md).
----- Example 4: Start a process in a maximized window -----
Start-Process -FilePath "notepad" -Wait -WindowStyle Maximized
```

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

Завершення будь-якого процесу здійснюється командлетом `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.

```

Администратор: Windows PowerShell
PS C:\Windows\system32> get-help Stop-Process -examples
ИМЯ
    Stop-Process
ОПИСАНИЕ
    Stops one or more running processes.

----- Example 1: Stop all instances of a process -----
PS C:\> Stop-Process -Name "notepad"

This command stops all instances of the Notepad process on the computer. Each instance of Notepad runs in its own process. It uses the Name parameter to specify the processes, all of which have the same name. If you were to use the Id parameter to stop the same processes, you would have to list the process IDs of each instance of Notepad.

----- Example 2: Stop a specific instance of a process -----
PS C:\> Stop-Process -Id 3952 -Confirm -PassThru

Confirm
Are you sure you want to perform this action?
Performing operation "Stop-Process" on Target "notepad (3952)".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help
(default is "Y"):y
Handles      NPM(K)      PM(K)      WS(K) VM(M)      CPU(s)      Id ProcessName
-----
41           2          996        3212   31          3952 notepad

This command stops a particular instance of the Notepad process. It uses the process ID, 3952, to identify the process. The Confirm parameter directs PowerShell to prompt you before it stops the process. Because the prompt includes the process name in addition to its ID, this is best practice. The PassThru parameter passes the process object to the formatter for display. Without this parameter, there would be no display after a `Stop-Process` command.

```

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.

```

user@WIN-9N95J4M4O7U: ~
user@WIN-9N95J4M4O7U:~$ ps
  PID TTY          TIME CMD
    8 tty1      00:00:00 bash
   79 tty1      00:00:00 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. 4.9 – Результати роботи команди ps (з різними аргументами)

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

```

user@WIN-9N95J4M4O7U: ~
top - 16:43:24 up 2 min, 0 users, load average: 0.52, 0.58, 0.59
Tasks:  4 total,  1 running,  3 sleeping,  0 stopped,  0 zombie
%Cpu(s):  2.3 us,  2.5 sy,  0.0 ni, 94.9 id,  0.0 wa,  0.3 hi,  0.0 si,  0.0 st
MiB Mem :  7948.6 total,  1377.4 free,  6347.2 used,  224.0 buff/cache
MiB Swap: 24576.0 total, 24171.1 free,  404.9 used. 1470.8 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
    1 root       20   0   8940    328   284  S   0.0   0.0   0:00.03  init
    7 root       20   0   8940    228   184  S   0.0   0.0   0:00.00  init
    8 user       20   0  18076   3604  3504  S   0.0   0.0   0:00.06  bash
   73 user       20   0  18920   2152  1528  R   0.0   0.0   0:00.12  top

```

Рис. 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 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. 4.11).

```
user@WIN-9N95J4M407U: ~
WIN-9N95J4M407U - IP 192.168.0.100/ Uptime: 0:22:29
CPU [ 4.9%] CPU \ 4.9% MEM - 83.7% SWAP - 2.9% LOAD 8-core
MEM [ 83.7%] user: 1.7% total: 7.76G total: 24.0G 1 min: 0.52
SWAP [ 2.9%] system: 3.0% used: 6.50G used: 720M 5 min: 0.58
idle: 95.1% free: 1.26G free: 23.3G 15 min: 0.59

NETWORK Rx/s Tx/s TASKS 4 (5 thr), 1 run, 3 slp, 0 oth sorted automatically
lo 0b 0b
wifi0 0b 0b
DefaultGateway 35ms

CPU% MEM% VIRT RES PID USER TIME+ THR NI S R/s W/s Command
1.0 0.6 433M 48.3M 1196 user 0:04 1 0 R ? ? /usr/bin
0.0 0.0 17.7M 3.51M 1183 user 0:00 1 0 S ? ? -bash
0.0 0.0 8.73M 328K 1 root 0:00 2 0 S ? ? //init
0.0 0.0 8.73M 228K 1182 root 0:00 1 0 S ? ? //init

High memory consumption
2022-03-14 17:03:23 EEST 2022-03-14 17:00:43 (ongoing) - MEM (85.0)
```

Рис. 4.11 – Результати роботи команди glances

Для управління процесами в Linux використовують наступні команди:

**kill** – посилає процесу сигнал завершення роботи;

**rkill** – 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. 4.12.

```
user@WIN-9N95J4M407U: ~  
user@WIN-9N95J4M407U:~$ ps -eF  
UID      PID  PPID  C   SZ   RSS  PSR  STIME TTY      TIME  CMD  
root      1    0    0  2235  328   0  16:40 ?        00:00:00 /init  
root     1182    1    0  2235  228   0  17:00 tty1    00:00:00 /init  
user     1183  1182  0  4519 3632   0  17:00 tty1    00:00:00 -bash  
user     1768  1183  0  4666 1892   0  17:34 tty1    00:00:00 ps -eF  
user@WIN-9N95J4M407U:~$ pgrep -u user  
1183  
user@WIN-9N95J4M407U:~$ pgrep -u root  
1  
1182  
user@WIN-9N95J4M407U:~$ kill 1183  
user@WIN-9N95J4M407U:~$
```

Рис. 4.12 – Результати роботи команд управління процесами

### ***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 `rgrep`? Наведіть приклади використання.

## 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 ВНУ, 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](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>