Міністерство освіти і науки України
Луцький національний технічний університет

СИСТЕМНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Методичні вказівки до лабораторних занять
для здобувачів першого (бакалаврського) рівня вищої освіти
освітньої програми «Комп'ютерна інженерія»
галузь знань 12 Інформаційні технології
спеціальності 123 Комп’ютерна інженерія
денної та заочної форм навчання

Луцьк 2025

УДК 004.75(07)
С-94

Рекомендовано до видання вченою радою факультету КІТ ЛНТУ,
протокол № _____ від «___ » _____ 20 25 року.

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри комп'ютерної інженерії та безпеки ЛНТУ, протокол № _____ від « _____ » _____ 20 25 року.

Завідувач кафедри КІБ _____ Тарас ТЕРЛЕЦЬКИЙ

Укладачі: _____ Оксана МІСКЕВИЧ, старший викладач, кафедри комп'ютерної інженерії та безпеки ЛНТУ

Рецензент: _____ Віктор КОШЕЛЮК, кандидат технічних наук, доцент кафедри комп'ютерних наук ЛНТУ

Відповідальний за випуск: _____ Тарас ТЕРЛЕЦЬКИЙ, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та безпеки ЛНТУ

С-94 Системне програмне забезпечення. Методичні вказівки до лабораторних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. Оксана МІСКЕВИЧ. Луцьк : ЛНТУ, 2025. 139 с

Методичне видання до лабораторних занять з дисципліни «Системне програмне забезпечення»: складене відповідно до діючої програми курсу.

Призначене для здобувачів вищої освіти спеціальності 123 Комп'ютерна інженерія освітньої програми «Комп'ютерна інженерія».

О.І.Міскевич 2025

ЗМІСТ

ВСТУП

Освітній компонент «Системне програмне забезпечення» є основоположним для підготовки фахівців в галузі інформаційних технологій. Вивчення цієї дисципліни дає здобувачам знання та практичні навички з впровадження та обслуговування системного програмного забезпечення.

Мета навчальної дисципліни полягає в тому, щоб забезпечити студентів глибокими теоретичними знаннями та практичними навичками розкриття сучасних наукових концепцій, понять, методів та технологій проектування та реалізації системного програмного забезпечення на підставі засвоєння алгоритмів системних служб, обробки інформації різних типів даних, вивчення принципів реалізації системного програмного забезпечення операційних середовищ та систем з використанням сучасних технологій програмування.

Програма курсу розподілена на два змістовні модулі, що забезпечують поступове поглиблення знань від основ програмного забезпечення до засобів синхронізації та керування пам'яттю в файлових системах Windows та Unix.

Лабораторні роботи з дисципліни «Системне програмне забезпечення» спрямовані на набуття практичних навичок з функціонування операційної системи в консольному режимі при роботі з файлами та каталогами; вивчення засобів та інструментів роботи з ОС Microsoft Windows; командна оболонка Windows Power Shell; робота з командами ОС LINUX; процеси в ОС UNIX і керування ними.

Метою виконання лабораторних робіт є набуття знань Microsoft Windows, Windows Power Shell, ОС LINUX, ОС UNIX.

Ці методичні вказівки охоплюють низку тем, передбачених робочою програмою освітнього компонента «Системне програмне забезпечення». Кожна лабораторна робота містить перелік рекомендованої літератури для опрацювання, теоретичний матеріал, приклади реалізації з детальним описом, завдання для виконання та перелік запитань для самоперевірки.

Дисципліна складається з двох змістових модулів. В першому змістовому модулі розглядається керування ресурсами операційних систем. В другому змістовому модулі розглядаються: автоматизація системних завдань та розробка СПЗ.

На лабораторних роботах кожен здобувач опрацьовує теоретичні відомості, виконуючи при цьому наведені в них приклади, а потім виконує завдання, оформляє звіт до лабораторної роботи та дає відповіді на питання викладача (усно, з переліку контрольних питань). Виконання наступних робіт може опиратися на результати попередніх, тому не рекомендовано їх виконувати в довільному порядку.

Оцінювання лабораторних робіт здійснюється за критеріями:

1. Теоретична підготовка (розуміння основних понять та вміння пояснити теоретичні концепції).

2. Практичне виконання завдань (повнота та точність).

3. Технічна реалізація (коректна організація коду).

4. Документація (наявність коментарів до програмного коду', якість оформлення звіту – чіткий та детальний звіт, в якому студент пояснює, які завдання виконувалися, які труднощі виникали, як були вирішені проблеми. Звіт має бути структурованим і зрозумілим).

5. Дотримання термінів – оцінюється, наскільки студент дотримується дедлайнів і вчасно здає лабораторну роботу.

6. Вміння захистити роботу – студент повинен продемонструвати здатність чітко і впевнено відповісти на питання щодо виконаних завдань, пояснити вибір рішень і відповісти на контрольні запитання.

7. Вміння виправляти помилки – наскільки студент здатний виявляти і виправляти помилки в програмному коді.

Для отримання максимального балу за лабораторне роботу заняття оцінюється ініціативність та креативність, наприклад чи застосовуються додаткові інструменти або методи, які не входять до завдання, але можуть покращити результат.

Ці критерії дають змогу об'єктивно оцінити навички студента, його здатність до самостійної роботи, розуміння теорії та застосування знань на практиці.

ЛАБОРАТОРНЕ ЗАНЯТТЯ №1
СТРУКТУРА ФАЙЛОВОЇ СИСТЕМИ WINDOWS. СЕРВІСНІ КОМАНДИ, РОБОТА З ФАЙЛАМИ, КАТАЛОГАМИ, ДЕРЕВАМИ

Мета роботи −ознайомитися з утилітою cmd.exe; вивчити команди для роботи з файлами і каталогами; навчитися створювати ієрархічну структуру через інтерфейс командного рядка Windows.

Література: [1, 2, 3, 7].

Теоретичні відомості

В консолі ОС Windows всі команди поділяються на дві групи:

– внутрішні (internal) команди містяться в командному процесорі – файл command.com

– зовнішні(external) кожна команда окремий виконуючий файл

Імена файлів реєструються на дисках у каталогах (чи директоріях). Каталог – це спеціальне місце на дискові, де зберігаються імена файлів, відомості про розмір файла, час його створення, атрибути (властивості) файла тощо. Якщо каталог зберігає ім'я якогось файла – можна сказати, що даний файл знаходиться у цьому каталозі. В кожному каталозі може бути багато файлів, але в одному каталозі не може міститися два і більше файлів з однаковими іменами.

Кореневий каталог. Кожен диск обов'язково має принаймні один головний (кореневий) каталог. В ньому реєструються файли та підкаталоги (каталоги 1-го рівня). В каталогах 1-го рівня реєструються файли та підкаталоги 2-го рівня і т.д. Таким чином утворюється ієрархічне дерево каталогів на дискові.

Підкаталоги і надкаталоги. Усі каталоги (крім кореневого) є файлами спеціального вигляду. Кожен каталог має своє ім'я, і він може бути записаний у будь-якому іншому каталозі. Якщо каталог А записаний у каталозі В, то це означає, що А – підкаталог каталога В, а В – надкаталог для каталога А.

Імена каталогів. Вимоги до імен каталогів такі самі, як і до імен файлів. Поточний каталог. Каталог, з файлами якого у даний час працює користувач, називається поточним каталогом.

Команди для роботи з дисками та каталогами

На початку вивчення низки консольних команд надзвичайно важливою є команда help. Якщо ввести цю команду просто, одним словом, то результатом буде виведена інформація про усі команди оболонки DOS. Якщо ж ввести команду поруч з іншою командою, то результатом буде детальна інформація, в тому числі про ключі, способи уведення зазначеної команди.

Запуск консолі здійснюється або гарячими клавішами Win + R, або через головне меню Windows (Пуск-Виконати-cmd).

За замовчуванням, командний процесор та файли зовнішніх команд знаходяться на системному диску, де інстальовано ОС в каталозі Windows\System32.

Наприклад, після введення команди help отримуємо результат (рис 1.1) :

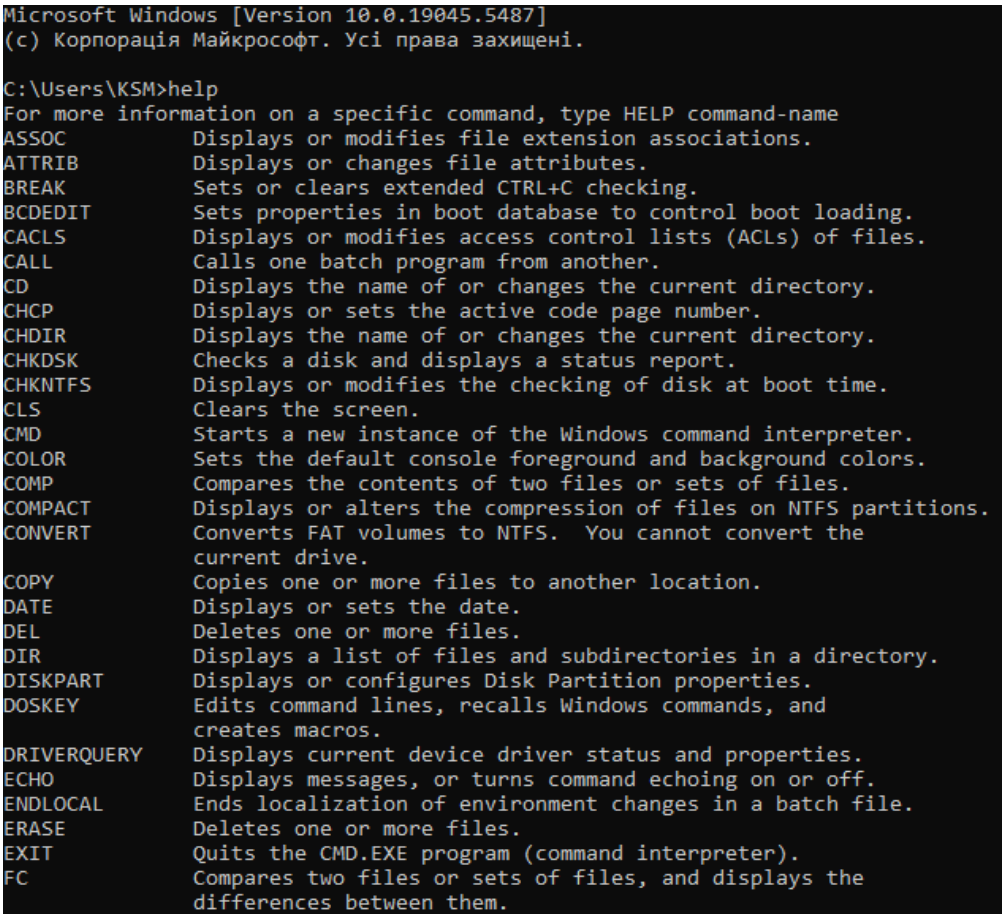


Рисунок 1.1 – Команда help

Після введення команди help copy результат (рис 1.2):

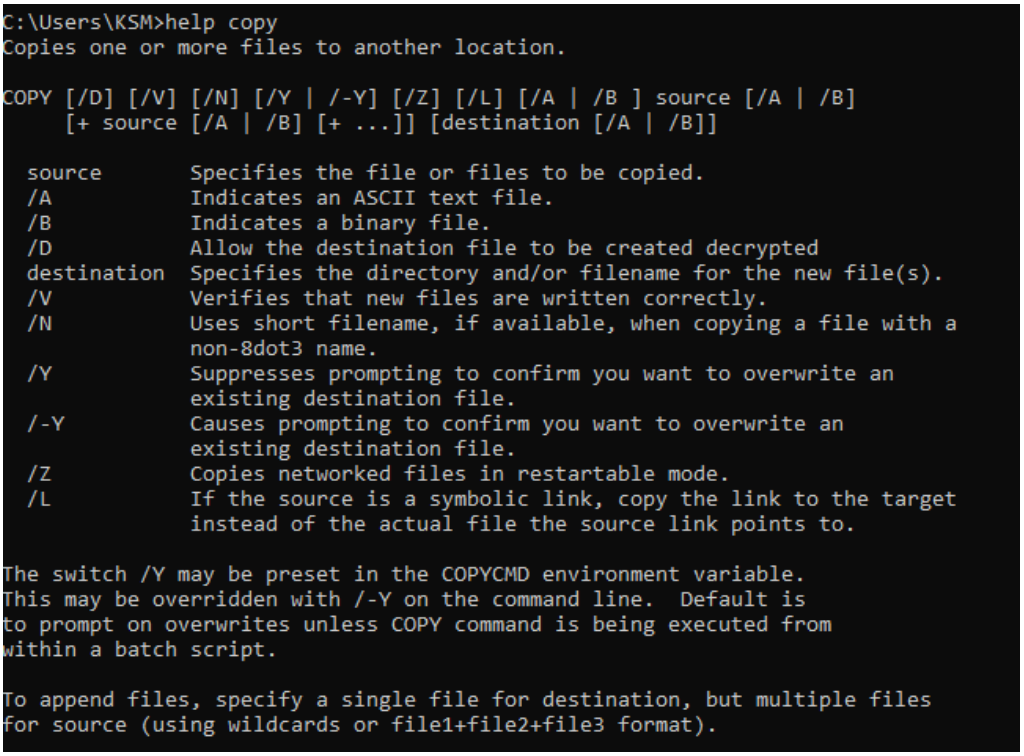


Рисунок 1.2 – Результат help copy

Команда перегляду каталогу

Дана команда призначена для перегляду вмісту каталога, тобто списку файлів та підкаталогів, що містяться в ньому.

Формат команди: dir [шлях] [ключі].

Параметри, які знаходяться в квадратних дужках не обов’язкові і при заданні команди можуть не використовуватись. Символи “[]” є умовними позначеннями і в командах ніколи не використовуються.

Ключі, які використовуються в команді перегляду каталогу:

/p–перегляд каталогу по екранних сторінках;

/w–перегляд списку файлів та підкаталогів в 5–ти колонках;

/s–перегляд каталогу та його підкаталогів всіх підрівнів;

/b–перегляд тільки списку файлів та підкаталогів, без вказання розміру, дати, часу створення та інших характеристик;

/q–додатково відображає власника файлів (людину яка його створила);

/4–відображає рік в даті файлів в чотирьох символьному форматі.

Команда створення каталогу має наступний формат: md [шлях\]ім’я каталогу або mkdir [шлях\] ім’я каталогу.

Команди зміни активності каталогу (переходу на інший каталог).

При роботі з деревом каталогів дуже важливим є переміщення по дереву каталогів, тобто зміна активності каталогу.

Формат команди – cd [/d назва диску] шлях або chdir [/d назва диску] шлях.

Для того, щоб у ієрархії каталогів перейти на рівень вище, використовується команда cd..

Для переходу в кореневий каталог використовуємо команду cd\

Команда знищення каталогу.

При роботі на ПК, з часом виникає необхідність знищувати інформацію (файли, каталоги) яка користувачу стала не потрібною. Інакше носій інформації через певний час повністю б заповнився. Для знищення каталогів в консольному режимі використовують команду:

rd[шлях\] ім’я каталогу[ключі] або rmdir[шлях\] ім’я каталогу[ключі]

Коли знищуємо каталог, що знаходиться в активному, то шлях до нього не вказується. Не можливо знищити каталог, якщо він є активним.

В команді rd можна використовувати такі ключі:

/s– знищити каталог з всіма його файлами та підкаталогами;

/q– використовують разом з попереднім ключем і забороняє вивід перепитування на знищення гілки дерева каталогів. В ранніх версіях Windows та в DOS ключ /s не використовувався, для знищення гілок дерева каталогів була присутня окрема зовнішня команда del tree.

Команда створення текстового файлу – copy con ім’я файла Enter.

Задавши цю команду, створюємо текст файлу, а в кінці натискуємо комбінацію клавіш Crtl+Z або F6 і тоді Enter.

Команда перегляду вмісту файлу – ця команда переглядає вміст файлу будь–якого типу в стандартних кодах. Якщо це звичайний текстовий файл, то можна буде прочитати текст записаний в ньому. В інших випадках буде виведено текст в машинних кодах.

Формат команди: type [шлях\] ім’я файлу Enter

Якщо переглядаємо файл в активному каталозі, то шлях до нього вказувати не потрібно.

Для знищення файлу в консолі існує спеціальна внутрішня команда, яка задається в такому форматі – del [шлях\] ім’я файлу [/ключі] Enter; erase [шлях\] ім’я файлу [/ключі] Enter.

В команді del можна використовувати такі ключі:

/p– проводиться перепитування на знищення кожного файлу;

/f– дозволяє знищення файла з атрибутами “тільки для читання” та “системний”. Якщо цей ключ не вказано, то при спробі знищення таких файлів буде повідомлення ACCESS DNEID (доступ заборонено) ;

/s– знищувати задані файли у всіх підкаталогах вказаного каталогу;

/q– не виводити перепитування при знищенні груп файлів у підкаталогах;

/a– знищувати файли тільки з вказаними атрибутами.

Наприклад, ключ /ah– дозволяє знищувати невидимі файли.

Команда виведення дерева каталогів. Для відтворення ієрархії створених файлів використовуємо команду: tree

Крім вище вказаних, у консолі використовуються команди:

– cls – очищення екрану. За цією командою відбувається очищення екрану від тексту інформації, при цьому залишається лише командний рядок, який розміщується у верхній частині екрану;

– date – видача інформації про системну дату та її зміна. Виводиться поточна дата і пропонується записати нову дату в форматі “день–місяць–рік” У випадку коли дату не потрібно міняти натискаємо Enter;

– time – видача інформації про системний час та його зміна. При виконанні цієї команди виводиться поточний час. При потребі зміни поточного часу вводимо новий час в форматі “гг.хх.сс”;

– ver – видає версію ОС;

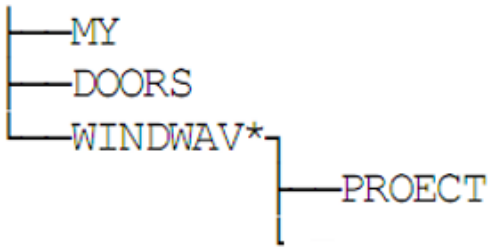
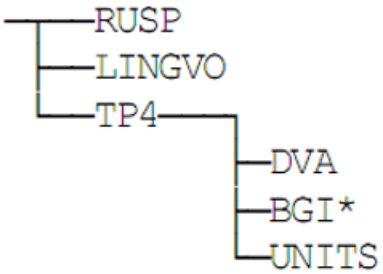
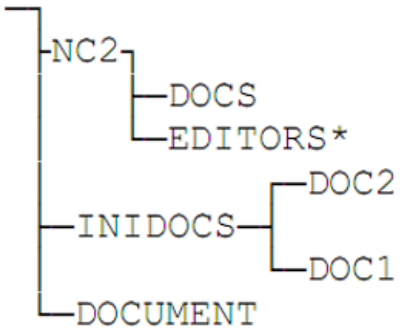
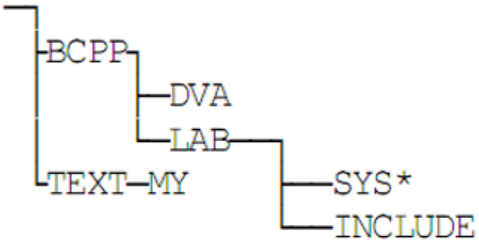
– vol – видає мітку та серійні номери диску. Мітка диску – це набір латинських літер або цифр і символів (що обов'язково починаються з літери), які задають дискове умовне ім’я;

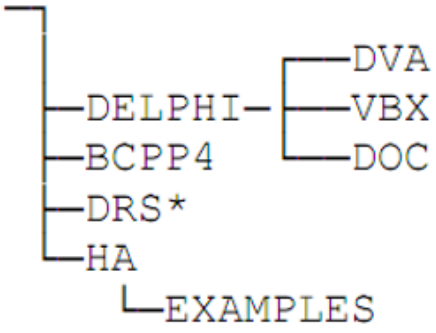
– verify on/off – вмикає (on) або вимикає (off) верифікацію інформації. Верифікація – це порівняння копій з оригіналом та ідентичність при копіюванні (переміщенні). Проводиться для підвищення надійності при записані інформації на диск, при виконанні операції копіювання (переміщення).

Індивідуальні завдання до лабораторного заняття №1

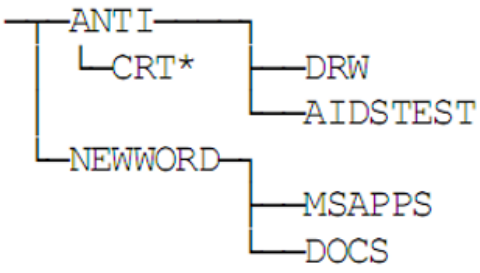
1. Створити каталог зі своїм прізвищем.
2. У папці з прізвищем створити файл з власним ім’ям (наприклад, adr.txt) з адресою проживання, інший файл – inform.txt з анкетними даними: ПІБ, група, дата народження.
3. Перейменуйте inform.txt на file2.txt, ввівши інформацію про свою спеціальність.
4. Перегляньте вміст файлу.
5. Перемістіть файл в корінь диску.
6. Згідно варіанту (номер у журналі групи) створити дерево каталогів, скрин консолі при виконанні даного завдання подати звітом з лабораторного заняття (Увага! При створенні каталогів у кожному варіанті є позначений зірочкою *, створюйте цей каталог без зірочки! Цей символ нам потрібен для наступної роботи)

Таблиця 1.1 – Варіанти завдань

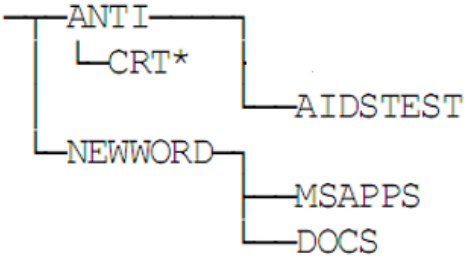
 1	 2
 3	 4



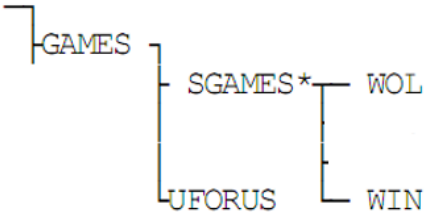
5



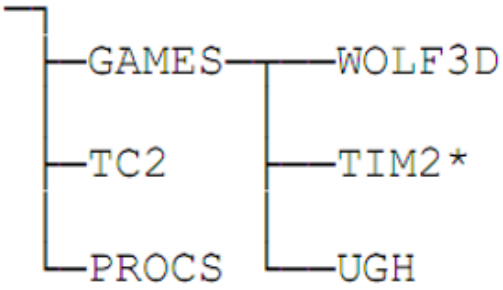
6



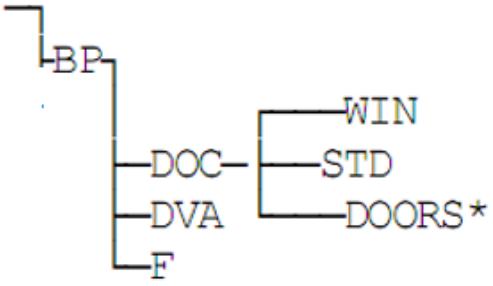
7



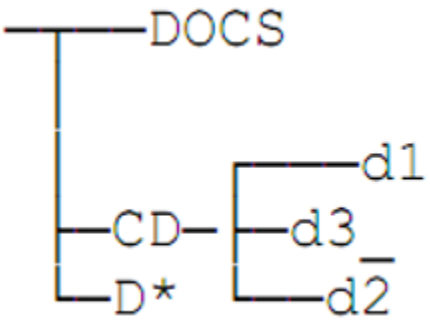
8



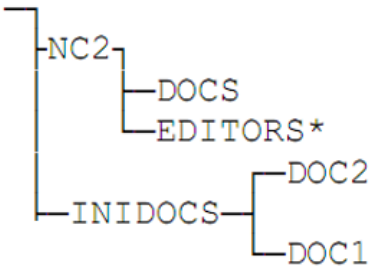
9



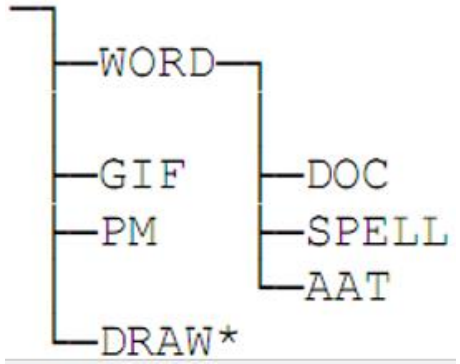
10



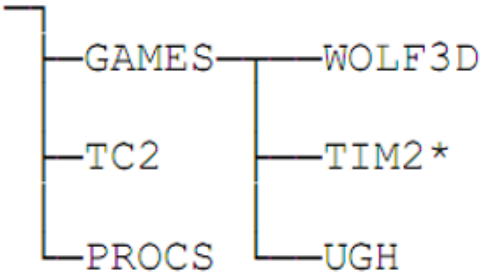
11



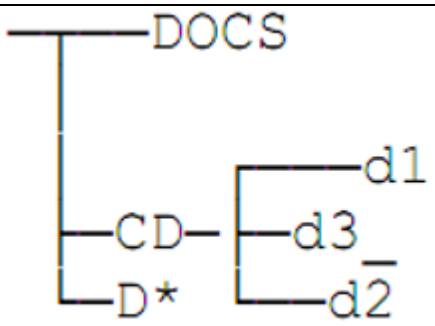
12



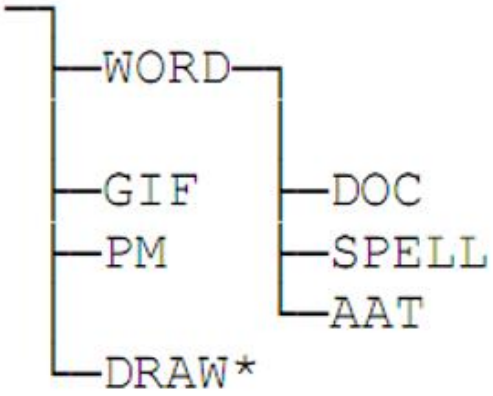
13



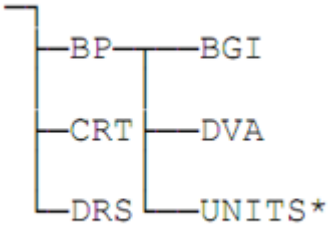
14



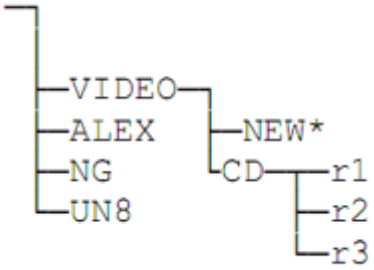
15



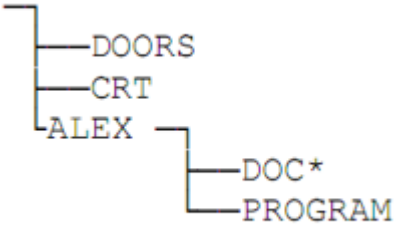
16



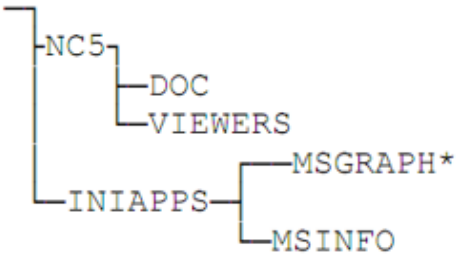
17



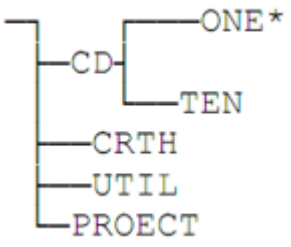
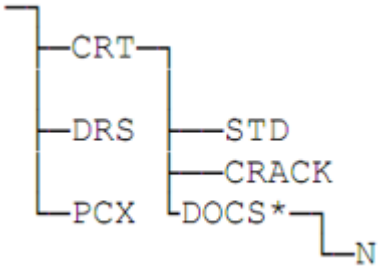
18



19



20

	
21	22

Вимоги до звіту

Звіт з лабораторного заняття повинен складатися з:

1. Титульний лист. (ПІБ, група, варіант)
2. Мета роботи.
3. Розв’язання зі скринами консолі при виконанні даного завдання

Контрольні питання

1. Як запустити командний рядок?
2. Як здійснюється перейменування файлів і папок?
3. Як переглянути конфігурація системних пристроїв?
4. Як відкрити і переглянути властивості миші, клавіатури та принтера з командного рядка?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №2
КОМАНДИ КОПІЮВАННЯ, ПЕРЕМІЩЕННЯ, ОБ'ЄДНАННЯ. МАСКИ ВВОДУ

Мета роботи – вивчити особливості копіювання та переміщення файлів і каталогів у командному рядку файлової системи Windows; навчитися оперувати масками вводу для здійснення контекстного пошуку в ОС.

Література: [1, 2, 3, 5]

Теоретичні відомості

В консольних командах можна використовувати спеціальні символи, що використовуються в парі із стандартними внутрішніми командами.

Найбільш поширеною є команда–символ перенаправлення вводу інформації.

Наприклад, коли задати команду: dir /p Enter – відбудеться звичайний перегляд каталогу Windows, а коли задати команду: dir /p> a.txt Enter – то вивід на екран результатів роботи не відбудеться, а список файлів і каталогів буде записано у файлі a.txt.

Причому, якщо на диску раніше уже був файл з таким іменем, то вся інформація з нього знищується, запишуться тільки результати виконання останньої команди. Коли на диску такого файлу ще не існувало, то він створиться.

Досить часто символ перенаправлення інформації використовують для перенаправлення результатів роботи команди на зовнішні периферійні пристрої, наприклад принтер: type robota.txt>prn.

Команди, що наведені вище, виводять результати перегляду каталогу (файлу) не на екран, а переправляють на принтер.

Не менш важливими командами, які підтримуються в консолі є внутрішні команди для роботи з файлами. Основною з них є, звичайно команда завантаження файлу на виконання, оскільки без використання цієї команди неможливо уявити жоден сеанс роботи ПК. Команда завантаження файлу на виконання має наступний формат:

[шлях\]ім'я файлу[.розширення][ключі]Enter

Якщо завантажуємо файл з активного каталогу, то »шлях» можна пропустити. «Шлях» також не вказують, коли файл, що завантажується знаходиться в кореневому

каталозі системного диску, або шлях до його каталогу записано в спеціальній команді файлу autexec.bat.

Розширення при завантаженні файлу вказувати не обов’язково і тому його практично завжди ігнорують. Варто також нагадати, що на ПК є лише три типи файлів, які можна запустити на виконання з консолі, це файли з розширенням: exe.com .bat (виконавчі та пакетні файли).

Маски вводу

В консолях використовують два види символів шаблону (масок), це

“*”– змінює будь-яку кількість символів

“?”– змінює не більше однієї літери.

Тут варто нагадати, що кожен файл у файловій системі має власне ім'я і розширення. Власне ім'я від розширення відділяється крапкою. Наприклад, datum.docx, command.com тощо.

Приклади запису шаблонів файлів через маски вводу: a*.exe – так записуються усі файли, імена яких починаються з літери «a», і з розширенням exe; b?.* – так записуються усі файли, імена які починаються з літери b і містять не більше двох літер у власному імені, з будь яким розширенням; ????.*– всі файли, імена яких містять не більше чотирьох літер, з довільним розширенням.

Якщо у команді del не вказати імені файлів, то це вважається шаблоном *.*, точніше знищення всіх файлів. У випадку, коли задано шаблон *.*, або не вказано ім'я взагалі, команда перепитує.

Команди копіювання файлів та груп файлів. Однією з основних найчастіше використовуваних команд консолі є команда копіювання файлів та груп файлів.

Формат команди: copy [шлях\] ім’я файла[шлях файла\] нове ім'я[ключі] Enter

шлях звідки– це шлях до каталогу, з якого потрібно копіювати файли, причому якщо файли копіюються з активного каталогу, то “шлях звідки” не вказується; нове ім'я– це ім'я яке буде мати файл після копіювання. Якщо воно не змінюється, то нове ім’я не вказується.

Якщо при копіюванні в деякий каталог там уже є файл з таким іменем, то ОС встановлює перепитування (overwrite) чи переписувати новий файл поверх існуючого (Owerrite proba.txt (Yes/No/All)?) пропонуючи:

Y(Yes) – файл перепишеться поверх існуючого

N(No) – пропустити копіювання даного файла

A(All) – переписати даний файл і всі наступні попередження

В команді copy можна задати такі ключі

/a– копіювати файли, поки команда не зустріне символ кінця текстового файлу (^Z)

/b– копіювати файли, ігноруючи символ кінця текстового файлу (^Z)

/v– проводити перевірку копії з оригіналом при копіюванні (верифікація інформації)

/y– коли при копіюванні ім'я файлу копії вже існує в цільовому каталозі, то він переписується поверх існуючого без перепитування

/d– дозволяє створювати зашифровані файли

/z– забезпечує копіювання мережевих файлів в режимі відновлення

/n–при копіюванні новий файл записувати в форматі DOS (ім'я до 8 символів)

В команді copy як і в командах ren та del можна застосовувати шаблони, для копіювання групи файлів.

Об’єднання файлів. За допомогою команди copy можна об’єднувати текст декількох файлів в один. Цю операцію називають склеюванням файлів. Склеювання відбувається за допомогою команди copy а саме:

copy[шлях 1\]ім'я1+[шлях 2\]ім'я2+[шлях 3\]ім'я3+…[шлях куди\]нове ім'я Enter

Якщо не вказати шлях 1, шлях 2, шлях 3, і тд. то відповідні файли беруться з активних каталогів. Коли не вказати нове ім'я, то автоматично приймається ім’я 1.

Команда перенесення файлів та груп файлів і перейменування каталогів

Інколи виникає необхідність перенесення одного чи декількох файлів з одного каталогу в інший. Операція перенесення файлів відрізняється від операції копіювання тим, що при перенесені файли знищуються а при копіюванні залишаються.

Формат команди: move [шлях звідки\]ім'я файла шлях куди\[ключі] Enter

Параметри в даній команді, аналогічні до описаних параметрів в команді copy. В команді move можна використовувати один з двох ключів

/y – коли при перенесенні ім'я файлу копії вже існує в кореневому каталозі, то буде виводитись перепитування для підтвердження перепису поверх існуючого;

/-y має протилежне значення до ключа «y» тобто не виводить перепитування.

Команда move має ще одну важливу функцію – перейменування каталогів.

Індивідуальне завдання до лабораторного заняття №2

1.Використовуючи команду перенаправлення вводу, записати у файл lab2zavd1.txt інформацію згідно варіанту. Продемонструвати результати роботи.

Таблиця 2.1 – Варіанти до виконання завдання 1

№ варіанту	Необхідна інформація	№ варіанту	Необхідна інформація
1	Довідка до команди переміщення	11	Довідка до команди копіювання
2	Системний час	12	Довідка до команди видалення каталогів
3	Довідка до команди видалення файлів	13	Вміст диску D
4	Вміст папки з Вашим прізвищем	14	Довідка до команди читання файлу
5	Вміст довідкової команди help	15	Вміст диску C
6	Мітку та серійний номер поточного диску	16	Системна дата
7	Версію операційної системи	17	Вміст папки Windows диску C
8	Перевірка системних файлів	18	Запланувати завдання
9	Просканувати диск	19	Поточний список усіх завдань, які виконуються на вашому ПК
10	Вміст папки User	20	Знайти ідентифікатор процесу

2. Згідно варіанту виконати контекстний пошук в будь якому каталозі з файлами на Вашому комп’ютері. Продемонструвати результати роботи.

Таблиця 2 .2 – Варіанти до виконання завдання 2

№ варіанту	Критерії пошуку	№ варіанту	Критерії пошуку
1	Усі файли, що мають у власному імені другу букву t	11	Файли та каталоги, що мають у власному імені дві букви і з будь яким розширенням

Продовження Таблиці 2.2

2	Файли та каталоги, що починаються буквами wi	12	Файли з розширенням txt
3	Файли та каталоги, що мають четверту букву t	13	Файли та каталоги, що мають першу букву t, а у розширенні першу букву j
4	Файли з 5 буквами у власному імені	14	Файли з іменем text
5	Усі файли з розширенням jpg	15	Файли з будь яким іменем і розширенням з двох букв
6	Файли з розширенням docx	16	Файли з 4 буквами у розширенні
7	Файли та каталоги, що починаються на букву a, з розширенням з двох букв	17	Файли з розширенням, яке починається на букву d
8	Визначити відмінності в тексті між двома файлами	18	Усі файли з розширенням pdf
9	Файли з розширенням, яке починається на букву d	19	Файли та каталоги, що мають другу букву t
10	Файли з розширенням, яке починається на букву p	20	Усі архівні папки

3. У індивідуальному завданні 4 попередньої роботи №1 у каталозі, що позначений зірочкою, створіть два текстові файли: zavd31.txt (своє прізвище та ім’я) та zavd32.txt (назву дисципліни). Об’єднайте ці два файли під назвою zavd3.txt та скопіюйте цей файл на рівень вище в дереві каталогів.

4. Спробуйте виконати копіювання іншим способом, за допомогою іншої утиліти Windows (скористайтесь help). Використайте щонайменше два ключі та поясніть їх використання. Продемонструвати результати роботи.

Контрольні питання

1. Як визначити відмінності в тексті між двома файлами?
2. Як відбувається перевірка системних файлів?
3. Що виконує команда taskkill –pid?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №3
ЗБЕРЕЖЕННЯ ДАНИХ ОПЕРАЦІЙНОЇ СИСТЕМИ. АРХІВАЦІЯ ТА ВІДНОВЛЕННЯ ДАНИХ У WINDOWS. ЛОКАЛЬНЕ РЕЗЕРВНЕ КОПІЮВАННЯ. СТВОРЕННЯ ОБРАЗУ ОПЕРАЦІЙНОЇ СИСТЕМИ

Мета роботи: вивчення методів архівації і відновлення даних у командному рядку файлової системи Windows. Навички роботи з командами порівняння, зміни системного запрошення консолі. Вміти змінювати системне запрошення командного рядка; навчитися порівнювати файли та каталоги файлової системи Windows; вміти архівувати файли та папки, створювати sfx-архіви як виконувані файли; вміти здійснювати локальне резервне архівування даних засобами адміністрування ОС Windows; вміти відновлювати дані, створювати образи операційної системи.

Література: [1, 2, 3 5]

Теоретичні відомості

Для того, щоб більш вільно почувати себе в деревах підкаталогів використовується зміна системної підказки. Це здійснюється за допомогою команди PROMPT. Для виведення імені робочого каталогу використовується запрошення: H>PROMPT PG де знак $ – ознака керуючого символу. Зазвичай використовуються такі символи: $p – видача імені робочого каталогу; $t – видача поточного часу; $d – видача поточної дати; $g – видача символу ">", і т.д.

Наприклад, PROMPT PG дає вид запрошення C:\Users>

Скасування запрошення здійснюється командою PROMPT без параметрів.

Команда порівняння файлів

Команда COMP дозволяє порівнювати між собою дискові файли. На екран видаються повідомлення про відмінності. Порівняння виконується байт за байтом, починаючи з перевірки останнього байта кожного файлу: чи є він символом EOF (End-offile). При виявленні більше 10 відмінностей команда COMP припиняє роботу.

Формат команди: COMP [ім'я диска:] [шлях до підкаталогу] [ім'я файла1] [ім'я диска:] [шлях до підкаталогу] [ім'я файла2].

Архівація та відновлення даних у Windows

В операційній системі «Windows» користувач легко може обробляти будь-які дані та переміщати інформацію відповідно до своєї необхідності. Однак, такий спрощений доступ, внаслідок зовнішнього впливу або помилкових дій самого користувача, може призвести до втрати або небажаного видалення важливої інформації з пристрою зберігання даних.

Архівація – це стискання одного, або більше файлів з метою економії пам'яті і розміщення їх в одному архівному файлі. Архівація даних – це зменшення фізичних розмірів файлів, в яких зберігаються дані, без значних інформаційних втрат.

Створення sfx-архівів

Досить часто при роботі над автоматичною установкою додатків виникає необхідність створити без ключовий саморозпакуваний (надалі будемо використовувати термін «SFX» – англ. self-extracting archive, так коротше) архів. Як правило, SFX архіви дозволяють значно заощадити місце на диску. Наприклад, якщо додаток не вимагає інсталювання, то його можна помістити в Program Files. Однак, аналогічної мети можна досягти, запакувавши додаток в SFX, який розпакує вміст в потрібну папку.

Область застосування SFX не обмежується додатками, що не вимагають установки. Можна без проблем архівувати додаток і конфігурувати SFX таким чином, що після розпакування буде запущена установка додатку з потрібними ключами. Ще однією перевагою SFX архівів є те, що в деяких випадках вони дають можливість не просто автоматизувати, але і повністю приховати процес і прогрес інсталяції від кінцевого користувача.

Можна розглядати створення SFX архівів (рис 3.1) за допомогою таких програм: WinRAR, 7-Zip, SFX Creator.

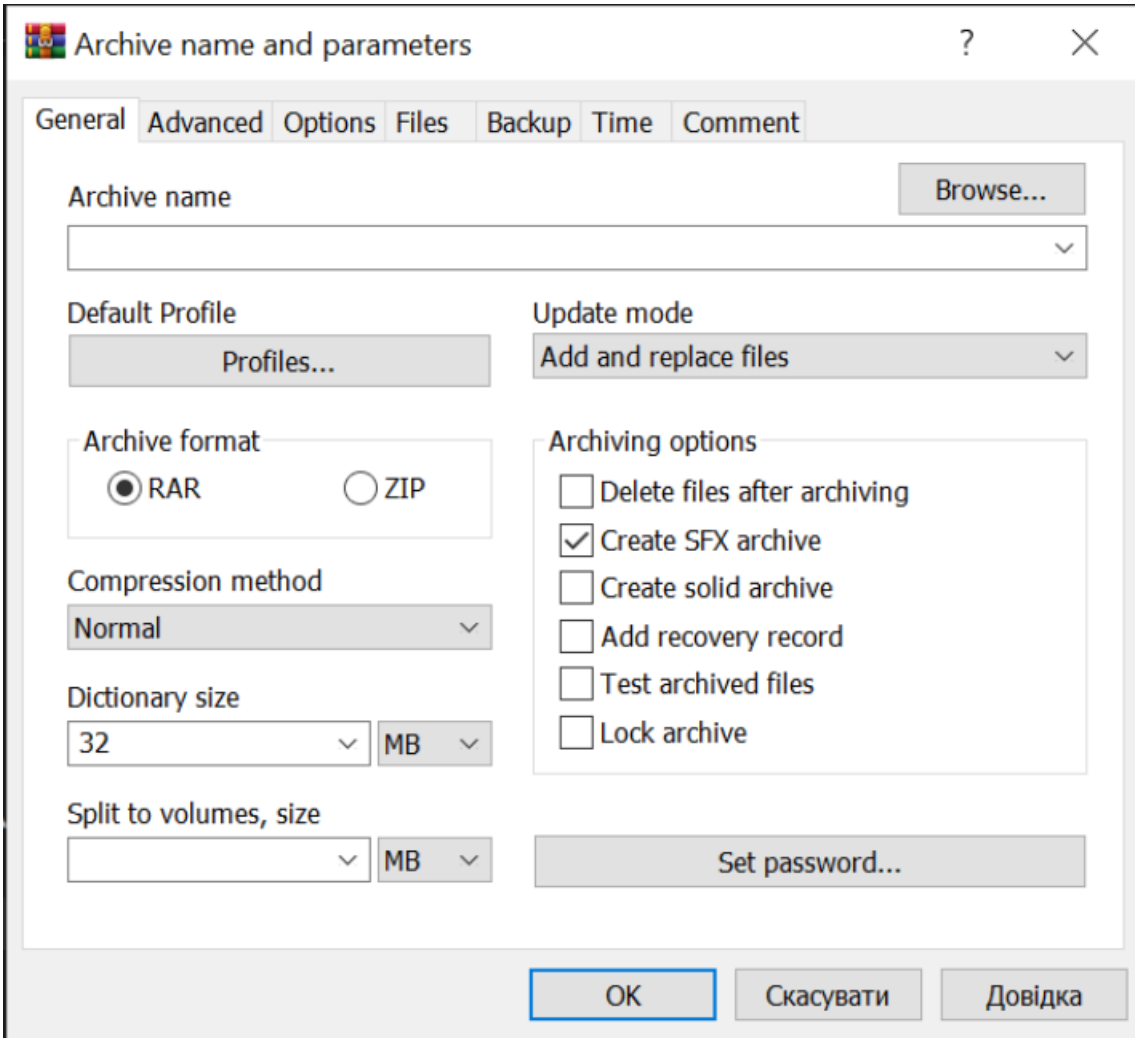


Рисунок 3.1 – Створення SFX архівів

Архівація та резервне копіювання засобами Windows

Архівація є фундаментальною частиною використання операційної системи. Нажаль, про це частото забувають (або взагалі не знають) більшість користувачів. Операційна система вбудованими засобами для адміністрування дозволяє архівувати дані таким чином, що в жодного користувача не повинно виникнути труднощів при виконанні цієї операції.

Переваги локального зберігання даних:

– простота впровадження, адміністрування та нарощування функціональності;

– прозорість для операційних систем;

– управління з однієї точки;

– неможливість рядовому працівнику отримати доступ до чужих даних, не маючи санкціонованого доступу.

Далі слід налаштувати параметри резервного копіювання (рис 3.2) та (рис3.3.):

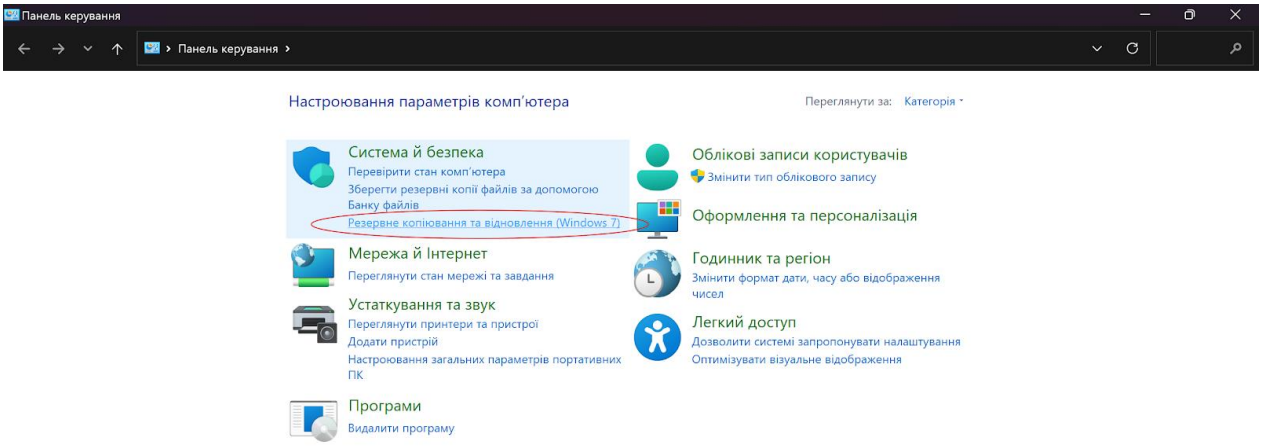


Рисунок 3.2 – Резервне копіювання у Панелі керування

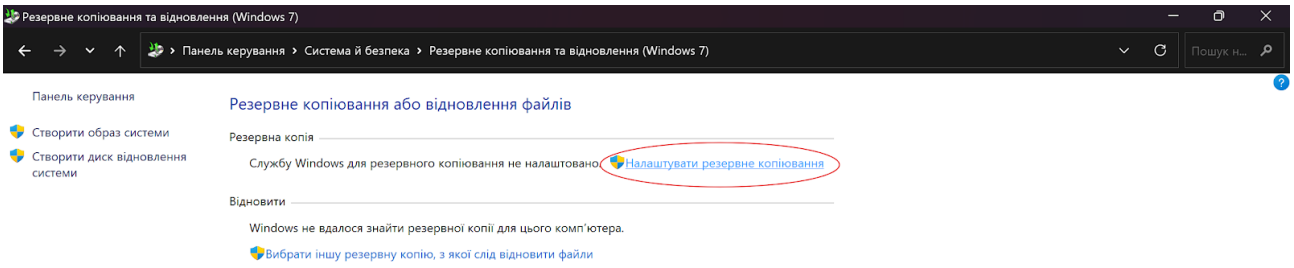


Рисунок 3.3 – Налаштування резервного копіювання

У разі резервного копіювання файлів за допомогою програми архівації Windows можна надати дозвіл ОС Windows вибирати об'єкти для резервного копіювання або самостійно вибрати окремі папки та диски, які потрібно архівувати.

Залежно від вибраних об'єктів, резервна копія містить компоненти, описані в наступних розділах:

Надати вибір Windows. Якщо дозволити Windows вибирати об'єкти для резервного копіювання, у резервну копію увійдуть такі компоненти: файли даних, збережені в бібліотеках, робочому столі та папках Windows за замовчуванням для всіх користувачів з обліковим записом на цьому комп'ютері.

Примітка. У резервну копію включаються лише локальні файли у бібліотеках. Якщо в бібліотеці є файли, збережені на диску, який розташований на іншому комп'ютері в мережі, в Інтернеті, на тому ж диску, де зберігається резервна копія, або

на диску, не відформатованому з використанням файлової системи NTFS, ці файли не увійдуть до резервної копії. За замовчуванням папки Windows включають: AppData, Контакти, Робочий стіл, Завантаження, Вибране, Посилання, Збережені ігри та Пошуки. Якщо диск, на якому зберігається резервна копія, відформатований з використанням файлової системи NTFS і на ньому достатньо місця, системний образ програм, Windows, всі драйвери та параметри реєстру також включаються до резервної копії. Цей образ можна використовувати для відновлення вмісту комп'ютера у разі відмови жорсткого диска або комп'ютера.

Примітка. Усі файли у відомих системних папках (тобто ті, що містять файли, які необхідні для запуску Windows) та відомі програмні файли (файли, які при встановленні програми визначаються в реєстрі як частина програми) не увійдуть до резервної копії, навіть якщо вони знаходяться у вибраній папці .

Якщо папка або диск не вибрано, вміст не включається до резервної копії. Програма архівації Windows не створює резервні копії таких об'єктів: програмних файлів (файли, які при встановленні програми визначаються у реєстрі як частина програми); файлів, що зберігаються на жорстких дисках, відформатованих із використанням файлової системи FAT; файлів у кошику; тимчасових файлів на дисках об'ємом менше ніж 1 ГБ.

Резервне копіювання сертифіката шифрованої файлової системи (EFS)

Шифрована файлова система (EFS) – це вбудований інструмент шифрування в Windows, який використовується для шифрування файлів і папок на дисках NTFS, щоб захистити їх від небажаного доступу.

EFS забезпечує прозоре шифрування та дешифрування файлів облікових записів користувачів за допомогою стандартних розширених алгоритмів шифрування. Будь-яка особа чи програма без відповідного ключа шифрування файлу не може відкрити жодних зашифрованих файлів і папок. Шифрування – це найнадійніший захист, який Windows пропонує для захисту особистих файлів і папок.

При шифруванні даних на комп'ютері необхідно передбачити спосіб відновлення цих даних у випадку, якщо щось станеться з ключем шифрування. Якщо ключ шифрування буде втрачено або пошкоджено, а спосіб відновлення даних відсутній, дані будуть втрачені.

Щоб гарантувати постійний доступ до зашифрованих даних, необхідно зробити резервні копії сертифіката та ключа шифрування. Якщо комп'ютер користується кількома особами або якщо для шифрування файлів використовується смарт-картка, потрібно створити сертифікат відновлення файлу.

Створення резервної копії сертифіката шифрування файлу PFX і ключа допоможе уникнути остаточної втрати доступу до зашифрованих файлів і папок, якщо оригінальний сертифікат і ключ буде втрачено або пошкоджено.

Якщо ви втратите доступ до своїх зашифрованих файлів і папок, ви не зможете знову відкрити їх, доки не зможете відновити сертифікат і ключ шифрування файлів, які використовуються з EFS.

Можна здійснювати локальне резервне архівування важливих елементів файлової системи ОС Windows для забезпечення цілісності інформації через резервну копію за розкладом (рис.3.4).

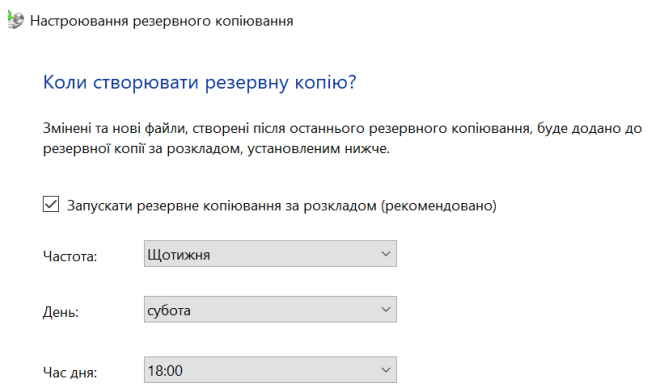


Рисунок 3.4 – Налаштування резервного копіювання за розкладом

Створення образу операційної системи

Образ диска – це повна, «фотографічна» копія оптичного диска. На відміну від простої копії диска його образ – це не просто набір папок, а файл, який містить інформацію про формат диска, завантажувальних даних, про його структуру і безпосередньо самі дані.

За замовчуванням образ операційної системи включає диски, необхідні для запуску Windows. Він також містить Windows та системні параметри, програми та файли. Образ системи можна використовувати для відновлення вмісту комп'ютера у разі відмови жорсткого диска або збою роботи комп'ютера.

Відновлення комп'ютера з образу системи повне: не можна вибрати окремі елементи для відновлення, вміст образу системи буде замінено всі поточні програми, системні параметри і файли. Хоча такий тип резервного копіювання включає особисті файли користувача, рекомендується регулярно створювати резервні копії файлів за допомогою програми архівації Windows, щоб за необхідності відновити окремі файли та папки.

Під час налаштування програми архівації Windows можна надати Windows вибір об'єктів для резервного копіювання (тоді буде увімкнено образ системи) або самостійно вибрати об'єкти для резервного копіювання та вказати, чи потрібно включати образ системи.

Якщо комп'ютер використовує кілька дисків або розділів, можна створити образ системи, який включає всі ці компоненти. Для цього виконайте дії, описані в розділі Резервне копіювання програм, системних параметрів та файлів.

Вибираючи один з способів збереження образу системи, варто пам’ятати про об’єм даних. Якщо, наприклад, образ включає понад 300 Гб даних, то таке розміщення зрозуміло, що доцільно, щоб було мережеве.

Відновлення даних

Розглянемо декілька прикладів відновлення втрачених даних в ОС Windows, за умови, якщо вони відсутні у Кошику та не збережено їх резервну копію.

1. Відновлення через Microsoft Store

Якщо не вдається знайти втрачений файл із резервної копії, а файл, приміром, був видалений комбінацією Shift+Del, скористайтеся відновленням файлів у Windows, яка є програмою командного рядка, доступною в Microsoft Store.

Скористайтеся цією програмою, щоб спробувати відновити втрачені файли, видалені з локального пристрою збереження даних (зокрема, внутрішні диски, зовнішні диски та USB-пристрої), і їх не можна відновити з кошика. Відновлення в хмарному сховищі та мережевих спільних файлах не підтримується.

Натисніть клавішу Windows, введіть «Відновлення файлів у Windows» в полі пошуку, а потім виберіть «Відновлення файлів у Windows».

Коли буде запропоновано дозволити програмі вносити зміни на вашому пристрої, натисніть кнопку Так.

У вікні командного рядка введіть команду в такому форматі:

winfr source-drive: destination-drive: [/switches]

Вихідні та цільові диски мають відрізнятися. Під час відновлення з диска операційної системи (часто C:) використовуйте фільтр /n <> та /y:<типу<(и)> перемикачів, щоб вказати файли або папку користувача.

Відновлення через програмне забезпечення

ПЗ компанії Hetman Software – це широкий асортимент програмного забезпечення, основним напрямом якого є відновлення даних із будь-яких носіїв інформації, яке зображено на рисунку 3.5.

Для відновлення даних скористаємося лінком для завантаження trial-версії Hetman Patrition Recovery: https://hetmanrecovery.com/uk/hard-drive-data-recovery-software (рис 3.6).

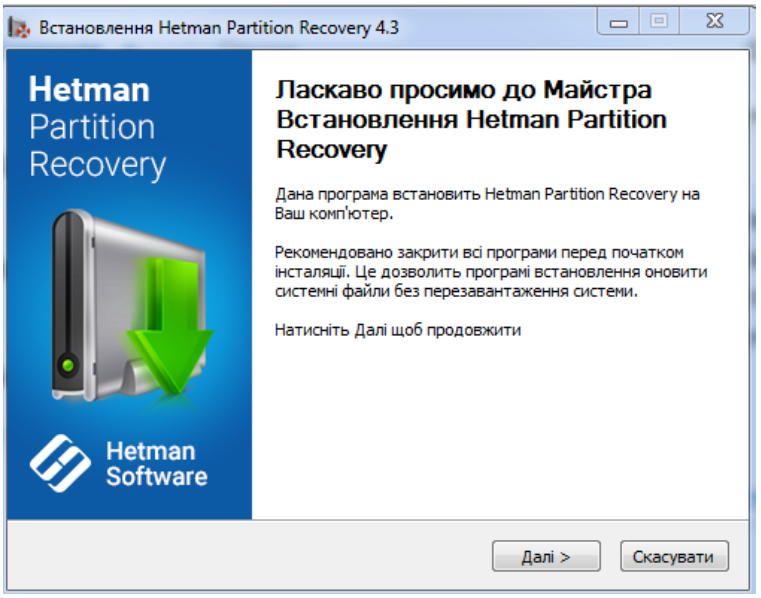


Рисунок 3.5 – Встановлення програми

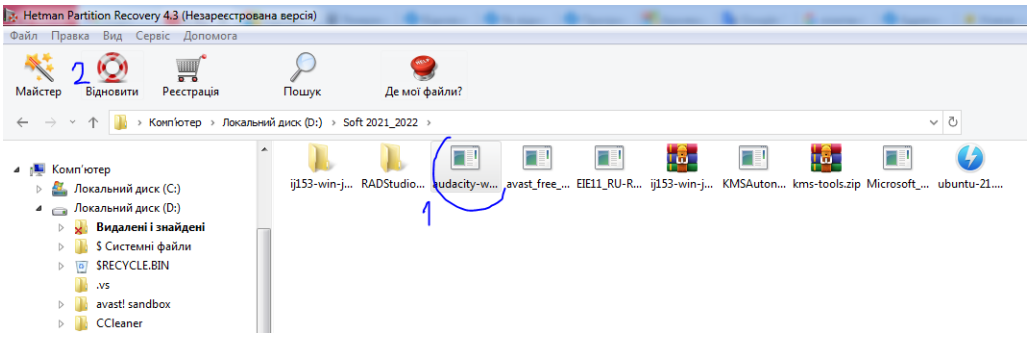


Рисунок 3.6 – Відновлення програмою

Індивідуальні завдання до лабораторного заняття №3:

1. Змініть системне запрошення, щоб воно виглядало у вигляді Вашого прізвища, на зразок нижче поданого скрину. Поясніть алгоритм. Продемонструйте результати роботи приклад якого зображено на рисунку 3.7.

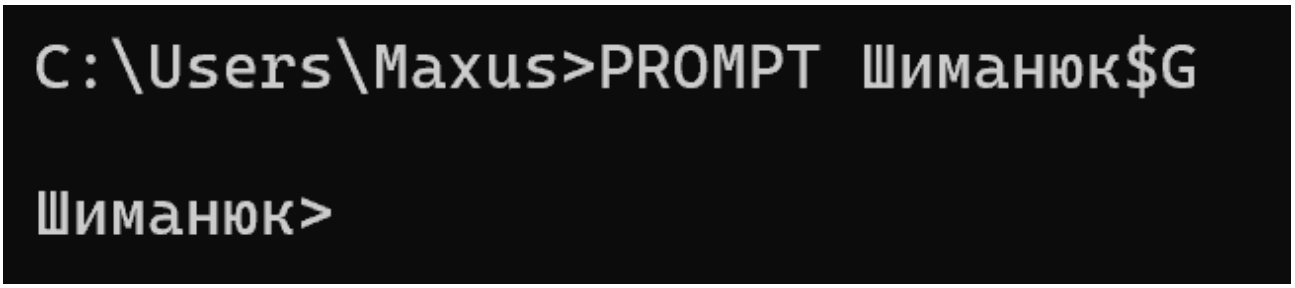


Рисунок 3.7 – Приклад виконання завдання

2. Порівняйте будь які два файли на своєму ПК консольним методом і зробіть висновки про результати порівняння.

3. Заархівуйте групу елементів sfx-архівом. Поясніть алгоритм покроково. Продемонструйте результати роботи.

4. Сформуйте покроковий алгоритм налаштування резервного копіювання на Вашому ПК. Налаштуйте архівацію однієї папки з часом архівування – щопонеділка о 18:00. Продемонструйте цей алгоритм.

5. Створіть на робочому столі Вашого ПК два порожні файли, видаліть їх (Shift+Del) та відновіть їх одним із відомих Вам способів.

Контрольні питання

1. Що таке образ диска і як ним користуватися?
2. Що таке резервне копіювання сертифіката шифрованої файлової системи (EFS)?
3. Яка команда використовується для порівняння файлів?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №4
КОМАНДИ ДЛЯ РОБОТИ З СИСТЕМОЮ І МЕРЕЖЕЮ

Мета роботи: вивчення методів консольного керування адміністрування операційною системою та команд роботи з мережею. Вміти визначати характеристики системи через консольні команди; вміти налаштовувати опції автозапуску, системних параметрів та компонент; навчитись отримувати інформацію про стан мережевого устаткування ПК та стан служб роботи Мережі.

Література: [1, 2, 3, 4, 5]

Теоретичні відомості

Команди керування операційною системою

Systeminfo – команда, яка дозволяє отримати докладну інформацію про програмну та апаратну конфігурацію комп'ютера.

У ній можна знайти багато корисних для користувача даних:

- інформація про «залізо»;
- версія операційної системи;
- список встановлених оновлень;
- дата встановлення та останнього завантаження ОС тощо.

Shutdown – інструмент для вимкнення або перезавантаження комп'ютера. На відміну від відповідних кнопок у меню "Пуск", підтримується розширена функціональність. Доступні відкладені дії за таймером, негайне відключення з примусовим закриттям запущених програм. Ключ /s відповідає за вимкнення комп'ютера, ключ /r – за перезавантаження. Щоб активувати дію команд за таймером, використовується ключ /t з бажаним часом у секундах. Наприклад, заплановане через хвилину перезавантаження буде виглядати так: shutdown /r /t 60. Є інструмент і графічний інтерфейс, який можна викликати командою shutdown /i.

Msinfo32 надає повний список апаратного забезпечення комп'ютера, ресурсів, компонентів, програмного забезпечення та драйверів. Інформація поділена за групами для зручної навігації.

Msconfig дозволяє змінювати налаштування запуску операційної системи, використовувати для нього додаткові параметри (на зразок безпечного режиму), включати/вимикати елементи автозавантаження та встановлені служби.

Команди консолі – мережеві утиліти

Ping – команда служить для перевірки зв'язку з віддаленим вузлом. Також виводить статистику про втрати пакетів та про час їх доставки. Наявна у всіх ОС. При роботі використовує протокол ICMP (Internet Control Message Protocol, протокол міжмережевих управляючих повідомлень).

Синтаксис команди для ОС Windows:

ping [-t] [-a] [-n лічильник] [-l довжина] [-f] [-i TTL] [-v тип] [-r лічильник] [-s кількість] [[-j список комп'ютерів] | [- k список комп'ютерів]] [-w інтервал] список розсилки

Параметри:

– t – повторює запити до віддаленого вузла до команди переривання;

– a – визначення адрес по назвах вузлів;

– n лічильник – зазначає кількість запитів, що посилаються.

– За замовчуванням – 4;

– l довжина – зазначає розмір пакету, що посилається. За замовчуванням – 32 байти, максимум – 65527;

– f – пересилаються пакети з ознакою заборони фрагментації (Do not Fragment). Пакети не будуть розриватися при проходженні шлюзів;

– i ttl – встановлює поле часу існування пакетів TTL (Time To Live);

– v тип – встановлює поле типу служби (Type Of Service) пакетів;

– r лічильник – записує маршрут посланих і повернутих пакетів в поле запису маршруту Record Route. Параметр лічильник задає кількість комп'ютерів в інтервалі от 1 до 9;

– s число – зазначає кількість ретрансляції на маршруті, де буде робитися відмітка часу;

– j список комп'ютерів – направляє пакети по маршруту, що задається параметром список комп'ютерів. Комп'ютери у списку можуть бути розділені проміжними шлюзами (вільна маршрутизація). Максимальна кількість, яка дозволяється протоколом IP, дорівнює 9;

– k список комп’ютерів – направляє пакети по маршруту, що задається параметром список комп’ютерів. Комп’ютери у списку не можуть бути розділені проміжними шлюзами (вільна маршрутизація). Максимальна кількість, яка дозволяється протоколом IP, дорівнює 9;

– w інтервал – вказує проміжок часу очікування (мс).

Список розсилки – зазначає список комп’ютерів, яким направляються запити (рис.4.1).

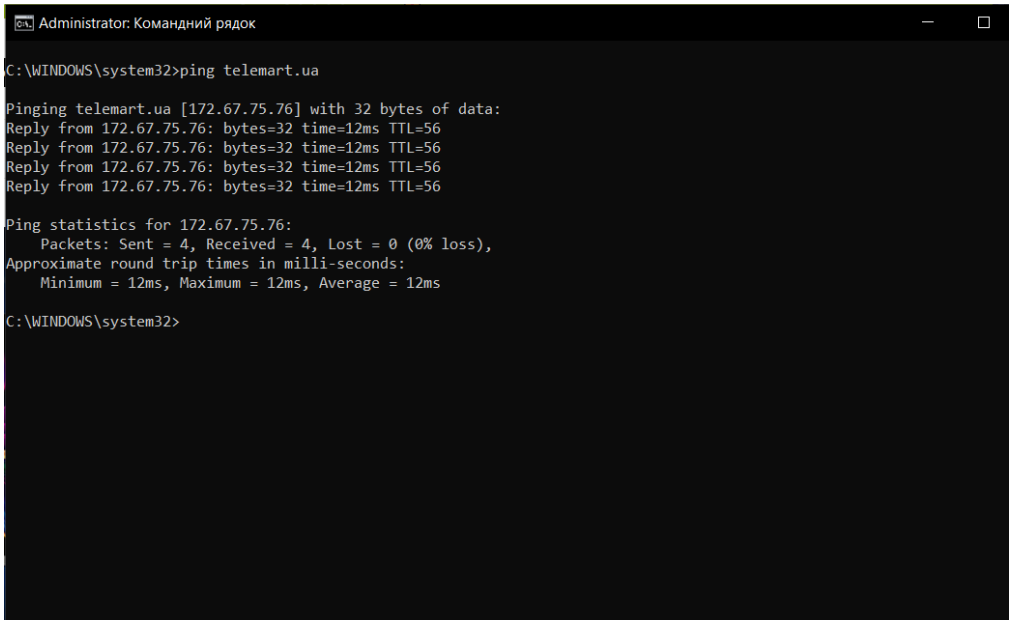


Рисунок 4.1 – Команда ping

За допомогою цих даних можна оцінити стабільність роботи з мережею чи конкретним сайтом. Команда підтримує як IP, так і інтернет-адреси. В останньому випадку IP-адреса ресурсу висвітиться автоматично після початку роботи. За замовчуванням здійснюється всього чотири обміни даними із сервером. Якщо потрібне постійне опитування до скасування процедури користувачем, після адреси потрібно використовувати команду -t.

Фактично пінгування означає тестове опитування (яке проводиться утилітою Ping), при якому здійснюється вимірювання часу проходження пакетів між комп'ютерами мережі, що дозволяє встановити відповідність між доменною і IP-адресою будь-якого пристрою мережі; так перевіряється якість і швидкість каналів зв'язку провайдерів.

Примітка. Скористайтесь командою ping /? для довідки про ключі програми.

Tracert – команда для трасування маршруту до певного інтернет-вузла. Утиліта надсилає запити на вказану адресу, а також команда ping. При цьому відображається інформація про всі проміжні маршрути, через які запити проходять шляхом до потрібного ресурсу. Також фіксується час кожної частки маршруту у мілісекундах (рис 4.2). Це дозволяє оцінити повний шлях трафіку та ділянку, на якій виникають найбільші затримки. Розташування проміжних вузлів зображено на рисунку 4.3. та довжина шляху за допомогою Карти Google (рис 4.4).

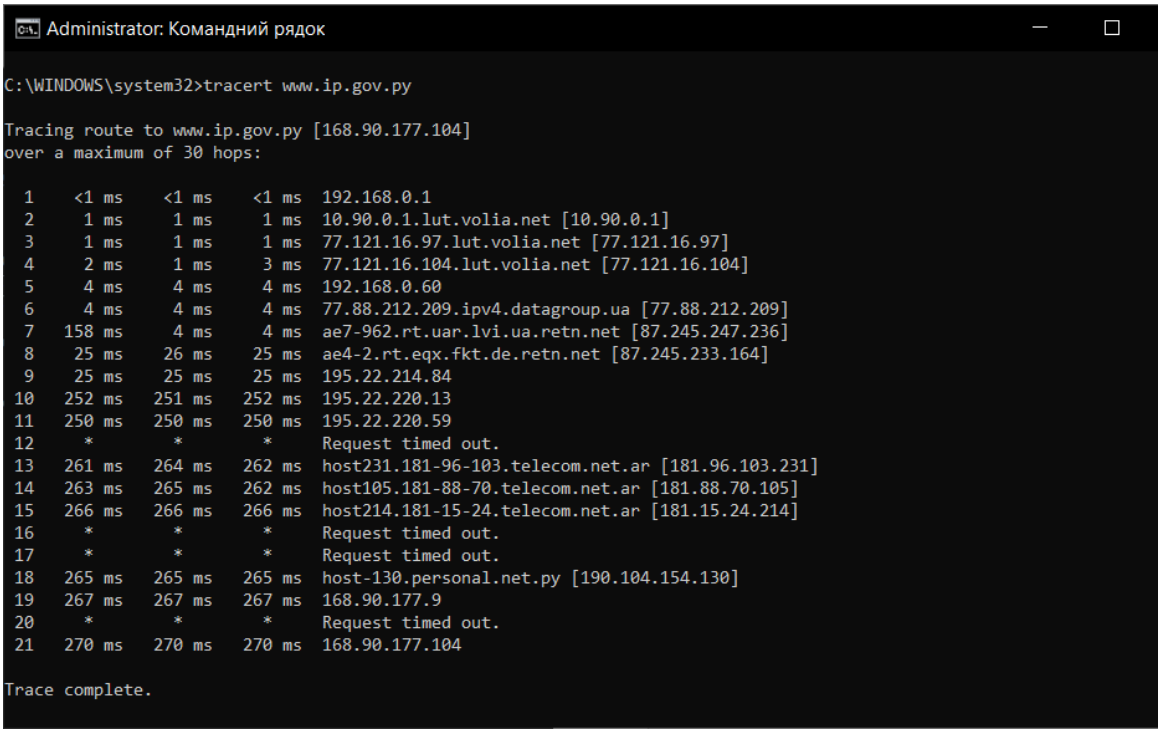


Рисунок 4.2 – Команда tracert

#	IP АДРЕСА	КРАЇНА	МІСТО	ПРОВАЙДЕР	ЧАСОВИЙ ПОЯС	КООРДИНАТИ
1	192.168.0.1					0, 0
2	10.90.0.1			IANA-ARIN		0, 0
3	77.121.16.97	Ukraine			Europe/Kyiv	50.4522, 30.5287
4	77.121.16.104	Ukraine			Europe/Kyiv	50.4522, 30.5287
5	192.168.0.60					0, 0
6	77.88.212.209	Ukraine			Europe/Kyiv	50.4522, 30.5287
7	87.245.247.236	United Kingdom			Europe/London	51.4964, -0.1224
8	87.245.233.164	United Kingdom			Europe/London	51.4964, -0.1224
9	195.22.214.84	Italy			Europe/Rome	43.1479, 12.1097
10	195.22.220.13	Italy			Europe/Rome	43.1479, 12.1097
11	195.22.220.59	Italy			Europe/Rome	43.1479, 12.1097
12	181.96.103.231	Argentina	Córdoba		America/Argentina/Cordoba	-31.429, -64.1756
13	181.88.70.105	Argentina			America/Argentina/Buenos_Aires	-34.6022, -58.3845
14	181.15.24.214	Argentina	General Piran		America/Argentina/Buenos_Aires	-37.1833, -57.7833
15	190.104.154.130	Paraguay	Asunción		America/Asuncion	-25.2869, -57.6511
16	168.90.177.9	Paraguay			America/Asuncion	-23.3333, -58
17	168.90.177.104	Paraguay			America/Asuncion	-23.3333, -58

Рисунок 4.3 – Розташування проміжних вузлів

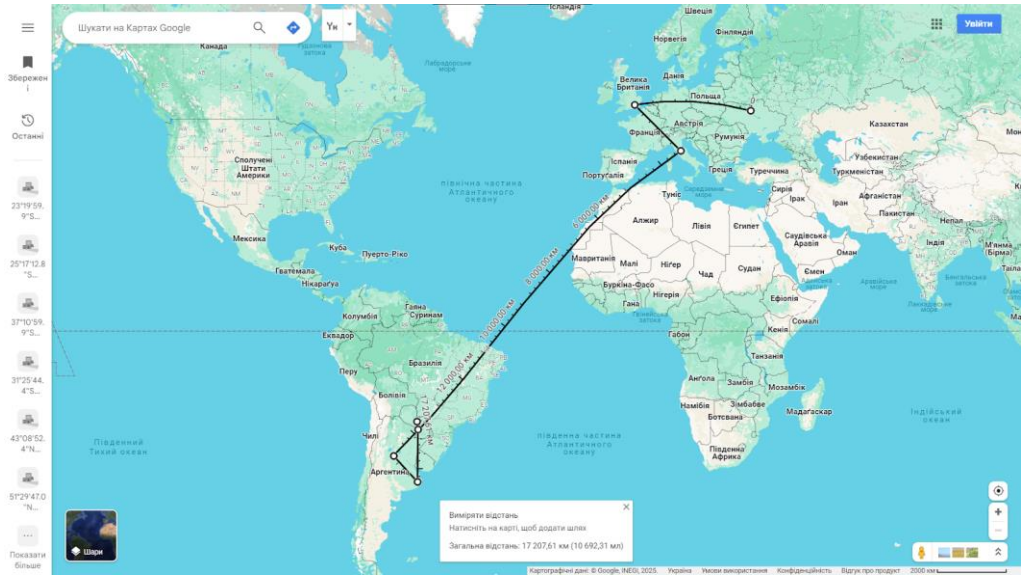


Рисунок 4.4 – Довжина шляху за допомогою Карти Google

Pathping – команда поєднує можливості команд ping та tracert. Вона дозволяє оцінити затримки передачі і втрати пакетів кожному ділянці маршруту до певному вузлу.

Ipconfig – засіб перегляду інформації про конфігурацію мережевих адаптерів (IP та MAC-адреси). Зазвичай використовується з ключем /all.

Індивідуальні завдання до лабораторного заняття №4:

1. За допомогою командного рядка Windows, визначіть версію BIOS та дату встановлення Windows на Вашому ПК. Продемонструйте результати (скрини).
2. Заплануйте вимкнення комп'ютера через X хвилин, де X- номер Вашого варіанту. Продемонструйте результати планування.
3. Командою консолі відкрийте список файлів автозавантаження на Вашому ПК. Продемонструйте результат (скрин).
4. Протестуйте шлях проходження пакетів від Вашого ПК до будь якого сайту. Оцініть ступінь втрати пакетів, час проходження. Продемонструйте результати.
5. За допомогою команди tracert в командному вікні Windows дослідіть маршрути до 3х комп'ютерів, розташованих в різних кінцях світу (див. варіанти внизу). Дізнайтеся і запишіть розташування проміжних вузлів. Перейдіть до кожної трасованої ділянки, в яких сигнал передавався на відстань більше 300 км. Вкажіть, між якими містами / країнами розташовані ці ділянки і оцініть приблизну довжину шляху. Вкажіть пройдені пакетами точки обміну трафіку.

Таблиця 4.1 – Варіанти для виконання завдання 5

Варіант	Країни
1.	Польща, США, Австралія
2.	Японія, Франція, Канада
3.	Німеччина, Іран, Мексика
4.	Італія, ОАЕ, Китай
5.	Іспанія, Ізраїль, Алжир
6.	Великобританія, Єгипет, Болгарія
7.	Словаччина, Індія, Бразилія
8.	Швеція, Венесуела, Північна Корея
9.	Швейцарія, Парагвай, Південна Корея
10.	Перу, Польща, Сингапур
11.	Словенія, Болівія, Тайвань
12.	Кіпр, Фінляндія, Аргентина
13.	Греція, Норвегія, Індонезія
14.	Іспанія, Ірландія, Малайзія
15.	Грузія, Бельгія, Таїланд
16.	Нідерланди, Данія, Папуа-Нова Гвінея
17.	Бельгія, Шрі-Ланка, Монголія
18.	Греція, Пакистан, Узбекистан
19.	Туреччина, Якутськ, Таджикистан
20.	Словаччина, Ірак, Краснодар
21.	Чехія, Грузія, Саудівська Аравія
22.	Угорщина, Азербайджан, Лівія
23.	Румунія, Сомалі, Ямайка
24.	Латвія, Кенія, Куба
25.	Литва, Ємен, Еквадор
26.	Естонія, Мадагаскар, Колумбія
27.	Казахстан, Ліберія, Болівія

Щоб дізнатись розташування проміжних вузлів, можна скористатись онлайн-сервісом для визначення геолокації IP, наприклад: https://semalt.tools/uk/ip-geolocation

Контрольні питання

1. Адресація вузлів комп’ютерної мережі. Види та типи адрес, які використовуються в сучасних мережах.
2. Призначення команди hostname.
3. Призначення та особливості застосування команди ipconfig.
4. Призначення команди ping. Приклади застосування.

ЛАБОРАТОРНЕ ЗАНЯТТЯ №5
ОСНОВИ АДМІНІСТРУВАННЯ ОС WINDOWS

Мета роботи – вивчити методи адміністрування ОС Windows. Вміти використовувати вбудовані механізми адміністрування операційної системи Windows; налаштовувати опції автозапуску через механізми адміністрування; розуміти принцип журналювання подій операційної системи; навчитись відновлювати елементи системи.

Література: [1, 2, 3, 4, 5]

Теоретичні відомості

Автозапуск автоматично виконує деякі функції за нас, передбачаючи наші бажання. Але з іншого боку, цей механізм таїть в собі загрозу, вірус проникнувши на носій інформації і прописавшись в файлі "autorun.inf" забезпечить собі «безперешкодний» пропуск на комп'ютер при підключенні пристрою до системи. Тому рекомендується відключити функцію автозапуску флешки і інших носіїв в системі. функція авто запуску в ОС сімейства Windows включена за замовчуванням.

Автовідтворення відкриваємо через Настроювання параметрів комп’ютера (рис5.1):

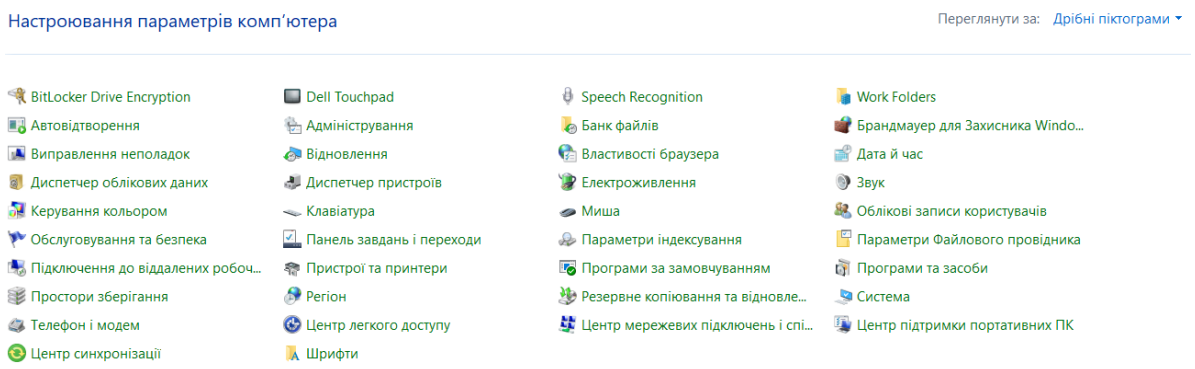


Рисунок 5.1 – Автовідтворення

У вікні (Автовідтворення) зніміть прапорець Використовувати автозапуск для всіх носії та пристроїв.

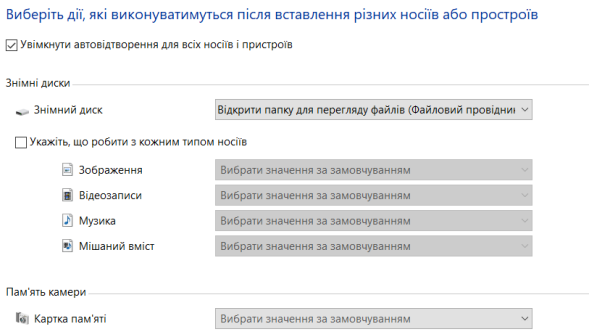


Рисунок 5.2 – Автовідтворення

Редактор локальної групової політики.

Виконайте команду Пуск – Виконати (WIN + R), або натисніть комбінацію клавіш в віконці наберіть команду gpedit.msc і клацніть на кнопці ОК отримаєте результат, який зображено на рисунку 5.3.

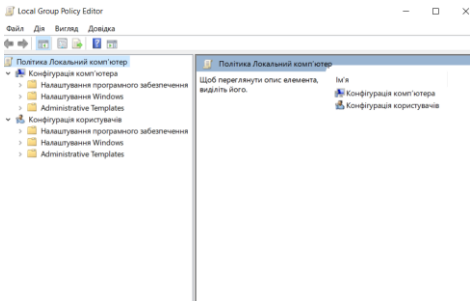


Рисунок 5.3 – gpedit.msc

В цьому вікні можна відключити або встановити системні параметри. Перейдіть послідовно по розділах: Конфігурація комп'ютер → Система (System), що зображено на рисунку 5.4.

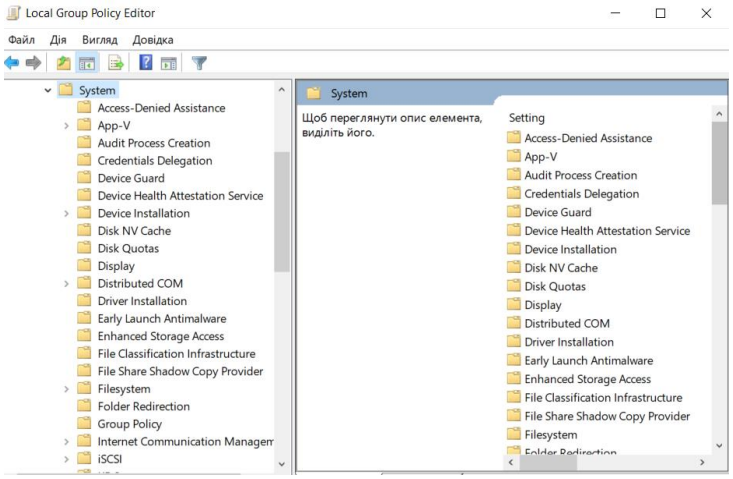


Рисунок 5.4 – Система

Реєстр Windows

Редактор реєстру є вбудованим компонентом операційної системи Windows і викликається шляхом введення команди Regedit.

Реєстр – це свого роду база даних Windows, в якій зберігається інформація про всіх параметрах ОС і конфігурації всіх встановлених в системі додатків і пристроїв. Він відповідає за функціонування і зовнішній вигляд операційної системи. У якій би ОС користувач не працював – кожне клацання мишею або натискання на кнопку клавіатури непомітно для користувача супроводжується десятками, а то і сотнями звернень до реєстру.

Реєстр являє собою системну базу даних, де операційна система, драйвери і всі програми зберігають свої налаштування. Фізично він складається з десятка файлів в каталогах, доступ до яких за допомогою файлових менеджерів, провідника і різних редакторів закритий. Вносити зміни до реєстру можуть тільки самі програми/драйвери і операційна система.

Користувач також може редагувати багато розділів цієї системної БД, і для цього існує інструмент – редактор реєстру Windows 10 (рис.5.5).

Редактор реєстру – це системне додаток, розташоване в каталозі «Windows».

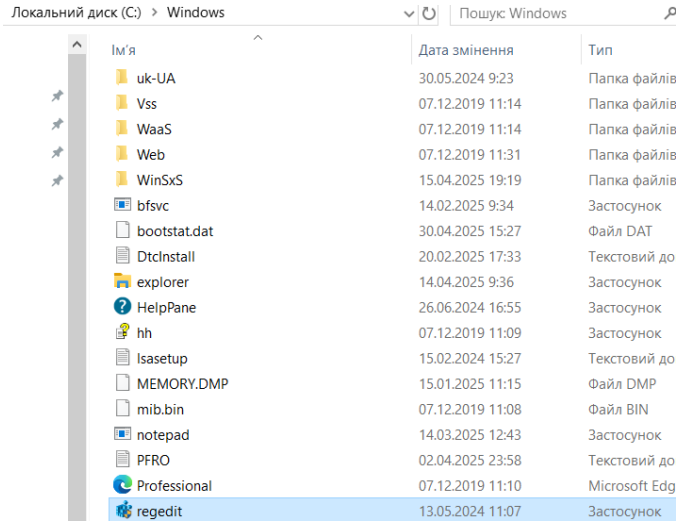


Рисунок 5.5 – Редактор реєстру

Реєстр складається з семи різних файлів. Шість з них знаходяться в папці Windows \ system32 \ config: system, software, sam, security, default, components.

Ці файли не мають розширення. Файл, який зберігає особисті настройки користувача, «прихований» в папці відповідної облікового запису, наприклад, в C: \ Documents and Settings \ <Ім'я користувача> і називається ntuser.dat.

Розділи реєстру зображені на рисунку 5.6 :

KEY_CLASSES_ROOT (HKCR) – найбільший розділ реєстру, в ньому міститься інформація про типи файлів і файлових асоціаціях.

HKEY_CURRENT_USER (HKCU) – містить налаштування облікового запису увійшов в систему користувача. Найбільш цікавим є підрозділ Software, який включає в себе налаштування всіх встановлених в системі додатків.

HKEY_LOCAL_MACHINE (HCLM) – містить абсолютно однакові для всіх користувачів системи параметри апаратної конфігурації комп'ютера.

HKEY_USERS (HKU) – тут зберігаються настройки (профілі) усіх користувачів комп'ютера. Якщо в систему увійшов інший користувач, відповідні підрозділи автоматично переміщаються в розділ «Б».

HKEY_CURRENT_CONFIG – містить відомості про налаштування всіх вбудованих компонентів комп'ютера і підключених до нього пристроїв.

Структура реєстру

Реєстр має ієрархічну структуру, він нагадує файлову систему жорсткого диска – з його каталогами, підкаталогами і файлами. Однак називаються елементи реєстру по іншому: верхній рівень ієрархії складають розділи (або ключі від англ. Key), кожен з яких може містити вкладені під-розділи, а також параметри. Саме в параметрах зберігається основний вміст реєстру, розділи служать лише для угруповання схожих по призначенням параметрів. Записи реєстру відповідають певному типу, який визначає область їх можливих значень, спосіб зберігання і обробки (всього передбачено 7 типів). До основних типів параметрів відносяться:

1. Строковий параметр REG_SZ і багаторядковий параметр REG_MULTI_SZ. За допомогою текстових параметрів можна, наприклад, поставити стартову сторінку веб-браузер.
2. Числові параметри містять значення будь-якої числової інформації. Приклад числового типу – тип REG_DWORD.
3. Для зберігання двійкових даних використовується тип REG_BINARY.

Змінювати записи реєстру або додавати нові можуть не тільки програми, але і самі користувачі. Наступний прийом впливає на швидкість завершення роботи Windows. За замовчуванням всім програмам на завершення роботи відводиться 20 с, але багатьом додаткам потрібно набагато менше часу. Можна дати Windows команду закривати програми по Після 4 с, щоб прискорити виключення ПК.

Тільки слід пам'ятати, найменша помилка може вивести з ладу як саму операційну систему, так і встановлені у ній додатки.

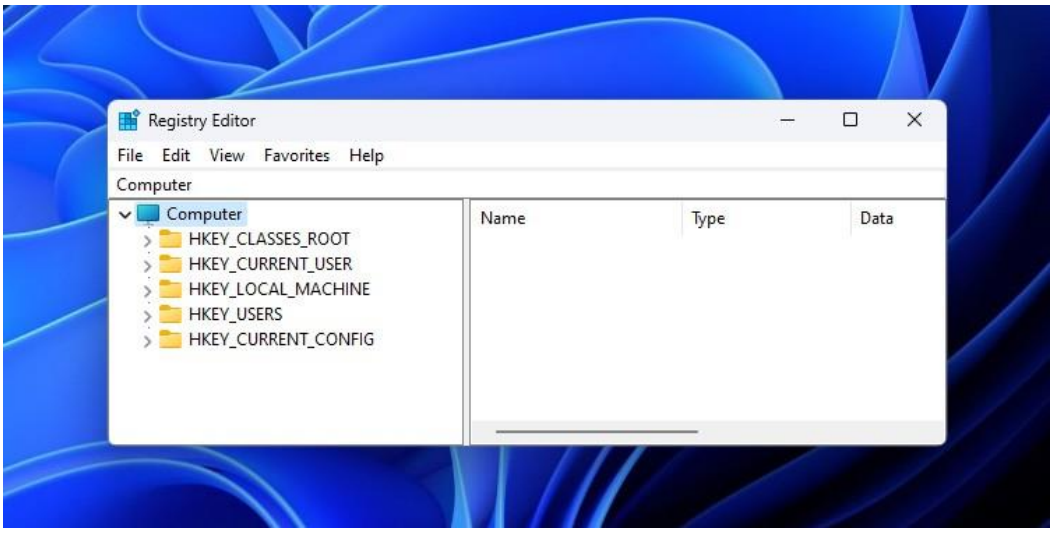


Рисунок 5.6 – Редактор реєстру

Чому Windows з часом починає працювати повільніше?

Чим довше використовуєте операційну систему, тим повільніше вона працює і завантажується. Чому це відбувається? Одна з причин – інтенсивна робота з додатками: установка нових програм, випробування демоверсій, деінсталяція непотрібного «софта».

При видаленні програми в реєстрі, як правило, залишаються її «сліди», особливо якщо деінсталятор додатки написаний розробником з помилками. Вже після декількох місяців роботи реєстру виявляється безнадійно захаращений цими «мертвими» записами – в результаті система вимагає більше часу на його завантаження. Допоможуть в даній ситуації утиліти для очищення системи – вони працюють, в тому числі і з реєстром Windows, відшукуючи в ньому помилкові записи і видаляючи їх. Надійно функціонують програми CCleaner, TuneUp Utilities і Glary Registry Repair.

Практична частина

1. У вікні редактора реєстру відкрийте розділ HKEY_LOCAL_MACHINE, а потім – підрозділи SYSTEM і CurrentControlSet.

2. Клацніть по підрозділу Control – справа з'являться всі параметри цього підрозділу. Виконайте подвійне клацання по параметру WaitToKillServiceTimeout.

3. Якщо у вікні вже зазначено значення 4000, то в цьому випадку рада буде для Вас закінчений. В іншому випадку введіть значення 4000, натисніть на кнопку OK і по X. Виконайте перезавантаження Windows – тепер система буде швидше завершувати свою роботу

Як зробити резервну копію реєстру Windows 10 вручну ?

1. Натисніть кнопку Пуск , введіть regedit.exe в полі пошуку, а потім натисніть клавішу Enter. Якщо відобразиться повідомлення із запитом ввести пароль адміністратора або підтвердити дію, введіть пароль або надайте відповідне підтвердження.

2. У редакторі реєстру знайдіть і клацніть розділ реєстру або підрозділ, для якого потрібно створити резервну копію.

3. Виберіть Пункт Файл > Експорт.

4. У діалоговому вікні Експорт файлу реєстру виберіть розташування, до якого потрібно зберегти резервну копію, а потім введіть ім'я файлу резервної копії в полі Ім'я файлу.

5. Натисніть кнопку Зберегти.

Якщо у Вас відкриті важливі вікна і з'явилася необхідність вимкнути або перезавантажити комп'ютер, в Windows є можливість зробити так, що при наступному включенні комп'ютера Ваші важливі вікна будуть на тому ж місці, де і були до перезавантаження виконавши відновлення колишніх вікон папок при вході в систему.

Індивідуальні завдання до лабораторного заняття №5:

Виконати запуск програмного забезпечення «Диспетчер завдань Windows» та перевірити його працездатність.

2. За допомогою програмного забезпечення «Диспетчера пристроїв» ОС Windows перевірити працездатність роботи пристроїв операційної системи.

3. Перевірити «Журнал подій» операційної системи Windows віртуальної машини. Переглянути кожний з журналів системи, ознайомитися з інформацією, що надається.

4. Перевірити працездатність програмного забезпечення «Системний моніторинг» та «Моніторинг ресурсів» операційної системи Windows.

5. Запустити програмне забезпечення стандартного брандмауера операційної системи Windows, переглянути роботу його компонентів.

6. Встановити програмне забезпечення «TeamViewer» та перевірити його роботу.

Контрольні запитання:

1. Призначення основних програмних компонентів операційної системи Windows щодо адміністрування її роботи.

2. Характеристика програмного забезпечення операційної системи Windows щодо віддаленого адміністрування.

3. Призначення «Журналу подій» операційної системи Windows, призначення журналів, за допомогою яких здійснюється моніторинг роботи.

ЛАБОРАТОРНЕ ЗАНЯТТЯ №6
СТРУКТУРА ФАЙЛОВОЇ СИСТЕМИ LINUX, ОСНОВНІ КОМАНДИ ТА КОМАНДИ РОБОТИ З ФАЙЛАМИ

Мета роботи – оволодіння практичними навичками роботи в системі UNIX. Знайомство із структурою файлової системи, основними командами роботи з файлами.

Література: [1, 2, 4, 7, 8]

Теоретичні відомості

Стандарт ієрархії файлової системи (FHS) визначає структуру файлових систем на Linux та інших UNIX-подібних операційних системах. Однак файлові системи Linux також містять деякі каталоги, які ще не визначені стандартом.

Якщо ОС UNIX встановлена на персональному комп'ютері, то на ньому підтримується певна кількість так званих віртуальних терміналів, між якими можна переключатись комбінаціями клавіш Alt+F#, де F# – одна з функціональних клавіш.

Розглянемо деякі основні команди. Слід зазначити, що немає штатного засобу, який надавав би користувачеві перелік доступних йому команд, тому основні команди необхідно пам'ятати.

Команда man форматує і відображає на терміналі сторінки довідкової системи. Відповідно до номерів розділів даються посилання на ту чи іншу сторінку довідника. Якщо є необхідність, можна вказати, в якому розділі треба шукати потрібну сторінку. Приклади використання (системні запрошення не показані): man 7 mdoc.

Команда who дозволяє визначити, хто ще працює в поточний момент в системі. Команда who am i нагадає вам, який ваш login.

Існують зручні команди визначення поточної дати й часу (date), а також виводу на екран календаря на будь-який місяць будь-якого року (cal).

Для того, щоби переглянути вміст текстового файлу, можна скористатись командою cat <ім'я файлу>, або more <ім'я файлу> (остання команда призначена для виводу інформації на екран посторінково, вона надає можливість "перегортати сторінки" вперед і назад). Існує команда wc <ім'я файлу> (word count – підрахувати

слова), яка дозволяє підрахувати кількість рядків (wc -l), слів (wc -w) і символів (wc -c) у файлі. Створити текстовий файл можна командою touch <ім’я файлу>.

Розглянемо особливості файлової системи UNIX. Вся файлова система поєднується в єдине дерево каталогів, які починаються з кореневого каталогу, що має позначення ‘/’. Всі зовнішні файлові системи (змінні носії інформації, мережеві диски і таке інше) монтуються у визначенні місця єдиного дерева файлової системи.

Як і в інших ієрархічних файлових системах, у файловій системі UNIX ім’я файлу повинно бути унікальним лише в межах одного каталогу. UNIX розрізняє великі і малі літери в назвах файлів. Для однозначної ідентифікації файлу в дереві каталогів слід указувати повний путь до файлу. Якщо путь починається з символу ‘/’ (наприклад, /usr/local/bin/cal), то він відраховується від кореневого каталогу (абсолютний путь), а якщо з іншого символу – то від поточного каталогу, тобто того, в якому користувач знаходиться в поточний момент (відносний путь). Крім того, поточний каталог позначається символом ‘.’ (крапка), каталог, що знаходиться на один рівень вище, тобто батьківський каталог – символом ‘..’ (дві крапки). Крім того, існує спеціальне позначення для так званого домашнього каталогу користувача, тобто каталогу, з якого він починає свою роботу – ‘~’ (тильда). Домашній каталог для кожного користувача також задається у файлі /etc/passwd, за замовчуванням це /home/<login>, або /usr/home/<login>.

Для переходу з каталогу в каталог існує команда cd <новий каталог> (change directory – змінити каталог). Якщо використати цю команду без параметрів, відбудеться перехід в домашній каталог користувача. Наприклад, якщо домашній каталог користувача /usr/home/user1, в поточний момент він знаходиться в каталозі /usr/local/samba, і бажає перейти в каталог /usr/local/bin, він може скористатись однією з наведених нижче команд:

cd /usr/local/bin

cd ../bin

cd ~/../../local/bin

Слід зазначити, що відносні путі слід використовувати з обережністю. Для того, щоби перевірити, в якому каталозі знаходиться користувач, можна скористатись командою pwd.

Перегляд вмісту каталогів здійснюється за допомогою команди ls, а розширений варіант цієї команди ls -l дає також інформацію з таблиці індексних дескрипторів. Щоби скопіювати файл, використовується команда cp <файл-джерело> <призначення>. Для перенесення файлу з каталогу в каталог, а також для перейменування файлу, використовується команда mv <файл-джерело> <призначення>. В обох командах в якості параметра <призначення> може задаватись каталог призначення або ім'я файлу призначення. Крім того, число параметрів може бути більше двох. В такому випадку всі параметри, крім останнього, розглядаються як список імен файлів-джерел, а останній параметр може бути лише каталогом призначення. Створити каталог можна командою mkdir, видалити файл – командою rm, видалити каталог – командою rmdir або rm -r.

Крім звичайних файлів існують різні типи спеціальних файлів. З одним із них ми вже познайомились – це каталоги. Ще одним типом спеціальних файлів є так звані посилання (англ. – link). В системі UNIX розрізняють два принципово різних типи посилань, хоча створюються вони однією командою – ln. Перший тип – це так звані жорсткі посилання. Фактично вони є абсолютно рівноправними новими іменами вже існуючого файлу. Після створення такого посилання система не розрізняє, яке ім'я було первинне, а яке було створене як посилання. Спроба видалити такий файл призводить до того, що одне з його імен (те, за яким ми видаляємо файл), знищується, а інші (як і сам файл) залишаються. Тільки після видалення останнього з імен фактично знищується сам файл. Другий тип посилання – символічне посилання, яке створюють командою ln –l. Це спеціальний тип файлу, який містить в собі ім'я того файлу (або каталогу), на який він посилається.

Більшість команд, що застосовуються по відношенню до посилання, діють безпосередньо на файл, на який посилання здійснене. При цьому деякі послідовності команд можуть привести до небажаних наслідків. Наприклад, маємо файл oldfile і бажаємо перейменувати його в newfile. Це можна зробити як командою: mv oldfile newfile.

так і послідовністю команд

cp oldfile newfile

rm oldfile

Результати будуть однакові. До речі, в одному командному рядку можна задати декілька команд, розділивши їх знаком ‘;’, ці команди будуть виконуватись послідовно: cp oldfile newfile; rm oldfile.

Тепер уявимо, що маємо файл targetfile і посилання на нього oldfile. Команда – mv oldfile newfile перейменує посилання, тобто тепер newfile буде посилатись на targetfile. Команда cp oldfile newfile скопіює не посилання, а сам файл targetfile, тобто під іменем newfile буде створено новий файл – копію targetfile. Наступна команда – rm oldfile знищить старе посилання, не пошкодивши при цьому файл targetfile. Тобто замість одного файлу з посиланням на нього у нас утворилися два ідентичних файла, які абсолютно не пов’язані між собою.

Індивідуальні завдання до лабораторного заняття №6

1. Завантажтеся в систему під вашим користувацьким ім'ям.
2. Виведіть системну дату.
3. Підрахуйте кількість рядків у файлі:

Варіант	Файл
1, 2, 4	/etc/passwd
3, 10	/etc/group
6, 9	/etc/profile
5, 7, 8	/etc/fstab

4. Виведіть на екран вміст відповідного файлу.
5. Виведіть календар на <1995+№варіанту> рік.
6. Виведіть календар на 1752 рік. Чи не помічаєте що-небудь цікаве у вересні? Поясніть.
7. Визначте, хто ще завантажений у систему.
8. Скопіюйте (скопіюйте, а не перемістіть, бо система перестане працювати коректно!) файли у ваш домашній каталог різними способами.

Таблиця 6.1 – Варіанти завдань

варіант	файл 1	файл 2
1	/bin/cat	/bin/at
2	/bin/cal	/bin/chmod
3	/bin/ls	/bin/chown
4	/bin/tee	/bin/file
5	/bin/more	/bin/gzip
6	/bin/date	/bin/gunzip
7	/bin/cp	/bin/ps
8	/bin/mv	/bin/csh
9	/bin/lpr	/bin/sh
10	/bin/find	/bin/ksh

9. Створіть каталог lab_6. Скопіюйте в цей каталог з вашого домашнього каталогу копію файлу 1, під ім'ям my_<ім’я файлу 1>.

10. Перемістіть в цей каталог з вашого домашнього каталогу копію файлу 2, перейменувавши його в my_<ім’я вихідного файлу 2>.

11. Створіть каталог lab_6_<№варіанту> і перейдіть в нього.

12. Скопіюйте в каталог lab_6_<№варіанту> файл з п.3 під ім'ям n<ім’я вихідного файлу>.

13. За допомогою команд cat і more перегляньте його вміст.

14. Перейдіть у свій домашній каталог та видаліть каталог lab_6_<№варіанту>.

Контрольні питання

1. Як переглянути вмісту каталогів?
2. Що виконує команда who?
3. Чи може Користувач працювати в системі, використовуючи одночасно кілька терміналів?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №7.
СИСТЕМА РОЗМЕЖУВАННЯ ДОСТУПУ В LINUX, ПРАВА ДОСТУПУ ДО ФАЙЛІВ І КЕРУВАННЯ НИМИ

Мета роботи – оволодіти практичними навичками керування правами доступу до файлів і їхній аналіз в ОС UNIX.

Література: [1, 2, 7, 8]

Теоретичні відомості

Складовою кожної захищеної системи є система розмежування доступу. В системі UNIX контролюється доступ до файлів. Кожний файл має свого власника, а також відноситься до визначеної групи. Вся детальна інформація про файл, що включає тип файлу, ідентифікатори власника файлу та його групи, розмір файлу, час останньої модифікації файлу, інформацію щодо прав доступу до файлу, а також про його розміщення (номери блоків), міститься не в каталогах, як це зроблено в багатьох інших файлових системах, в тому числі FAT, а в системній таблиці так званих індексних дескрипторів (i-node). Безпосереднє звернення до цієї таблиці заборонене. Каталоги містять лише ім’я файлу та індекс – посилання на відповідний запис в таблиці індексних дескрипторів. Саме завдяки такій організації, кожний файл в файловій системі UNIX може мати кілька абсолютно рівноправних імен (жорстких посилань), що в загальному випадку містяться в різних каталогах. Інформацію з таблиці індексних дескрипторів можна переглянути командою ls –l. При цьому для кожного файлу інформація видається у вигляді одного рядка. Перший символ – тип файлу: ‘–’ – звичайний файл (текстовий або бінарний), d –каталог (directory), l – символічне посилання (link), c – символьний пристрій (character device), b – блочний пристрій (block device). Наступні дев’ять символів описують права доступу до файлу (див. далі). Далі надається інформація про кількість жорстких посилань на файл, власника файлу, групу, розмір файлу, дату останньої модифікації і останнє поле – ім’я файлу.

UNIX реалізує дискреційну модель розмежування доступу, в якій для кожного файлу визначається, які права мають всі користувачі на доступ до файлу. Для цього з

кожним файлом асоціюється спеціальна інформація, що містить ідентифікатор власника файлу, ідентифікатор групи файлу і права доступу до файлу. Права доступу поділяються на 3 частині: права власника, права групи і права всіх інших. У кожному класі користувачів виділено по 4 біти. Перші (молодші) три біти відповідають правам читання, запису й виконання, відповідно. Для каталогів право виконання трактується як право доступу до таблиці індексних дескрипторів на читання і запис, не маючи цього права неможливо зробити поточним цей каталог чи будь-який з його підкаталогів, неможливо ознайомитись і змінити права доступу до об'єктів цього каталогу, можна тільки переглядати його вміст, якщо є право читання. Навіть маючи право запису, без права виконання не можна змінити вміст каталогу. Навпаки, якщо є право на виконання, але не встановлено право на читання для каталогу, то неможливо переглянути вміст каталогу, але можна заходити в його підкаталоги чи звертатись до файлів, що містяться в ньому, якщо знати їхні імена.

Четвертий біт має різне значення в залежності від того, в якій групі прав доступу він установлений. Для групи прав власника цей біт називається SUID (Set-User-ID-on-execution bit), і якщо він установлений для файлу, що виконується, то цей файл виконується для будь-якого користувача із правами власника цього файлу. Якщо четвертий біт встановлений у групі прав доступу членів групи (SGID – Set-Group-ID-on-execution bit), то ця програма виконується для будь-якого користувача із правами членів групи цього файлу. Для каталогів SGID визначає, що для усіх файлів, створюваних у цьому каталозі, ідентифікатор групи буде встановлений такий же, як у каталогу (ці правила залежать від версії UNIX).

Четвертий біт у групі прав доступу всіх інших користувачів називається Sticky, на сьогоднішній день він використовується тільки для каталогів і визначає, що користувачі, які мають право на запис в каталог, не мають права видаляти чи перейменовувати файли інших користувачів у цьому каталозі. Це необхідно для каталогів загального використання, наприклад /tmp.

Права доступу до файлів задаються при створенні файлу і в подальшому можуть бути змінені командою chmod <нові права> <файл(и)>

<нові права> задаються двома способами. Перший – символьний. Використовуються такі позначення: u – власник файлу (user), g – група (group), o – всі інші (others), a – всі категорії користувачів одночасно (all). Після категорії

користувачів задається дія: ‘+’ – додати права до існуючих, ‘-’ – відібрати права (якщо існують), ‘=’ – встановити права замість існуючих. Далі позначаються права: r – зчитувати (Read), w – записувати (Write), x – виконувати (eXecute). Можна формувати список зміни прав, розділяючи окремі категорії користувачів комами (без пробілів). Наприклад, команда chmod u+rw,g=rx,o-w my_file встановить для користувача права на зчитування і записування (право на виконання для користувача буде залежати від того, чи було воно встановлено раніше), для групи встановить права на зчитування і виконання (незалежно від того, які права були раніше), а для всіх інших гарантовано заборонить записування (права на зчитування і виконання будуть встановлені в залежності від того, чи були вони встановлені раніше).

Другий – числовий спосіб задавання прав доступу – зручніше використовувати, коли треба встановлювати нові права незалежно від попередньо встановлених. При цьому використовується представлення у восьмеричній системі числення, яке легко зрозуміти з наведеної таблиці:

Таблиця 7.1 – Права доступу

Восьмеричне представлення	Двійкове представлення	Права доступу
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

Наприклад, команда: chmod 751 my_file встановлює такі права доступу до файлу my_file: власнику – зчитування, записування, виконання, групі – зчитування і виконання, а всім іншим – лише виконання. Команда ls -l покаже для цього файлу таку інформацію про права доступу: -rwxr-x--x (перший “мінус” означає, що це звичайний файл).

За замовчуванням для всіх нових файлів, що створюються, встановлюються такі права: для файлів 666, тобто rw-rw-rw-, для каталогів – 777, тобто rwxrwxrwx (ми вже бачили вище, що наявність права на виконання дуже важлива для каталогів). Є особлива команда – umask, яка дозволяє зменшити права доступу, що встановлюються для нових файлів. Параметром цієї команди є восьмерична маска, аналогічна тій, що

використовується при числовому способі задавання параметрів команди chmod, але у випадку команди umask права доступу, що задані нею, будуть відніматись від прав доступу, що були задані по замовчуванню для вже утвореного файлу. Команда umask не може додавати прав, тому нові файли ніколи автоматично не отримають право на виконання, в разі необхідності його треба буде додати вручну. Дія команди umask розповсюджується на всі файли всіх типів, що утворюються протягом поточної сесії користувача після цієї команди. Наприклад, після команди umask 123 всі файли будуть утворюватись з параметрами доступу rw-r--r-- (644), а всі каталоги – rw-r-xr-- (654). За замовчуванням рекомендується використовувати команду umask 22 (інтерпретується як umask 022).

Для зміни власника файлу існує команда chown. З міркувань безпеки, ця команда дозволяє встановлювати власником файла будь-кого лише системному адміністратору (root'у). Інші користувачі можуть лише привласнити файл собі, якщо вони мають на це права, встановлені правами доступу до файлу і каталогу.

Індивідуальне завдання до лабораторного заняття №7

1. Створіть каталог lab_7.
2. Скопіюйте в каталог lab_7 файл /bin/cat під назвою my_cat
3. За допомогою файлу my_cat, що знаходиться в каталозі lab_7, перегляньте уміст файлу .profile (Ви знаходитесь у домашньому каталозі).
4. Перегляньте список файлів у каталозі lab_7. Потім перегляньте список усіх файлів, включаючи приховані, з повною інформацією про файли. Зверніть увагу на права доступу, власника, дату модифікації файлу, що ви тільки-но скопіювали. Потім перегляньте цю інформацію про оригінальний файл (той, який копіювали) і порівняйте два результати.
5. Змініть права доступу до файлу my_cat так, щоб власник міг тільки читати цей файл.
6. Переконайтеся в тім, що ви зробили ці зміни і повторіть п.3.
7. Визначте права на файл my_cat таким чином, щоб Ви могли робити з файлом усе, що завгодно, а всі інші – нічого не могли робити.
8. Поверніться в домашній каталог. Змініть права доступу до каталогу lab_7 так, щоб ви могли його тільки читати.

9. Спробуйте переглянути простий список файлів у цьому каталозі. Спробуйте переглянути список файлів з повною інформацією про них. Спробуйте запустити і видалити файл my_cat з цього каталогу.

10. Поясніть отримані результати. Результати виконання п.8 можуть бути різними в різних версіях UNIX, зокрема, Linux і FreeBSD. Прокоментуйте отримані результати у висновках.

11. Завантажтесь в систему, користуючись обліковим записом іншого користувача. (Вам потрібно знати пароль цього користувача.) Якщо Ви не маєте прав адміністратора, попросіть адміністратора створити для Вас тимчасово інший обліковий запис. Спробуйте отримати доступ до Вашого каталогу lab_7. Перевірте, чи правильно зроблено завдання попереднього пункту. Створіть каталог lab_7_2.

12. Знову завантажтесь в систему, користуючись своїм обліковим записомі. Спробуйте зробити власником каталогу lab_7 іншого користувача. Спробуйте зробити себе власником каталогу lab_7_2. Поясніть результати.

13. Зайдіть у каталог lab_7. Зробіть так, щоб нові створені файли і каталоги діставали права доступу згідно Таблиці. Створіть новий файл і каталог і переконайтеся в правильності ваших установок.

Таблиця 7.1 – Варіанти завдань

Варіант	Права для файлів	Права для каталогів
1	644	754
2	664	774
3	6-4	7-5
4	62-	73-
5	644	745
6	664	764
7	6-4	715
8	62-	63-
9	644	744
10	664	765
11	644	755
12	62-	754
13	644	715
14	644	7-5
15	6-4	745
16	664	7-5
17	6-4	754
18	644	715
19	644	7-4
20	62-	73-

Контрольні питання

1. Що виконує команда chmod u+rw,g=rx,o-w my_file?
2. Що виконує команда chmod 751 my_file
3. Що виконує команда chown?
4. Що дає права доступу 777? Чи варто його використовувати?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №8.
КОМАНДНА ОБОЛОНКА SHELL: СТАНДАРТНІ ПОТОКИ ВВОДУ/ВИВОДУ

Мета роботи – ознайомитися з командною оболонкою Shell та оволодіти практичними навичками перенаправлення стандартних потоків.

Література [1, 2, 3, 5, 6]

Теоретичні відомості

Shell (або «шелл», «командна оболонка») – це не тільки командний інтерпретатор, який забезпечує інтерфейс взаємодії між користувачем та ядром операційної системи, але й своєрідна мова програмування, в якій присутні такі конструкції, як оператори умовного розгалуження, цикли, змінні та багато іншого.

У попередніх роботах ми вже познайомились з командними оболонками (shell). У цій роботі розглянемо прийоми професійної роботи з командними оболонками, а саме використання змінних оточення і створення командних файлів.

Командний файл, або сценарій (також дуже часто кажуть “скрипт” від англійського script – сценарій) є текстовим файлом, який оформлено з дотриманням певних правил, і який містить команди, у найпростішому випадку повністю аналогічні тим командам, що вводяться з клавіатури. Командна оболонка здатна запускати такий файл на виконання і послідовно виконувати команди, що містяться в ньому. Для користувача, що запустив цей сценарій, його виконання буде виглядати як виконання звичайної програми.

Зверніть увагу на розбіжності у різних програмних оболонках shell, які суттєві для програмування. Під час виконання роботи впевніться, в якій із програмних оболонок Ви працюєте (зазвичай, для FreeBSD це csh чи tcsh, а для Linux – bash, який є розвитком sh), і яка буде запускатись для виконання Вашого командного файлу (це визначається першим рядком Вашого командного файлу). Уважно прочитайте правила використання операторів if і формування перевірки відповідної умови. Зверніть увагу на команду test.

Важливою можливістю командних оболонок (усіх) є обробка так званих пакетних файлів.

Кожна командна оболонка в процесі свого запуску робить налаштування системного оточення (виконує ініціалізацію системних змінних та змінних командної оболонки), для чого використовує визначені файли. Їх щонайменше два – глобальний і локальний (деякі оболонки можуть використовувати більшу кількість конфігураційних файлів). Глобальні конфігураційні файли для всіх оболонок знаходяться в каталозі /etc. Локальні конфігураційні файли для всіх оболонок знаходяться в домашньому каталозі користувача. Імена конфігураційних файлів, як правило, закінчуються на ‘rc’. Локальні файли роблять “прихованими”, щоби вони не заважали користувачу в його повсякденній роботі (приховані файли мають ім’я, що починається з символу ‘.’, команда ls без параметрів їх не показує). Приклади конфігураційних файлів: для sh – /etc/profile і ~/.profile, для csh – /etc/cshrc і ~/.cshrc, для tcsh – /etc/cshrc і ~/.cshrc.

Часто в якості параметру деякої команди нам треба вказати не один файл, а кілька файлів, назви яких мають певні спільні риси. В таких випадках використовують так звані маски пошуку. Спеціальний символ ‘?’ в масці означає один будь-який символ, а спеціальний символ ‘*’ – будь-яку послідовність символів. Наприклад, команда ls /bin/???? виведе на екран всі файли з каталогу /bin, імена яких складаються з чотирьох символів, а команда ls /etc/d* виведе на екран всі файли з каталогу /etc, імена яких починаються з літери d. Також можна задати список символів, наприклад, маска [abc]??? задає ім’я з чотирьох літер, перша з яких – a, b чи c.

Для пошуку файлів за певними ознаками можна використовувати команду find. Перший параметр цієї команди (обов’язковий) – це каталог, з якого починається пошук (наприклад, / – кореневий каталог), далі – параметр, що задає ознаку пошуку (наприклад, -name – пошук файлів, імена яких відповідають заданій масці, –atime – пошук файлів, дата модифікації яких відповідає заданій умові), далі іде власне маска пошуку, а далі – дія. Найтиповіша дія – –print, вивід результатів пошуку на екран. Якщо цей параметр не вказати, пошук відбуватись буде, а от результатів його видно не буде.

Наприклад:

find / -name "*.c" –print.

Ще одна можливість оболонок – перенаправлення потоків введення-виведення. Як правило, більшість команд (утиліт) приймає інформацію з клавіатури, або з файлу, якщо його вказано як параметр, і виводить результати на екран. Однак, фактично вони працюють із так званими стандартними потоками введення і виведення, які пов'язані з певними файлами. Файл в UNIX-подібних системах розглядається як потік байт. Оскільки пристрої в UNIX-подібних системах розглядаються як файли, а операції введення і виведення – як зчитування і записування у відповідні файли, це дозволяє легко переводити вхідний і вихідний потоки з файлів на пристрої чи навпаки.

Стандартний потік введення за умовчанням зчитується з клавіатури. Якщо в якості параметру вказано ім'я файлу, то замість стандартного введення відповідна утиліта буде організовувати вхідний потік з указаного файлу (але так діють не всі команди!) Вихідних потоків є два – стандартний потік виведення (за умовчанням у сучасних системах – на екран монітору), і потік повідомлень про помилки (за умовчанням – туди ж).

Оболонка дає змогу перенаправити потоки у заданий файл. Символ '<' перенаправляє вхідний потік. Після цього символу очікується ім'я файлу або пристрою, з якого буде братись вхідний потік.

Наприклад, команда cat без параметрів очікує введення з клавіатури, і кожний рядок передає на екран монітора. Команда cat my_file замість введення з клавіатури виведе на екран вміст файлу my_file, якщо такий існує, і повідомлення про помилку, якщо такого не існує.

Перенаправлення введення-виведення широко використовується у двох випадках. Перший – це запуск утиліт у фоновому режимі. Щоб робота фонових утиліт не заважала роботі користувача з терміналом, слід так перенаправити потоки введення- виведення, щоб вони працювали лише з файлами. Другий – це використання спеціальних команд-утиліт, які призначені саме для того, щоби прийняти певну інформацію з одного файлу, обробити її, а результат записати у другий файл. Такі утиліти називаються фільтрами. Утиліта cat, варіанти використання якої з перенаправленням потоків було розглянуто вище – це простіший фільтр. Він практично не обробляє інформацію, лише може зчіплювати кілька файлів в один. Інші корисні фільтри: cut, grep, sort.

Утиліта cut переглядає вхідний файл, і виділяє з кожного його рядка інформацію за ознаками розміщення в певних колонках або полях. Наприклад, рядки файлу /etc/passwd розділяються на поля за допомогою символу ‘:’. Перше поле – login, п’яте поле – інформація про користувача. Якщо ми хочемо надрукувати лише цю інформацію, ми можемо дати команду:

cut –d: –f1,5 < /etc/passwd

Ключ –d задає символ-роздільник полів (у цьому випадку ‘:’), ключ –f – список полів, що треба роздрукувати (у цьому випадку 1 і 5).

Утиліта grep виводить лише ті рядки, в яких зустрічається заданий рядок пошуку. Утиліта sort виконує сортування вхідного потоку, наприклад, за абеткою. Докладніше про ці та інші фільтри вам слід дізнатися з довідкової системи man.

Індивідуальне завдання до лабораторного заняття №8

1. Перейдіть у каталог /bin. Перегляньте список усіх файлів, що починаються із символу, який визначено в таблиці індивідуальних завдань.
2. Перегляньте список файлів, імена яких складаються з визначеної у таблиці індивідуальних завдань кількості символів.
3. Перегляньте список файлів, імена яких починаються із символів, які визначено в таблиці індивідуальних завдань. Зробіть це декількома способами.
4. Створіть в Вашому домашньому каталозі підкаталог lab_8 і перейдіть в нього.
5. За допомогою команди cat створіть файл my_text і запишіть у нього кілька рядків. Потім за допомогою команди cat допишіть у нього ще кілька рядків.

Таблиця 8. 1 – Варіанти індивідуальних завдань

варіант	п.1	п.2	п.3	п.6
1	a	2	a, b, c, d	/bin
2	b	3	e, f, g, h	/usr
3	c	4	i, j, k, l	/usr/bin
4	d	5	m, n, o, p	/home
5	f	2	q, r, s, t	/var

6	g	3	u, v, w	/home
7	h	4	x, y, z	Ваш домашній каталог
8	k	5	a, d, k, l	/tmp (або /var/tmp)
9	l	3	m, g, y	/sbin
10	n	2	x, z, r, q	/usr/sbin
11	r	5	t, r, e	/var
12	m	3	u, s, a	/home
13	q	4	m, e, z	Ваш домашній каталог
14	z	2	t, o, d	/home
15	p	6	x, z, q	/var
16	s	5	m, g, y, a	/usr/bin
17	r	4	t, o, d, y	/sbin
18	u	2	u, a, r, z	/home
19	w	3	a, d, k	/usr/bin
20	y	6	a, d, l	/var

6. Підрахуйте кількість файлів у каталозі, визначеному з таблиці індивідуальних завдань, при цьому зберігши список файлів у файлі filelist, використовуючи команду tee.

7. Починаючи з Вашого домашнього каталогу, виведіть на екран у повному форматі назви усіх файлів і каталогів, що починаються на ‘m’. При цьому перед виводом кожної назви на екран повинен виводитися запит на його підтвердження.

8. Починаючи з кореневого каталогу, виведіть на екран імена всіх каталогів, що останній раз змінювалися більш 10 днів назад.

9. Виведіть на екран тільки час, що повертається командою date.

10. Виведіть на екран список усіх користувачів системи, тобто перші поля кожного рядка файлу /etc/passwd (роздільник полів – символ ‘:’).

11. Виведіть на екран імена усіх файлів у каталозі /bin, що містять слова Software чи software. Потік помилок при цьому не повинний виводитися на екран.

Контрольні питання

1. Що таке Shell?
2. Коли використовується перенаправлення введення-виведення?
3. Що виконує утиліта cut?

4. Перенаправлення стандартного виводу?
5. Перенаправлення стандартного вводу?
6. Перенаправлення стандартного виводу помилок?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №9
КОМАНДНА ОБОЛОНКА SHELL: ФІЛЬТРИ І КОНВЕЄРИ

Мета роботи – Оволодіння практичними навичками роботи з фільтрами і організації конвеєрів.

Література [1, 2, 5,6]

Теоретичні відомості

Фільтри командної оболонки (shell) – це команди, які приймають вхідні дані, обробляють їх і виводять результат. Вони часто використовуються для обробки текстових файлів і потоків даних. Ось кілька популярних фільтрів у Unix-подібних системах:

grep: Використовується для пошуку тексту за шаблоном.

grep "pattern" filename

awk: Потужний інструмент для обробки та аналізу текстових файлів.

awk '{print $1}' filename

sed: Потоковий редактор, який використовується для пошуку та заміни тексту.

sed 's/old/new/g' filename

sort: Сортує рядки у файлі.

sort filename

uniq: Видаляє дублікати рядків у відсортованому файлі.

sort filename | uniq

cut: Вирізає певні поля або символи з кожного рядка.

cut -d',' -f1 filename

tr: Перетворює або видаляє символи.

tr 'a-z' 'A-Z' < filename

wc: Підраховує кількість рядків, слів і символів у файлі.

wc filename

Ці фільтри можна комбінувати за допомогою конвеєрів (pipes) для створення складних командних ланцюжків. Наприклад:

cat filename | grep "pattern" | sort | uniq

В Unix-подібних операційних системах конвеєр – це механізм міжпроцесної взаємодії, що використовує передачу пов*ід*омлень. Є набором поєднаних між собою процесів, які забезпечують передачу даних так, що стандартний вихідний потік кожного процесу (stdout) безпосередньо з'єднується зі стандартним вхідним потоком (stdin) наступного. Наступний процес починається, поки перший ще виконується і вони виконуються одночасно. Концепцію було запропоновано Дугласом Макілроєм в Bell Labs, колисці Unix, в процесі його розробки та формування філософії інструментизації. Назву дано за аналогією зі справжніми конвеєрами. Ключовою функцією цих конвеєрів є «приховування нутрощів» (Ritchie & Thompson, 1974). Це забезпечує системі чистоту за простоту.

Існує можливість перенаправити вихідний потік однієї утиліти безпосередньо у вхідний потік іншої, без використання тимчасових файлів. Це так звані конвеєри. В системі UNIX всі утиліти, що поєднані в конвеєр, запускаються паралельно і обробляють інформацію по мірі її надходження. Конвеєр утворюється за допомогою символу ‘|’ таким чином:

util1 | util2 | util3

При утворені конвеєра окремо перенаправляти вхідні й вихідні потоки на проміжних стадіях не можна – це буде або сприйнято як синтаксична помилка, або результат може бути непередбачуваним.

Приклад конвеєру:

ps –al | grep root | more

Команда ps з ключами –al направить у вихідний потік список всіх процесів у системі, grep root вибере з них лише ті, які виконуються від імені root’a, more забезпечить вивід їх на екран посторінково. Інший приклад:

cat /etc/passwd | cut –d: –f1,5 | more

Ця команда зробить те ж саме, що й приклад з командою cut, що розглядався раніше, але вивід на екран буде посторінковим.

Якщо проміжні результати на якійсь із стадій конвеєра бажано зберегти, можна скористатись командою tee my_file. Ця команда візьме вхідний потік, передасть його

без змін у вихідний потік і одночасно продублює у файл my_file1. Наприклад, так можна модифікувати один із розглянутих вище прикладів:

ps –al | grep root | tee my_file | more

Тепер ми не лише побачимо на екрані посторінково виведений список всіх процесів root'а, але й збережемо його у файлі my_file.

Наприклад, щоб отримати список файлів в поточній директорії, зберегти лише стрічку ls виводу, що має рядок "key" (grep) і показати результат на сторінці, що прогортується (less), користувач може написати в командний рядок терміналу наступне:

```
ls -l | grep key | less
```

Команда ls –l виконується як процес, вивід (stdout) якого передається вводу (stdin) процесу команди grep key, так само з командою less. Кожен процес приймає ввід від попереднього і передає свій вивід наступному стандартними потоками. Кожен | вказує командній оболонці під'єднати стандартний вивід команди зліва до стандартного вводу команди справа через механізм взаємодії між процесами, що називають неіменованим каналом, вбудованим в операційну систему. Канали є односторонніми: дані переходять каналом лише зліва направо.

Нижче наведений приклад конвеєра, що реалізовує перевірку орфографії для веб-сторінки, на яку вказує посилання.

```
curl "https://en.wikipedia.org/wiki/Pipeline_(Unix)" |
sed 's/[^a-zA-Z ]/ /g' |
tr 'A-Z ' 'a-z\n' |
grep '[a-z]' |
sort -u |
comm -23 – <(sort /usr/share/dict/words) |
less
```

1. curl отримує html вміст вебсторінки (для деяких систем використовують wget).

2. sed замінює всі символи (з вмісту вебсторінки), що не є пробілами чи літерами, на пробіли (зі збереженням переходу на новий рядок)

3. tr змінює всі великі літери на малі і конвертує пробіли у переходи на новий рядок (кожне «слово» тепер в окремому рядку).

4. grep приймає лише ті рядки, які містять щонайменше одну малу літеру алфавіту, таким чином виключаючи пусті рядки.

5. sort сортує список «слів» в алфавітному порядку, а -u видаляє дублікати.

6. comm шукає стрічки, що є однаковими в обидвох файлах, -23виключає вивід стрічок, що є унікальними в другому файлі та однаковими в обох файлах, виводячи лише стрічки, унікальні для першого файлу. – на місці назви файлу вказує процесу comm використовувати стандартний потік вводу (в даному випадку з конвеєра). sort /usr/share/dict/words сортує вміст файлу words за алфавітом (за вимогами процесу comm). <(...)виводить результат виконання в тимчасовий файл (використовуючи процесне заміщення), який зчитує comm. Результатом є список слів (стрічок), не знайдених в /usr/share/dict/words

7. less дозволяє користувачеві розділити результат на сторінки.

Індивідуальне завдання до лабораторного заняття №9

1. Підрахуйте кількість файлів у каталозі, визначеному з таблиці індивідуальних завдань, використовуючи і не використовуючи конвеєри. Детально порівняйте результат.

Таблиця індивідуальних завдань

варіант	каталог	варіант	Каталог
1	/bin	11	/var
2	/usr	12	/home
3	/usr/bin	13	Ваш домашній каталог
4	/home	14	/home
5	/var	15	/var
6	/home	16	/usr/bin
7	Ваш домашній каталог	17	/sbin
8	/tmp (або /var/tmp)	18	/home
9	/sbin	19	/usr/bin
10	/usr/sbin	20	/var

2. Відсортуйте конфігураційний файл Вашого shell (.profile, .cshrc) відповідно до кодової таблиці ASCII так, щоб при цьому ігнорувалися пробіли на початку рядків.

3. Навести власний приклад конвеєра, який здійснює перевірку.

Контрольні питання

1. Що таке конвеєр?
2. Що містить директорія / bin?
3. В якому каталозі містяться основні спільні бібліотеки?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №10-11
РЕДАКТОР VI

Мета роботи – оволодіти практичними навичками роботи з редактором Vi. Вивчити: завантаження редактора і вихід з нього; завантаження і збереження файлів; редагування тексту; виконання команд UNIX.

Література [1, 2, 7, 8]

Теоретичні відомості

Текстові редактори присутні в усіх операційних системах. Вони використовуються для створення і редагування текстових файлів. Текстові файли можуть містити документи, програмні коди і конфігураційні параметри.

Документи можуть містити ознаки форматування (стиль і розмір шрифту, поля, відступи, тощо). Іноді такі ознаки можна розмістити безпосередньо в текстовому документі, використовуючи деякий визначений формат розмітки. Прикладами є формати TEX, вже забутий ChiWriter, а також сучасні HTML і XML. Деякі формати документів відходять від використання текстових файлів, як, наприклад, формати документів Microsoft Office. В будь-якому випадку, для редагування документів переважно використовують специфічний клас редакторів – Word Processors.

Для власних потреб операційної системи більше значення мають класичні текстові редактори, важливою ознакою яких є те, що під час редагування вони відображають весь вміст файлу, не приховуючи спеціальні символи чи ключові слова, а під час збереження файлу також не додають від себе додаткових керуючих символів. Це необхідно для можливості роботи з файлами, що містять коди програм або перелік конфігураційних параметрів. Система UNIX дуже інтенсивно використовує текстові файли. Будь-який текстовий файл може розглядатись системою як послідовність команд (так званий пакетний файл, або файл сценарію, див. Лабораторну роботу №6). Крім того, налаштовування практично всіх програм, як тих, що входять до складу операційної системи, так і прикладних програм, здійснюється за допомогою текстових файлів.

Тому ефективні засоби редагування текстових файлів мають виключне значення для UNIX. Одним із таких засобів є редактор vi. Він традиційно входить до поставки

будь-якої системи UNIX, і майже напевно присутній в кожній системі (звичайно, адміністратор може його видалити, якщо віддає перевагу іншому редактору, але заради зручності інших користувачів він не повинен так діяти).

Редактор vi пропонує могутній набір операцій для редагування тексту, заснований на визначеній множині мнемонічних команд. Більшість команд викликаються натисканням одиночних клавіш і виконують прості функції редагування. На відміну від більшості сучасних редакторів, vi взагалі не має системи меню. Не має він і вбудованої системи підказок. Для роботи з цим редактором необхідно запам’ятати команди. Однак для тих, хто знає команди, ефективність роботи в цьому редакторі неперевершена.

Редактор vi відкриває "вікно" розміром з екран дисплея, у якому ви можете редагувати ваш файл. При редагуванні забезпечується зворотний візуальний зв'язок (ім'я vi – скорочення від слова "visual"). Нижній рядок екрану використовується як інформаційний і командний рядок.

Редактор vi тісно співпрацює із строковим редактором ex. vi і ex – це той самий редактор: імена vi і ex ідентифікують скоріше особливий інтерфейс користувача, чим функціональне розходження. Розходження в інтерфейсі користувача, однак, зовсім разюче. ex – могутній строчно-орієнтований редактор, схожий з редактором ed. Однак, і в ex, і в ed візуальне коректування екрана термінала обмежено, а команди вводяться з командного рядка. На відміну від них, vi – це екранно-орієнтований редактор, влаштований таким чином, що те, що ви бачите на екрані, точно відповідає вмісту файлу, який ви редагуєте.

Запуск редактора

vi [option...] [command...] [filename...]

view [option...] [command...] [filename...]

Команда view аналогічна vi, за винятком того, що автоматично встановлюється опція “тільки зчитування” (-R). При використанні view файл змінюватися не може.

Крім вже згаданої опції –R, у командному рядку vi припустимі інші опції, з яких слід відзначити -r (не плутати великі і маленькі літери!). Ця опція використовується при відновленні, коли мало місце ушкодження редактора чи всієї системи. vi –r

<filename> відшукує останню збережену версію зазначеного файлу. Якщо файл не визначений, то ця опція виводить список збережених файлів.

vi не виконує ніяких операцій редагування безпосередньо над зазначеним вами файлом. Замість цього він працює з копією вашого файлу, що знаходиться в буфері редагування. Коли ви активізуєте vi з одним аргументом – ім'ям файлу, цей файл копіюється в тимчасовий буфер редагування. Редактор запам'ятовує ім'я файлу, визначеного при виклику, таким чином, що пізніше він може скопіювати вміст буфера редагування назад у зазначений файл. Вміст заданого файлу не змінюється доти, поки всі зміни не будуть скопійовані назад у первісний файл.

Приклади команд, що можна застосовувати для входу в редактор vi:

vi	Редагує порожній буфер редагування
vi <file>	Відкриває для редагування зазначений файл
vi +123 <file>	Відкриває для редагування зазначений файл і переходить в ньому на рядок з номером 123
vi +/<word> <file>	Відкриває для редагування зазначений файл і шукає перше входження слова “word”

Режими роботи

У Vi існує три окремих режими:

Командний режим: у цьому режимі сигнал із клавіатури інтерпретується як команда редагування.

Режим вставки: перейти в цей режим можна набором будь-яких команд вставки, приєднання, відкриття, підстановки, зміни чи заміщення, що існують в vi. У цьому режимі символи, набрані на клавіатурі, вставляються в буфер редагування.

Режим переключення в 'ex': У vi команди – це одиночні клавіші. У ex командами є рядки тексту, завершені натисканням клавіші RETURN. vi має спеціальну команду "escape", що дозволяє перейти до більшості строчно-орієнтованих команд редактора ex. Для використання режиму переключення в ex наберіть символ ‘:’. Цей символ відобразиться в командному рядку як покажчик на наступну команду редактора ex. Більшість команд обробки файлу виконується в режимі переключення в ex (наприклад, команди читання з файлу і запису з буфера редагування назад у файл).

Команди переключення між режимами

ESC	Переводить з режиму вставки в командний режим. Якщо не впевнені, в якому режимі знаходитесь – краще зайвий раз натиснути ESC
	Команди вставки: переводять з командного режиму в режим вставки. При застосуванні команд вставки текст не знищується.
I i	Команди 'i' та 'I' – “insert”. Команда 'i': починає вставляти текст перед символом, що розташований під курсором. Для вставки нового рядка натисніть RETURN. Команда 'I' починає вставляти текст із початку поточного рядка.
A a	Команди 'a' та 'A' – “append”. Команда 'a' працює точно так, як команда 'i', тільки вставка тексту починається після курсору, а не перед ним. Це є однією з можливостей додавання тексту в кінець рядка. Команда 'A' починає вставку тексту наприкінці поточного рядка.
O o	Команди 'o' та 'O' – “open”. Відкривають новий рядок і вставляють текст. Команда 'o' відкриває новий рядок під поточним рядком, команда 'O' відкриває новий рядок над поточним рядком. Після того, як новий рядок відкритий, обидві команди працюють аналогічно команді 'I'.
	Команди переходу в режим редактора ‘ex’.
Q	Переводить з командного режиму vi в режим редактора ex. Після виконання цієї команди буде продовжуватись редагування того самого файлу. Ви можете повернутися в режим редактора vi, набравши 'vi' у редакторі ex.
:	Переводить в режим тимчасового переключення в 'ex' для виконання однієї команди редактора ex. В інформаційному рядку з'являється двокрапка ‘:’ як підказка-вказівка для введення ex-команди. Ви можете ввести ex-команду, закінчити її символами RETURN чи ESC, після чого цю команду буде виконано. Потім вам запропонують натиснути клавішу RETURN для повернення в командний режим vi.

Команди виходу

Існує кілька шляхів виходу з редактора vi. Якщо ви знаходитесь в режимі вставки, то перший крок – перейти в командний режим (клавіша ESC). Далі можна використовувати такі команди:

ZZ	Здійснює вихід з командного режиму vi. Вміст буфера редагування записується у файл, що редагувався, тільки за умови, що були зроблені які-небудь зміни. При цьому ви повертаєтесь в ту командну оболонку, з якої був запущений редактор vi.
:x	Діє так само, як і ZZ. Фактично означає перехід в режим 'ex' з наступним виконанням команди x.
:q	Намагається здійснити вихід з редактора vi без запису з буфера редагування у файл. Однак, якщо з моменту останньої команди 'w' до буферу редагування вносились зміни, то vi видає попереджуюче повідомлення, і вихід з редактора не здійснюється. vi також видає діагностику, якщо в списку аргументів залишилися імена ще невідредагованих файлів. Зазвичай ви хочете зберегти усі ваші зміни; для цього вам належить набрати команду

	':w'.
:q!	На відміну від попередньої, ця команда скасовує сеанс редагування. Знак оклику вказує редакторові vi на необхідність безумовного виходу. Вміст буфера редагування не зберігається.
:wq	Переходить в режим 'ex', зберігає зміни буфера редагування в поточний файл (команда w), після чого здійснює вихід з редактора (команда q).
:wq!	Скасовує перевірку, що звичайно виконується перед командою 'w'. Наприклад, якщо ви володієте файлом, але не маєте дозволу на запис у нього, команда ':wq!' дозволить вам змінити вміст.

Увага! Для виходу з редактора vi (як і з інших програм UNIX) не можна використовувати комбінацію клавіш CTRL+Z, оскільки ця комбінація не завершує програму, а лише призупиняє її.

У режимі вставки можуть використовуватися такі символи:

BKSP	У поточному рядку повертає курсор назад на один символ. Останній символ, набраний перед BKSP, видаляється з вхідного буфера, але залишається на екрані дисплея.
Ctrl-V	Відміняє спеціальне значення наступного набраного символу. Використовуйте Ctrl-V для вставки керуючих символів.
Клавіші курсору	Можуть діяти нормально (переміщати курсор) або ні (вводити якісь незрозумілі символи) в залежності від версії редактора і налаштовувань терміналу

Переважна більшість команд, що діють в командному режимі vi, в режимі вставки діяти не можуть, оскільки викликають вставку в буфер редагування набраних літер.

Команди редактора vi, що діють в командному режимі

Далі описані лише основні, найнеобхідніші команди. Більшість команд можуть використовувати лічильник, що стоїть перед ними і показує число повторів команди. Цей параметр у наступних описах команд не заданий, але він мається на увазі, якщо не скасований яким-небудь аргументом, що стоїть перед ним. Коли редактор vi одержує команду неправильного формату, він сигналізує про це.

Переміщення курсору

Команди керування курсором дозволяють вам переміщати курсор по файлу. Вони дуже важливі тому, що деякі команди з інших груп (див. далі) використовують команди переміщення курсору як аргумент, що дозволяє визначити блок тексту. Поточна позиція курсору розглядається як початок блоку, а позиція курсору, яку він займе в разі виконання команди переміщення, – як кінець блоку (або навпаки, якщо

переміщення здійснюється не вперед, а назад). Виконання таких команд відбувається лише після введення команди переміщення курсору.

l **SPACEBAR** →	Переміщає курсор на один символ вперед. Якщо заданий лічильник, то переміщає вперед на зазначене число символів. Ви не можете переміститися за межу кінця рядка.
h **BKSP** ←	Переміщає курсор на один символ назад. Якщо заданий лічильник, то переміщає назад на зазначене число символів. Ви не можете переміститися за межу початку рядка.
+ **RETURN**	Переміщає курсор вниз на початок наступного рядка.
J **Ctrl-N** ↓	Переміщає курсор вниз на один рядок, залишаючи його в тому ж стовпчику.
–	Переміщає курсор на початок попереднього рядка. Якщо заданий лічильник, то курсор переміщається вгору на зазначене число рядків.
K **Ctrl-P** ↑	Переміщає курсор на один рядок вгору, залишаючи його в тому ж стовпчику. Якщо заданий лічильник, то курсор переміщається на зазначене число рядків.
O	Переміщає курсор на перший символ поточного рядка.
^	Переміщає курсор на перший символ рядка, такий, що не є міткою табуляції чи пробілом. Це дуже зручно при редагуванні тексту з форматованими відступами (наприклад, тексту програми).
$	Переміщає курсор у кінець поточного рядка. Зверніть увагу, що курсор знаходиться над останнім символом у рядку.
w **b** **e**	Переміщає курсор вперед на початок наступного слова (**w**), назад на початок поточного слова (**b**), або на останній символ поточного слова (**e**), де слово визначене як рядок буквено-числових символів, розділених пунктуацією, мітками табуляції, символами нового рядка чи пробілами.
%	Переміщає курсор на узгоджуючий роздільник, яким можуть бути кругла, операторна чи фігурна дужки. Це дуже зручно при узгодженні пар вкладених круглих, операторних і фігурних дужок (наприклад, при редагуванні тексту програми).

Команди екрана

Команди екрана не є командами керування переміщенням курсору, і не можуть використовуватися як роздільники об'єктів тексту. Однак команди екрана здійснюють переміщення тексту і дуже зручні для сторінкової організації чи "прокручування" інформації з файлу на екрані дисплея.

Ctrl-U **Ctrl-D**	"Прокручує" екран на половину вікна нагору (**Ctrl-U**) чи вниз (**Ctrl-D**).
Ctrl-F **Ctrl-B**	Посторінково перегортає екран уперед та назад. Якщо можливо, то між сторінками зберігаються два нерозривні рядки. Заданий попереду лічильник указує число сторінок, на яке треба пересунути вперед чи назад.
Ctrl-G	Показує стан редактора **vi** в інформаційному рядку: ім'я файлу, що редагується, чи був він змінений, номер поточного рядка, число рядків у файлі і відсоток файлу (у рядках), що передує місцеві перебування курсору.
Ctrl-R **Ctrl-L**	Перемальовує вміст екрана. Використовуйте цю команду для витирання будь-яких системних повідомлень, що можуть накладатися на інформацію, що міститься на вашому екрані. Зверніть увагу, що системні повідомлення не впливають на файл, що редагується вами.

Найбільш універсальна команда видалення тексту використовує як оператор виконання клавішу 'd'. Цей оператор видаляє текстові об'єкти, обмежені поточною позицією курсору і командою переміщення курсору. Видалення фактично відбувається після завершення вводу команди переміщення курсору. Вилучений текст завжди продовжує зберігатися в буфері.

x	Видаляє символ, що знаходиться під курсором.
<*n*>**x**	Видаляється <*n*> символів вправо від символу, що стоїть під курсором.
X	Видаляє символ, розташований перед курсором.
<*n*>**X**	Видаляється <*n*> символів у зворотному напрямку, починаючи із символу, що стоїть перед курсором.
d <*mov*>	Видаляє текстовий об'єкт. Команда '**d**' бере як аргумент команду переміщення курсору <*mov*>. Якщо <*mov*> задає переміщення в межах рядка, то здійснюється видалення від курсору до кінця текстового об'єкта, обмеженого аргументом. Видалення в прямому напрямку (вперед) видаляє символ, розташований під курсором; видалення в зворотному напрямку (назад) не виконується. Якщо <*mov*> задає переміщення на інший рядок, то видалення здійснюється з поточного рядка, включаючи його самого, до текстового об'єкта. Наприклад: **dl** знищує символ, що стоїть під курсором; **d5l** знищує 5 символів, починаючи з того, що стоїть під курсором; **d3J** знищує 3 рядки, починаючи від поточного рядка вниз
dd	Видаляє цілі рядки.
D	Видаляє усі символи від позиції курсору, включаючи її, до кінця поточного рядка.

Переміщення в межах файлу:

– k – перемістити курсор вгору;

– j – перемістити курсор вниз;
– h – перемістити курсор ліворуч;
– l – перемістити курсор праворуч.

Переміщення тексту здійснюється командами видалення, копіювання і вставки текстових блоків. Для цього зарезервовано 9 буферів видалення (позначених цифрами від 1 до 9), 26 буферів для копіювання (позначених літерами від a до z) і один “безіменний” буфер, який використовується за замовчуванням. Якщо необхідно вказати певний буфер, перед будь-якою командою копіювання або вставки вводяться подвійні лапки ("), а за ними – позначення буфера. Для команд вставки (put) позначення буфера може бути цифрою від 1 до 9 або літерою від a до z. Для команд копіювання (yank) позначення буфера може бути літерою від a до z або від A до Z. Якщо ви копіюєте текст у буфер з ім'ям 'A' замість 'a', то текст додається до вмісту буфера 'a' (так само для будь-якого іншого буфера). При видаленні вилучений текст автоматично заноситься в стек буферів, пронумерованих від 1 до 9, тобто щойно видалений блок поміщається в буфер 1, блок з буферу 1 переміщається у буфер 2 і так далі. Щойно вилучений текст також розміщається в “безіменному” буфері.

Таким чином, в буферах 1-9 знаходяться блоки тексту, які було вилучено, в буферах a-z – блоки тексту, які було скопійовано, а в безіменному буфері – блок тексту, який було вилучено під час останньої операції або скопійовано без зазначення буфера.

Наприклад, команда "4p поміщає вміст буфера видалення з номером 4 у ваш буфер редагування під поточним рядком. Якщо ім'я буфера-джерела не зазначено, то текст вставляється з "безіменного" буфера. Таким чином, дуже просто видалити текст, потім пересунути курсор у те місце, куди ви хочете вставити вилучений текст, і після цього вставити текст у нове місце за допомогою команди 'p' або 'P'.

Пойменовані буфери найбільш зручні для збереження сукупності декількох частин тексту, які ви хочете мати постійно напоготові для пізнішого доступу до цих текстів, або їхнього переміщення і перерозміщення. Наприклад, для копіювання рядка з файлу в буфер 'a' наберіть "ayy. Для того, щоб помістити цей текст назад у файл, наберіть "ap.

Слід зазначити, що вміст пойменованих буферів не руйнується при переключенні файлів. Тому ви можете видалити чи скопіювати текст у буфер,

відкрити новий файл і потім виконати команду 'p'. Вміст буферів губиться при виході з редактора, тому будьте обережні.

["<*buf*>**]p**	Вставляє текст із буфера <*buf*> в буфер редагування під поточним рядком чи після курсору, в залежності від того, містить буфер повний чи рядок ні.
["<*buf*>**]P**	Вставляє текст із буфера <*buf*> в буфер редагування або над поточним рядком, або перед курсором, у залежності від того, містить буфер повний рядок чи ні.
["<*buf*>**]y** <*mov*>	Копіює текст із буфера редагування в буфер <*buf*>. Аргумент <*mov*> обмежує об'єкт, що копіюється (аналогічно команді видалення '**d**' – див. вище.
["<*buf*>**]yy** **["**<*buf*>**]Y**	Копіюють один рядок, чи, якщо заданий лічильник, кілька рядків.

Відміна/повтор операцій

u	Скасовує результат останньої команди вставки чи видалення. Після виконання команди вставки команда 'u' видаляє вставлений текст, а після команди видалення – вставляє текст назад.
U	Відновлює поточний рядок у його первісному стані, перш ніж він був відредагований, незалежно від того, скільки операцій редагування ви здійснили з того моменту, як ви перейшли на нього.
.	Повторює останню команду вставки чи видалення.

Команди пошуку здійснюють пошук тексту, що відповідає заданому регулярному виразу і знаходиться в буфері редагування, у прямому чи зворотному напрямку.

/[<*pat*>**]**↵ **?[**<*pat*>**]**↵	Здійснює пошук тексту, що відповідає шаблону <*pat*>, у прямому (**/**) чи зворотному (**?**) напрямку. Рядок є дійсним регулярним виразом. Якщо шаблон <*pat*> не заданий, то для пошуку використовується останнє значення <*pat*>, що використовувалось раніше. (↵ – натискання клавіші "**RETURN**")
n N	Повторює останню команду пошуку. Команда '**n**' повторює пошук у тому ж напрямку, що й остання команда пошуку, команда '**N**' – у протилежному напрямку.
f <*ch*> **F** <*ch*>	Знаходить символ <*ch*> у поточному рядку. Команда '**f**' виконує пошук у прямому напрямку, команда '**F**' – у зворотному.
t <*ch*> **T** <*ch*>	Встановлює курсор над символом <*ch*>.
; **,**	Знак "**;**" повторює пошук останнього символу. Знак "**,**" змінює напрямок пошуку на протилежний.

Команди редактора ex

Введення двокрапки (:) при перебуванні в командному режимі видає підказку-запрошення (:) у рядку стану. Це підказка для введення команди, доступної в строковому редакторі ex.

В основному, ex-команди дозволяють вам записувати до і читати з файлів, переключатись в shell чи змінювати файли, що редагуються. Більшість цих команд виконує дії, що впливають на "поточний" за замовчуванням файл. Поточним звичайно вважається файл, що ви вибрали при запуску редактора vi, хоча поточний файл може бути змінений командою file (f) чи командою next (n).

Як імена більшості ex-команд використовуються англійські слова, а доступною для використання абревіатурою є початкові букви цих слів. В подальших описах згадуються тільки абревіатури як найчастіше використовувана форма запису команди.

Загальна форма команди ex досить складна. Для всіх ex-команд використовується наступний формат:

[address][command][!][parameters][count][flags]

Усі частини необов'язкові, в залежності від команди та її опцій.

Більшість команд використовує адреси, що стоять попереду і визначають рядки, над якими необхідно виконувати дії. Ряд команд також може містити наступний за ними лічильник count, що показує число рядків, що захоплюються при виклику команди. Числа count при необхідності округляються вбік меншого значення. Таким чином, команда '10p' показує десятий рядок у буфері (тобто, 10 – це адреса), а команда 'move 5' переміщає поточний рядок за рядок з номером 5 (тобто, 5 – це лічильник).

Деякі команди також використовують параметри, розміщені після імені команди. Ряд команд має варіанти. Інша форма команди викликається встановленням знаку оклику (!) безпосередньо за ім'ям команди.

Далі команди ex розглянуті в мінімальному обсязі: лише ті команди, які часто викликаються з vi і суттєво доповнюють можливості останнього. Крім того, розгляд обмежено командами, що стосуються всього файлу.

Команди запису

Команди запису дозволяють вам переписувати весь ваш буфер редагування або його частину в поточний або будь-який інший файл·

w [<*file*>**]**	Записує зроблені зміни у файл <*file*>, показуючи кількість записаних рядків і символів. Якщо параметр <*file*> не заданий, буфер записується в поточний файл. Якщо ім'я <*file*> визначене, то буфер редагування записується в цей файл. Редактор здійснює запис у файл тільки якщо це поточний файл і він редагується, або якщо файл не існує (в останньому випадку файл створюється). В інших випадках для запису ви повинні задати команду у зміненій формі – **'w!'**.
w! <*file*>	Скасовує перевірку звичайної команди **write** і здійснює запис у будь-який файл, що дозволений з боку системи.
w >> <*file*>	Додає вміст буфера до кінця існуючого файлу. Попередній вміст файлу не руйнується.

Команди зміни поточного файлу редагування

Для редагування файлу, іншого від того, що редагується в поточний момент, ви можете використовувати один з варіантів команди 'e'.

e <file>	Використовується для початку редагування нового файлу. Редактор спочатку перевіряє, чи був модифікований буфер з моменту використання останньої команди 'w'. Якщо це було зроблено, то видається попередження, і команда переривається. Якщо ні – команда видаляє вміст буфера редагування, робить файл <file> поточним і висвітлює нове ім'я файлу. Після перевірки, що цей файл дійсний (тобто не є бінарним файлом, каталогом або пристроєм), редактор читає файл у свій буфер. Якщо читання файлу виконане без помилок, у рядку стану з'являється число прочитаних рядків і символів. Поточним рядком спочатку вважається перший рядок файлу.
e! <file>	Такий виклик скасовує повідомлення про модифікації, що були зроблені і не записані з буфера редагування, викликаючи, тим самим, скасування всіх змін, що були виконані перед редагуванням нового файлу.
e +n <file>	Змушує редактор почати редагування не з першого, а з n-го рядка. Аргумент n може бути також командою редактора, що не містить пробілів, наприклад, +/pattern.

Команди читання

Команди читання дозволяють вам читати текст у ваш буфер редагування з будь-якого місця (тобто, дозволяє вставити певний текст у визначене місце). Текст, що ви читаєте, повинен складатися, принаймні, з одного рядка, чи бути або файлом, або вхідною інформацією команди.

r [<file>]	Поміщає копію тексту з заданого файлу <file> в буфер редагування після поточного рядка. Якщо файл не заданий, то використовується поточне ім'я файлу. Якщо буфер редагування порожній, то це трактується як команда 'e'. Коли команда 'r' завершується успішно, то видається статистика, подібна до тієї, яка супроводжує виконання команди 'e'. Після команди 'r' поточним вважається останній прочитаний рядок.
r! <cmd>	Читає вихідну інформацію команди <cmd> у буфер після визначеного рядка.

Команди закінчення роботи

Існує кілька шляхів виходу з редактора vi. Деякі переривають сеанс редагування, інші записують вміст буфера редагування перед виходом, треті попереджають вас, якщо ви вирішили вийти без записування буфера редагування. Ці шляхи були розглянуті вище. Зазначимо додатково лише те, що ex-команди wq, wq! і x дозволяють зробити запис у файл, відмінний від поточного (наприклад, wq <file>).

Команди переключення в shell

Часто виникає необхідність тимчасово вийти з редактора і виконати звичайні команди системи UNIX. Також може виникнути необхідність змінити робочий каталог, щоби редагування не впливало на цілісність іншого робочого каталогу. Нижче описані необхідні для цього операції:

cd <dir>	Зазначений каталог <dir> стає вашим поточним каталогом. Якщо каталог не визначений, то в якості імені цільового каталогу використовується домашній каталог користувача.
sh	Запускається новий shell, у якому ви можете виконати будь-яку кількість команд. Для повернення в vi натисніть Ctrl-D.
!<command>	Залишок рядка після знака “!” передається в shell як команда для виконання. В текст команди <command> замість символів '%' і '#' підставляються імена поточного файлу й останнього файлу, що редагувався, а символ '!' заміняється на текст попередньої команди. Ці підстановки повторюються на екрані, але запам’ятовується рядок цієї команди без підстановок.

Якщо з моменту останньої зміни в буфері редагування запис файлу не здійснювався, то перед виконанням команди видається попередження. Коли команда виконається, висвітлюється знак “!”.

Якщо ви використовуєте оболонку C-shell і встановлюєте змінну prompt для виводу підказки-вказівки prompt>, необхідної для роботи з інтерактивними shell, то при використанні вами вищенаведених команд prompt розглядається як ім'я файлу. Це може привести до виникнення несподіваних ситуацій. Щоб уникнути їх,

використовуйте значення prompt, призначуване за замовчуванням, яке визначено у файлі /usr/lib/mkuser/mkuser.cshrc.

Отже, основні Команди редагування VI:

– i – Вставити за курсором (переходить у режим вставки);
– a – Запис після курсору (переходить у режим вставки) ;
– A – Записати в кінці рядка (переходить у режим вставки) ;
– ESC – завершити режим вставки;
– u – скасувати останню зміну;
– U – скасувати всі зміни в усьому рядку;
– Відкрити новий рядок (переходить у режим вставки) ;
– dd – видалити рядок;
– 3dd – видалити 3 рядки;
– D – Видалити вміст рядка після курсору;
– C – Видалити вміст рядка після курсору та вставити новий текст. Натисніть клавішу ESC, щоб завершити вставлення;
– dw – Видалити слово;
– 4dw – Видалити 4 слова;
– cw – змінити слово;
– x – видалити символ біля курсору;
– r – символ заміни;
– R – перезаписати символи, починаючи з курсору;
– s – Підставте один символ під курсором і продовжуйте вставляти;
– S – замінити весь рядок і почати вставляти на початку рядкам;
– ~ – Змінити регістр індивідуального символу.

Збереження та закриття файлу:

– Shift+zz – зберегти файл і вийти
– :w – збережіть файл, але залиште його відкритим
– :q! – Вийдіть з vi і не зберігайте зміни
– :wq – зберегти файл і вийти

Індивідуальне завдання до лабораторного заняття №10-11

1. Завантажтеся в систему під Вашим користувацьким ім'ям.
2. Створіть новий текстовий файл text за допомогою редактора vi. Наберіть два абзаци тексту. Текст повинен містити Ваше прізвище (наприклад, у вигляді підпису). Запишіть файли під іменами text і text1, вийдіть із редактора.
3. Установіть на файл text1 права доступу так, щоб Ви могли тільки читати цей файл, але не модифікувати його.
4. Завантажте файл text у редактор, скопіюйте перші 2 рядка тексту в буфер і вставте їх у кінець тексту.
5. Запишіть файл під тим же ім'ям.
6. Не виходячи з редактора, завантажте файл text1 і, попередньо відкривши новий shell і змінивши права доступу на файл, запишіть файл, не виходячи з редактора.
7. Користуючись пойменованими буферами, перенесіть 3 рядки тексту з першого файлу в другий. Збережіть зміни. Вийдіть з редактора.
8. Відкрийте у редакторі файл **text** і в кінець його додайте уміст файлу **text1**.
9. Запишіть отриманий файл як **text2** і, не виходячи з редактора, видаліть файли **text** і **text1**.

Контрольні питання

1. Як користуватися редактором Vi?
2. Що необхідно, щоб виконати команди редагування?
3. В яких режимах працює редактор Vi?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №12
ПРОЦЕСИ В ОС LINUX І КЕРУВАННЯ НИМИ

Мета роботи – оволодіти практичними навичками роботи з процесами – створення і знищення, керування процесами і їхній аналіз. Вивчити: поняття процесу і його характеристики; вивід на екран списку процесів і його аналіз; фонові й активні процеси; пріоритет процесів і його зміна; відправлення сигналів процесам, організація перехоплення сигналів; виконання завдань у системі в заданий час і з заданою періодичністю.

Література [1, 2, 7, 8]

Теоретичні відомості

UNIX – багатозадачна система з розділенням часу. Це означає, що в системі одночасно виконується багато процесів. Кожний процес асоційований з певним користувачем, від імені якого цей процес діє. Для того, щоби переглянути список процесів, існує команда ps. Ця команда має багато ключів-модифікаторів, які визначають, яку саме інформацію про процеси повинна виводити команда. Слід зазначити, що в різних системах UNIX значення цих ключів може суттєво відрізнятись. Типові ключі: -a виводить інформацію про всі процеси, а не лише про процеси даного користувача, -l та -x виводять розширену інформацію про процес.

Кожний процес в системі має свій унікальний ідентифікатор – PID. За цим ідентифікатором можна звертатись до процесу. Крім того, кожний процес виникає не сам по собі – він має так званий батьківський процес, що характеризується ідентифікатором PPID (parent process ID). Таким чином утворюється ієрархія процесів, що бере початок від початкового процесу init.

Створення нового процесу у системі UNIX виконується у два етапи. Спочатку системний виклик fork() викликає “розщеплення” поточного процесу на два тотожні (різниця буде лише у тому, що процес-“нащадок” має інші PID і PPID). Далі виконується системний виклик exec(), який “підміняє” контекст поточного процесу іншим контекстом. Розглянемо типовий приклад, коли один процес має запустити на виконання інший –командна оболонка sh виконує команду ls, для чого утворює новий

процес, в якому виконується програма /usr/bin/ls. Для цього програміст, який пише код sh, повинен передбачити такі кроки:

Якщо під час синтаксичного аналізу командного рядка виявлено, що необхідно запустити на виконання деяку команду, то виконується виклик fork().

Наступна за fork() інструкція перевіряє значення, що повернув виклик. Якщо виклик fork() був успішний, то в результаті його виконання утворюється новий процес, тотожній батьківському, і операційна система буде виконувати їх обидва (в режимі розділення часу). Для процесу-“нащадка” значення, що повертає fork(), дорівнює 0, для батьківського процесу повертається значення PID нащадка.

Якщо це процес-нащадок, то виконується виклик exec(), якому в якості параметра передається ім’я файлу програми, яку необхідно запустити на виконання (в нашому прикладі – /usr/bin/ls). Якщо виклик був успішний, то на цьому виконання коду sh припиняється, і на його місце завантажується код ls.

Якщо це батьківський процес, то його дії залежать від того, які параметри були у командному рядку. Зокрема, якщо ls було запущено у пріоритетному режимі то виконання sh призупиняється до завершення процесу-нащадка, якщо ж ls було запущено у фоновому режимі, то sh видає необхідні повідомлення про запуск фонового процесу і продовжує свою роботу, тобто знову приймає і редагує командний рядок.

Процеси можуть взаємодіяти між собою за допомогою так званих сигналів. Існує обмежена кількість сигналів, які мають свої числові ідентифікатори і мнемонічні позначення. Сигнали діють як переривання, тобто вони призупиняють процес, до якого вони направлені, і викликають відповідний обробник сигналу. Деякі із сигналів мають чітко визначене значення і обробляються системним обробником. Інші можуть перехоплюватись процесом, тобто процес може встановлювати для цих сигналів свій обробник. Звичайно, існують певні узгодження щодо призначення певних сигналів, і програмістам слід дотримуватись їх при розробці своїх обробників.

Користувач також може відправити процесу сигнал, для цього існує команда kill. Формат команди: kill [-<сигнал>] <PID>

<сигнал> – це мнемонічне або числове позначення сигналу (наприклад, STOP, TERM, CONT, 9), а <PID> –ідентифікатор процесу, якому посилають сигнал. Якщо не вказати параметр <сигнал>, то буде відправлено сигнал завершення процесу TERM

(15). Цей сигнал може перехоплюватись процесом, але існує сигнал KILL (9), який не перехоплюється і безумовно знищує процес (якщо у користувача є достатньо для цього прав). Таким чином можна зупинити будь-який свій процес, якщо над ним втрачене керування (або процес “завис”, що у системі UNIX трапляється дуже нечасто, або користувач не знає, яку команду процес може сприйняти). Для цього слід зайти з іншої консолі (віртуального або фізичного терміналу) і дати команду kill -9 <PID>, де <PID> можна дізнатись за допомогою попередньої команди ps.

Команда ps без аргументів дає список лише процесів даного користувача і (в деяких системах) лише пов’язаних з конкретним терміналом. При роботі в графічній оболонці, а також при роботі на персональному комп’ютері, де підтримується певна кількість так званих віртуальних терміналів, користувач може працювати в системі, використовуючи одночасно кілька терміналів (наприклад, на персональних комп’ютерах між ними можна переключатись комбінаціями клавіш Alt+F#, де F# – одна з функціональних клавіш). Однак в часи створення системи UNIX такі можливості не передбачались. Передбачалось, що користувач має доступ лише до одного терміналу. Тому було розроблено систему завдань і фонового виконання процесів, щоби користувач міг одночасно виконувати різні задачі.

Коли користувач дає команду з консолі, в системі запускається процес, або кілька процесів. Якщо в командному рядку утворюється конвеєр (наприклад, ls –l | wc –l), то всі процеси (у нашому прикладі ls і wc) запускаються одночасно. Разом вони складають так зване завдання. Завдання пов’язано з терміналом. Поки воно не завершиться, користувач не має можливості вводити наступну команду. Це так званий пріоритетний (foreground) режим виконання завдання. Щоби під час виконання завдання мати можливість запускати інше завдання, перше з них слід запустити в так званому фоновому (background) режимі. Для того, щоби запустити завдання в фоновому режимі, слід завершити рядок команди символом “&” (після пробілу). При цьому стандартний ввід за замовчанням назначається порожньому файлу /dev/null. Слід врахувати, що завдання в фоновому режимі може намагатись здійснювати вивід на екран, заважаючи при цьому виводу пріоритетного процесу (спробуйте працювати, запустивши у фоновому режимі команду ping). Тому слід подбати, щоби фонові завдання здійснювали вивід у файли. Завдання, що було запущено у пріоритетному режимі, можна перевести у фоновий. Для цього необхідно спочатку призупинити

виконання завдання (комбінація клавіш CTRL-Z). Далі можна поновити виконання завдання у пріоритетному режимі (команда fg) або у фоновому режимі (команда bg). Завдання мають свої номери. Переглянути їх можна за допомогою команди jobs.

Важливою можливістю є запуск певних завдань за розкладом. Це реалізується за допомогою програми-демона crond, який переглядає свої файли crontab (окремі для кожного користувача) і запускає завдання згідно розкладу від імені того користувача, в файлі якого це завдання задано. Для зміни розкладу завдань треба відредагувати файл crontab і після цього перезапустити crond, що може зробити лише root та ті користувачі, яким це право надано. Останнім рекомендується для цього користуватись командою crontab, яка після редагування автоматично перезапускає crond.

Користувачу дозволено виконувати команду crontab тільки за умови, що його ім’я зустрічається в файлі cron.allow і не зустрічається в файлі cron.deny (в різних системах Unix ці файли можуть знаходитись в різних каталогах, наприклад /usr/lib/cron/ або /etc/cron.d/). Якщо обидва файли відсутні, то тільки root може користуватись командою crontab. Якщо cron.allow не існує, cron.deny існує, але не містить імен, то використовувати команду crontab дозволено усім. Файли cron.allow і cron.deny повинні містити по одному імені в рядку.

Існує спеціальна команда at, яка призначена для одноразового виконання заданої команди в будь-який заданий час. Правила дозволу і заборони користування командою at аналогічні crontab (файли at.allow і at.deny). Завдання, що задані командою at, виконуються тим же демоном crond.

Індивідуальні завдання до лабораторного заняття №12

1. Перегляньте список процесів користувача (Вас).
2. Перегляньте повний список процесів, запущених у системі. При цьому гарантуйте збереження інформації від "утікання" з екрана (якщо процесів багато). Зверніть увагу на ієрархію процесів. Простежте через поля PID і PPID всю ієрархію процесів тільки-но запущеної Вами команди, починаючи з початкового процесу init. Зверніть увагу на формування інших полів виводу.
3. Запустіть ще один shell. Перегляньте повний список процесів, запущених вами, при цьому зверніть увагу на ієрархію процесів і на їхній зв'язок з терміналом. Використовуючи команду kill, завершіть роботу в цьому shell’і.

4. Перегляньте список задач у системі і проаналізуйте їхній стан.

5. Запустіть фоновий процес командою

6. find / -name "*.c" -print > file 2> /dev/null & 3.

7. Визначте його номер. Відправте сигнал призупинення процесу. Перегляньте список задач у системі і проаналізуйте їхній стан. Продовжить виконання процесу. Знову перегляньте список задач у системі і проаналізуйте його зміну. Переведіть процес в активний режим, а потім знову у фоновий. Запустіть цей процес із пріоритетом 5.

8. Виведіть на екран список усіх процесів, запущених не користувачем root.

9. Організуйте вивід на екран календаря <1996+№варіанту> року через 1 хвилину після поточного моменту часу.

10. Організуйте періодичне (щоденне) видалення в домашньому каталозі усіх файлів з розширенням *.bak і *.tmp.

Контрольні питання

1. Що таке процес у Linux?
2. Що таке унікальний ідентифікатор і його призначення?
3. Яким чином можна зробити запуск певних завдань за розкладом?
4. У чому різниця між процесом і потоком?
5. Як перевірити запущені процеси у системі?
6. Що таке PID та PPID?
7. Як змінити пріоритет процесу?
8. Як перевірити рівень пріоритету процесу?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №13
ПРОФЕСІЙНА РОБОТА З КОМАНДНИМИ ОБОЛОНКАМИ

Мета роботи – оволодіти практичними навичками професійної роботи з командною оболонкою shell – використання змінних і створення командних файлів. Вивчити: організацію умовного виконання командного рядка, угруповання команд у командному рядку; використання змінних shell; організація командних файлів: передача параметрів, уведення значень, умовні розгалуження і цикли; арифметичні обчислення в shell. Розробити алгоритм рішення відповідно до завдання. Скласти програми рішення завдань. Підготувати тест для перевірки програм.

Література [1, 2, 7, 8]

Теоретичні відомості

У попередніх роботах ми вже познайомились з командними оболонками (shell). У цій роботі розглянемо прийоми професійної роботи з командними оболонками, а саме використання змінних оточення і створення командних файлів.

Командні файли

Командний файл, або сценарій (також дуже часто кажуть “скрипт” від англійського script – сценарій) є текстовим файлом, який оформлено з дотриманням певних правил, і який містить команди, у найпростішому випадку повністю аналогічні тим командам, що вводяться з клавіатури. Командна оболонка здатна запускати такий файл на виконання і послідовно виконувати команди, що містяться в ньому. Для користувача, що запустив цей сценарій, його виконання буде виглядати як виконання звичайної програми.

Зверніть увагу на розбіжності у різних програмних оболонках shell, які суттєві для програмування. Під час виконання роботи впевніться, в якій із програмних оболонок Ви працюєте (для Linux – bash, який є розвитком sh), і яка буде запускатись для виконання Вашого командного файлу (це визначається першим рядком Вашого командного файлу). Уважно прочитайте правила використання операторів if і формування перевірки відповідної умови. Зверніть увагу на команду test.

Важливою можливістю командних оболонок (усіх) є обробка так званих пакетних файлів.

Індивідуальні завдання до лабораторного заняття №13

1. Створіть командний файл, який буде виконувати завдання пп. 1-3

п.1 Перейдіть у каталог /bin. Перегляньте список усіх файлів, що починаються із символу, який визначено в таблиці індивідуальних завдань.

п.2 Перегляньте список файлів, імена яких складаються з визначеної у таблиці індивідуальних завдань кількості символів.

п.3 Перегляньте список файлів, імена яких починаються із символів, які визначено в таблиці індивідуальних завдань. Зробіть це декількома способами.

варіант	п.1	п.2	п.3
1	a	2	a, b, c, d
2	b	3	e, f, g, h
3	c	4	i, j, k, l
4	d	5	m, n, o, p
5	f	2	q, r, s, t
6	g	3	u, v, w
7	h	4	x, y, z
8	k	5	a, d, k, l
9	l	3	m, g, y
10	n	2	x, z, r, q
11	r	5	t, r, e
12	m	3	u, s, a
13	q	4	m, e, z
14	z	2	t, o, d
15	p	6	x, z, q
16	s	5	m, g, y, a
17	r	4	t, o, d, y
18	u	2	u, a, r, z
19	w	3	a, d, k
20	y	6	a, d, l

2. Модифікуйте конфігураційний файл Вашого shell, щоб Ваше системне запрошення мало вигляд "Hello, <Ваше ім’я>", а перед виводом системного запрошення на початку роботи shell друкував на екрані велике вітання з інформацією про систему, на якій Ви працюєте, поточну дату і час.

3. Запишіть у файл ~/lab_6/res_3 список файлів у каталозі, ім'я якого зазначено в системній змінний x, якщо він існує; у противному випадку в цей файл повинен записатися рядок "Error". При цьому на екрані не повинно з'явитися повідомлення від команди ls у випадку відсутності каталогу.

4. Створіть командний файл, що виконує наступне: у випадку, якщо файл my_file існує, то виводить його вміст; у противному випадку виводить повідомлення "File <ім’я файлу> not found". Перед виводом вмісту файлу повинен друкуватися заголовок "Вміст файлу <ім’я файлу>". При цьому на екрані не повинні з'явитися повідомлення, видавані системними командами. Використуйте механізм умовного виконання команд. Виконайте цей командний файл у всіх трьох режимах запуску. Перевірте його роботу при наявності файлу my_file і при його відсутності.

5. Модифікуйте попередній командний файл так, щоб він очікував введення імен файлів із клавіатури. При виводі результату повинно бути підставлене правильне ім’я файлу.

6. Модифікуйте попередній командний файл, щоби він брав ім’я файлу з параметру командної строки, якщо параметр заданий, і очікував введення імені файлу із клавіатури, якщо параметр не заданий.

7. Додайте до попереднього командного файла перевірку кількості параметрів командної строки й послідовну обробку всіх заданих параметрів.

Контрольні питання

1. Що таке командна оболонка (shell) у Windows?
2. Які основні відмінності між cmd.exe та PowerShell?
3. Що таке batch-файл і яке його призначення?
4. Яке розширення мають командні файли у Windows?
5. Як виконується командний файл?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №14
АВТОМАТИЗАЦІЯ АДМІНІСТРАТИВНИХ ЗАВДАНЬ ЗА ДОПОМОГОЮ BATCH-СКРИПТІВ

Мета роботи – створення batch-скриптів з елементами адміністрування ОС Windows. Вміти створювати сценарії для роботи з елементами адміністрування операційної системи Windows та організовувати автозавантаження власних сценаріїв за таймінгом.

Література [1, 2, 3,4]

Теоретичні відомості

Наступні відомості з написання batch-скриптів рекомендується відтворювати в ході опрацювання теоретичного матеріалу.

Запуск програм

Для запуску програм з командного рядка використовуємо:

start ["заголовок"] [команда/програма] [параметри]

(Довідку про команду можна отримати виконавши start /?)

Наприклад,

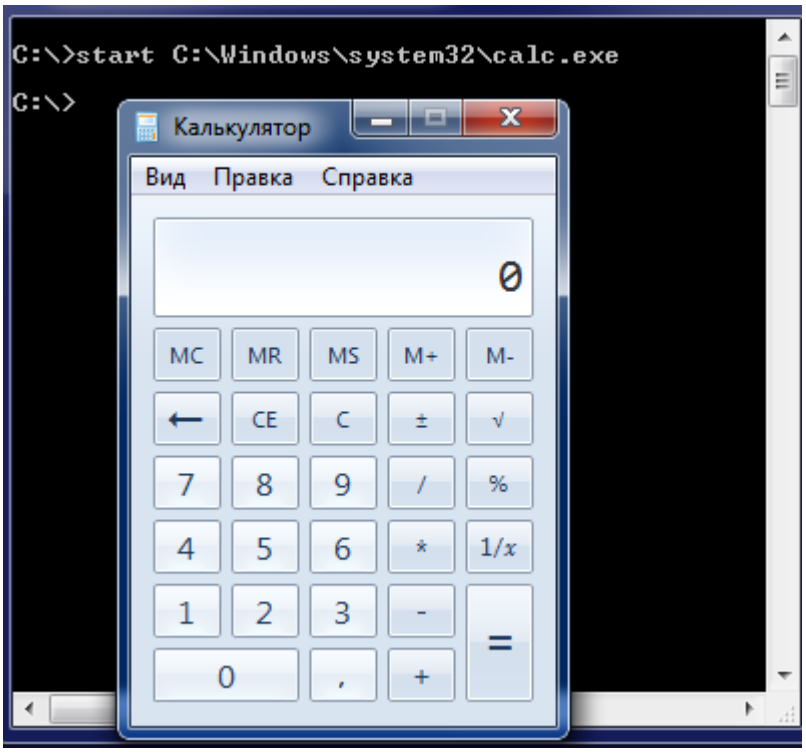


Рисунок 14.1 – Запуск програми Калькулятор

Тепер ще пара важливих моментів. Припустимо, ви створили батник, який запускає три програми і вам потрібно на якийсь час виключити запуск однієї з трьох. Це можна зробити видаленням рядка або його коментуванням в блокноті. Перший спосіб – вандальний, а другий дивіться нижче:

start firefox start jetaudio rem start defraggler

У цьому випадку вимкнено запуск встановленої в системі програми defraggler.exe. Коментують рядки, вказуючи команду rem на початку рядка.

Всі файли .bat виконуються у вікні консолі. Щоб воно зникало після закінчення команд, не забувайте в кінці батнику в блокноті писати команду exit.

Наприклад, start firefox start jetaudio rem start defraggler exit

Тепер розглянемо, як за допомогою bat файлу запустити кілька програм з певними налаштуваннями, або в автоматичному режимі встановити програму, щоб не витрачати час на відповіді типу «Ви згодні з умовами ліцензійної угоди?» та не натискати зайві кнопки. Є кілька способів запуску програм за допомогою bat-файлу. Найперший – це коротка команда на запуск встановленої у системі програми.

Насправді, start firefox – це не завжди працює. Тому такий прийом можна цілком застосовувати на якійсь конкретній системі, але як універсальне рішення він не підходить. Якщо є мета змусити працювати bat файл скрізь і завжди, потрібно скористатися повними шляхами:

start C:\"Program Files\"Mozilla Firefox\firefox.exe

Також в bat-файлі обов'язково має бути присутня команда на завершення:

start C:\"Program Files\"Mozilla Firefox\firefox.exe exit

Запуск програм у bat-файлах із параметрами (ключами)

Програму можна не просто запускати, а давати додаткові команди при запуску. Наприклад, можна дати команду запускатися програму згорнутою:

start /min D:\FileZilla\FileZilla.exe exit

Ключ /min вказується через слеш після основної команди (команда/ключ). Основною командою наразі є start . Правда, ключ min працює тільки в половині випадків, тому що відноситься саме до команди запуску start, а не до програм, які ця команда запускає. Взагалі ключів існує дуже багато, і набори ключів різних програм

можуть суттєво відрізнятися. Є, щоправда, кілька спільних. Наприклад, ключ довідки (/? або /help).

Як перенаправити результат виконання команд у файл

Часто, при виконанні складного bat-файла в автоматичному режимі, перевірити результати його роботи буває важко з багатьох причин. Тому простіше записувати результати роботи команд bat файлу текстовий файл (лог-файл). а потім аналізувати правильність роботи bat файлу за цим логом.

Перенаправити результат роботи команд bat файлу до лог-файлу досить просто. Далі буде показано як це можна зробити.

Створіть bat-файл наступного змісту (скопіюйте ці рядки в Блокнот і збережіть файл з розширенням bat):

```
@echo off
echo Start %time%
echo Create test.txt
echo test>C:\test.txt
echo Copy Test.txt to Old_test.txt
copy C:\test.txt C:\Old_test.txt
echo Stop %time%
```

Перший рядок відключає виведення самих команд. Таким чином, у лог-файл будуть записані лише результати виконання.

Другий рядок записує в лог-файл час початку пакетного файлу.

Третій рядок записує до лог-файлу пояснення того, що наступна команда створить файл test.txt

Команда з четвертого рядка створює файл test.txt з кореня диска C. Файл створюється для прикладу. Ця команда записує у файл C:\test.txt слово test

П'ятий рядок виводить у лог-файл пояснення, що наступна команда копіює файл з одного місця в інше.

Команда в шостому рядку копіює створений файл C:\test.txt файл C:\Old_test.txt, тобто створюється копія файлу під новим ім'ям.

Останній, сьомий рядок містить команду виведення часу завершення роботи пакетного файлу. У сумі із записом в лог-файл часу початку роботи пакетного файлу ці значення часу дають можливість оцінити час роботи пакетного файла.

Збережіть цей пакетний файл під назвою, наприклад, 1.bat

Якщо просто запустити цей файл, отримаємо таке:

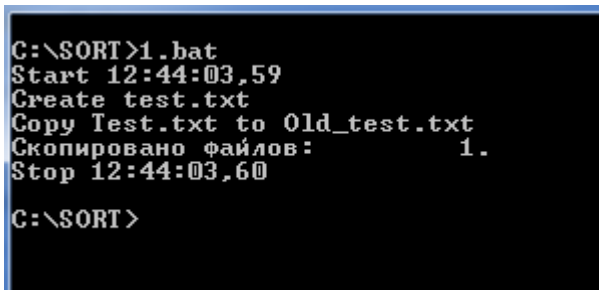


Рисунок 14.2 – Пакетний файл

Припустимо, що звіт про роботу пакетного файлу ми хотіли б зберігати в окремій папці і кожен день записувати звіт з новим ім'ям файлу, щоб була можливість в будь-який день звернутися до логів за попередні дні. Причому ім'я лог-фалу хотілося б мати у вигляді дати роботи пакетного файлу. Щоб це реалізувати створимо на диску C (наприклад) папку з ім'ям LOG, тобто. повний шлях до неї виглядатиме, як

C:\LOG

Створений пакетний файл 1.bat запускатимемо наступною командою:

1.bat>C:\LOG\%date%.txt

Якщо пакетний файл буде запускатися з Планувальника, потрібно вказати повний шлях з bat-файлу. Пам'ятайте, якщо в прописаному шляху є пробіли, то треба використовувати або лапки, або формат

"C:\Program Files\1.bat">C:\LOG\%date%.txt

C:\Progra~1\1.bat>C:\LOG\%date%.txt

Після запуску файлу 1.bat у папці C:\LOG буде створено файл з ім'ям, що дорівнює даті запуску bat-файлу, наприклад, 16.04.2022.txt Це і буде звіт про роботу пакетного файлу 1.bat

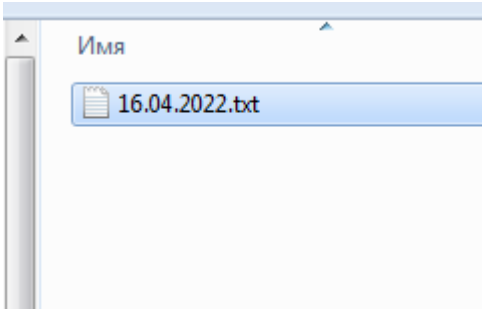


Рисунок 14. 3 – Створення файлу

Запуск bat-файлу, приклад якого показаний у попередньому лістингу, призведе до створення лог-файлу такого змісту:

C:\SORT>1.bat

Start 13:23:03,27

Create test.txt

Copy Test.txt to Old_test.txt

Скопійовано файлів: 1.

Stop 13:23:03,28

Таким чином, для виконання перенаправлення результатів роботи bat-файлу в лог-файл потрібно використовувати символ перенаправлення > Синтаксис такий:

Шлях\Ім'яФайлу.bat>Шлях\Ім'яЛогФайлу.txt

Розширення файлу лог-файлу може бути будь-яким. При бажанні звіт про виконання пакетного завдання можна оформити навіть у вигляді сторінки html.

Як автоматично відповісти на запит про підтвердження

Деякі команди вимагають підтвердження потенційно небезпечної дії. Наприклад, такі команди, як format або del, попередньо запросять підтвердження на подальше виконання. Якщо одна з цих команд виконується в пакетному файлі, запит на підтвердження зупинить виконання пакетного файлу і він чекатиме від користувача вибору одного із запропонованих варіантів. Причому, якщо результат виконання пакетного файлу перенаправлено в лог-файл, то користувач не побачить запиту на підтвердження і batch-скрипт виглядатиме завислим.

Для виправлення таких неприємностей можна перенаправити потрібну відповідь до команди. Тобто, виконати зворотну дію для перенаправлення результатів роботи команди у файл.

Подивимося на прикладі, як виглядає запит на підтвердження потенційно небезпечної дії. Створимо на диску, наприклад, папку Folder. Створимо в ній або скопіюємо в неї два будь-які файли. Далі відкриємо командний рядок і виконаємо наступну команду:

Ця команда повинна видалити всі файли із зазначеної папки. Але попередньо буде видано запит на підтвердження наступного змісту:

C:\Folder*, Продовжити?

Виконання команди буде зупинено доти, доки не буде натиснуто або клавішу Y, або клавішу N.

При виконанні пакетного файлу в автоматичному режимі його виконання зупиниться.

Щоб уникнути цього використовуємо перенаправлення. Перенаправлення здійснюється за допомогою символу |.

Вертикальна риска говорить про те, що замість виведення символу на екран його треба «віддати» команді, що слідує за символом. Перевіримо роботу перенаправлення. Виконайте в командному рядку таку команду:

echo Y|del C:\Folder

На екрані буде відображено запит на підтвердження видалення всіх файлів у папці Folder, але вже з позитивною відповіддю (Y). Усі файли з папки Folder будуть видалені.

Будьте обережні з цією командою, видаляйте лише після впевненості, що це можна зробити без загрози втрати інформації.

При виконанні пакетного файлу на екрані, крім результатів роботи команди, виводяться і самі команди. Щоб вимкнути виведення команд, можна використовувати символ @. Щоб не виводити одну команду на екран, можна поставити знак @ на початку цієї команди.

Іноді при виконанні пакетного файлу виникає потреба запустити інший пакетний файл. Причому, у деяких випадках, виконання основного пакетного файлу має бути призупинено, поки виконується допоміжний файл, а інших допоміжний файл повинен працювати паралельно з основним.

Наприклад створимо два bat файли. Один з ім'ям 1.bat і містить лише одну команду. Другий з ім'ям 2.bat і також містить одну команду.

Тепер запустимо файл 1.bat Відкриється вікно, у якому буде запропоновано натиснути будь-яку клавішу для продовження після натискання якої вікно закриється. Таким чином, виклик з одного пакетного файлу іншого за допомогою команди call зупиняє виконання пакетного файлу, доки не завершиться виконання пакетного файлу, викликаного командою call.

В іншому випадку, треба запустити з bat файлу або програму, або інший пакетний файл, не перериваючи виконання основного пакетного файлу. Таке нерідко

буває потрібно зробити, наприклад, примусово відкривши блок роботи пакетного файлу, запланованого на ніч, щоб зранку, користувач міг проконтролювати правильність його виконання. Для цього використовується команда start.

Індивідуальні завдання до лабораторного заняття №14

1. Напишіть batch-скрипт для запуску розгорнутої версії блокноту. Продемонструйте написаний скрипт і результат його роботи.
2. Напишіть batch-скрипт, який видаляє з папки з Вашим іменем усі файли зображень. Скрипт має виконуватись максимально приховано (без коментарів в консолі, з наступним закриттям консолі). Організуйте автозавантаження такого скрипта. Продемонструйте написаний файл, його розміщення і результати роботи.
3. Напишіть batch-скрипт, який копіює файли з вибраної папки на Вашому комп’ютерів в папку LOG. Скрипт назвіть своїм прізвищем. Продемонструйте написаний скрипт і результат його роботи.
4. Організуйте запуск одного сценарію, що створений сьогодні на лабораторній роботі, через інший сценарій.

Контрольні питання

1. Для чого потрібна команда @echo off?
2. Що робить команда set у batch-скриптах?
3. Яке призначення команди if у скриптах?
4. Як реалізується цикл у командному файлі?
5. Що таке змінна середовища і як її використати?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №15
BATCH-СКРИПТИНГ ДЛЯ КЕРУВАННЯ ЕЛЕМЕНТАМИ ФАЙЛОВОЇ СИСТЕМИ

Мета роботи – оволодіти практичними навичками написання Batch-скриптів у Windows. Вивчити основні команди керування папками та файлами, умовні конструкції та цикли; виконати мережеву діагностику, використовуючи файли з розширенням .bat.

Література [1, 2, 4, 6]

Теоретичні відомості

Batch-скрипти в Windows – це прості текстові файли з розширенням .bat або .cmd, які виконують послідовність команд командного рядка. Їх можна використовувати для автоматизації задач, пов'язаних з файловою системою: створення/видалення папок, копіювання/переміщення файлів, перейменування, перевірка наявності тощо.

Використовується для автоматизації задач: керування файлами, налаштування системи, запуск програм тощо.

Основними елементами файлової системи є: файл, папка (директорія) та шлях.

Основні команди для керування папками та файлами:

– створення папки – mkdir C:\MyText;
– видалення папки – rmdir /S /Q C:\MyText;
– копіювання файлів – copy file1.txt D:\Backup\;
– копіювання папок – xcopy C:\Data D:\Backup /E /I /Y.

Перевірка існування

```
if exist C:\file.txt
(
echo Файл знайдено
)
```

Отримання вводу від користувача

```
set /p USER=Введіть ім’я:
echo Привіт, %USER%!
```

Умовні конструкції	if "%VAR%"=="test" (
	  echo Так
	) else (
	  echo Hi
	)

Цикл for	for %%f in (*.txt) do (echo Файл: %%f)

Принципово просто створити пакетний файл. Єдине, що потрібно змінити, це те, що ви вводите в Блокнот. Щоб запустити декілька команд, ви вводите кожну з них на окремому рядку, а командний файл запускатиметься кожен.

Наприклад, скажімо, ми хочемо написати пакетний файл, який виконує кілька команд мережевої діагностики. Ми можемо захотіти побігти ipconfig / all , щоб переглянути інформацію про мережу, ping google.com щоб перевірити, чи відповідають сервери Google, та tracert google.com щоб запустити traceroute на google.com і подивитися, чи є якісь проблеми на шляху.

У найосновнішій формі ми можемо просто помістити всі ці команди в пакетний файл, один за іншим, так:

ipconfig / все пінг google.com tracert google.com PAUSE

Коли ми запускаємо цей файл, ми побачимо вихід кожної команди відразу за іншим. Але це не обов'язково є ідеальним способом написання пакетного файлу.

Наприклад, можна додати рядки коментарів. Будь-яка лінія, що починається з a :: є рядком коментарів і не буде виконано. Це робить їх корисним способом пояснити, що відбувається у файлі, для тих, кого ви можете надати – або для вашого майбутнього я, хто може забути, чому ви ввели ту чи іншу команду.

Ви також можете додати команду “ECHO OFF” до початку файлу. Зазвичай це додано до початку більшості пакетних файлів. Коли ви робите це, самі команди не будуть надруковані в командний рядок, але результати будуть. Наприклад, ви побачите деталі мережного підключення, але не рядок "ipconfig / all". Більшість людей не бажають бачити команди, так що це може очистити вихід.

Ось так це може виглядати:

:: Цей пакетний файл перевіряє наявність проблем з мережним підключенням. ECHO OFF :: Переглянути дані про мережеве підключення ipconfig / all :: Перевірте,

чи Google.com доступний ping google.com :: Запустіть traceroute, щоб перевірити маршрут до Google.com tracert google.com PAUSE

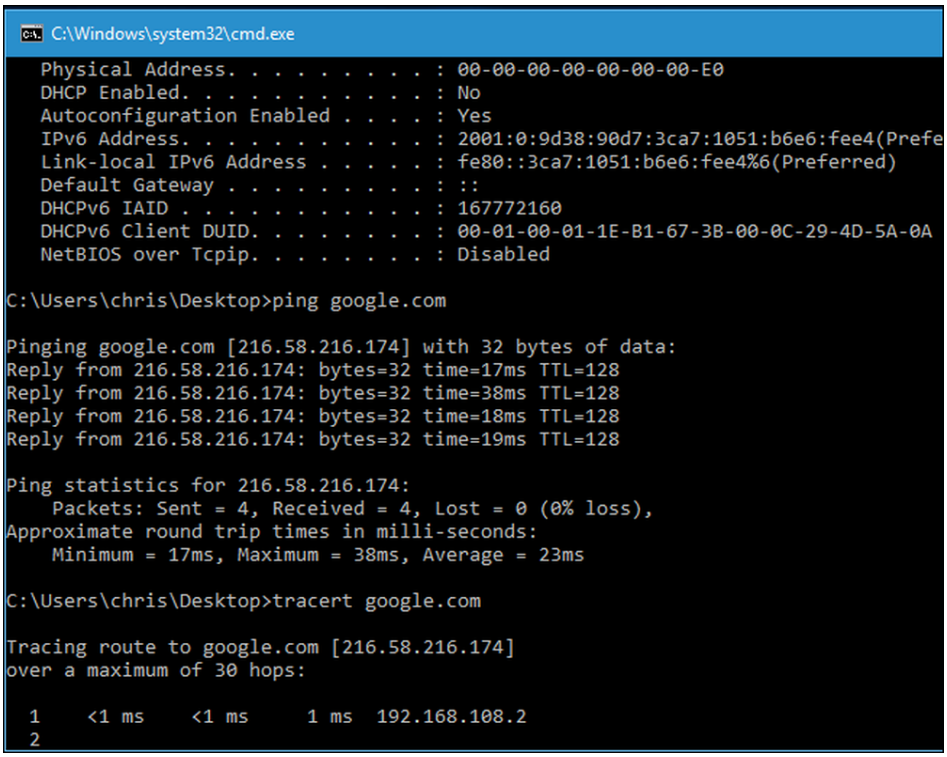


Рисунок 15.1 – Пакетний файл команди ping

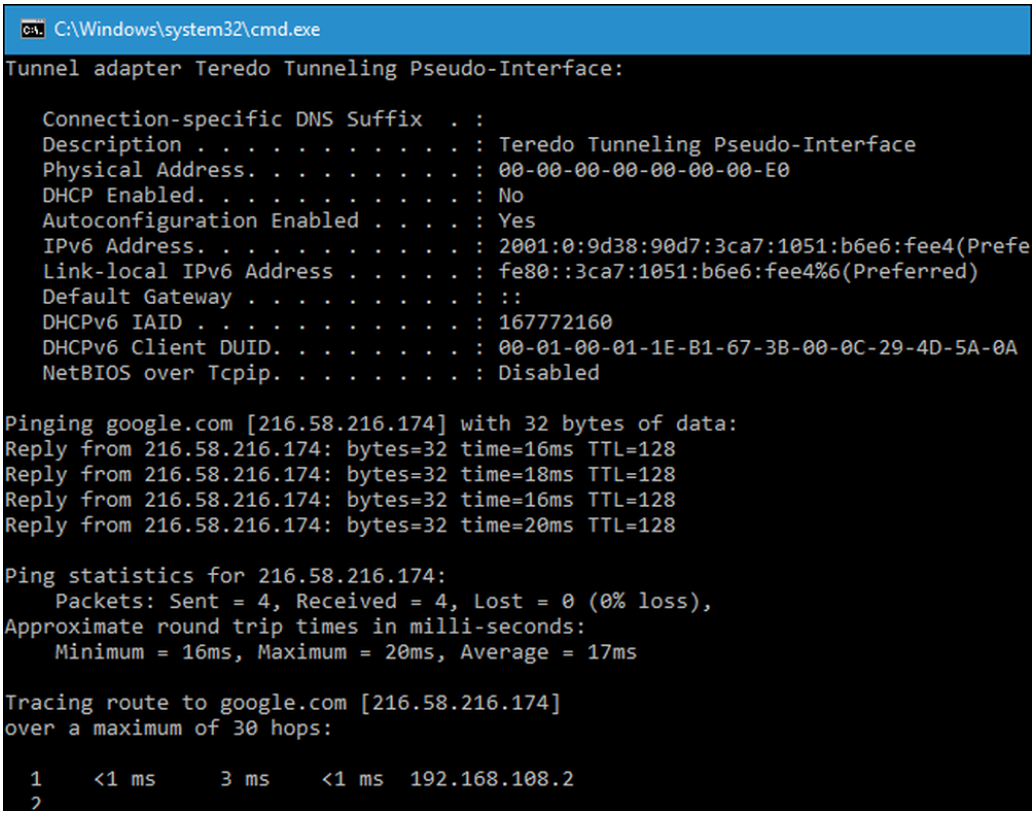


Рисунок 15.2 – Пакетний файл команд

Існують і інші напрямки, у яких можна виконати командний файл. Наприклад, вам може знадобитися, щоб ваш пакетний скрипт виконував наведені вище команди, а потім скидав вихідні дані до текстового файлу, який можна переглянути пізніше. Для цього потрібно скористатися >> Оператор після кожної команди додасть свій висновок до текстового файлу. Оскільки ми збираємося прочитати вихідний текст з текстового файлу, ми можемо пропустити ПАУЗА команду.

:: Цей пакетний файл перевіряє наявність проблем з мережевим підключенням :: та зберігає вихідні дані у файлі .txt. ECHO OFF :: Перегляд деталей мережного підключення ipconfig / all >> results.txt :: Перевірте, чи є Google.com доступним пінг google.com >> results.txt :: Виконайте traceroute, щоб перевірити маршрут до Google.com tracert google. com >> results.txt

Після запуску вищезазначеного сценарію ви знайдете файл з назвою results.txt у тій же папці, що й пакетний файл з виходом команд. Вікно командного рядка автоматично закриється після запуску пакетного файлу.

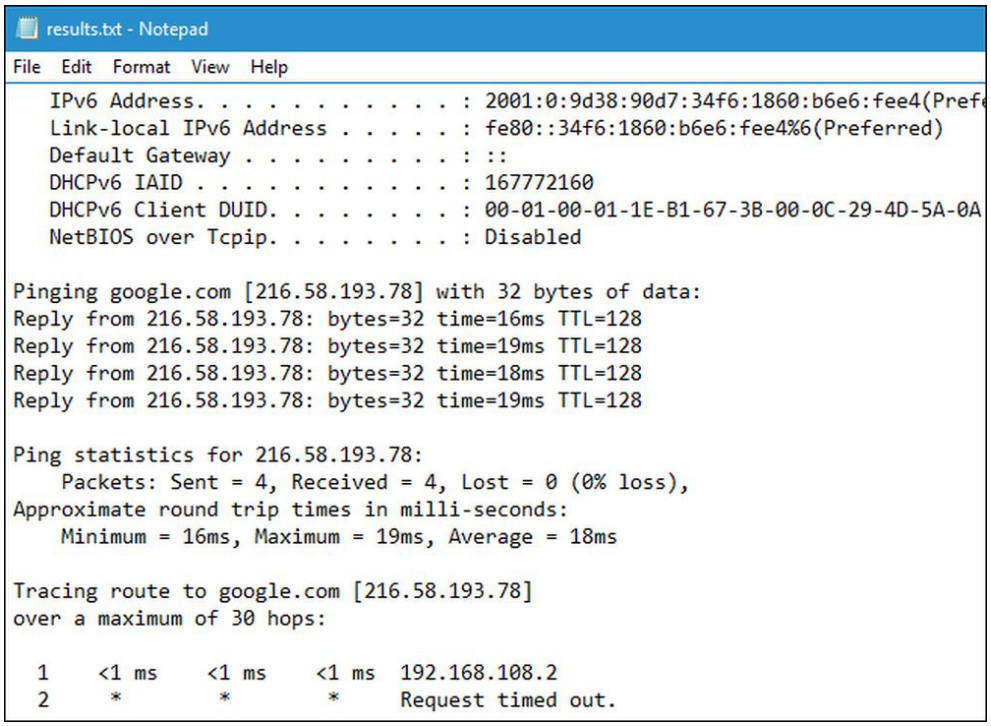


Рисунок 15.3 – results.txt

Наведений вище приклад спирається на фактичну друк інформації в командний рядок, щоб користувач міг його прочитати. Тим не менш, багато командних файлів розроблено для запуску неінтерактивно. Наприклад, у вас може бути пакетний файл,

який видаляє кілька файлів або каталогів, коли ви двічі клацнете на ньому. Вам потрібно просто скористатися del команда для видалення файлів або deltree команда для видалення каталогів. Пам'ятайте, що ви використовуєте ті самі команди, що й у вікні командного рядка.

В принципі, це точка більшості пакетних файлів – просто запуск декількох команд один за одним. Проте пакетні файли можуть бути значно складнішими. Наприклад, ви можете використовувати оператори «IF» разом з командою «GOTO», щоб перевірити значення чогось, а потім перейти до різних ліній залежно від результату. Це більше схоже на написання реальної невеликої програми, ніж швидкий і брудний сценарій. Це одна з причин, чому файли .bat іноді називаються "пакетними програмами". Якщо ви хочете зробити щось більш складне, ви знайдете безліч посібників для виконання конкретних речей з пакетним програмуванням онлайн. Але тепер, ви знаєте основи, як кинути простий один разом.

Переваги використання Batch-скриптів:

1. Простота (не потребує стороннього ПЗ);
2. Швидкість написання;
3. Можливість автоматизації типових задач;
4. Вбудована підтримка в Windows.

Обмеження:

1. Примітивний синтаксис;
2. Обмежені логічні оператори;
3. Важче читати при складних умовах;
4. Не підтримує сучасне програмування як PowerShell або Python.

Індивідуальні завдання для лабораторного заняття №15

1. Створити файл (.bat), який виконує кілька популярних мережевих діагностичних команд у Windows, які описані в лабораторній роботі
2. Створити підпапку з назвою поточної дати та копіювати туди вміст певної директорії.
3. Знайти всі файли, що перевищують певний розмір (наприклад, 100 МБ), і записати їх імена у файл.
4. Замінити пробіли в іменах усіх файлів на підкреслення.

5. Видалити всі файли старші за 7 днів (використати forfiles).

Контрольні питання

1. Які можливості PowerShell надає для роботи з мережею (перевірка підключень, сканування, DNS тощо)?
2. Що таке модулі PowerShell і як ними користуватися? Як імпортувати новий модуль?
3. Як перевірити наявність оновлень Windows і встановити їх через PowerShell?
4. Що таке PowerShell Remoting і як його налаштувати для віддаленого адміністрування?
5. Як використовувати PowerShell для резервного копіювання файлів або реєстру?
6. Які засоби безпеки слід враховувати при використанні PowerShell (Execution Policy, цифрові підписи, доступи)?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №16
РЕЗЕРВНЕ КОПІЮВАННЯ

Мета роботи – навчитись створювати сценарії з використанням лексем batch; вміти створювати скрипти для захисту даних ОС Windows, організовувати резервне збереження інформації. Вміти створювати вибірку з файлів, виведення масивів, індивідуальних сценаріїв резервного копіювання у Windows.

Література [1, 2, 3, 4]

Теоретичні відомості

Звісно, що для написання серйозних і містких програмних продуктів потрібно володіти усіма лексемами, операторами, структурами, підпрограмами, утилітами тощо вибраної мови програмування. Без цих знань використання будь якої мови, в тому числі і скриптової мови batch, досить поверхневе. На досконале вивчення мови у розробників ідуть місяці, роки, можливо і більше). Метою лабораторних робіт по batch скриптах, як і класично у навчанні, є ознайомлення для подальшого зацікавлення і навчання практичним навичкам для елементарного розуміння.

Приміром, після сьогоднішньої лабораторної роботи, кожен студент повинен вміти створити власний продукт для захисту своїх даних при роботі з ОС Windows. Складність і універсальність розробки може бути різною. Але, розуміння функціонування скриптів, способи їх автоматичного відтворення, налаштування даних, функціоналу, програмування таймінгу – першочергове завдання.

Отож, головним завданням сьогоднішньої лабораторної роботи є використання скриптової мови для створення дієвого індивідуального продукту для захисту даних при роботі з ОС Windows.

Для цього необхідно розуміти наступні лексеми і структури.

Умовні лексеми

IF Обробка умов у пакетних програмах

Синтаксис:

if [not] рядок1==рядок2 команда [else вираз]

При обробці змінних важливо враховувати, що вони можуть виявитись порожніми. Тому, щоб команда порівняння залишилася коректною, операнди краще укладати в лапки: if "%1"=="input.txt". Тому, розширений синтаксис:

if [/i] рядок1 оператор_порівняння рядок2 команда [else вираз]

Розширений синтаксис дозволяє використовувати реєстро-незалежне (/i) порівняння з такими операторами:

Оператор	Опис
EQU	дорівнює
NEQ	не дорівнює
LSS	<
LEQ	<=
GTR	>
GEQ	>=

Перевірка існування файлу:

if [not] exist ім'я_файлу команда [else вираз]

Наприклад:

if exist $result$.txt del $result$.txt

Перевірка коду повернення:

if [not] errorlevel число команда [else вираз]

Умова вірна, якщо попередньо оброблена команда завершилася з кодом, рівним або більшим за зазначене число.

Наприклад, нижче наведений скрипт s.bat, що знаходиться в папці C:\SORT копіює усі файли з C:\SORT\LOG. Імена файлів записуються в консолі:

```
@echo off
rem "цей скрипт копіює всі файли у всіх підкаталогах"
rem вихідного каталогу (%SORT) до іншого каталогу (%LOG)
copy %SORT  /s /e %LOG
if errorlevel 4 echo Insufficient memory, invalid drive specified, or syntax error
if errorlevel 2 echo Press CTRL+C to stop copying
if errorlevel 0 echo Copying was successful
```

Цикли та ключі

Команда for дозволяє організувати виконання однотипних дій, що повторюються. Можна використовувати її для того, щоб вивести на екран числа від одного до десяти, як показано на наступному прикаді:

for /l %%i in (1,1,10) do echo %%i

Ключі:

Цикл For /L використовується для завантаження ряд цифр (range of numbers). Синтаксис:

FOR /L %%variable IN (start, step, end) DO command

@rem Or:

FOR /L %%variable IN (start, step, end) DO (

command

)

start: Перше значення змінної

step: Після кожного повтора (iteration) значення змінної буде додавати 'step'.

end: Останнє значення.

Наприклад, скрипт виведення чисел від 1 до 8 з кроком 2 та результат виконання:

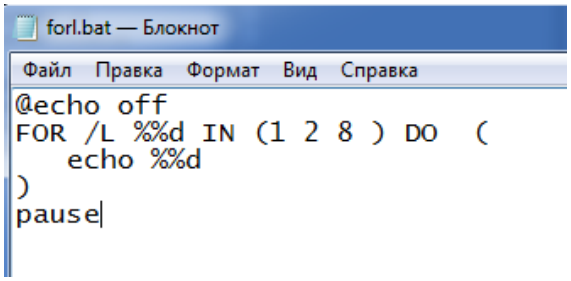


Рисунок 16. 1 – Скрипт виведення чисел

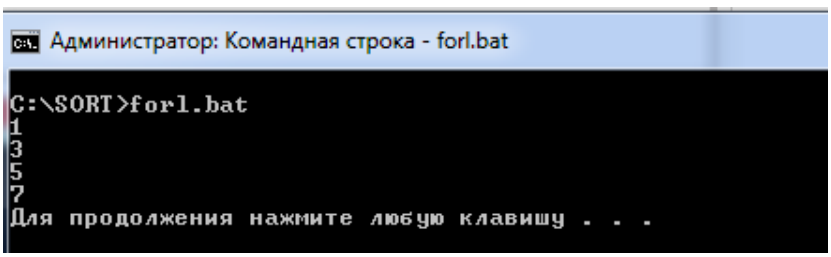


Рисунок 16. 2 – Виведення чисел через командну стрічку

Цикл For/F є складним циклом, але сильним. Він читає вміст файлу чи кількох файлів. Після цього аналізує їх за вмістом. Вміст фала є текстом, він буде розділено на кілька маленьких текстів, кожен уривок тексту називається Token. Правило за замовчуванням для поділу ґрунтується на пробілах (white space). Але можна кастомізувати правило поділу за допомогою параметра ["delims=xxx"]. Цикл For /F так само використовується для аналізу змісту рядка (string), або виконання набору команд.

Синтаксис:

```
FOR /F ["options"] %%variable IN ( set_of_filenames ) DO command
FOR /F ["options"] %%variable IN ( set_of_filenames ) DO (
    command
)
FOR /F ["options"] %%variable IN ("Text string to process") DO command
FOR /F ["options"] %%variable IN ("Text string to process") DO (
    command
)
```

де

set_of_filenames: Список з одного чи декількох файлів.

options: Опції, наприклад: "delims=, tokens=1,2,4"

Наприклад, для вибірки з файлу log1.txt

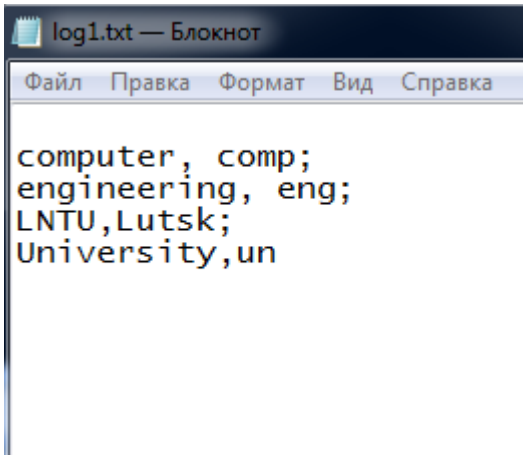


Рисунок 16. 3 – файл log1.txt

використовуємо скрипт:

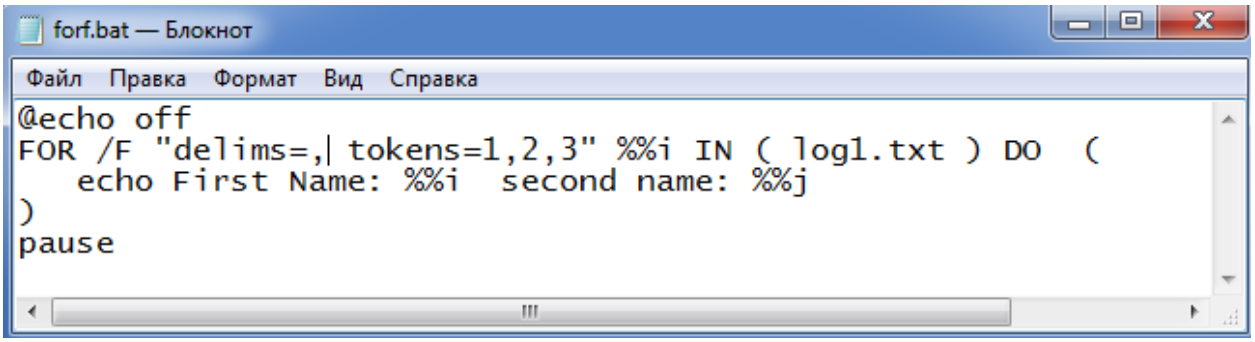


Рисунок 16. 4 – файл fort.bat

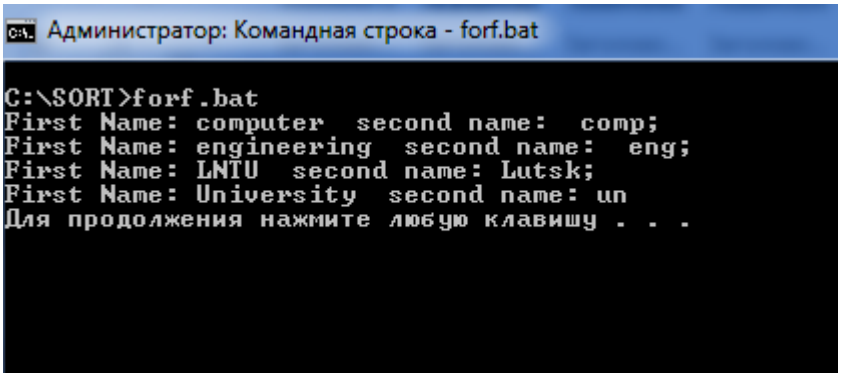


Рисунок 16. 5 – Результат виконання

Резервне копіювання

Програм для створення резервних копій (backup, бекап) є безліч.

Є платні, є безкоштовні. У деяких використовуються «майстри», які дозволяють вказати масу параметрів, не вдаючись до складного налаштування. Однак, сам процес резервування, насправді – звичайне копіювання. Є звичайно варіанти на кшталт: архівування, відстеження змін тощо, але для більшості випадків сам алгоритм зводиться до простих дій: вибір вихідних даних (каталог, файли); вибір куди копіювати (каталог); безпосередньо копіювання. Так ось для того, щоб виконати всі ці операції, зовсім не обов'язково вдаватися до допомоги сторонніх програм. Усе необхідне вже є у Windows. Для копіювання файлу/каталогу використовується команда copy. Вона має багато параметрів, от наприклад вона вказується так:

copy "D:\myfiles*.*" "j:\backup\myfiles*.*"

Ця команда скопіює каталог "d:\myfiles" у "j:\backup\myfiles".

Індивідуальне завдання до лабораторного заняття №16

Створіть файл резервного копіювання для свого робочого комп'ютера. Передбачте максимально необхідні завдання, реалізуйте їх програмно (скриптом). Рядки скрипта закоментуйте. Використайте лексеми, змінні, умови, автоматичне створення з датою. Представте сформований файл та результати його виконання. (складністю написання скрипта оцінюється вищим балом).

Можливий Функціонал скрипта:

– Копіює вміст ключових папок користувача:
  Documents
  Desktop
  Pictures
– Створює резервну папку з датою створення.
– Створює лог-файл.
– Перевіряє існування джерел та створює резервні директорії автоматично.
– Пропускає системні або тимчасові файли.

Контрольні питання

1. Що таке batch-файл? Як його створити у Windows?
2. Які основні лексеми (команди) використовуються в batch-файлах для копіювання файлів і папок?
3. Як за допомогою batch-файлу автоматизувати процес резервного копіювання на інший диск або флешку?
4. Як у batch-файлі можна виводити дату та час резервного копіювання у назві файлу чи папки?
5. Що таке змінні середовища в контексті batch-файлів? Як ними скористатися у скриптах резервного копіювання?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №17
АВТОМАТИЗАЦІЯ АДМІНІСТРАТИВНИХ ЗАВДАНЬ ЗА ДОПОМОГОЮ POWERSHELL

Мета роботи – навчитись працювати в оболонці PoweShell. Вміти застосовувати функціонал PoweShell для реалізації завдань щодо адміністрування операційної системи Windows.

Література [1, 2, 5]

Теоретичні відомості

PowerShell – це оболонка командного рядка та мова сценаріїв, два в одному. Вона була розроблена як обробник завдань, який використовує командлети, які потрібно виконувати користувачам. В PowerShell можна виконувати команди на локальних або віддалених комп'ютерах. Також ви можете виконувати такі завдання, як керування користувачами та автоматизація робочих процесів.

Запуск здійснюється різними способами, можна через відому вам консоль, а можна (як у Windows10), через Головне меню:

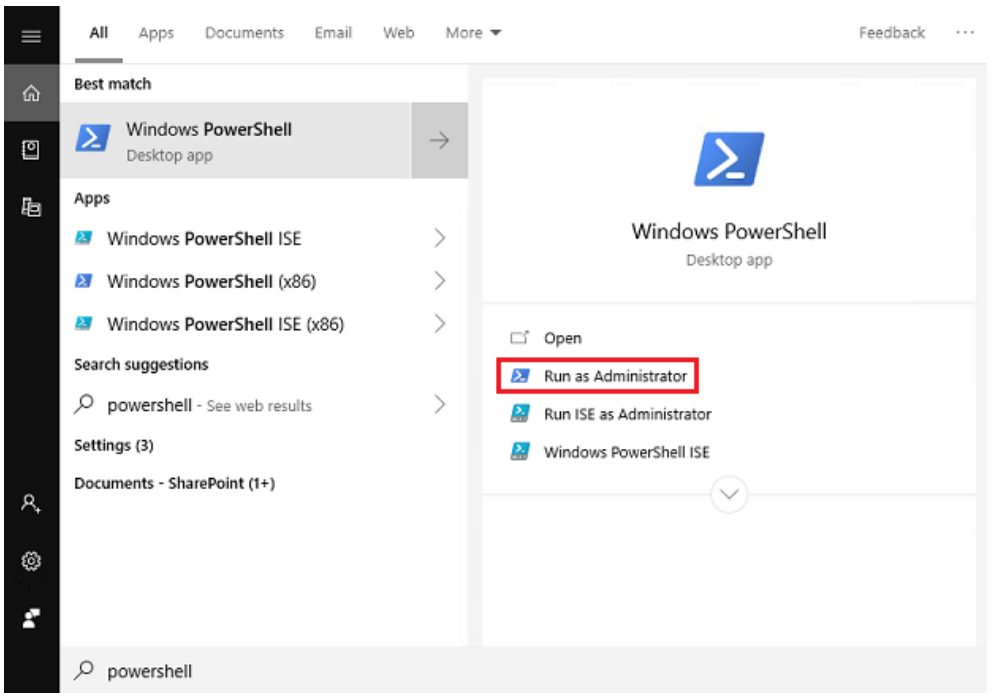


Рисунок 17. 1 – Запуск PowerShell

Windows PowerShell має так звані командлети (cmdlets).

Це спеціалізовані класи .NET, які реалізують функціональність PowerShell.

Вони мають назви у відповідності «дія – об'єкт» (дієслово-іменник, англ. Verb-Noun).

В командлетах для визначення дій використовуються наступні ключові слова:

Add – додати;

Clear – очистити;

Enable – включити;

Disable – виключити;

New – створити;

Remove – видалити;

Start – запустити;

Stop – зупинити;

Export – експортувати;

Import – імпортувати.

Get – отримати дані

Set – задати, призначити

Clear – очистити

Наприклад, Get-Help буквально означає «Отримати-Допомога» або в контексті PowerShell – «Показати-Довідку». Це аналог команди help в консолі. Такі «подібності» не випадкові, в PowerShell аналоги команд консолі називаються аліясами (Alias) і їх можна викликати за допомогою Get-Alias (рис. 17.2).

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 «;».

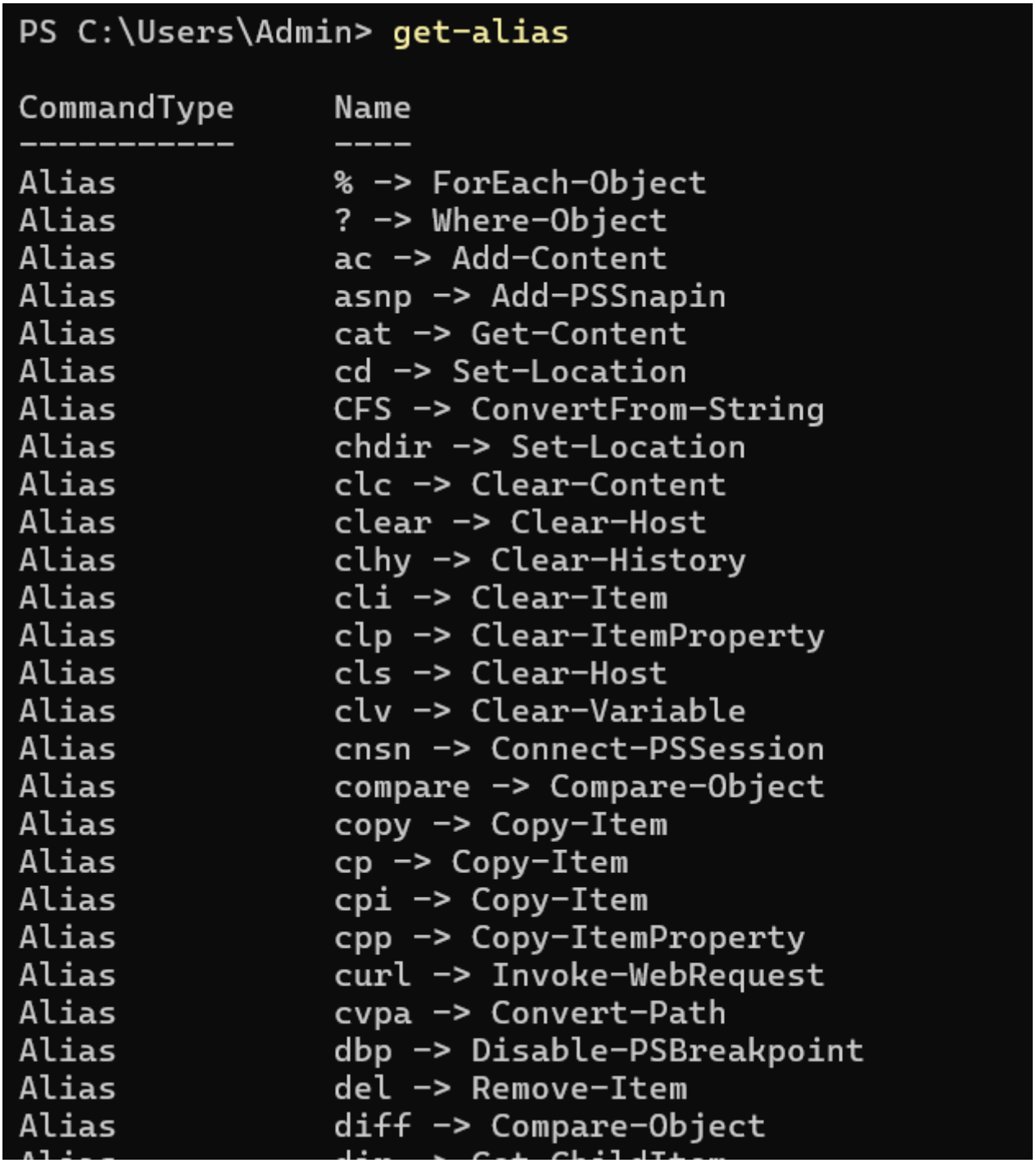


Рисунок 17. 2 – Запуск Get-Alias

Для пошуку об'єкта та дії використовується командлет Get-Command. Усі командлети PowerShell можна викликати за допомогою Get-Command (спробуйте це зробити). Показати довідку про нього можна наступним чином:

Get-Help Get-Command

Робота з процесами ОС Windows 10 має інструменти для отримання списку доступних командлетів управління процесами. А саме команда Get-Command -Noun Process виводить цей список:

Рисунок 17. 3 – Запуск Get-Command -Noun Process

Компоненти

У PowerShell застосовуються деякі функції, характерні для традиційних оболонок:

Вбудована довідкова система. Більшість оболонок мають подібність довідкової системи, з якої можна отримати додаткові відомості про команду. Наприклад, можна дізнатися, що робить команда та які параметри вона підтримує. Довідкова система PowerShell не тільки надає відомості про команди, але також інтегрується зі статтями довідки в Інтернеті.

Конвеєр. Традиційні оболонки використовують конвеєр для послідовного виконання безлічі команд. Вихідні дані однієї команди є вхідними даними наступної команди. У PowerShell реалізована ця концепція, характерна для традиційних оболонок, проте вона відрізняється, оскільки працює з об'єктами поверх тексту. З цією функцією ви ознайомитеся докладніше в рамках цього модуля.

Псевдоніми: Псевдоніми – це альтернативні імена, які можна використовувати для виконання команд. PowerShell підтримує використання загальних псевдонімів, таких як cls (очищення екрана) та ls (виведення списку файлів). Тому нові користувачі можуть використовувати свої знання про інші платформи і необов'язково запам'ятовувати ім'я PowerShell для виконання знайомих команд.

PowerShell має ряд відмінностей від традиційної оболонки командного рядка:

Вона працює з об'єктами поверх тексту. В оболонці командного рядка необхідно виконувати сценарії, вихідні та вхідні дані яких можуть відрізнятися, тому ви можете витрачати час на форматування вихідних даних та отримання потрібної вам інформації. У PowerShell як вхідні і вихідні дані використовуються об'єкти. Це означає, що ви витрачаєте менше часу на форматування та вилучення.

Вона містить командлети. Команди PowerShell називаються командлетами (вимовляється як командлети). На відміну від багатьох інших оболонок командлети

PowerShell створюються на основі загального середовища виконання, а не окремих файлів, що виконуються. Ця особливість забезпечує однаковість при обробці параметрів та поведінці конвеєра.

Як правило, командлети приймають об'єкти як вхідні дані, і повертаються також об'єкти. Основні командлети PowerShell вбудовані в .NET Core і є відкритими. Ви можете розширити PowerShell за допомогою додаткових командлетів, скриптів та функцій спільноти та інших джерел, а також створити власні командлети у .NET Core або PowerShell.

Вона містить багато типів команд. Командами PowerShell можуть бути власні виконувані файли, командлети, функції, сценарії або псевдоніми. Кожна команда, що виконується, належить одному з цих типів. Слова команди та командлети часто взаємозамінні, оскільки командлет – це тип команди.

Пошук команди за допомогою Get-Command

Якщо виконати командлет Get-Command, з'явиться список усіх команд, встановлених у PowerShell. Оскільки доступні тисячі команд, необхідно відфільтрувати відповідь, щоб можна було швидко знайти потрібну команду.

Щоб відфільтрувати список, пам'ятайте про стандарт іменування командлетів ("дієслово-іменник"). Наприклад, у команді Get-Random Get є дієсловом, а Random – іменником. Використовуйте прапорці для вибору дієслова або іменника у потрібній команді. Заданий прапорець передбачає значення, яке є рядком. У цей рядок можна додати символи, які відповідають шаблону, щоб переконатися, що, наприклад, значення прапора має починатися або закінчуватися певним рядком.

У прикладах нижче показано, як використовувати прапори для фільтрації списку команд:

-Noun. -Noun вказує на частину імені команди, пов'язану з іменником. Тобто, він призначений для всіх елементів після дефісу (-). Ось типовий пошук команди на ім'я:

Get-Command -Noun a-noun*

Ця команда виконує пошук всіх командлетів, іменник у яких починається з a-noun.

-Verb. -Verb вказує на частину імені команди, яка пов'язана з дієсловом. Можна поєднувати прапори -Noun та -Verb, щоб створити ще більш детальний пошуковий запит і тип. Нижче наведено приклад:

Get-Command -Verb Get -Noun a-noun*

Тепер ми уточнили умови пошуку, щоб вказати, що дієслово в команді має бути Get, а іменник – a-noun.

Виконайте команду Get-Command із прапором -Noun. Вкажіть File*, щоб знайти все, що пов'язане із файлами:

При перших сеансах роботи рекомендується подивитися і порівняти результати виконання кількох команд, уже відомих користувачеві, наприклад по роботі з інтерпретатором команд «Cmd.exe». Практично всі команди інтерпретатора мають аналоги з тими ж іменами (псевдонімами), але уявлення даних відрізняється, іноді дуже значно. Перш за все, слід відзначити рівень деталізації інформації. подивимося результат виконання команди dir в середовищі PowerShell:

```
PS C:\Users\User> dir


    Каталог: C:\Users\User


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         16.03.2019     21:42            .android
d----         25.02.2019     19:25            .AndroidStudio3.3
d----         28.02.2019     15:19            .borland
d----         26.02.2019     20:35            .dotnet
d----         16.03.2019     16:14            .eric6
d----         16.03.2019     15:52            .idlerc
d----         03.12.2019     23:40            .imagej
d----         20.03.2020     22:55            .nuget
d----         16.03.2019     16:23            .PyCharmCE2018.3
d----         20.03.2020     22:54            .templateengine
d----         28.01.2019     13:01            .thumbnails
d----         11.04.2019     11:26            ARISExpress
d-r--         25.10.2016      9:05            Contacts
d-r--         23.09.2020     21:09            Desktop
d-r--         11.07.2020      9:45            Documents
d-r--         21.03.2020     18:36            Downloads
d-r--         25.10.2016      9:05            Favorites
d----         16.09.2020     17:54            Khr
d-r--         28.05.2019     20:31            Links
d-r--         15.04.2019     20:50            Music
d-r--         29.04.2020     19:59            Pictures
d----         16.03.2019     16:26            PycharmProjects
d-r--         25.10.2016      9:05            Saved Games
d-r--         25.10.2016      9:05            Searches
d----         26.02.2019     22:10            source
d----         07.08.2019     23:57            SpineTrial
d-r--         22.05.2020     20:06            Videos
-a---         28.02.2019     14:49            319 ia_oasis.log
-a---         28.02.2019     12:12           4593 regwizard.log
-a---         28.02.2019     14:49           1482 sanct.log
-a---         06.09.2019     18:38           1626 txt111
-a---         06.09.2019     18:39           1674 txt222.txt


PS C:\Users\User>
```

Рисунок 17.4 – Результат виконання команди dir

У середовищі PowerShell є специфічна команда (командлет) Get-Childltem, яка також має додаткове ім'я (псевдонім) dir. Робота цієї команди без параметрів

представлена на рисунку вгорі. У новій редакції команди dir з'явився стовпець Mode, що відображає можливі режими використання програмних коштів, дуже схожі на режими Unix. режими визначаються окремо для каталогів і файлів. Повний перелік характеристик, виведених з різним командам, можна подивитися за допомогою командлет Get-Member.

Командний рядок PowerShell крім набору і виконання команд надає користувачеві можливість обчислень арифметичних виразів різної складності. У найпростішому випадку вона забезпечує обчислення як калькулятор. після запису вираження в командному рядку і натискання клавіші «Enter» результат обчислення відображається на наступному рядку. кілька простих прикладів наведено нижче на рисунку 17.5:

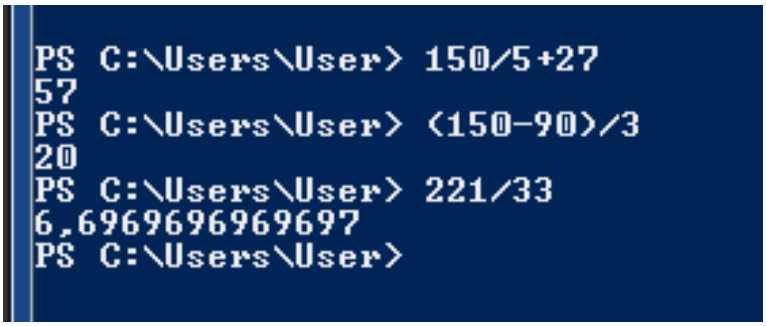


Рисунок 17.5 – Результат обчислення арифметичних виразів

У більш складних випадках вираження можуть включати різні математичні функції. Їх реалізація забезпечується шляхом звернення до бібліотек класів платформи .NET, зокрема до методам класу System.Math.

При складних обчисленнях може знадобитися збереження проміжних результатів в якихось елементах пам'яті. Для цього слід простими засобами визначити ім'я змінної і визначити її значення. Імена змінних повинні починатися знаком $. Запис тільки імені змінної після знака долара означає звернення до виведення її значення:

На перших сеансах роботи користувачам буде корисне використання команд-псевдонімів cls (очищення екрану дисплея) і cd (Зміна каталогу), аналогічних по роботі з інтерпретатором команд cmd.exe. Функціональність цих команд залишається такою ж.

Індивідуальні завдання до лабораторного заняття №17

1. Викликати узагальнену довідку по пакету PowerShell, набравши в командному рядку Get-Help без параметрів. переглянути довідкові дані по команді help. Ознайомитися з контекстом команд. перша команда видає лише однієї сторінки довідку, а остання команда дає багатосторінкову довідку.

2. Показати всі розділи довідкової системи, набравши команду GetHelp *. Параметр * є шаблоном, що позначає «будь-яке поєднання символів ».

3. Ознайомитися зі структурою PowerShell за переліком розділів довідки, набравши по дві команди, зазначені в якості прикладів, по кожному з розділів. Подивитися, як змінюється зміст довідкових даних, якщо в команду довідки включаються параметри – detailed або -full.

4. Переглянути довідку по Командлети Get-Process, яка відображає процеси, активізовані в локальному комп'ютері користувача. Для цього набираємо в командному рядку команду PS З: \ users \ student> Get-Help Get-process -full Ознайомитися з переліком характеристик процесів, активізованих в комп'ютері.

5. Переглянути довідку по Командлети Get-Process, набравши команду PS C: \ users \ student> Get-process \? Порівняти отриману довідку з попередніми даними пункту.

6. Перегляньте роботу засобів PowerShell за вказаними псевдонімами: cd, ls, copy, del, dir, echo, erase, more, popd, pushd, ren. Вкажіть ці аліаси.

7. Випробувати роботу PowerShell в режимі калькулятора для обчислення простих арифметичних виразів: п'ять арифметичних виразів.

8. Випробувати роботу PowerShell в режимі калькулятора для обчислення простих змінних: п'ять змінних і одна змінна підсумкова.

Контрольні питання

1. Що таке Windows PowerShell і чим вона відрізняється від командного рядка (CMD)?

2. Які основні компоненти має PowerShell (оболонка, мова сценаріїв, модулі тощо)?

3. Що таке cmdlet? Наведіть приклади типових cmdlet-ів для адміністрування.

4. Як переглянути список усіх доступних командлетів у PowerShell?
5. Як вивести список процесів, що виконуються, за допомогою PowerShell?
6. Як керувати службами Windows за допомогою PowerShell?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №18
РОБОТА З ДИСКАМИ В POWERSHELL

Мета роботи – навчитись працювати з дисками в оболонці PoweShell. Вміти застосовувати функціонал PoweShell для реалізації завдань щодо адміністрування операційної системи Windows

Теоретичні відомості

Одним з найважливіших ресурсів комп'ютерних систем є ресурс пам'яті. З появою багатоядерних мікропроцесорів значення цього ресурсу ще більш зростає і виходить на передній план.

Фундаментальним положенням будь-якої ОС є управління даними, здійснюване з боку файлової системи. файлова система являє собою дерево вкладених каталогів (папок) і файлів. У командній оболонці PowerShell поняття диска, файлу і папок значно розширені і практично еквівалентні однойменною поняттям Unix і Linux ОС. Як файли можуть виступати не тільки дані, що знаходяться на зовнішніх носіях, але і фізичні і логічні пристрої, диски, їх розділи і т. п. Це значно спрощує роботу ОС і дозволяє засобами файлових систем контролювати роботу будь-яких сховищ даних: як локальних, так і мережевих. Крім того, використовуючи в якості псевдонімів назви керуючих операторів ОС, відмінних від Windows, можна керувати по-різному організованими даними.

У кожному сеансі роботи користувачеві необхідно знати, які ресурси пам'яті не тільки його комп'ютера, але і мережевих сховищ доступні. Обсяг одержуваної інформації про ресурси може бути дуже великим.

Список дисків, доступних користувачеві з середовища PowerShell, можна отримати командою Get-PSDrive (рис. 18.1):

Інформація про доступні сховища є вихідною для роботи з ресурсом пам'яті. За командою Get-PSDrive повідомляється точне позначення диска (Root), його ім'я (Name), ім'я провайдера (Provider), що підтримує цей диск, і поточна локалізація (CurretLocation). Крім цих даних вказуються доступні функції, псевдоніми, змінні

оточення і т. п. В PowerShell вбудовані специфічні провайдери, що забезпечують доступ до спеціальних сховищ:

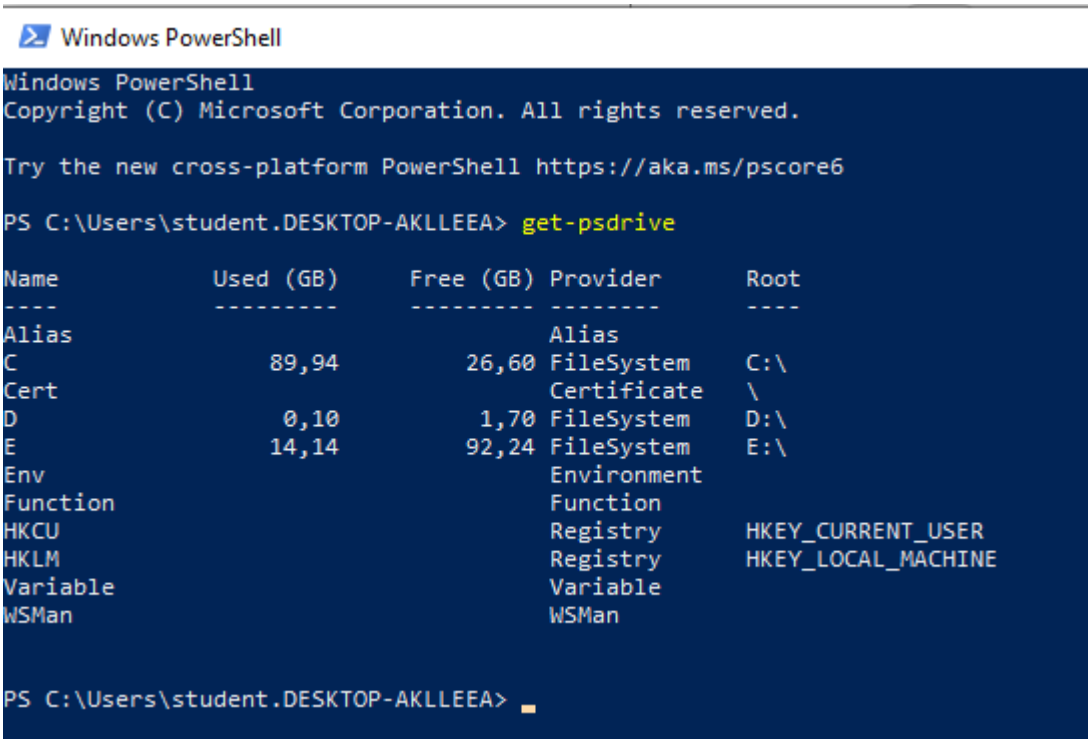


Рисунок 18.1 – Список дисків

Alias - для доступу до псевдонімів PowerShell,

Certificate – для використання сертифікатів X509 цифрового підпису,

Environment – для змінних середовища Windows,

FileSystem – для звернення до файлової системі,

Function – для звернення до функцій PowerShell,

Registry – для звернення до реестру Windows (гілки реєстру HKCU – поточного користувача і HKLM – локальної машини),

Variable – для змінних PowerShell.

Провайдер PowerShell – це NET-додаток, що надає користувачам оболонки доступ до сховищ в єдиному форматі, що нагадує формат звичайних дисків файлової системи. Робота з представленими дисками практично нічим не відрізняється від роботи зі звичайною файловою системою. Навігація по різним дискам, перегляд їх вмісту, звернення до елементів даних виконується за допомогою команд: командлетів Get-Location і Set-Location з їх псевдонімами pwd, cd, chdir, si. Всі переміщення по дисках здійснюються командлетом GetLocation, або за допомогою його псевдоніма cd

(chdir – повне ім'я), як і в файловій системі Windows з використанням інтерпретатора «cmd.exe».

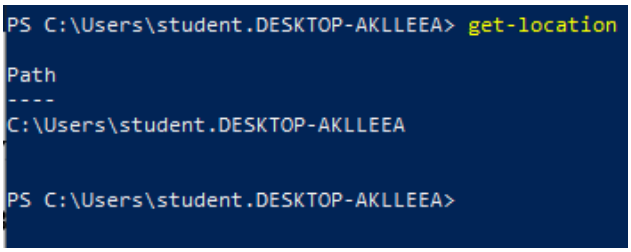


Рисунок 18.2 – Get-Location

Ця команда відображає поточне розташування на поточному диску Windows PowerShell. Наприклад, якщо користувач працює в каталозі Windows на диску C:, команда відобразить шлях до цього каталогу.

Примітка. Якщо ввести в PS

get-help get-location –examples,

то отримаємо довідку по команді get-location з прикладами

Файлова система в PowerShell керує лише фізичними і логічними дисками (C: \, D: \, E: \). Всі інші сховища керуються власними провайдерами. Сама файлова система контролює тільки частину простору, іменованого дисками. Очевидно, що можливості PowerShell набагато ширші від програми «Провідник» ОС Windows.

Список всіх провайдерів оболонки може бути отриманий командлети: Get-PSProvider:

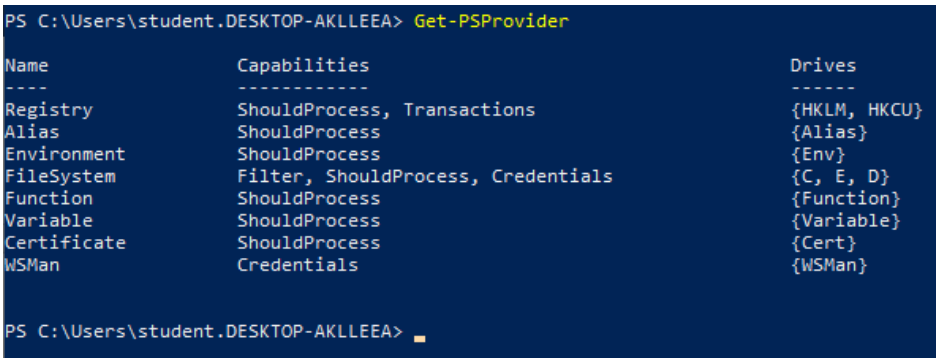


Рисунок 18.2 – Get-PSProvider

Навігація по дисках PowerShell нічим не відрізняється від типовою роботи файлової системи. Тут також зберігається поняття робочого або поточного каталогу.

Шлях до цього каталогу встановлює вже згадуваний командлет Get-Location (псевдонім cd) без параметрів.

Створення нових дисків (сховищ)

Для прикладу можна розв’язати таку задачу: створимо новий диск всередині папки user, який буде містити каталог з ім'ям Ivan.

Після цього досить знову набрати команду Get-PSDriver і переконатися, що з'явився новий диск, доступ до якого забезпечує файлова система.

При роботі з файловою системою користувач повинен вміти створювати каталоги (папки) і файли, копіювати їх і переміщати по власним бажанням. У середовищі PowerShell для цього є всі необхідні засоби.

Вивчення будь-якої файлової системи починають з команд, що забезпечують отримання списку каталогів і файлів. У PowerShell для цих завдань призначений командлет Get-Childltem і його псевдонім dir, більш звичний для користувачів персональних комп'ютерів. Відмінною особливістю нових засобів є їх розширена функціональність.

Розглянемо кілька типових прикладів їх застосування. Використання параметра -Recurse (рекурсія) дозволяє відобразити зміст будь-якого диска з його каталогами і підкаталогами. команда:

Dir '[папка]' -Recurse

дозволяє переглянути повний вміст каталогу «папка», тобто всі файли, що входять в каталог (рис. 18.3).

Розглянемо приклад, який ілюструє використання більшого числа полів в форматі командлет. Нехай, наприклад, в каталозі C:\SORT потрібно знайти всі doc-файли, імена яких починаються буквою «t» (рис. 18.4).

Вирішенням цього завдання може служити команда:

dir -Recurse -Filter t*.docx -Path C:\SORT

огляду на, що синтаксис оболонки PowerShell не критичний в щодо великих і малих літер, а також допускає скорочення слів до декількох символів, що забезпечують однозначне розуміння термінів, запис можна скоротити:

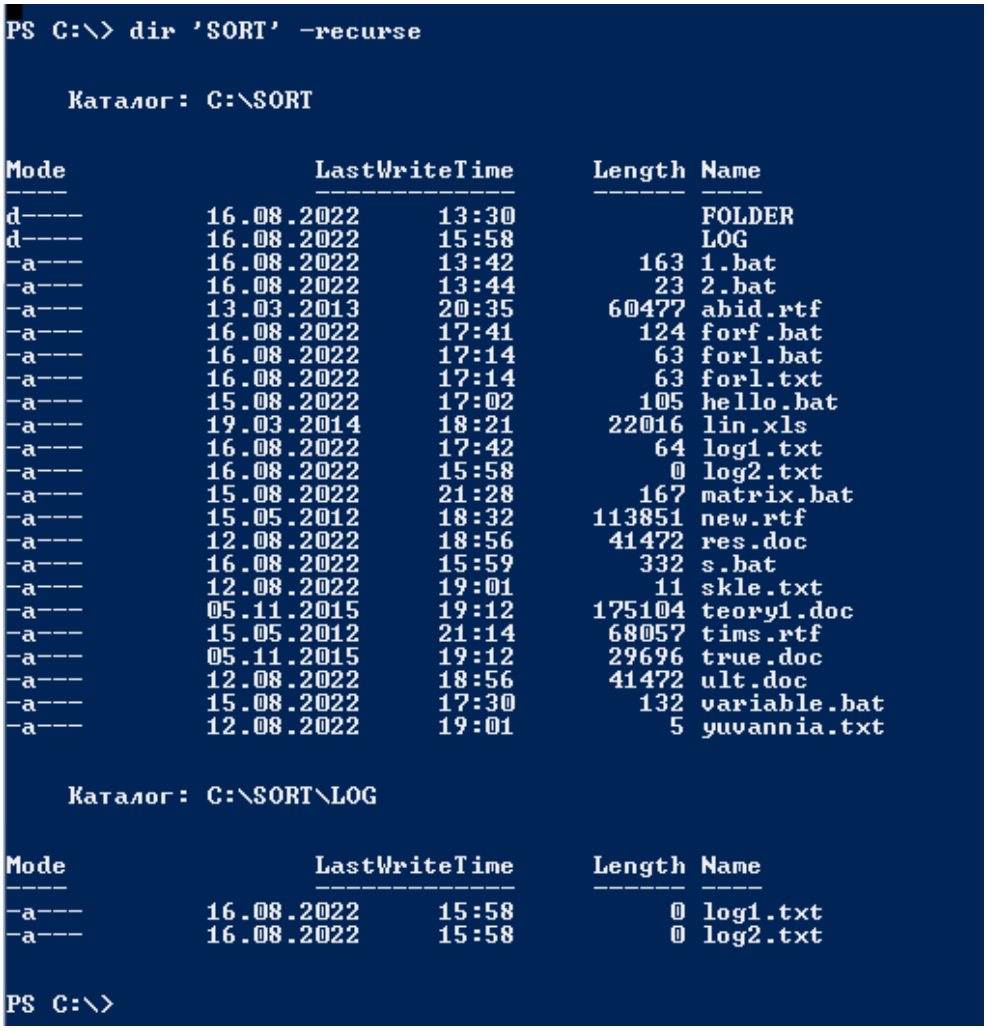


Рисунок 18.3 – Сортування

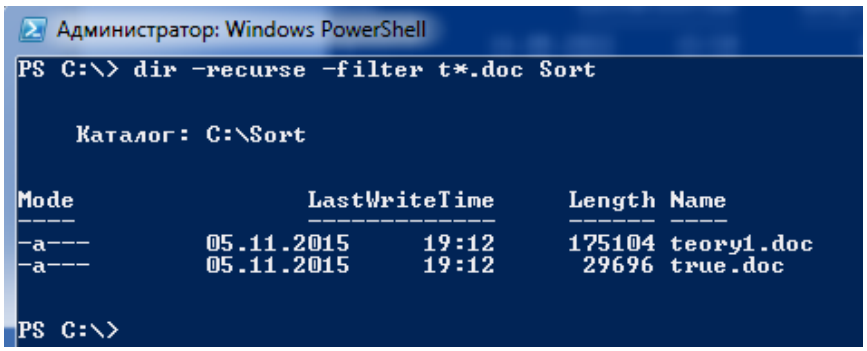


Рисунок 18.4 – Сортування

У цьому виразі псевдонім dir замінює ім'я командлета GetChildItem, ключ -r є скороченням -Recurse. Цей ключ поширює дію команди не тільки на вказаний каталог, але і на всі його підкаталоги. Параметр фільтр -fi (-Filter) з аргументом t*.doc задає маску файлів для пошуку. Ще один параметр -Path з аргументом C:\SORT визначає шлях до досліджуваного каталогу. Ім'я параметра з ключовим словом -Path годі й записувати, якщо з контексту вираз не має інших значень.

Інший процедурою файлової системи PowerShell є створення нових каталогів і файлів. Ці функції виконує командлет New-Item. Як параметри потрібно вказати місце розміщення створюваного об'єкта. Для цього в параметрі -Path прописується повний шлях до каталогу, в якому створюється об'єкт, а в параметрі -Type вказується тип об'єкту – «directory» або «file». Створимо в каталозі диска «C:\SORT» новий каталог з ім'ям CYBER2023:

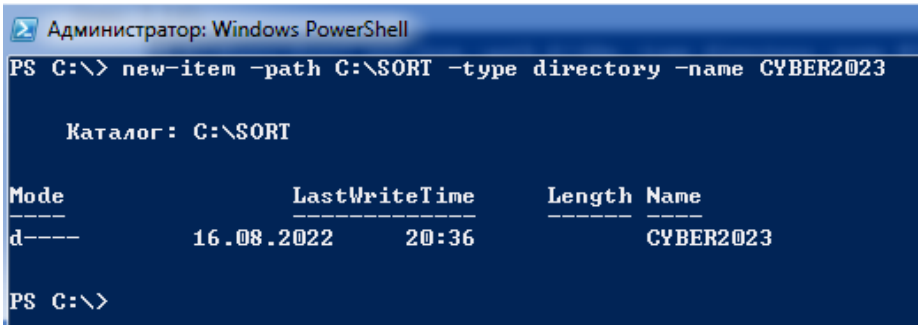


Рисунок 18.5 – Створення нового каталога

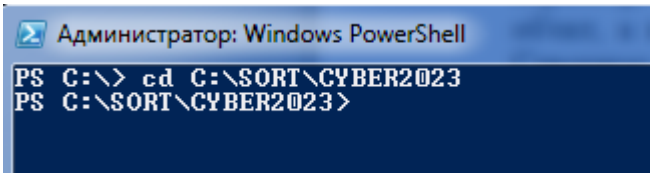


Рисунок 18.5 – Перехід у створений каталог

Оголосимо і створимо текстовий файл Лабораторна11. Оскільки в параметрі -Path шлях до файлу не вказано (можна опустити і назву параметра), то за замовчуванням файл створюється в поточному каталозі:

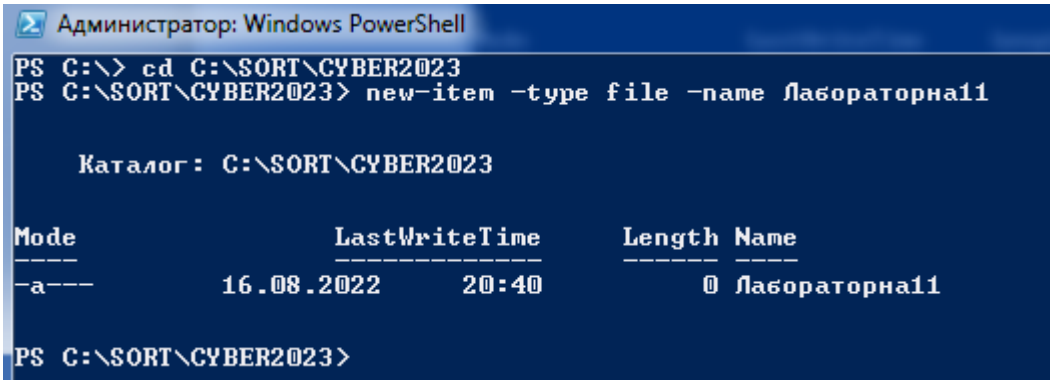


Рисунок 18.6 – Створення текстового файлу

Індивідуальні завдання до лабораторного заняття №18

1. Запишіть команду пошуку у вашому каталозі файлів (з використанням масок вводу) з командлетом GetChildItem. Продемонструйте результат.
2. Відсортуйте усі файли вибраної вами папки та створіть там кілька нових елементів6 каталоги чи файли. Продемонструйте результат.
3. Відоразіть список дисків, доступних у вашій операційній системі через командлети. Продемонструйте результат.
4. Наведіть приклад використання одного зі специфічних провайдерів, що забезпечують доступ до спеціальних сховищ у операційній системі. Поясніть командлети, ключі, продемонструйте результат.

Контрольні питання

1. Як здійснюється робота з файлами та каталогами у PowerShell?
2. Що таке скрипт PowerShell і як його створити та виконати?
3. Як обробляти помилки у PowerShell-скриптах?
4. Як автоматизувати запуск PowerShell-скриптів через Планувальник завдань (Task Scheduler)?
5. Як використовувати PowerShell для встановлення або видалення програм чи компонентів Windows?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №19
АДМІНІСТРУВАННЯ WINDOWS У СЕРЕДОВИЩІ POWESHELL

Мета роботи – створювати скрипти для адміністрування Windows у середовищі PoweShell. Вміти застосовувати функціонал PoweShell для реалізації завдань щодо адміністрування операційної системи Windows

Література: [1, 2, 3, 5]

Теоретичні відомості

Скрипт – це просто набір команд, збережених в текстовий файл (з розширенням .ps1), які PowerShell може зрозуміти і виконати в заданій послідовності. Єдине попередження полягає в тому, що на відміну від командного рядка, протокол безпеки за замовчуванням запобігає виконанню всіх сценаріїв.

Це означає, що при подвійному натисканні .ps1 файлу в системі Windows 10 нічого не станеться, і якщо ви намагаєтеся виконати скрипт в PowerShell, ви отримаєте повідомлення про помилку: «Не вдалося завантажити вихідний файл, тому що заборонено виконання сценаріїв в цій системі». Проте, запускати сценарії на вашому пристрої досить просто. Вам просто потрібно включити правильну політику виконання.

1 спосіб Створення скрипта за допомогою блокнота

Щоб створити сценарій PowerShell за допомогою блокнота, виконайте наступні дії:

Відкрийте програму «Блокнот».

Створіть або вставте сценарій. Наприклад:

Write-Host "« Вітаємо! Ваш перший скрипт успішно виконаний »"

Вищенаведений скрипт просто виводить на екрані фразу «Вітаємо! Ваш перший скрипт успішно виконаний ».

Збережіть файл під будь-якою зручною назвою, наприклад, first_script.ps1

2 спосіб Створення сценарію за допомогою інтегрованого середовища сценаріїв

Консоль **PowerShell ISE** можна використовувати для кодування сценаріїв в Windows 10. Інтегроване середовище сценаріїв є складним інструментом, але ви можете почати роботу з допомогою цих кроків:

Відкрийте системний пошук і введіть запит Windows PowerShell ISE, клацніть правою кнопкою миші верхній результат, і виберіть Запуск від імені адміністратора або виберіть відповідний параметр в правій колонці

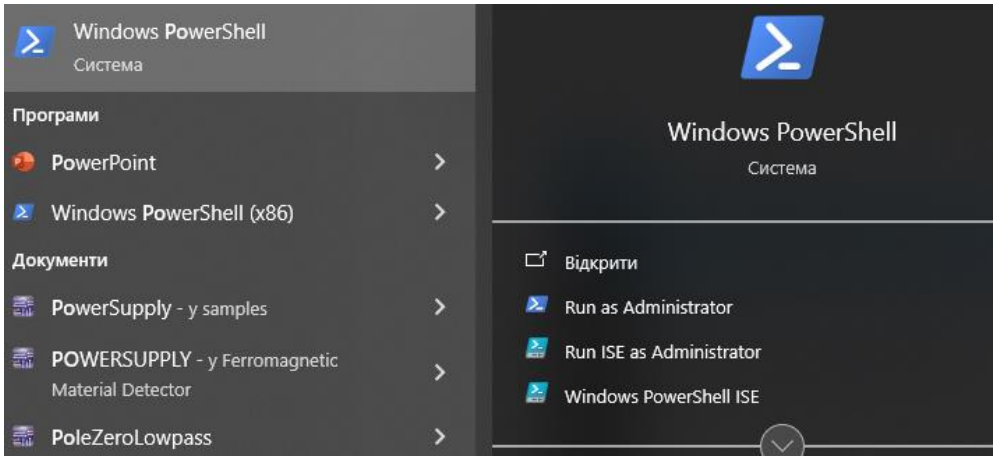


Рисунок 19.1 – Запуск від імені адміністратора

У PowerShell ISE створіть порожній файл .ps1, в якому можна створити або вставити скрипт. Наприклад:

Write-Host "« Вітаємо! Ваш перший скрипт успішно виконаний »"

Відкрийте меню Файл і натисніть кнопку Зберегти.

Введіть назву сценарію. Наприклад, first_script_ise.ps1

Збережіть скрипт.

Як тільки Ви виконали ці кроки за допомогою Блокнота або PowerShell ISE, сценарій готовий до запуску, але він не буде виконаний. Це відбувається тому, що параметри PowerShell за замовчуванням завжди налаштовані на блокування виконання будь-якого сценарію.

Запуск файлу сценарію PowerShell

Щоб запустити файл сценарію в PowerShell, необхідно змінити політику виконання, виконавши такі дії:

Відкрийте пошук і введіть PowerShell, клацніть правою кнопкою миші в верхній результат і виберіть Запуск від імені адміністратора.

Введіть наступну команду, щоб дозволити виконання скриптів і натисніть клавішу Enter:

Set-ExecutionPolicy RemoteSigned

Вкажіть тип A та ще раз натисніть клавішу Enter.

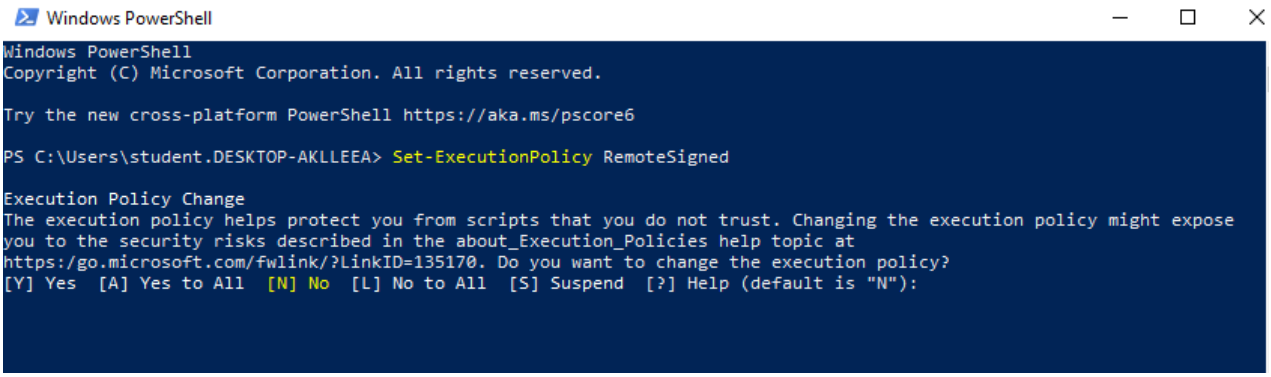


Рисунок 19.2 – Дозвіл на виконання від імені адміністратора

Введіть наступну команду для запуску скрипта і натисніть клавішу Enter:

& "C: \ PATH \ to \ SCRIPT \ first_script.ps1"

У наведеній вище команді обов'язково змініть PATH\to\SCRIPT на шлях розташування вашого скрипта.

Після виконання цих кроків сценарій буде запущений, і якщо він був створений правильно, ви повинні побачити його висновок без проблем.

PowerShell в Windows 10 включає чотири політики виконання:

Restricted – зупиняє виконання скрипта.

RemoteSigned – запускає скрипти, створені на пристрої. Однак, сценарії, створені на іншому комп'ютері, не будуть запускатися, якщо вони не містять підпису довіреного видавця.

AllSigned – всі скрипти будуть працювати до тих пір, поки вони підписані надійним видавцем.

Unrestricted запускає будь скрипт без будь-яких обмежень.

У наведених вище кроках ми використовували команду, щоб дозволити запуск локальних скриптів в Windows 10. Однак, якщо ви не плануєте регулярно виконувати скрипти, можна відновити налаштування за замовчуванням, використовуючи ті ж інструкції, але на попередньому кроці, обов'язково використовуйте Set-ExecutionPolicy Restricted команду!

1. Написання скрипта

Скрипт PowerShell (не важливо якої версії) – це текстовий файл (як уже вказано, з розширенням * .ps1)

Ось приклад простого Power Shell сріпта (файл systemInfo.ps1):

Retrieve WMI object for the operating system

Get-WmiObject Win32_OperatingSystem

Цей файл можна створювати і редагувати, наприклад, в FAR Manager.

Зверніть увагу, що FAR Manager хоч і може працювати в консолі Power Shell, але виконує з-під себе скрипти в оточенні звичайної Windows-консолі cmd. Тобто, FAR Manager можна використовувати тільки для створення і редагування PowerShell скриптів, але не для запуску. Але перш ніж розчаровуватися, прочитайте пункт 3.

2. Запуск сріпта

Скрипт потрібно виконувати з консолі Power Shell, а не зі звичайної консолі Windows. В консолі Power Shell необхідно перейти в каталог, де лежить скрипт (командами cd), і потім запустити сам скрипт, обов'язково прописавши перед ним символи ". \". Наприклад, маємо шлях до файлу скрипта d: \ work \ systemInfo.ps1. Тоді команди запуску будуть виглядати так:

d:

cd \

cd work

. \ SystemInfo.ps1

або так (просто вказується повний шлях до скрипта):

d: \ work \ systemInfo.ps1

Швидше за все, при запуску скрипта з'явиться наступна помилка:

```
Не удается загрузить файл D:\work\systemInfo.ps1, так как выполнение скриптов запрещено для данной системы. Введите "get-
help about_signing" для получения дополнительных сведений.
строка:1 знак: 18
    + CategoryInfo : NotSpecified: (:) [], PSSecurityException
    + FullyQualifiedErrorId : RuntimeException
```

Неможливо завантажити файл D:\work\systemInfo.ps1, так як виконання скриптів заборонено для даної системи. Введіть "get-help about_signing" для отримання додаткових відомостей.

рядок: 1 знак: 18

+ CategoryInfo: NotSpecified: (:) [], PSSecurityException

+ FullyQualifiedErrorId: RuntimeException

Ця помилка з'являється через те, що за замовчуванням в Power Shell включена максимальна політика безпеки, яка дозволяє виконувати команди PowerShell в командному рядку, але не дозволяє в тій же командному рядку виконати скрипт з командами PowerShell.

Щоб дозволити виконання PowerShell скриптів, потрібно створити .bat файл, наприклад enableScript.bat наступного змісту:

powershell -Command Set-ExecutionPolicy RemoteSigned

Цей * .bat файл можна виконати в будь-який консолі: хоч в PowerShell, хоч у звичайній cmd. Після виконання цього файлу, PowerShell-скрипти стануть запускатися в консолі PowerShell.

3. Запуск PowerShell-скрипта зі звичайної Windows-консолі cmd

Скрипт PowerShell можна виконати і зі звичайної консолі Windows. Для цього можна скористатися командою:

Powershell -File ./systemInfo.ps1

Таким чином можна виконувати скрипти прямо з FAR Manager, і вони будуть працювати. Тут є невелика тонкість. Параметр -File спрацьовує тільки на локальних шляхах, навіть якщо шлях зазначений відносний "./". Тобто, якщо .ps1 – файл лежить на локальному диску C: або D:, то такий виклик буде працювати. Але, якщо спробувати виконати скрипт, розташований на доменному ресурсі, то скрипт не буде знайдений. Можливо, це виправлять в наступних версіях PowerShell.

Індивідуальні завдання до лабораторного заняття №19

Напишіть два власних скрипта в PowerShell. Поясніть їх вміст, призначення. Продемонструйте виконання.

Контрольні питання

1. Як запустити PowerShell від імені адміністратора?
2. Як підключитися до віддаленого комп’ютера через PowerShell Remoting?
3. Як отримати облікові дані через PowerShell (Get-Credential)?
4. Як обмежити виконання скриптів у системі?

ЛАБОРАТОРНЕ ЗАНЯТТЯ №20
РОЗРОБКА КОНСОЛЬНИХ ДОДАТКІВ ДЛЯ ОС WINDOWS: РОЗРОБКА КОНСОЛЬНОЇ УТИЛІТИ З МЕНЮ КОМАНД У СЕРЕДОВИЩІ VISUAL STUDIO

Мета роботи – ознайомитися з принципами створення системної консольної утиліти з меню команд; навчитися реалізовувати обробку команд, інкапсульовувати логіку в класи, використовувати структури даних для зберігання результатів.

Література [1, 2, 3, 4]

Теоретичні відомості

Системне програмне забезпечення включає низькорівневі утиліти, які взаємодіють з операційною системою, апаратним забезпеченням або забезпечують інфраструктуру для прикладних програм. Прикладом є файлові менеджери, командні інтерпретатори, системні монітори.

Консольна утиліта – це типова форма системної програми, що запускається з командного рядка й виконує певну операцію: опрацювання аргументів, облік стану, взаємодія з файлами, пам’яттю чи користувачем.

Такі утиліти можуть бути побудовані у вигляді командного меню, де кожна команда виконує певну дію. Для організації коду доцільно використовувати класи, внутрішні стеки/черги/вектори, модульність.

Консольний додаток – це програмне забезпечення, що працює у середовищі командного рядка, без графічного інтерфейсу. Такі додатки використовуються для обчислень, обробки тексту, файлових операцій тощо. У середовищі Visual Studio створення консольного додатку передбачає налаштування проекту типу *Console App*, написання коду, компіляцію та запуск.

Розглянемо приклад реалізації консольної утиліти «Командний журнал».

Завдання. Реалізувати консольну утиліту, яка імітує обробник простих команд користувача. Кожна введена команда зберігається у стек, доступні операції – додати команду, переглянути останню, вивести всі, очистити історію.

Програмний код для реалізації наступний:

```
#include <iostream>
#include <stack>
#include <string>
using namespace std;

class CommandLogger {
private:
   stack<string> history;

public:
   void addCommand(const string& cmd) {
      history.push(cmd);
      cout << "Команду збережено: " << cmd << endl;
   }

   void showLastCommand() {
      if (!history.empty()) {
         cout << "Остання команда: " << history.top() << endl;
      } else {
         cout << "Історія порожня." << endl;
      }
   }

   void showAllCommands() {
      stack<string> temp = history;
      cout << "Журнал команд (останнє → перше):" << endl;
      while (!temp.empty()) {
         cout << temp.top() << endl;
         temp.pop();
      }
   }

   void clear() {
      while (!history.empty()) history.pop();
      cout << "Журнал очищено." << endl;
   }
};

int main() {
   CommandLogger logger;
   int choice;
   string cmd;

   do {
      cout << "\n1 – Ввести команду\n2 – Показати останню\n3 – Всі команди\n4 – Очистити\n0 – Вихід\nВаш вибір: ";
      cin >> choice;
      cin.ignore();  // очистка буфера

      switch (choice) {
         case 1:
            cout << "Введіть команду: ";
```

```
            getline(cin, cmd);
            logger.addCommand(cmd);
            break;
        case 2:
            logger.showLastCommand();
            break;
        case 3:
            logger.showAllCommands();
            break;
        case 4:
            logger.clear();
            break;
      }
   } while (choice != 0);

   return 0;
}
```

Результат виконання:

```
1 – Ввести команду
2 – Показати останню
3 – Всі команди
4 – Очистити
0 – Вихід
Ваш вибір: 1
Введіть команду: mkdir new_folder
Команду збережено: mkdir new_folder
```

Індивідуальні завдання до лабораторного заняття № 20

Розробити консольну утиліту системного типу (на зразок команд оболонки, облікових менеджерів, симуляторів системних команд) згідно нижче поданого варіанту, яка має:

– текстове меню;

– клас(-и) з інкапсульованою логікою;

– внутрішнє зберігання даних (черга, стек, масив);

– базову взаємодію з користувачем.

1. Утиліта керування файлами: зберігає імена файлів, дозволяє додати, видалити, переглянути список.

2. Менеджер симуляції процесів: додає «процеси» у чергу, завершує їх, показує список.

3. Система обліку логів: додає повідомлення у стек, виводить історію.

4. Утиліта «echo-історія»: зберігає текст, який вводився, і повторює на запит.

5. Менеджер підключень: вводиться IP-адреса, фіксується, доступні перегляд/видалення.

6. Калькулятор з історією операцій.

7. Менеджер запуску команд: вводяться команди (не виконуються), ведеться журнал.

8. Імітатор системних повідомлень: додавання повідомлення, категорії, фільтрація.

9. Утиліта «Check-in»: фіксує, хто «увійшов у систему», веде список.

10. Список відкритих портів (імітація): додаються/видаляються номери портів.

11. Імітатор менеджера користувачів: додає логіни, виводить список, очищає.

12. Планувальник завдань: додає завдання з мітками часу.

13. Історія команд із позначенням часу введення.

14. Утиліта для перевірки допустимості імені файлу (без обробки файлів).

15. Список завантажених модулів (вручну додаються «модулі» – назви).

16. Облік помилок системи: вводиться код помилки, пояснення, фіксується.

17. Менеджер «служб»: моделює запуск/зупинку служб (статус ON/OFF).

18. Журнал роботи користувача (імена, дії, час).

19. Менеджер симульованих потоків.

20. Утиліта паролів: додає логін+хеш, не зберігає відкрито паролі.

21. Утиліта контролю пам’яті: імітація розміру використаної пам’яті.

22. Облік підключень до пристроїв (назва пристрою, час підключення).

23. Імітатор мережевих пакетів: додавання/обробка/перегляд.

24. Журнал оновлень програмного забезпечення.

25. Утиліта перевірки цілісності даних (імена «файлів» і хеш-значення).

26. Командна оболонка з обмеженим набором «команд».

27. Менеджер груп користувачів: додавання/виведення учасників.

28. Утиліта планування запуску задач: зберігає назви, час запуску.

29. Список «гарячих клавіш» (Ctrl+X, Alt+F4 тощо), з описом.

30. Симулятор логу системних подій: додаються події, сортуються.

ЛАБОРАТОРНЕ ЗАНЯТТЯ №21-22
РОЗРОБКА ФАЙЛОВОГО МЕНЕДЖЕРА В СЕРЕДОВИЩІ VISUAL STUDIO

Мета роботи – ознайомитися з базовими можливостями роботи з файловою системою в C++ під Windows; навчитися створювати консольні файлові менеджери з можливістю перегляду, створення, видалення файлів і тек.

Література [1, 2, 3, 4]

Теоретичні відомості

Операційна система Windows надає API та стандартні бібліотеки мови C++ для роботи з файлами, папками та директоріями. Основними задачами системних програм, які працюють із файловою системою, є перевірка наявності файлів і папок, перегляд вмісту директорій, створення, перейменування, видалення файлів і папок.

Читання та запис у файли.

Для реалізації простих файлових операцій у C++ можна використовувати бібліотеки:

<fstream> – для роботи з файлами;

<filesystem> (з C++17) – для роботи з файловою системою (папки, шлях, типи тощо).

Приклад оголошення:

#include <filesystem>

namespace fs = std::filesystem;

Розглянемо приклад реалізації консольного файлового менеджера:

Завдання. Розробити файловий менеджер, де меню повинно містити такі категорії: показати вміст поточної директорії, перейти до іншої директорії, створити файл, створити папку, видалити файл/папку, вихід.

Програмний код реалізації (C++):

```
#include <iostream>
#include <filesystem>
```

```
#include <fstream>
using namespace std;
namespace fs = std::filesystem;

class FileManager {
private:
    fs::path currentPath;

public:
    FileManager() {
        currentPath = fs::current_path();
    }

    void showDirectoryContents() {
        cout << "Вміст директорії: " << currentPath << endl;
        for (const auto& entry : fs::directory_iterator(currentPath)) {
            cout << (fs::is_directory(entry) ? "[DIR]  " : "       ");
            cout << entry.path().filename().string() << endl;
        }
    }

    void changeDirectory(const string& dir) {
        fs::path newPath = currentPath / dir;
        if (fs::exists(newPath) && fs::is_directory(newPath)) {
            currentPath = fs::canonical(newPath);
            cout << "Перехід до: " << currentPath << endl;
        } else {
            cout << "Невірна директорія." << endl;
        }
    }

    void createFile(const string& filename) {
        ofstream file(currentPath / filename);
        if (file) cout << "Файл створено: " << filename << endl;
        else cout << "Помилка створення файлу." << endl;
    }

    void createDirectory(const string& dirname) {
        if (fs::create_directory(currentPath / dirname))
            cout << "Папка створена: " << dirname << endl;
        else
            cout << "Помилка створення папки." << endl;
    }

    void deleteEntry(const string& name) {
        fs::path target = currentPath / name;
        if (fs::exists(target)) {
```

```
            fs::remove_all(target);
            cout << "Елемент видалено: " << name << endl;
        } else {
            cout << "Елемент не знайдено." << endl;
        }
    }
};

int main() {
    FileManager manager;
    int choice;
    string name;

    do {
        cout << "\n1 – Показати вміст\n2 – Перейти до папки\n3 – Створити файл\n4 – Створити папку\n5 – Видалити\n0 – Вихід\nВаш вибір: ";
        cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1:
                manager.showDirectoryContents();
                break;
            case 2:
                cout << "Назва папки: ";
                getline(cin, name);
                manager.changeDirectory(name);
                break;
            case 3:
                cout << "Назва файлу: ";
                getline(cin, name);
                manager.createFile(name);
                break;
            case 4:
                cout << "Назва папки: ";
                getline(cin, name);
                manager.createDirectory(name);
                break;
            case 5:
                cout << "Назва елемента для видалення: ";
                getline(cin, name);
                manager.deleteEntry(name);
                break;
        }
    } while (choice != 0);

    return 0;
```

}

Результат виконання:
1 – Показати вміст
2 – Перейти до папки
3 – Створити файл
4 – Створити папку
5 – Видалити
0 – Вихід
Ваш вибір: 1
[DIR] Documents
      notes.txt

Індивідуальні завдання до лабораторного заняття №21-22

Реалізуйте власну консольну файлову утиліту файлового менеджера, яка виконує хоча б 4 файлові операції (створення, читання, запис, перейменування, видалення, навігація), має меню взаємодії та використовує класи.

Підкресліть індивідуальність розробки (зазначте її при захисті лабораторної роботи) – навігація стрілками, динамічне меню, історія навігації (можливість вернутись назад в каталог за стрілкою), вбудований пошук файлів тощо.

Лабораторна робота оцінюється, по перше, з урахуванням завершеності завдання, та по друге, індивідуальної складності розробки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Системне програмне забезпечення : конспект лекцій для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології» спец. 123 Комп'ютерна інженерія денної та заоч. форм навч. / уклад. Н. А. Христинець. Луцьк : ЛНТУ, 2024. 116 с.

2. Христинець Н.А. Системне програмне забезпечення: методичні вказівки до самостійної роботи для здобувачів першого (бакалаврського) рівня освіти освітньої програми «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології» спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання. Луцьк : ЛНТУ, 2023. 52 с.

3. Khrystynets N., Melnyk K., Lavrenchuk S., Miskevych O., Kostiuchko S. Multiprocessing as a Way to Optimize Queries. Advances in Transdisciplinary Engineering, 2024. №48. pp. 455–464. URL: https://doi.org/10.3233/ATDE231357 (дата звернення: 15.04.2025)

4. Дячук С. Ф. Windows 2010: навчальний посібник. Тернопіль : ТНТУ імені Івана Пулюя, 2021. 144 с. URL: http://surl.li/untbx (дата звернення: 1.05.2025).

5. PowerShell Documentation. Official product documentation for PowerShell. Microsoft Corporation. URL: https://learn.microsoft.com/en- us/powershell/ (Last accessed: 2.05.2025).

6. Mwiinga P. Operating System. Eden university : Real Deal, 2023. 39p. URL:https://www.researchgate.net/publication/372132620_operating_system (Last accessed: 2.05.2025).

7. Черевик В.М., Танцюра Л.І., Коротков С.С., Сосновий В.О. Операційна ситема Linux: принципи роботи з файловою системою. Київ : ДУТ, 2021. 147 с. URL: https://duikt.edu.ua/uploads/l_2226_57735395.pdf (дата звернення: 2.05.2025).

8. Посібник Ubuntu. URL: https://ubunlog.com/uk/керівництво-ubuntu/ (дата звернення: 2.05.2025).

С-94 **Системне програмне забезпечення.** Методичні вказівки до лабораторних занять для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Комп'ютерна інженерія» галузь знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія денної та заочної форм навчання / уклад. О.І. Міскевич. Луцьк : ЛНТУ, 2025. 139 с.

Комп’ютерний набір: О. І. Міскевич

Редактор: О. І. Міскевич

Підп. до друку «___» _____ 2025р.
Формат 60х84/16. Папір офс. Гарнітура Таймс.
Ум. друк. арк. ____. Тираж 10 прим. Зам. ____

Відділ іміджу та промоції
Луцького національного технічного університету
43018, м. Луцьк, вул. Львівська, 75
Друк – ВІП ЛНТУ