МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ МЕНЕДЖМЕНТУ

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

ОПЕРАЦІЙНІ СИСТЕМИ ТА СИСТЕМНЕ ПРОГРАМУВАННЯ

методичні рекомендації для практичних занять та самостійної роботи здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» денної форми здобуття вищої освіти

Миколаїв
2025

УДК 004.451
О-60

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету (протокол № 1 від 28 серпня 2025 року)

Укладачі:

С. І. Тищенко – к.п.н., доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. Ю. Пархоменко – к.ф.-м.н., доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

С. І. Ємельянов – PhD, старший викладач кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

Т. С.Кучмійова – к.е.н., доцент, доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. О. Жебко – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

Ю.В. Волосюк - к.т.н., доцент, заступник сільського голови з питань відбудови та цифрової трансформації Шевченківської територіальної громади Миколаївської області;

А. М.Коломієць – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету

Рецензенти

Р. В. Кураченко – Head o Production / Learning Experience at Interaction Design Foundation (IxDF)

О. С. Садовий – канд. техн. наук, доцент, завідувач кафедри агроінженерії Миколаївського національного аграрного університету

ЗМІСТ

ВСТУП

Без операційних систем неможливо ефективно вирішувати багато практичних завдань у різних сферах діяльності людини. Вони забезпечують управління апаратними ресурсами, виконання програм, організацію зберігання та обміну даними, що є необхідною основою для сучасних інформаційних технологій. Водночас системне програмування дозволяє розробляти програмне забезпечення, яке працює безпосередньо з операційною системою та апаратним забезпеченням, забезпечуючи високий рівень контролю, продуктивності та надійності.

Дисципліна «Операційні системи та системне програмування» вивчається здобувачами вищої освіти спеціальності 122 «Комп'ютерні науки» на другому курсі і є обов'язковою компонентою освітньо-професійної програми «Комп'ютерні науки».

Мета дисципліни: формування у студентів фундаментальних знань про архітектуру, функції та принципи роботи операційних систем, а також розвиток навичок системного програмування для ефективної взаємодії з ОС та управління ресурсами комп'ютерної системи.

Завдання дисципліни: є ознайомлення студентів із основами архітектури операційних систем, вивчення їхніх функціональних компонентів та принципів взаємодії між користувачем, прикладними програмами й системними ресурсами. Дисципліна спрямована на формування вміння аналізувати роботу операційних систем, вибирати оптимальні методи управління ресурсами комп'ютера, застосовувати засоби системного програмування та використовувати їх для розробки прикладних рішень. Особливу увагу приділено проведенню практичних робіт, що дозволяють відпрацювати навички взаємодії з ядром ОС, дотриманню правил побудови ефективних і безпечних системних процесів та використанню інструментів для діагностики й оптимізації роботи операційної системи.

Предмет дисципліни: архітектура, функціональні механізми та принципи роботи операційних систем, а також методи й засоби системного програмування, що забезпечують взаємодію програмного забезпечення з апаратними та програмними ресурсами комп'ютерної системи.

Під час вивчення дисципліни «Операційні системи та системне програмування» студенти не лише опановують теоретичні основи архітектури сучасних операційних систем і принципи системного програмування, а й активно розвиватимуть практичні навички роботи з різними операційними системами (Windows, Linux, macOS та спеціалізованими платформами), включаючи їх встановлення, глибоке налаштування, адміністрування, управління процесами та

ресурсами, моніторинг, а також діагностику й усунення несправностей. Такий комплексний підхід забезпечить формування високого рівня професійної компетентності майбутніх фахівців у галузі інформаційних технологій, що дозволить їм ефективно проектувати, впроваджувати та супроводжувати складні програмно-апаратні системи в реальних виробничих умовах.

ЗМІСТОВИЙ МОДУЛЬ 1. ОСНОВИ ОПЕРАЦІЙНИХ СИСТЕМ ТА ЇХ АРХІТЕКТУРА

Лабораторна робота №1-4

Основи операційних систем та їх архітектура

Мета роботи – одержання практичних навичок розбиття дисків на розділи, їх форматування, створення дерева директорій, об'єктів файлової системи (ФС) в оболонці bash ОС Linux.

- Робота виконується: на основній або віртуальній машині.
- Вивчаються команди bash: fdisk, mkfs, ls, tree, lsblk, ln.

Теоретичні відомості

Оболонка bash та вхід користувача у систему

Оболонка bash призначена для роботи в командному рядку. Сама по собі bash не виконує ніяких прикладних задач. Її функції полягають в наступному:

- забезпечує виконання всіх додатків: пошук виконуваних програм, їх запуск та організацію введення/виведення;
- відповідає за роботу зі змінними оточення і виконує деякі перетворення (підстановки) аргументів;
- включає в себе просту мову програмування, що робить її потужним інструментом користувача.

Комбінації клавіш bash:

- Ctrl +Alt+T – виклик Терміналу;
- Ctrl +Alt+Fn – перехід на n віртуальну консоль (n=1..6);
- Ctrl +Alt+F7 – повернення до графічного режиму;
- Ctrl + L – очистити екран і помістити поточну команду у верхньому рядку екрана;
- Shift+PgUp і Shift+PgDown – переглянути кілька сторінок екранного виведення;
- Ctrl + C – перервати виконання запущеної команди;
- Ctrl + D – вихід з оболонки bash;
- Tab – автодоповнення.

Довідка

Для отримання довідки з команд bash можна використати вбудовану довідкову систему:

- команда – **help**;
- **man** команда;
- **info** команда.

Запрошення

Вхід у систему починається із запиту імені і пароля користувача, так як тільки після авторизації в системі можна виконувати будь-які дії:

login: ім’я_користувача

password: пароль_користувача

У тому випадку, коли користувачеві дозволено увійти в систему, на екрані терміналу з'явиться запрошення, наприклад таке:

[my@dog my] $

У квадратних дужках дається підказка користувачеві:

- до символу "@" вказується ім'я користувача,
- після – ім'я комп'ютера;
- ім'я директорії, в якій знаходиться у даний момент користувач (шлях до директорії не виводиться);
- символ "$" говорить про те, що в систему увійшов звичайний користувач.

Щоб відрізняти звичайного користувача від адміністратора, запрошення на екрані виглядає трохи по-іншому:

[root@dog root] #

Замість символу "$" використовується символ "#". Інформація у квадратних дужках формується так само, але в якості домашньої директорії для адміністратора (root) виступає директорія /root.

Команди для отримання тимчасових прав root наведено у таблиці 1.

Таблиця 1. Команди для тимчасового отримання прав адміністратора

su	заміна поточного користувача на вказаного
sudo	тимчасове отримання прав root

Ярлик файлу

Кожен файл і директорія в операційній системі Linux має ярлик (етикетку, атрибути, attributes), який описує вказаний файл:

-rw-r-r-- 1 user users 6043 2008-01-11 1:28 myfile

Запис про файл має 8 полів:

- перше поле складається із десяти символів, перший символ визначає тип файлу, другий-десятий символи визначають права доступу до файлу;
- друге поле вказує на кількість жорстких зв’язків файлу;
- третє та четверте поля вказують на власника файлу та його групу;
- п’яте поле містить розмір файлу у байтах;
- шосте та сьоме поля показують дату останнього доступу до файлу, час модифікації і час зміни атрибутів відповідно;
- останнє восьме поле містить назву файлу. Для перегляду ярликів файлів використовується команда ls. Основні команди для роботи із об’єктами ФС

наведено у таблиці 2.

Таблиця 2. Основні команди для роботи із об’єктами ФС

pwd	виводить повний шлях від кореневої директорії до поточної робочої директорії
ls	виводить список файлів
file	виводить тип даних, які розміщено у файлі
mkdir	створити директорію
mv	перейменувати (перемістити) файл
rmdir	видалити директорію
rm	видалити файл
cd	перехід у директорію
touch	створення порожнього файлу
cp	копіювання файлу
echo	виведення тексту або значення змінних на екран
cat	виведення на екран вмісту файлу
tree	виведення дерева директорій
diff	порівняти файли
grep	шукати у файлах

Типи файлів

У Linux тип файлу не визначається за його розширенням, як у Windows. Хоча маркування файлів за допомогою крапки і трьох букв допускається і використовується у Linux, особливо для стандартних файлів, що створюються різними додатками, але тільки для зручності користувачів. Наприклад, для файлів малюнків у Linux використовують маркування типу: jpg, gif, tiff, wmf, а для текстових файлів широко використовується мітка txt.

Класифікація файлів у Linux:

- – звичайні файли або рядові файли (regular). Такі файли можуть містити різні дані аналогічно файлам в MS-DOS, в тому числі і двійковий код програм.

d – директорії (directory). Ці файли призначені для зберігання списків файлів і піддиректорій.

b, c – файли пристроїв (devices). Імена таких файлів можна знайти в директорії /dev, вони складаються, наприклад, з двох номерів: номери класу пристрою і порядкового номеру в даному класі. Напряму з цими файлами користувачі справи не мають, хіба що, коли доводиться у налаштуваннях пристроїв обирати номер порту.

s, p – спеціальні файли: сокети (sockets) та іменовані канали (named pipes). Такі файли призначені для обміну даними між процесами.

l – символічні посилання (symlinks). Основне призначення символічних

посилань в тому, щоб не копіювати кожен раз файл в ту чи іншу директорію, а вказувати її «адресу» у файлі символічним посиланням.

Крім перерахованих раніше типів файлів можна побачити файли і директорії , імена яких починаються із символу «крапка». Це законні імена в Linux, такі файли і директорії є прихованими. Так до прихованих файлів потрапляють імена поточної та батьківської директорій (. та .. відповідно). Основні команди для роботи із директоріями та файлами наведено у таблицях 3 та 4 відповідно.

Таблиця 3. Основні команди для роботи з директоріями

ls [параметри] [директорія / файл]	Виведення списку файлів.
ls	виводить вміст поточної директорії
ls /usr/local /etc/default	виводить вміст директорій /usr/local і /etc/default.
ls -l /usr/local	виводить детальну інформацію про вміст директорії /usr/local.
ls -F	Позначає в списку - виконувані файли зірочкою "*", - директорії – похилій рисою "/", - символічні посилання – символом "@"
ls n*	пошук усіх файлів, що починаються на "n"
ls *n	пошук усіх файлів, що закінчуються на "n"
ls -a	виводить приховані файли
mkdir mydir	створити директорію
mkdir -p mydir1 /mydir2	створити цілий ланцюжок директорій
mv oldDir newDir	перейменувати директорію oldDir у newDir
rmdir директорія	видалення непотрібної директорії
rmdir –rf директорія	рекурсивне видалення вмісту директорії без підтвердження
cd mydir	переміщення у директорію
cd, cd ~	перехід у домашню директорію користувача
cd /	перехід у кореневу директорію

Таблиця 4. Основні команди для роботи із файлами

touch myfile	створення порожнього файлу
rm myfile	видалення файлу слід користуватися акуратно, тому що в ряді випадків може не послідувати питання системи про підтвердження користувачем згоди на вилучення.
mv oldName newName	перейменовує файл oldName у newName
mv file1 file2 fileN mydir	переміщує зазначені файли file1, file2 і fileN у директорію mydir
cp myfile yourfile	копіює myfile в yourfile
cp file1 file2 fileN mydir	копіює вказані файли file1, file2 і fileN у директорію mydir
diff myfile1 myfile2	порівняти файли
grep root /etc/passwd	пошук у /etc/passwd рядка root

Права доступу до файлів

Права доступу до файлів використовуються ядром Linux, яке за ними визначає, що може робити з даним файлом конкретний користувач: читати, змінювати або запускати файл (програму) на виконання. У Linux права доступу до файлу визначені для трьох рівнів користувачів:

- для власника файлу, який створив цей файл (user);
- для користувачів, що входять до групи, наприклад, група співробітників, що працює над одним проектом (group);
- для всіх інших користувачів (others).

І тільки адміністратор (root) має право проводити будь-які операції з файлами. Користувач для входу в систему реєструється, вводячи свій пароль та ім'я. Після цього він отримує доступ до ФС. Для прискорення швидкодії системи використовуються унікальні ідентифікатори користувача UID і групи GID. Відповідно у атрибутів файлів є шаблон, який визначає приналежність файлу конкретному користувачеві та групі. Якщо користувач не є власником файлу і не входить до складу групи, то він відноситься до інших (others), які мають дуже мало прав, а то й зовсім ними не володіють.

Враховуючи такий принцип поділу користувачів, для опису прав доступу до файлів використовується шаблон: user group others. У багатьох випадках для полегшення розуміння того, як розподіляються права, використовується запис атрибутів файлу у вигляді символів «x», «w» і «r» або «1», «2», «4» відповідно:

- право виконання (--x), скор. від execute – виконувати;
- право редагування (-w-), скор. від write – писати;
- право читання (r--), скор. від read – читати.

Зауважимо, що для запуску виконуваного файлу користувач повинен мати

право на його виконання, а для виконання скрипта – на читання та виконання. Все, що було сказано про атрибути файлів, стосується і директорій, але зміст прав дещо інший:

- якщо у користувача є право на читання (r) директорії, то це означає, що в даній директорії він може вивести на екран зміст цієї директорії;
- щоб записувати і видаляти в директорії файли, користувач повинен мати право запису (w);
- право на виконання (x) для директорії означає, що користувач може увійти в директорію і переглянути її вміст.

Зауважимо, що для того, щоб писати та читати із директорії, потрібно обов'язково мати право на її виконання. Тому парні права на директорії обов'язково повинні бути непарними. Для зміни прав доступу до об'єктів ФС використовується команда chmod. Право змінювати дозволи має лише власник об'єкту та адміністратор.

В табл.5 наведено основні команди для роботи із вмістом файлів.

Таблиця 5. Основні команди для роботи із вмістом файлів

echo	виведення тексту або значення змінних на екран
echo> myfile "текст"	перенаправлення виведення за допомогою символу ">": коли файл із таким ім'ям вже існує, то він буде перезаписаний.
echo>>test "текст додається"	перенаправлення виведення за допомогою символу ">>": новий текст буде додано в кінець файлу
echo "текст на принтер" | lp	передача стандартного виведення однієї команди на стандартний вхід іншої за допомогою символу "|" : текст треба роздрукувати на принтері
cat myfile	виведення інформації з існуючого файлу
cat> myfile	введення тексту з клавіатури Для завершення введення тексту треба натиснути комбінацію клавіш Ctrl + D
cat newfile >> myfile	дописати зміст файлу newfile у файл myfile

Монтування файлових систем (пристроїв)

Монтування – підключення дочірньої файлової системи (пристрою) до єдиного дерева батьківської файлової системи в певній логічній точці (точці монтування).

Із загальної точки зору будь-який пристрій, наприклад, дисковий

накопичувач: перш, ніж до нього можна звернутися (читати або записати дані), має бути змонтовано в будь-яку точку файлової системи. Причому точка монтування може бути довільною. Для забезпечення сумісності різних дистрибутивів Linux один з одним і для спрощення взаємодії користувачів у файловій структурі завжди існує директорія /mnt.

У цій директорії за традицією монтуються всі зовнішні накопичувачі у

вигляді окремих директорій. Тут же намагаються монтувати і розділи Windows, хоча це необов'язково, наприклад, диск C: – /mnt/windows.

За винятком ФС, для яких є спеціальні настройки в /etc/fstab, монтування (демонтування) здійснює тільки root.

Імена зовнішніх пристроїв

Так як будь-який зовнішній пристрій ототожнюється в Linux з файлом, то в імені цього файлу є ланцюжок символів, що характеризує його особливості. Причому прийнято включати в ім'я пристрою назву директорії /dev.

Імена вінчестерів пов'язані з інтерфейсом, що використовується та типом пристроїв. Якщо диск має стиль MBR, то використовується така нумерація: 1,2,3,4 – для первинних розділів, 5 і далі – ... для логічних розділів.

Тобто, /dev/sda1 – первинний розділ, /dev/sda5 – перший логічний розділ на першому диску. Якщо диск має стиль GPT, всі розділи якого є логічними, то використовується наскрізна нумерація розділів. Основні команди для роботи із розділами дисків наведено у табл.6.

Таблиця 6. Основні команди для роботи із розділами дисків

fdisk <ім'я пристрою>	редагувати таблицю розділів
	Опції:
	`Command (m for help): m`
	`Command action`
	`a toggle a bootable flag`
	`b edit bsd disklabel`
	`c toggle the dos compatibilit`
	`d delete a partition`
	`l list known partition types`
	`m print this menu`
	`n add a new partition`
	`o create a new empty DOS part`
	`p print the partition table`
	`q quit without saving changes`
	`s create a new empty Sun disk`
	`t change a partition's system`
	`u change display/entry units`
	`v verify the partition table`

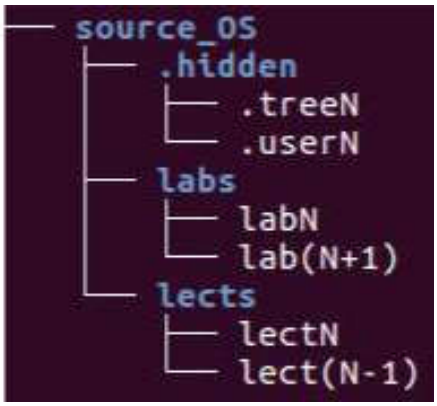
mkfs <розділ>	створити файлову систему на розділі -t – вказати тип ФС -c – перевіряти на збійні блоки -j – створювати ext3 -b – задати розмір блоку (1024/2048/4096) байтів
mount <опції> <пристрій> <точка монтування>	монтування
umount <пристрій>	демонтування
lsblk -f	переглянути список монтованих ФС
blkid	отримати список UUID розділів
df <файл пристрою>	отримати інформацію про вільний простір опції:

	-h – у людському форматі -i – скільки використано індексних дескрипторів
du <директорія>	отримати інформацію про використаний простір у директорії Опції: -h – в людському форматі -s – без інформації про піддиректорії

Завдання на лабораторну роботу

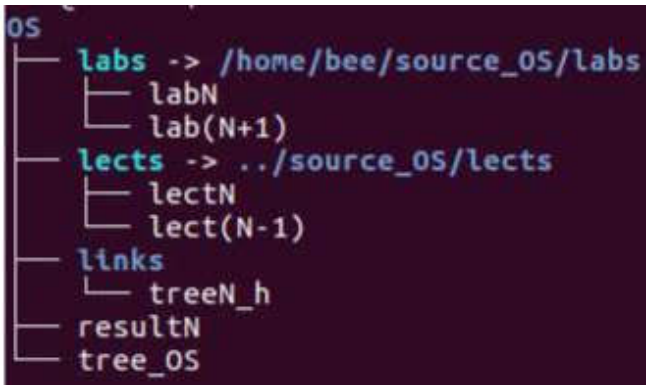
1. Створення об’єктів ФС

1.1. У домашній директорії користувача створити директорію source_OS із відповідним вмістом та правами доступу 755 (rwxrwxr-x) для директорій та 664 для файлів (rw-rw-r--), де N – номер варіанта.



Файли labN, lab(N+1) наповнити рядками labN, lab(N+1) відповідно; .userN – іменем поточного користувача, .treeN – деревом source_OS із відображенням прихованих файлів.

1.2. У домашній директорії користувача створити директорію OS із відповідним вмістом:



Файли labs, lects – символічні посилання на відповідні директорії; treeN_h – жорстке посилання на файл source_OS/.hiddeen/.treeN, tree_OS містить дерево директорії, файл resultN містить результати виконання лабораторної роботи.

2. Створення, форматування та монтування розділів диску.

3. Переміщення об’єктів ФС на монтовані розділи та перевірка

властивостей зв’язків.

4. Підготовка файлів звіту resultN.txt, lsN.txt

Послідовність виконання завдання на прикладі Ubuntu 16.04.2

1. Створення об’єктів ФС

Від імені користувача:

1.1. У домашній директорії користувача створюємо директорію source_OS та наповнюємо її.

source_OS
├———.hidden
├——— labs
└——— lects

1.1.0. Встановлюємо права доступу на створені файли та директорії 002:

umask 002

Ця маска передбачає права доступу на створені файли та директорії 775 (rwxrwxr-x).

1.1.1. Створюємо дерево директорій source_OS:

```
bee@home:~$     mkdir source_OS
bee@home:~$     cd source_OS
bee@home:~/source_OS$     mkdir lects labs .hidden
```

Можна скористатися опцією -p команди mkdir і створити дерево, виконавши лише одну команду.

Перевірка:

```
bee@home:~/source_OS$ ls -l
total 8
drwxrwxr-x 2 bee bee 4096 лис 29 12:08 labs
drwxrwxr-x 2 bee bee 4096 лис 29 12:08 lects
```

Пояснення:

- перший файл labs є директорією;
- права доступу labs 775 – всі дозволи для власника, всі дозволи для групи власника та інші користувачі мають право на читання і вхід у директорію;
- кількість жорстких посилань на директорію – 2. Коли створюється нова порожня директорія, для неї створюється 2 жорсткі посилання: одне ім’я міститься у батьківській директорії, інше – в самій створеній директорії («.»). Якщо директорія містить піддиректорії, кількість її жорстких зв’язків збільшується на 1 (запис про неї «..» міститься у дочірній піддиректорії). Жорсткими посиланнями на директорії оперує виключно ОС;
- власник директорії (bee);
- група власника (bee);
- розмір директорії (4096b);
- дата та час створення директорії;

➢ назва директорії (labs).

Щоб переглянути приховані файли, використовуємо опцію –a:

```
bee@home:~/source_OS$ ls -la
total 20
drwxrwxr-x  5 bee bee 4096 лис 29 12:08 .
drwxr-xr-x 19 bee bee 4096 лис 29 12:08 ..
drwxrwxr-x  2 bee bee 4096 лис 29 12:08 .hidden
drwxrwxr-x  2 bee bee 4096 лис 29 12:08 labs
drwxrwxr-x  2 bee bee 4096 лис 29 12:08 lects
```

1.1.2. У директорії labs створюємо файли labN і lab(N+1), де N – номер варіанта за списком, записуючи в кожен файл рядок із назвою файлу (lab N, lab(N+1) відповідно):

```
bee@home:~/source_OS$ cd labs
```

```
bee@home:~/source_OS/labs$ echo labN > labN
bee@home:~/source_OS/labs$ echo 'lab(N+1)' > lab\(N+1\)
```

Перевірка:

```
bee@home:~/source_OS/labs$ cat labN
labN
bee@home:~/source_OS/labs$ cat lab\(N+1\)
lab(N+1)
```

1.1.3. Знаходячись у директорії labs, створюємо в lects файли lectN, lect(N-1), використовуючи відносні шляхи:

```
bee@home:~/source_OS/labs$ touch ../lects/lectN ../lects/lect\(N-1\)
```

Перевірка:

```
bee@home:~/source_OS/labs$ ls -l ../lects
total 0
-rwxrwxrwx 1 bee bee 0 тра 11 12:15 lectN
-rwxrwxrwx 1 bee bee 0 тра 11 12:15 lect(N-1)
```

1.1.4. У директорії .hidden створюємо файли .userN, .treeN, записуючи в файл логін користувача (для .userN) та дерево директорій source_OS (для .treeN).

Щоб переглянути дерево директорій, потрібно встановити утиліту tree:

 # apt-get install tree.

```
bee@home:~/source_OS/.hidden$ whoami > .userN
bee@home:~/source_OS/.hidden$ tree -a /home/bee/source_OS > .treeN
```

Перевірка:

```
bee@home:~/source_OS/.hidden$ ls -la
total 16
drwxr-xr-x 2 bee bee 4096 бер 26 15:35 .
drwxr-xr-x 5 bee bee 4096 бер 26 16:36 ..
-rw-rw-r-- 1 bee bee  237 бер 26 16:43 .treeN
-rw-rw-r-- 1 bee bee    4 бер 26 16:43 .userN
bee@home:~/source_OS/.hidden$ cat .userN
bee
bee@home:~/source_OS/.hidden$ cat .treeN
/home/bee/source_OS
├── .hidden
│   ├── .treeN
│   └── .userN
├── labs
│   ├── labN
│   └── lab(N+1)
└── lects
    ├── lectN
    └── lect(N-1)
```

1.2. У домашній директорії користувача створюємо директорію OS та наповнюємо її:

```
OS
├── labs -> /home/bee/source_OS/labs
│   ├── labN
│   └── lab(N+1)
├── lects -> ../source_OS/lects
│   ├── lectN
│   └── lect(N-1)
├── links
│   └── treeN_h
├── resultN
└── tree_OS
```

1.2.1. Створюємо директорію OS, links:

```
bee@home:~$ mkdir OS OS/links
```

Перевірка:

```
bee@home:~$ tree OS
OS
└── links
```

1.2.2. В OS створюємо символічні посилання на директорію /home/bee/source_OS/labs (за повним шляхом), /home/bee/source_OS/lects (із використанням відносного шляху):

```
bee@home:~/OS$ ln -s /home/bee/source_OS/labs labs
bee@home:~/OS$ ln -s ../source_OS/lects lects
```

Зверніть увагу, що розмір файлу-посилання – довжина шляху до відповідного об’єкта.

```
bee@home:~/OS$ ls -l
total 4
lrwxrwxrwx 1 bee bee   24 лис 29 12:29 labs -> /home/bee/source_OS/labs
lrwxrwxrwx 1 bee bee   18 лис 29 12:29 lects -> ../source_OS/lects
drwxrwxr-x 2 bee bee 4096 лис 29 12:27 links
```

1.2.3. У файл результатів resultN розмістимо ім’я користувача та номер варіанта:

```
bee@home:~/OS$ cat ../source_OS/.hidden/.userN > resultN
bee@home:~/OS$ echo 'N' >> resultN
```

Перевірка:

```
bee@home:~/OS$ cat resultN
bee
N
```

1.2.4. Додамо у файл результатів resultN зміст попередньо створених файлів labN, lab(N+1), скориставшись символічним зв’язком на файл та іменем файлу.

Спершу додамо 2 розділювача рядків (важливо для автоматичної перевірки лабораторної роботи):

```
bee@home:~$ echo -e >> resultN && echo -e >> resultN
bee@home:~/OS$ cat labs/labN >> resultN
bee@home:~/OS$ cat ../source_OS/labs/lab\(N+1\) >> resultN
```

1.2.5. У links створюємо файл treeN_h, який є жорстким посиланням на файл treeN:

```
bee@home:~$ ln source_OS/.hidden/.treeN OS/links/treeN_h
```

Перевірка:

```
bee@home:~$ cat OS/links/treeN_h
/home/bee/source_OS
├── .hidden
│   ├── .treeN
│   └── .userN
├── labs
│   ├── labN
│   └── lab(N+1)
└── lects
    ├── lectN
    └── lect(N-1)
```

1.2.6. Перевіримо, що файли пов’язані жорстким зв’язком, вказують на одній й ті ж дані. Побачити це можна, порівнявши номери індексних дескрипторів цих файлів. Для цього проведемо пошук у рекурсивному виведенні вмісту директорій OS, source_OS (повне виведення, перегляд прихованих файлів, виведення індексного дескриптору):

```
bee@home:~$ ls -laiR OS source_OS | grep treeN
283485 -rw-rw-r-- 2 bee bee  237 бер 26 16:43 treeN_h
283485 -rw-rw-r-- 2 bee bee  237 бер 26 16:43 .treeN
```

Допишемо цей результат у файл resultN, спершу додавши 2 розділювачі рядків:

```
bee@home:~$ echo -e >> resultN && echo -e >> resultN
bee@home:~$ ls -laiR OS source_OS | grep treeN >> OS/resultN
```

1.2.7. У директорії OS створюємо файл tree_OS, який містить дерево директорій OS (опція –l в утиліті tree дозволяє відображати вміст директорії, на яку вказує символічне посилання):

```
bee@home:~$ tree -l OS > OS/tree_OS
```

Перевірка:

```
OS
├── labs -> /home/bee/source_OS/labs
│   ├── labN
│   └── lab(N+1)
├── lects -> ../source_OS/lects
│   ├── lectN
│   └── lect(N-1)
├── links
│   └── treeN_h
├── resultN
└── tree_OS
```

1.3. Створення директорій – точок монтування.

Створюємо директорії – точки майбутнього монтування в домашній директорії користувача:

```
bee@home:~$ mkdir /home/bee/mount/
bee@home:~$ mkdir /home/bee/mount/NTFS
bee@home:~$ mkdir /home/bee/mount/EXT4
```

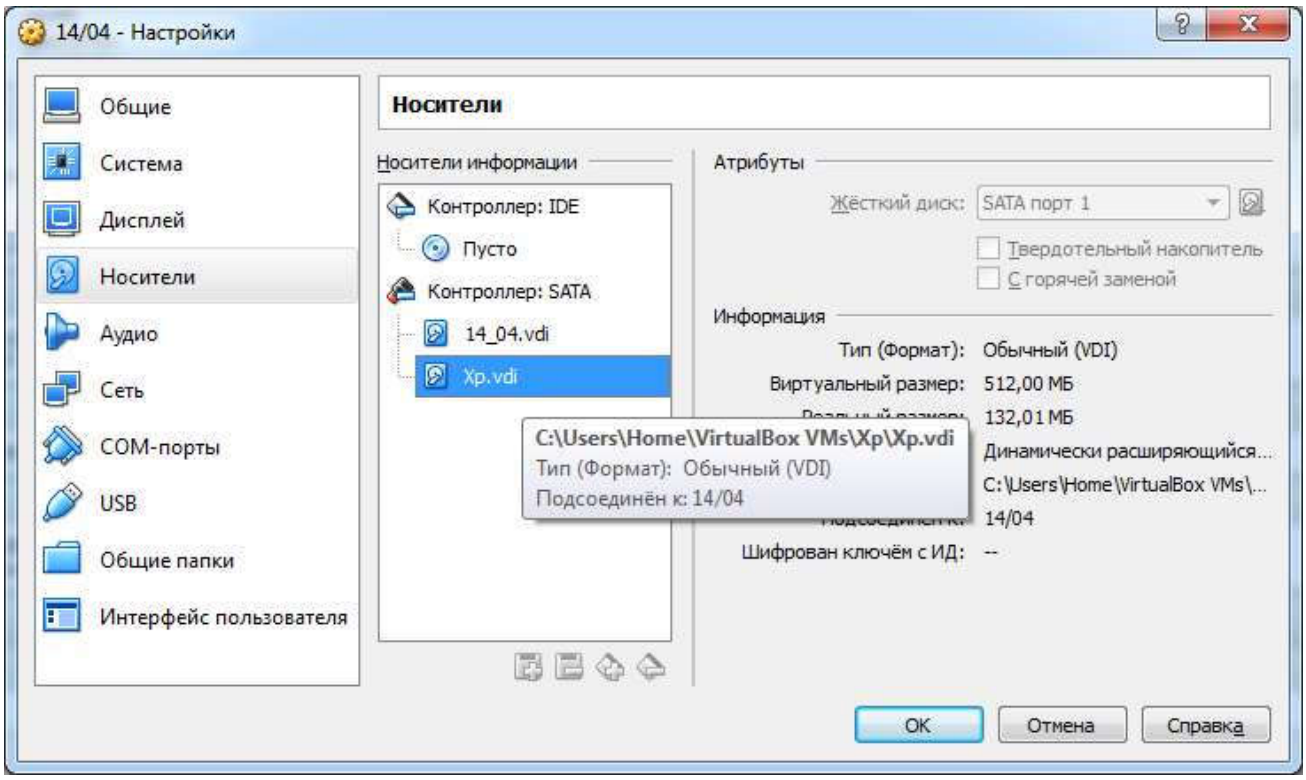
Перевірка:

```
bee@home:~$ tree mount
mount
├── EXT4
└── NTFS
```

2. Створення, форматування та монтування розділів диску

1. Перш ніж приступити до роботи з розділами, складіть для себе план дій (прочитайте інструкцію до кінця). 2. Не змінюйте розділи, які використовуються в даний момент. Для створення розділів рекомендується використати:

- віртуальний жорсткий диск (якщо працюємо у віртуальній машині);
- usb.



3. Будьте уважні! Команда fdisk не вносить ніяких змін без Вашого підтвердження, проте багато інших програм можуть вносити зміни негайно.

4. Для використання інструментів роботи з розділами Ви повинні володіти привілеями користувача root.

Від імені root:

2.1. Створення розділів на диску за допомогою *fdisk.*

2.1.1. Будемо використовувати утиліту fdisk. Подивитися список дисків підключених до комп'ютера можна командою:

#fdisk –l

```
Disk /dev/sda: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0002fa22

Device     Boot    Start      End  Sectors  Size Id Type
/dev/sda1  *        2048 15204351 15202304  7,3G 83 Linux
/dev/sda2       15206398 16775167  1568770  766M  5 Extended
/dev/sda5       15206400 16775167  1568768  766M 82 Linux swap / Solaris


Disk /dev/sdb: 512 MiB, 536870912 bytes, 1048576 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

Пояснення:

У прикладі маємо 2 диска /dev/sda, /dev/sdb.

Перший диск /dev/sda має ємність 8 Гб, 16777216 секторів; розмір сектору

– 512 б; це диск із MBR (disk label type=dos); ідентифікатор диску 0x0002fa22.

Він має 1 первинний (основний, /dev/sda1) та 1 розширений (/dev/sda2) розділ, який в свою чергу складається із 1 логічного диску (/dev/sda5).

Основний розділ позначений як завантажувальний (або активний). Це дозволяє завантажувати комп'ютер із зазначеного розділу за допомогою стандартного MBR-запису DOS. Цей прапор не впливає на роботу завантажувачів LILO і GRUB.

Стовпці Start і End показують початковий і кінцевий сектори кожного розділу. Ці значення не повинні перекриватися і в загальному випадку повинні бути послідовними, без проміжків.

Стовчик Sectors показує кількість секторів розміром у 512 байтів, що містяться в розділі. Для більшості дисків розмір сектора дорівнює 512 байт.

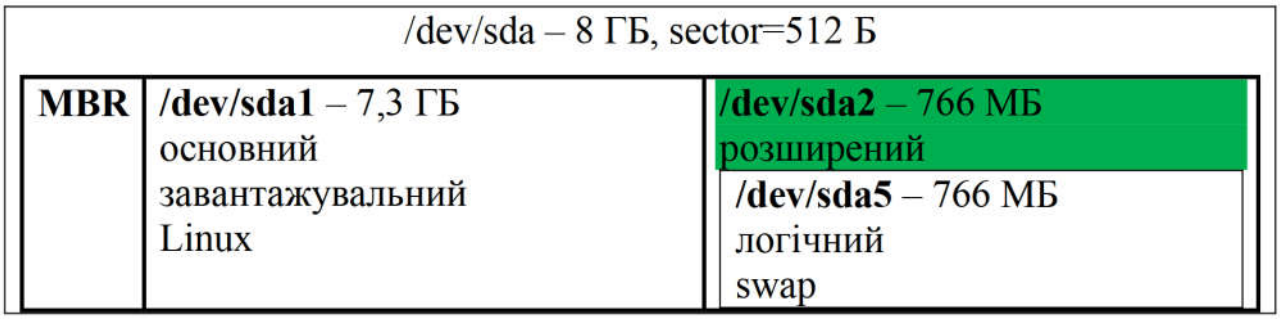
Максимальна ємність розділу дорівнює числу секторів*розмір сектору.

Поле Id визначає призначення розділу. Тип 82 вказує на розділ підкачки Linux, тип 83 – на розділ для зберігання даних, а тип 5 – на розширений

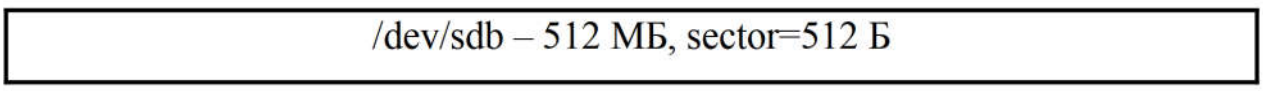
розділ. Існує приблизно 100 різних типів розділів (можна переглянути при виконанні fdisk->l).

Тип розділу можна розглядати як мітку, яка вказує операційній системі на його заплановане призначення.

Структура диску /dev/sda має такий вигляд:



Другий диск /dev/sdb має ємність 512 Мб, на ньому 1048576 секторів по 512 байт. Диск не розмічений (таблиця розділів та самі розділи на ньому не створені).



2.1.2. Створення розділів на sdb. Будемо використовувати стиль розмітки диску GPT.

2.1.2.1. Використовуємо інтерактивну утиліту fdisk:

#fdisk /dev/sdb

Буде запропоновано ввести `m` для довідки, вводимо і дивимося, які є можливості по роботі з диском.

Опції, які стосуються створення стилю розмітки диску:

```
Create a new label
 g   create a new empty GPT partition table
 G   create a new empty SGI (IRIX) partition table
 o   create a new empty DOS partition table
 s   create a new empty Sun partition table
```

Для створення GPT розмітки диску необхідно використати опцію g:

```
Command (m for help): g
Created a new GPT disklabel (GUID: 5C086E26-16E3-472E-9343-BAFD5724DBC0).
```

2.1.2.2. Нам необхідно ввести `n`, тобто додати новий розділ. При цьому нам буде запропоновано ввести номер розділу, перший та останній сектори цього розділу.

Для першого розділу початковий сектор підтверджуємо, а останній вказуємо із розрахунку, що диск ділиться на 2 рівні розділи.

```
Command (m for help): n
Partition number (1-128, default 1): 1
First sector (34-1048542, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-1048542, default 1048542): 524271

Created a new partition 1 of type 'Linux filesystem' and of size 255 MiB.
```

Для другого розділу початковий і останній сектори просто підтверджуємо.

```
Command (m for help): n
Partition number (2-128, default 2): 2
First sector (524272-1048542, default 524288):
Last sector, +sectors or +size{K,M,G,T,P} (524288-1048542, default 1048542):

Created a new partition 2 of type 'Linux filesystem' and of size 256 MiB.
```

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p):
```

Якщо ми хочемо створити диск із MBR, спершу обираємо опцію o, створюємо новий розділ (опція n). При цьому буде необхідно обрати тип розділу, його номер та параметри розмітки (e – розширений розділ, p – основний розділ). Відмітимо, що основних розділів може бути максимально 4, вони нумеруються від 1 до 4, а розширений – лише один (він може створюється замість одного із основних розділів і може мати довільну нумерацію від 1 до 4), логічні розділи всередині розширеного нумеруються починаючи із 5.

2.1.2.3. Якщо Ви хочете щось змінити, можете виконати команду d для видалення одного або декількох розділів, а потім створити їх заново. Якщо конфігурація Вас влаштовує, виконайте команду v для контрольної перевірки конфігурації, а потім команду w для запису змін у таблицю розділів і виходу з програми. Якщо змін зберігати не треба, вказуємо q.

Збережемо результати розбиття диску:

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

2.1.3. Для початку роботи з новими значеннями таблиці розділів перезавантаження комп'ютера не потрібно.

Командою *#fdisk –l /dev/sdb* переглядаємо оновлений список розділів диску sdb:

```
root@home:~# fdisk -l /dev/sdb
Disk /dev/sdb: 512 MiB, 536870912 bytes, 1048576 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: FAD6C2EC-3877-4707-9BBA-DEB50DF5CBCA

Device       Start     End Sectors  Size Type
/dev/sdb1     2048  524271  522224  255M Linux filesystem
/dev/sdb2   524288 1048542  524255  256M Linux filesystem
```

У команді fdisk можна скористатися режимом експерта (fdisk->x) для більш тонкого налаштування параметрів таблиці та розділів (наприклад, змінити послідовність розділів в таблиці розділів).

```
DOS (MBR)
 b   move beginning of data in a partition
 i   change the disk identifier

Geometry
 c   change number of cylinders
 h   change number of heads
 s   change number of sectors/track

Generic
 p   print the partition table
 v   verify the partition table
 d   print the raw data of the first sector from the device
 D   print the raw data of the disklabel from the device
 f   fix partitions order
 m   print this menu
```

2.2. Для створення файлової системи скористаємося командою **mkfs.**

Формат її простий:

#mkfs.файлова система пристрій.

Нам необхідно відформатувати наш /dev/sdb1 в ФС NTFS, dev/sdb2 в ext4. Для цього виконаємо наступні команди:

 # mkfs.ntfs /dev/sdb1

#mkfs.ext4 /dev/sdb2.

Щоб подивитися, які ще файлові системи можна використовувати наберіть mkfs і двічі натисніть на знак табуляції. Отримаєте приблизно наступне:

mkfs (+2 знака TAB)

mkfs mkfs.ext2 mkfs.ext4dev mkfs.vfat

mkfs.bfs mkfs.ext3 mkfs.minix mkfs.xfs

mkfs.cramfs mkfs.ext4 mkfs.msdos

Щоб дізнатися, які файлові системи підтримуються вашою системою, використовуйте команду ls –l /sbin/mk *.

```
bee@home:~$ ls -l /sbin/mk*
lrwxrwxrwx 1 root root      8 тра 26  2016 /sbin/mkdosfs -> mkfs.fat
-rwxr-xr-x 1 root root 106392 жов 30  2015 /sbin/mke2fs
-rwxr-xr-x 1 root root  10592 тра 27  2016 /sbin/mkfs
-rwxr-xr-x 1 root root  27224 тра 27  2016 /sbin/mkfs.bfs
-rwxr-xr-x 1 root root  35456 тра 27  2016 /sbin/mkfs.cramfs
lrwxrwxrwx 1 root root      6 жов 30  2015 /sbin/mkfs.ext2 -> mke2fs
lrwxrwxrwx 1 root root      6 жов 30  2015 /sbin/mkfs.ext3 -> mke2fs
lrwxrwxrwx 1 root root      6 жов 30  2015 /sbin/mkfs.ext4 -> mke2fs
lrwxrwxrwx 1 root root      6 жов 30  2015 /sbin/mkfs.ext4dev -> mke2fs
-rwxr-xr-x 1 root root  27128 тра 26  2016 /sbin/mkfs.fat
-rwxr-xr-x 1 root root  76912 тра 27  2016 /sbin/mkfs.minix
lrwxrwxrwx 1 root root      8 тра 26  2016 /sbin/mkfs.msdos -> mkfs.fat
lrwxrwxrwx 1 root root      6 лют 18  2016 /sbin/mkfs.ntfs -> mkntfs
lrwxrwxrwx 1 root root      8 тра 26  2016 /sbin/mkfs.vfat -> mkfs.fat
-rwxr-xr-x 1 root root  18848 бер 16  2016 /sbin/mkhomedir_helper
-rwxr-xr-x 1 root root  84080 лют 18  2016 /sbin/mkntfs
-rwxr-xr-x 1 root root  72944 тра 27  2016 /sbin/mkswap
```

Як бачимо, деякі утиліти є символічними посиланнями на інші утиліти. Щоб переглянути специфічні параметри для певної утиліти, можна використати довідку:

```
bee@home:~$ mkfs.ntfs --help
```

```
Usage: mkntfs [options] device [number-of-sectors]

Basic options:
    -f, --fast                      Perform a quick format
    -Q, --quick                     Perform a quick format
    -L, --label STRING              Set the volume label
    -C, --enable-compression        Enable compression on the volume
    -I, --no-indexing               Disable indexing on the volume
    -n, --no-action                 Do not write to disk
```

За допомогою опції -L і символьного рядка можна задати мітку для файлової системи. Цю мітку можна використовувати замість імені пристрою при монтуванні файлової системи. При внесенні певних змін, які також необхідно відображати у файлах налаштувань, мітка дозволяє ізолювати ім'я пристрою, якому вона фактично відповідає. Довжина мітки обмежена 16 символами.

На заміну міткам прийшов універсальний унікальний ідентифікатор ФС (Universally Unique Identifier, UUID). UUID – це 128-бітовий ідентифікатор, який зазвичай відображається у вигляді 32 шістнадцяткових цифр, розділених дефісами. Для більшості файлових систем Linux UUID генерується автоматично при їх форматуванні. Для перегляду UUID розділу, який ми тільки що відформатували, використовуйте команду blkid (її можна запустити без прав суперкористувача). Ідентифікатори UUID мають більшу унікальністю в порівнянні з мітками і особливо корисні при використанні пристроїв із гарячою заміною, наприклад, USB- накопичувачів:

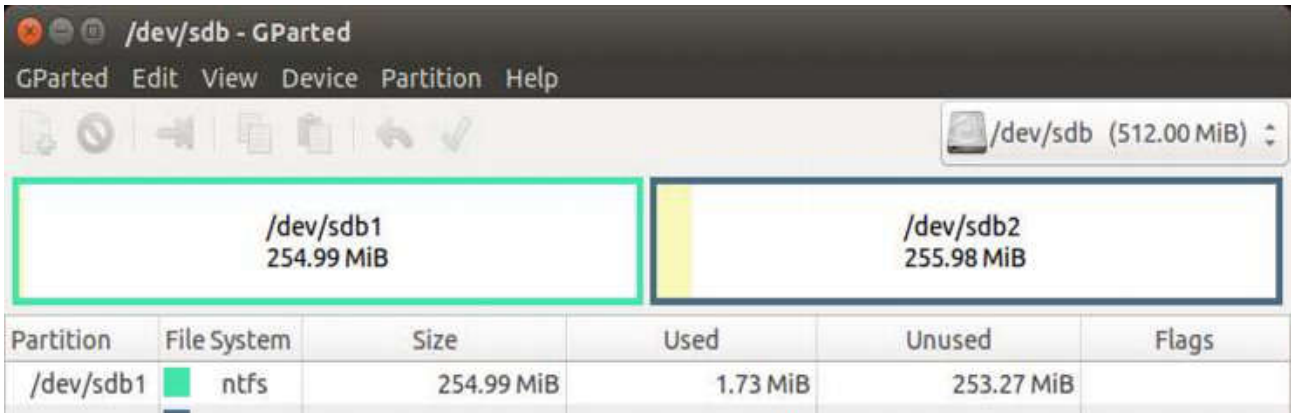
```
root@home:~# blkid
/dev/sda1: UUID="991fdca8-10fb-4f93-9c92-5d7f63d8fdcd" TYPE="ext4" PARTUUID="0002fa22-01"
/dev/sda5: UUID="48549743-e0d6-434c-99ca-42e31bc93c67" TYPE="swap" PARTUUID="0002fa22-05"
/dev/sdb1: UUID="4138DBDE00935D68" TYPE="ntfs" PARTUUID="d4b48880-1956-4652-87d9-f192732932b1"
/dev/sdb2: UUID="b82f744f-7f45-49a5-8782-45f28ecdbe95" TYPE="ext4" PARTUUID="24c90ec2-80d7-4aed-9f86-5f5caed3f8d4"
```

Відмітимо, що і диск, і кожен із розділів має свої унікальні ідентифікатори (GUID та PARTUUID відповідно). GUID надається при виборі gpt стилю розмітки диска, PARTUUID – при створенні розділу, це ідентифікатор розділу, який використовується саме для ідентифікації розділу, наприклад у GPT).

Результат роботи із створення та форматування розділів диску можна переглянути в утиліті GParted, яку необхідно попередньо встановити:

apt-get install gparted.

Ось результат нашого розбиття у GParted:



2.3. Монтування відформатованого розділу командою mount.

Щоб змонтувати розділ у певну директорію в ієрархічній файловій системі, відповідна директорія обов'язково повинна існувати. Відмітимо, що файлові об'єкти, які існують у директорії, при виконанні монтування в неї будуть приховані, а при відмонтовуванні ФС знову стануть доступні.

2.3.1. Виконуємо монтування:

```
root@home:~# mount /dev/sdb1 ~bee/mount/NTFS
root@home:~# mount /dev/sdb2 ~bee/mount/EXT4
```

/dev/sdb1 – власне, сам розділ, який ми підключаємо; /home/bee/mount/NTFS – точка монтування (місце, куди підключається наш розділ);

~bee – домашня директорія відповідного користувача.

2.3.2. Переглянемо список монтованих пристроїв, виконавши команду mount або виводячи файл /etc/mtab:

У результаті серед записів про монтовані пристрої відобразиться запис про щойно монтований розділ:

/dev/sdb1 /home/bee/mount/NTFS fuseblk
rw,relatime,user_id=0,group_id=0,allow_other,blksize=4096 0 0
/dev/sdb2 /home/bee/mount/EXT4 ext4 rw,relatime,data=ordered 0 0

2.3.3. Список монтованих ФС можна переглянути за допомогою команди *lsblk –f:*

```
root@home:~# lsblk -f
NAME   FSTYPE LABEL UUID                                 MOUNTPOINT
sda
├─sda1 ext4         991fdca8-10fb-4f93-9c92-5d7f63d8fdcd /
└─sda5 swap         48549743-e0d6-434c-99ca-42e31bc93c67 [SWAP]
sdb
├─sdb1 ntfs         4138DBDE00935D68                     /home/bee/mount/NTFS
└─sdb2 ext4         b82f744f-7f45-49a5-8782-45f28ecdbe95 /home/bee/mount/EXT4
sr0
```

Допишемо цей результат у файл resultN, спершу додавши 2 розділювача рядків:

```
bee@home:~$ echo -e >> resultN && echo -e >> resultN
bee@home:~$ lsblk -f >> OS/resultN
```

Демонтувати розділ можна командою umount, вказавши відповідний розділ або точку монтування:

#umount /dev/sdb1.

Для автоматичного монтування ФС при завантаженні ОС або для задання правил монтування пристроїв використовується файл /etc/fstab, який потрібно редагувати від імені root. Зауважимо, що в цьому випадку краще використовувати UUID розділу, оскільки імена пристроям надаються при їх підключенні, а UUID змінюється при створенні ФС.

3. Переміщення об’єктів ФС на підмонтовані розділи та перевірка властивостей зв’язків

Від імені користувача:

3.1. Переглядаємо дозволи на директорії – точки монтування:

```
bee@home:~$ ls -l mount
total 5
drwxrwxrwx 1 root root 4096 лис 29 13:31 EXT4
drwxrwxrwx 3 root root 1024 лис 29 13:34 NTFS
```

Користувач має право на запис, читання і виконання в цій директорії .

Якщо права Вашого користувача відрізняються, їх потрібно змінити командою:

chmod 777 EXT4

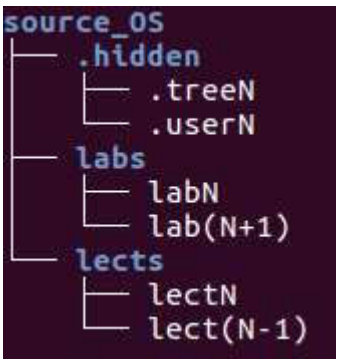
3.2. Переміщуємо директорію source_OS із її вмістом у директорію *mount/EXT4.*

Переміщуємо директорію OS із її вмістом у директорію *mount/EXT4:*

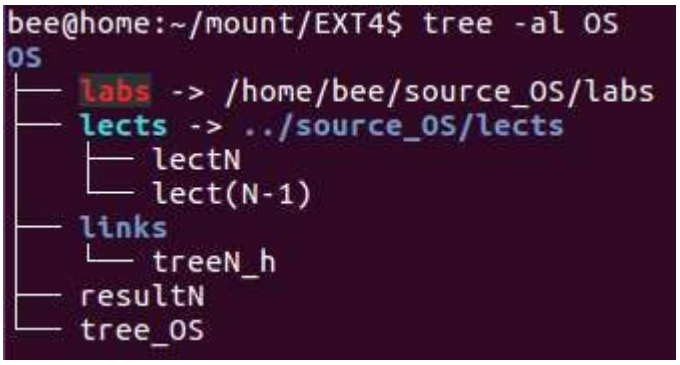
```
bee@home:~$ cd mount/EXT4
bee@home:~/mount/EXT4$ mv ~/source_OS/ .
bee@home:~/mount/EXT4$ mv ~/OS/ .
```

3.3. Перевіряємо, чи збереглися властивості переміщених символічних та жорстких зв’язків.

3.3.1. Переглянемо дерево source_OS:

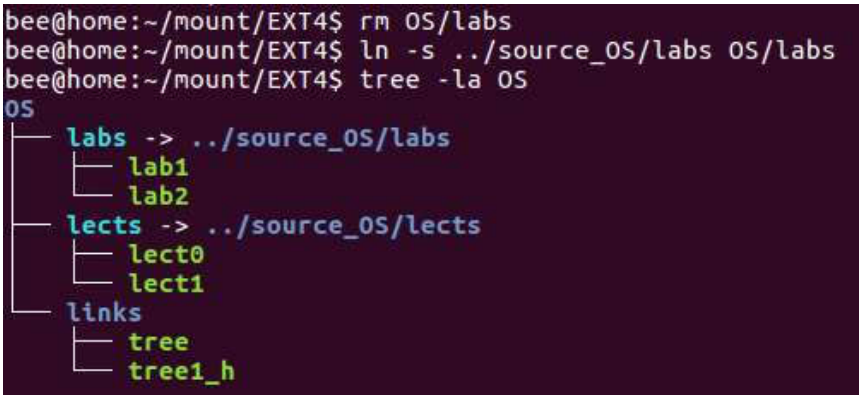


3.3.2. Переглянемо дерево OS:

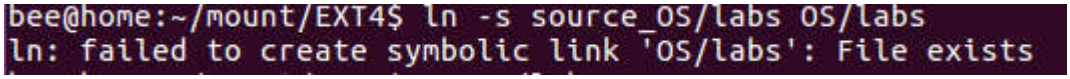


Із результату видно, що при переміщенні в іншу директорію одне із символічних посилань стало «зламаним» (виділено червоним кольором).

3.3.3. Щоб виправити помилку, потрібно видалити це посилання та створити нове із правильною адресою зв'язування:

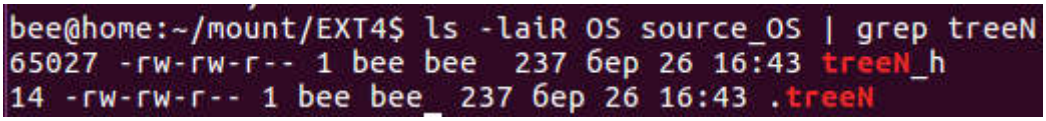


Якщо ми спробуємо «переписати» посилання, коли воно існує, то система видасть попередження і не виконає дію:



3.3.4. Порівняємо файли source_OS/.hidden/.treeN та OS/links/treeN_h - при переміщенні на інший розділ вони втрачають жорстке зв'язування.

3.3.4.1. Перевіримо і-ноди відповідних файлів: виконуємо пошук записів про файли treeN рекурсивно в директорії mount, включаючи приховані файли, і виводимо номери індексних дескрипторів та повний запис про знайдені файли:



Як бачимо, номери індексних дескрипторів відрізняються, отже файли втратили зв’язок.

Допишемо цей результат у файл resultN, спершу додавши 2 розділювача рядків:

```
bee@home:~$ echo -e >> resultN && echo -e >> resultN
bee@home:~/mount/EXT4$ ls -laiR OS source_OS | grep treeN >>OS/resultN
```

3.3.4.2. Допишемо у файл OS/links/treeN_h вміст файлу source_OS/.hidden/.userN та дерево директорій OS:

```
bee@home:~/mount/EXT4$ cat source_OS/.hidden/.userN >> OS/links/treeN_h
bee@home:~/mount/EXT4$ tree -l OS >> OS/links/treeN_h
```

3.3.4.3. Порівняємо файли .treeN та treeN_h утилітою diff:

```
bee@home:~/mount/EXT4$ diff source_OS/.hidden/.treeN OS/links/treeN_h
12a13,26
> bee
> OS
> ├── labs -> ../source_OS/labs
>     ├── labN
>     └── lab(N+1)
> ├── lects -> ../source_OS/lects
>     ├── lectN
>     └── lect(N-1)
> ├── links
>     └── treeN_h
> ├── resultN
> └── tree_OS
>
> 3 directories, 7 files
```

Файли відрізняються. Допишемо цей результат у файл resultN, спершу додавши 2 розділювача рядків:

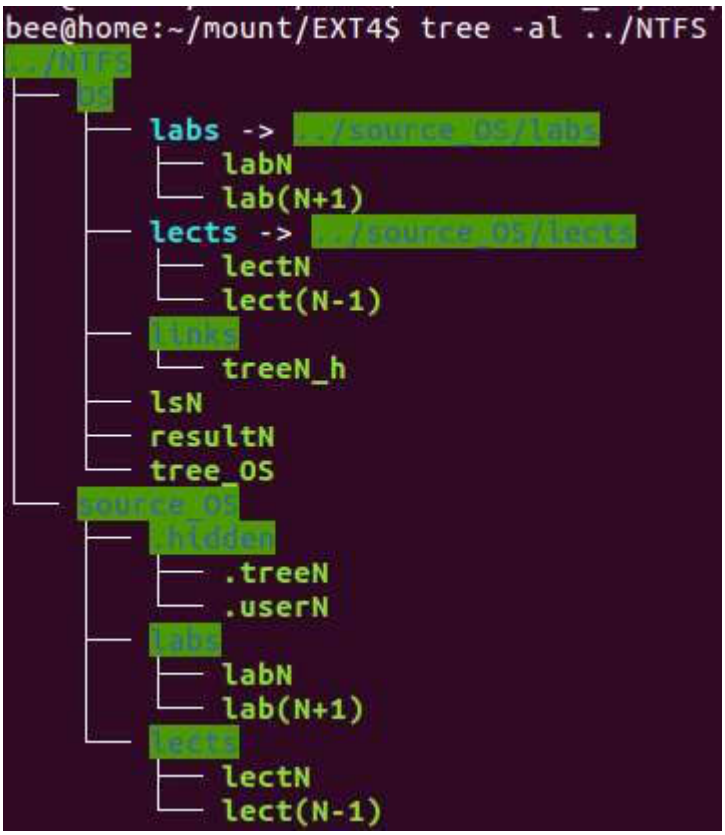
```
bee@home:~$ echo -e >> resultN && echo -e >> resultN
bee@home:~/mount/EXT4$ diff source_OS/.hidden/.treeN OS/links/treeN_h >> OS/resultN
```

3.3.5. Виконаємо рекурсивний перегляд вмісту директорій OS та source_OS і запишемо результат у файл OS/lsN:

```
bee@home:~/mount/EXT4$ ls -lRia OS source_OS >> OS/lsN
```

3.4. Перемістимо директорії на розділ із *NTFS* і переконаємося, що властивості зв’язків не втрачено:

```
bee@home:~/mount/EXT4$ mv ./* ../NTFS/
```

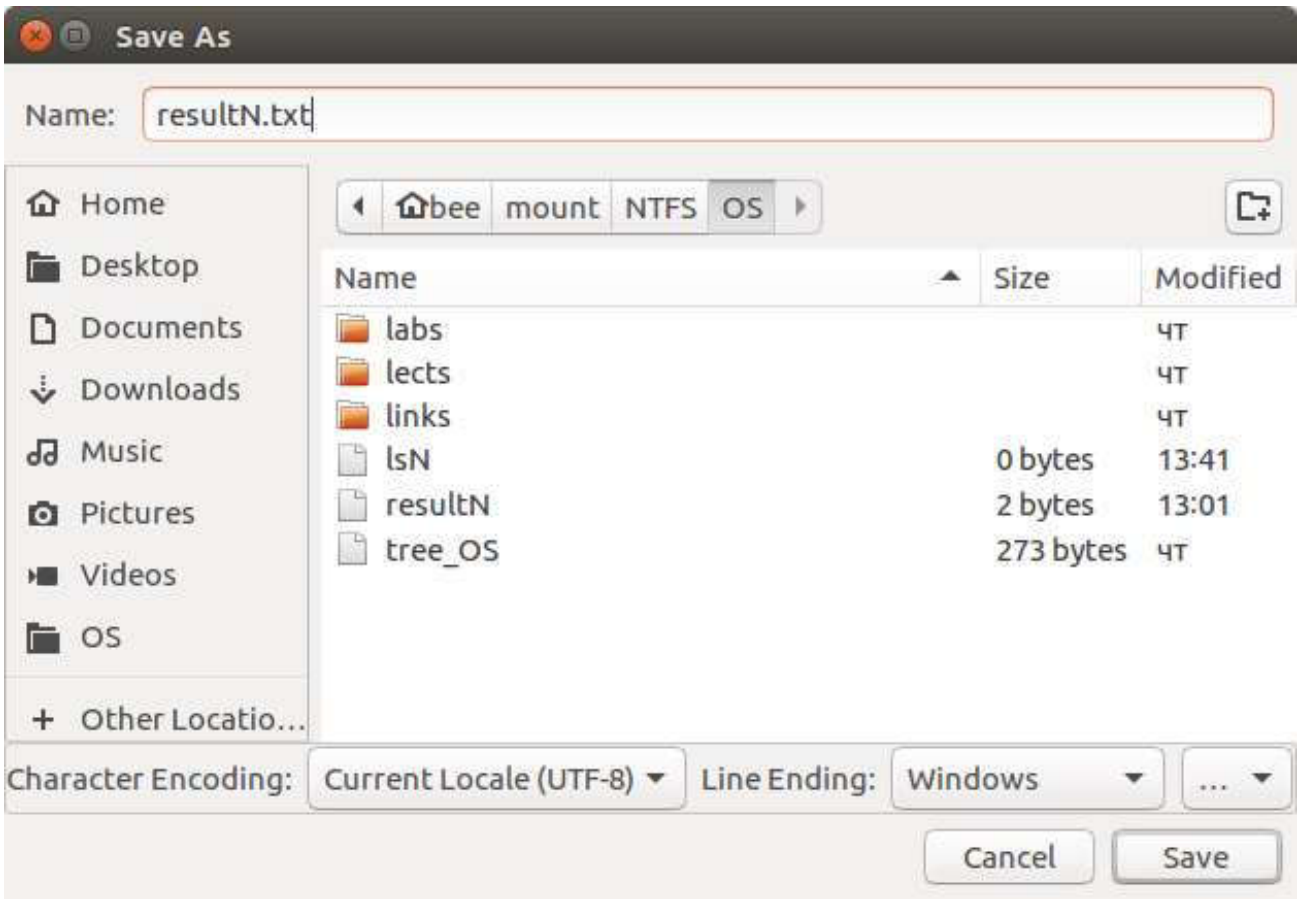


4. Підготовка файлів результатів resultN.txt, lsN.txt

4.1. Відкриємо файли результатів resultN, lsN у текстовому редакторі gedit у фоновому режимі:

```
bee@home:~/mount/NTFS/OS$ gedit resultN lsN &
[1] 2272
```

4.2. Збережемо файли у форматі txt в кодуванні utf-8 із закінченням рядків як у Windows (resultN.txt, lsN.txt відповідно):



Вимоги до оформлення результатів роботи

1. Електронний звіт про лабораторну роботу повинен мати назву N.doc та містити:

1) титульний аркуш;

2) висновки з виконання лабораторної роботи.

До захисту друкується пункт 1 електронного звіту.

2. Файл результату result.txt має бути у кодуванні utf-8 (Windows) і складатися із 6 структурних елементів, які відокремлені 2 порожніми рядками:

1) завдання 1.2.3;

2) завдання 1.2.4;

3) завдання 1.2.6;

4) завдання 2.3.3;

5) завдання 3.3.4.1;

6) завдання 3.3.4.3.

3. В файлі lsN.txt (utf-8, Windows) міститься результат виконання завдання 3.3.3.5.

4. На Google Drive у відповідну папку викладаються файли resultN.txt, lsN.txt, N.doc.

Додаткові практичні завдання

1. Визначити стиль розділів диску (MBR/GPT).

2. Показати і прокоментувати, які розділи створені на диску.

3. Визначати розмір блоку.
4. Переглянути точки монтування розділів.
5. Монтувати та демонтувати розділ (від імені адміністратора).
6. Створити файлові системи (форматування).
7. Розбити диски на розділи.
8. Відобразити серійний номер диску, UUID розділу.
9. Встановити, відобразити, знищити жорсткі та символічні зв’язки.
10. Попрацювати із файлами та директоріями.

ЗМІСТОВИЙ МОДУЛЬ 2. СИСТЕМНЕ ПРОГРАМУВАННЯ: ПРОЦЕСИ, ПОТОКИ І СИГНАЛИ

Лабораторна робота №5-6

Системне програмування: процеси, потоки і сигнали

Мета роботи:одержання практичних навичок створення, моніторингу, зміни пріоритетів процесів і завдань та надсилання їм сигналів в оболонці bash ОС Ubuntu.

- Робота виконується: на основній або віртуальній машині із ОС Ubuntu.
- Вивчаються команди: ps, top, pstree, bg, fg, nice, renice, kill, killall, tee.

Теоретичні відомості

Процеси в Linux

Кожному процесу в ОС Linux призначаються числові ідентифікатори (особисті номери) в діапазоні від 1 до 65535 (PID – Process Identifier) і ідентифікатори батьківського процесу (PPID – Parent Process Identifier). PID є ім'ям процесу, за яким ми можемо ідентифікувати процес в операційній системі при використанні різних засобів перегляду і управління процесами.

PPID визначає родинні стосунки між процесами, які в значній мірі визначають його властивості і можливості. Інші параметри, які необхідні для роботи програми, називають "оточення процесу". Одним з таких параметрів є керуючий термінал, його мають далеко не всі процеси.

Батьком всіх процесів в ОС Ubuntu 16.04.2 є процес systemd (в попередніх версіях ОС Linux це був процес init) . Його PID завжди 1, PPID – 0 . Всі процеси ОС Linux можна уявити собі у вигляді дерева, в якому коренем буде процес systemd. Цей процес хоч і не є частиною ядра, але виконує в системі дуже важливу роль.

Процеси, імена яких поміщено в квадратні дужки, наприклад "[keventd]" – це процеси ядра. Ці процеси управляють роботою системи, а точніше такими її частинами, як менеджер пам'яті, планувальник часу процесора, менеджери зовнішніх пристроїв тощо.

Решта процесів є процесами користувача, запущеними або з командної оболонки (терміналу), або з графічної оболонки, або під час ініціалізації системи.

Інтерактивний та фоновий режими виконання процесів

При звичайному запуску в оболонці bash процес працює на передньому плані, тобто процес "прив'язується" до терміналу, з якого він запущений, зчитуючи введення з цього терміналу і здійснюючи на нього виведення.

Такий режим роботи процесів називається інтерактивними.

Процес може виконуватися у фоновому режимі (не будучи пов'язаним із терміналом, тобто не виконуючи операції читання-запису на термінал), якщо при

його створенні вказати амперсант:

```
bee@bee-VirtualBox:~$ sleep 2000 &
[1] 3088
```

Для прикладу розглядається unix-утиліта sleep, яка виконує затримку на зазначений час, а після вичерпання цього часу завершується.

При створенні фонового процесу у оболонці bash в квадратних дужках вказується номер завдання в системі, а поряд – його PID.

Системні фонові процеси у Linux називаються демонами. Зазвичай, вони містять літеру d в кінці своєї назви (systemd, keventd).

Для виведення процесу із фонового режиму використовується команда fg і процес отримує фокус керування (володіє потоками введення-виведення терміналу):

```
bee@bee-VirtualBox:~$ fg
sleep 2000
```

Для призупинення інтерактивних процесів використовується комбінація клавіш Ctrl+Z – процесу надсилається сигнал SIGSTOP:

```
^Z
[1]+  Stopped                 sleep 2000
```

При цьому в терміналі відображається запис про зупинене завдання: в квадратних дужках його номер, статус (призупинено), відповідна команда (без амперсанта, оскільки процес був інтерактивним).

Для переведення призупиненого процесу у фоновий режим та його відновлення використовується команда bg:

```
bee@bee-VirtualBox:~$ bg
[1]+ sleep 2000 &
```

Моніторинг процесів та завдань

Команда jobs

Для перегляду списку фонових завдань використовується команда jobs:

```
bee@home:~$ jobs
[1]   Running                 sleep 2000 &
[2]+  Stopped                 sleep 3000
[3]   Running                 sleep 4000 &
[4]-  Running                 sleep 5000 &
```

У її виведенні знаком «+» позначено процес, з якими працювали останнім, «-» – процес, з яким працювали передостаннім.

Опція –l дозволить переглянути відповідні PID фонових завдань.

Команда ps

Для перегляду списку процесів в Linux призначена команда ps:

ps [PID] options.

Стилі опцій команди

Команда має декілька стилів подання опцій: GNU-версія цієї програми, що

входить до складу Linux, підтримує опції в стилі трьох різних типів:

- в стилі Unix98: ps [-опції] (використовується дефіс);
- в стилі BSD: ps [опції] (без дефіса);
- в стилі GNU-версії: ps [-- довге ім’я опції [--довге ім’я опції] ...] (використовується два дефіса).

Без параметрів ps показує всі процеси, які були запущені протягом поточної сесії та асоційовані з даними терміналом, за винятком демонів.

Виводиться ідентифікатор процесу (PID), ідентифікатор терміналу (TTY), витрачений до даного моменту час ЦП (TIME) і ім'я команди (CMD).

```
bee@bee-VirtualBox:~$ ps
  PID TTY          TIME CMD
 2228 pts/0    00:00:00 bash
 2241 pts/0    00:00:00 ps
```

Якщо потрібна інша інформація, слід користуватися відповідними опціями. Перелік опцій та пояснень до них можна прочитати в довіднику із команди, скориставшись утилітою *man*: *man ps.*

Опції, які найчастіше використовуються, наведено у таблиці 1.

```
bee@bee-VirtualBox:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2228  2220  0  80   0 -  7054 wait   pts/0    00:00:00 bash
0 S  1000  3088  2228  0  80   0 -  3211 restar pts/0    00:00:00 sleep
0 R  1000  3120  2228  0  80   0 -  3916 -      pts/0    00:00:00 ps
```

При вказівці опції -f утиліта ps намагається визначити ім'я команди і аргументи, з якими був створений процес. Якщо це не вдається, ім'я процесу виводиться так само, як і при відсутності опції -f, тільки у квадратних дужках (процеси ядра не мають виконуваних файлів, їх назва подається саме так).

```
bee@bee-VirtualBox:~$ ps -f
UID          PID  PPID  C STIME TTY          TIME CMD
bee         2228  2220  0 11:25 pts/0    00:00:00 bash
bee         3088  2228  0 12:23 pts/0    00:00:00 sleep 2000
bee         3122  2228  0 12:40 pts/0    00:00:00 ps -f
```

Таблиця 1. Деякі опції команди ps

Опції	Значення
-e	вивести інформацію про всі процеси
-a	вивести інформацію про всі процеси, пов’язані із терміналом
-t список_терміналів	видавати інформацію тільки про процеси, асоційовані з терміналами із заданого списку
-p список PIII процесів	видавати інформацію тільки про процеси із вказаними pid
-u список_иіІ_користувачів	видавати інформацію тільки про процеси вказаних користувачів

-C	вивести інформацію про процеси, які породжені певною командою (для всіх користувачів та терміналів)
-f	генерувати повний лістинг (full – повний) процеси ядра позначаються в квадратних дужках
-l	генерувати лістинг в довгому форматі (long – розширений вивід)
-o	включити в лістинг певні поля
-H	відображення ієрархії процесів
--sort	сортувати виведення за певними полями
ps -ax | grep httpd	виведення інформації про конкретний процес

В таблиці 2 наводяться заголовки колонок видачі ps і їх зміст.

Таблиця 2. Заголовки колонок видачі команди ps

Колонка	*Зміст*
Р (I)	Прапори (шістнадцяткові), логічна сума яких дає відомості про процес
8 (I)	Статус процесу: Б – Очікує на введення / виведення (зазвичай очікує запису на диск) 8 – Сплячий: очікує завершення події. Я – Виконуваний або Готовий: стоїть у черзі на виконання. 2 – Стан "зомбі": процес завершений, але батьківський процес не чекає цього. Т – трасований: процес зупинений сигналом, так як батьківський процес трасує його. Х – Зростаючий: процес очікує отримання більшого обсягу основної пам'яті.
	Поруч з покажчиком статусу можуть стояти додаткові символи з наступного набору (Б8Б формат): W – процес не має резидентних сторінок; <- Високо-пріоритетний процес; N – низько-пріоритетний процес; Ь – процес має сторінки, заблоковані в пам'яті. з – є лідером сеансу 1 - багатопоточний + Знаходиться в групі процесів переднього плану
ОБ (С I)	ідентифікатор власника процесу; при вказівці опції -С видається вхідне ім'я користувача
РІБ	ідентифікатор процесу
РРІБ (С 1)	ідентифікатор батьківського процесу
%сри	частка часу центрального процесора (у відсотках), виділеного даному процесу

%MEM	частка реальної пам'яті (у відсотках), яка використовується даним процесом
c (C 1)	частка виділеного планувальником часу ЦП
8TIME (C	час запуску процесу (години: хвилини: секунди). Якщо процес запущено більш ніж 24 години назад, видається місяць і день запуску

Таблиця 2. Заголовки колонок видачі команди ps (продовження)

Колонка	*Зміст*
PRI (l)	пріоритет процесу; більше число означає менший пріоритет
NI (l)	поправка до пріоритету
ADDR (l)	адреса процесу в пам'яті
SZ (l)	розмір (в блоках по 512 байт) образу процесу в пам'яті, який він потребує
VSZ	розмір ВАП процесу у Кб (VIRT у top)
RSS	розмір пам’яті процесу, який він займає на даний час (RES у [to]P)
WCHAN (l)	адреса події, яку очікує процес – системний виклик, який привів до блокування процесу. У активного процесу ця колонка порожня.
TTY	керуючий термінал (зазвичай – термінал, з якого був запущений процес). Якщо такого немає, видається символ «?»
TIME	витрачений процесом час ЦП
TIME+	сумарний витрачений процесом час ЦП
COMMAND	ім'я програми; якщо вказана опція -f, то виводиться повне ім'я команди і її аргументи.

Букви l або f в дужках означають, що ця колонка з'являється відповідно при довгому або повному форматі видачі; відсутність букв означає, що дана колонка виводиться завжди. Відзначимо, що опції -l і -f впливають тільки на формат видачі, але не на список процесів, інформацію про яких буде надано. Процес, який закінчив виконання своєї програми і має батьківський процес, який ще не дочекався завершення, як ім'я програми отримує <defunct>.

Команда top

На відміну від ps, команда top відображає стан процесів і їх активність "в реальному режимі часу".

Команда top має кілька підкоманд, найбільш корисні з яких подані в табл.3.

Таблиця 3. Підкоманди команди top

Підкоманда	*Значення*
h	help – виведення довідкової інформації
q	quit – завершення роботи команди top
f	field – додавання або видалення видимих полів
o	сортування порядку виведення інформації.

F	вибір полів, за якими виконується сортування
n	number – кількість процесів у виведенні
s	час оновлення виведення
c	command – відображення імені команди і аргументів
r	renice – зміна пріоритету обраного процесу
k	kill – надсилання сигналу обраному процесу

Виведення команди top містить 2 блоки: статистика роботи системи та статистика роботи процесів.

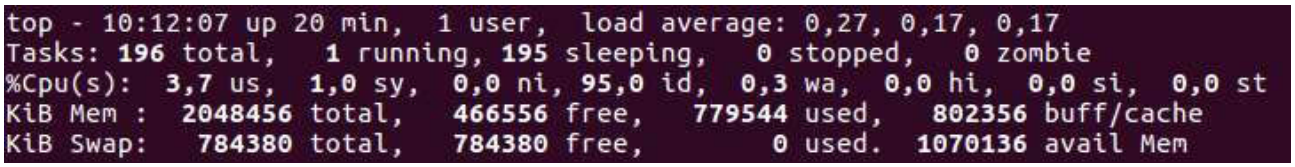
Перший блок має 5 рядків:

1 рядок – загальна інформація (top);

2 рядок – статистика процесів (task);

3 рядок – статистика використання центрального процесора (CPU);

4 і 5 рядки – статистика використання пам'яті (memory usage).



Нижче наведено опис перших 3 рядків результатів команди top, які описують стан системи:

Перший рядок виводить дані по порядку:

- поточний час (10:12:07);
- час роботи системи (20 min);
- кількість відкритих сесій користувачів (1 користувач);
- середнє завантаження системи (середнє навантаження: 0,27, 0,17, 0,17), три значення відповідають завантаженню в останню хвилину, п'ять хвилин і п'ятнадцять хвилин відповідно.

Другий рядок виводить наступні дані:

- загальна кількість процесів в системі (196 total);
- кількість працюючих в даний момент процесів (1 running);
- кількість процесів, які очікують подій (195 sleeping);
- кількість зупинених процесів (0 stopped);
- кількість процесів, які очікують батьківський процес для передачі статусу завершення (0 zombie).

Третій рядок виводить наступні дані:

- % використання CPU процесами користувача (3,7% us);
- % використання CPU системними процесами (1,0% sy);
- % використання CPU процесами з пріоритетом, підвищеним за допомогою nice (0,0% ni);
- % часу, коли CPU не використовується (95,0% id);
- % використання CPU процесами, які очікували завершення операцій

введення-виведення (0,3% wa);

- % використання центрального процесора обробниками апаратних переривань (0,0% hi – Hardware IRQ);
- % використання центрального процесора обробниками програмних переривань (0,0% si – Software Interrupts);
- кількість ресурсів CPU "запозичених" у віртуальної машини гіпервізором для інших завдань; це значення дорівнюватиме нулю на настільних комп'ютерах і серверах, які не використовують віртуальні машини (0.0% st – Steal Time (запозичений час)).

Повну інформацію про команду top можна знайти на man-сторінках. Другий блок виведення команди top стосується статистики роботи процесів. На рис. 10 наведено приклад, в якому процеси відсортовано за кількістю використаної віртуальної пам'яті (%MEM) в порядку зменшення.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1734	bee	20	0	1242520	200216	73048	S	2,0	9,8	0:59.20	compiz
1657	bee	20	0	1019412	52916	41892	S	0,0	2,6	0:00.33	unity-setti+
1951	bee	20	0	986720	111064	28200	S	0,0	5,4	0:05.22	gnome-softw+
1919	bee	20	0	880808	52312	42444	S	0,3	2,6	0:01.06	nautilus
1949	bee	20	0	857104	59496	21880	S	0,0	2,9	0:00.16	evolution-c+

Перемикати режими відображення (сортувати) можна за допомогою наступних гарячих клавіш:

<Shift> + <N> – за PID;

<Shift> + <A> – за віком;

<Shift> + <P> – за використанням ЦП;

<Shift> + <M> – за використанням пам'яті;

<Shift> + <T> – за часом виконання.

Запам’ятати комбінації клавіш зручно за відповідними словами: N – number, A – age, P – processor, M – memory, T – time.

Керування пріоритетом процесів

В кожного процесу існує параметр, який називається пріоритетом (PRI). Планувальник ОС визначає, який із запущених процесів виконуватиметься першим, який буде наступним і т.д. на основі значення пріоритету процесу. Значення поправки до пріоритету (NI – *niceness, «люб’язність»*) дозволяє процесам підвищувати або зменшувати власний пріоритет.

Завданням з великим чисельним значенням пріоритету дістається менше процесорного часу. Тому працюють вони, як правило, повільніше і споживають набагато менше системних ресурсів. Пріоритет нового процесу дорівнює пріоритету процесу-батька.

В операційних системах Linux використовується система пріоритетів (NI), яка має 40 рівнів, починаючи з -20 (найвищий пріоритет) і закінчуючи 19 (нижчий пріоритет).

Процеси, запущені звичайними користувачами, зазвичай мають пріоритет 0.

Команда ps може показати пріоритет процесу (наприклад, значення nice або NI) за допомогою опції -l.

Команда nice показує пріоритет за замовчуванням.

Запуск процесу із певним пріоритетом

Команда *nice -n число command* дозволяє вказувати пріоритет, з яким виконуватиметься процес після запуску. Від'ємне значення пріоритету може задавати лише суперкористувач (root):

sudo nice -n 10 sleep 1000.

```
bee@home:~/mount/NTFS/OS$ sleep 2000 &
[2] 2600
bee@home:~/mount/NTFS/OS$ ps -f| grep sleep
bee        2600  2000  0 14:28 pts/18   00:00:00 sleep 2000
bee        2610  2000  0 14:29 pts/18   00:00:00 grep --color=auto sleep
```

Зміна пріоритету процесу

Команда *renice -n число PID* дозволяє зменшувати пріоритет виконуваного процесу, збільшуючи його поправку до пріоритету.

Змінювати пріоритет процесу дозволено лише користувачу, який запустив процес та адміністратору.

Якщо потрібно пришвидшити виконання процесу, потрібно вказати від'ємну поправку до пріоритету. Виконати таку зміну можна лише від імені адміністратора (root):

sudo renice -n -10 PID.

Надсилання сигналів процесам

Під час роботи процесу ядро контролює його стан і в разі виникнення непередбаченої ситуації керує процесом за допомогою надсилання йому сигналу. Користувач теж може надсилати сигнали, але тільки своїм процесам. При надходженні сигналу процес може скористатися дією за замовчуванням, або, якщо у нього є обробник сигналу, він може перехопити або ігнорувати сигнал. Сигнали SIGKILL і SIGSTOP неможливо ні перехопити, ні ігнорувати, оскільки вони адресовані не процесу, а планувальнику ОС.

Команда *kill -SIGNAL pid* надсилає *SIGNAL* процесу з ідентифікатором *pid*. Якщо сигнал не вказаний, команда посилає процесу SIGTERM.

Команда *killall -s SIGNAL процес* надсилає сигнал всім процесам з іменем *процес*. Якщо сигнал не вказаний, надсилається SIGTERM.

Сигнали для цих команд необхідно вказувати без SIG . Для отримання відповідності цифрового вигляду і імені сигналу використовується опція -l команди *kill*. В таблиці 4 наведено опис деяких сигналів, що існують в ОС Ubuntu.

Звичайні користувачі можуть надсилати сигнали тільки тим процесам, для яких вони є власниками. Привілейований користувач root може посилати сигнали будь-яким процесам.

Якщо в команді kill скористатися ідентифікатором процесу (PID), рівним -1, то зазначений в команді сигнал буде надісланий всім процесам даного користувача. Коли привілейований користувач надсилає сигнал із PID=-1, він розсилається всім процесам, за винятком системних. Якщо цим сигналом буде SIGKILL, то у простих користувачів будуть втрачені всі відкриті ними, але не збережені файли даних.

Таблиця 4. Деякі сигнали в ОС Ubuntu

№	*Назва сигналу*	*Значення*
2	SIGINT	Сигнал надсилається ядром всім процесам при натисненні клавіші переривання (*<CTRL>+<C>*)
9	SIGKILL	Сигнал, при отриманні якого виконання процесу припиняється
15	SIGTERM	Сигнал зазвичай представляє певне попередження, що процес незабаром буде знищений. Цей сигнал дозволяє процесу відповідним чином "підготуватися до смерті" – видалити тимчасові файли, завершити необхідні транзакції і так далі. Команда kill за замовчуванням відправляє саме цей сигнал
18	SIGCONT	Сигнал надсилається для продовження призупиненого процесу
19	SIGSTOP	Отримання сигналу викликає зупинку виконання процесу
20	SIGTSTP	Сигнал надсилається всім процесам поточної групи при натисненні клавіш <CTRL>+<Z> . Отримання сигналу викликає зупинку виконання процесу

Перенаправлення потоку введення-виведення

Запуск декількох команд з передачею вихідного потоку наступній програмі реалізується за допомогою "|":

ps -ax | more.

Запуск команди із записом вихідного потоку в файл реалізується за допомогою ">" та ">>":

ps -ax> test.txt – створить новий файл, або перепише існуючий;

ps -ax>>test.txt – додасть виведення в кінець файлу.

Завдання на лабораторну роботу

1. Спостереження за процесами та побудова ієрархії процесів

1.1. Записати у файл результатів resultN (N – № варінту, n – остання цифра номеру варіанту) логін користувача, n, PID поточної оболонки.

1.2. Переглянути список процесів, які є у системі, і записати їх у файл

procN.

1.3. Через поля PID и PPID простежити всю ієрархію процесів поточної оболонки.

1.4. Побудувати дерево процесів, які визначені у попередньому пункті.

Результат виконання вивести на екран і дописати в файл resultN.

1.5. Переглянути список процесів вашого користувача.

1.6. Вивести список процесів вашого користувача у вигляді дерева (команда pstree).

1.7. Вивести на екран та у файл результатів resultN інформацію про процеси вашого користувача (поля S, PID, UID, параметр сортування, CMD), сортовані відповідно до завдання (параметр сортування взяти із табл. 5 відповідно з останньою цифрою номеру варіанту). Необхідно вивести (n+2) перші рядки лістингу.

2. Керування фоновими процесами та наданням процесам пріоритету

2.1. Запустити у фоні виконання 3 завдань із вказаними пріоритетами:

- yes fg (пріоритет 10);
- yes bg (пріоритет 10);
- yes nice (пріоритет 0); 2.2.

Змінити пріоритети завдань та вивести інформацію про них у resultN:

- yes fg – пріоритет 10+n,
- yes bg – пріоритет –(10+ n).

3. Команда top: моніторинг процесів, надсилання сигналів, зміна пріоритетів

3.1. Налаштувати команду top і визначити процеси, які використали найбільше часу ЦП:

- оновлення результатів кожні 5 секунд;
- кількість процесів у виведенні – n+2;
- команда повинна бути записана у повному форматі;
- стовпчики для відображення – PID, USER, PRI, NI, S, %CPU, %MEM, TIME+, COMMAND;
- процеси сортовано за TIME+ .

3.2. Знищити процес “yes bg”, який займає найбільше процесорного часу.

3.3. Змінити пріоритет процесу “yes nice” на 10+n.

3.4. Вивести на екран та у файл результатів resultN інформацію про процеси, породжені за допомогою команди yes.

4. Надсилання сигналів

4.1. Зупинити процес “yes fg”, надіславши йому сигнал SIGSTOP (19).

4.2. Знищити процес “yes fg”, надіславши йому сигнал SIGKILL (9).

4.3. Знищити процес “yes nice”, скориставшись номером завдання.

4.4. За допомогою клавіш Ctrl + D завершити дочірній bash.

5. Підготовка файлів звіту resultN.txt, procN.txt Файли результатів resultN, procN необхідно відкрити у текстовому редакторі gedit і зберегти у форматі txt в кодуванні utf-8 із закінченням рядків як у Windows.

Варіанти завдання до лабораторної роботи №8

Таблиця 5. Параметри сортування для команди ps

Остання цифра №варіанту, n	*параметр*	*сортування*
0.	pcpu	за зростанням
1.	pcpu	за спаданням
2.	c	за зростанням
3.	c	за спаданням
4.	pmem	за зростанням
5.	pmem	за спаданням
6.	vsz	за зростанням
7.	vsz	за спаданням
8.	time+	за зростанням
9.	time+	за спаданням

Послідовність виконання завдання на прикладі Ubuntu 16.04.2

1. Спостереження за процесами та побудова ієрархії процесів

1.1. Записати у файл результатів resultN (N – № варінту, n – остання цифра номеру варіанту) логін користувача, n, PID поточної оболонки.

1.1.1. Запишіть у файл результатів resultN, де N – №варінту логін свого користувача (приклад виконано для варіанту №5):

```
bee@home:~$ whoami | tee result
bee
```

1.1.2. Додайте до файлу результатів resultN номер свого варіанту:

```
bee@home:~$ echo '5' >>result
```

1.1.3. Виведіть на екран та додайте до файлу результатів resultN *PID* поточної оболонки.

Для визначення *PID* поточного процесу використовується змінна середовища $:

```
bee@home:~$ echo $$ | tee -a result
2873
```

1.2. Перегляньте список процесів, які є у системі і запишіть їх у файл procsN.

```
bee@home:~$ ps -el > procs5
```

1.3. Через поля PID и PPID простежте всю ієрархію процесів поточної оболонки. Результат виконання виведіть на екран і допишіть в файл resultN.

Для виконання цього завдання необхідно використовувати опції –p для вказування PID процесу, який нас цікавить, та –o для вказування стовпчиків, які необхідно вивести.

```
bee@home:~$ ps -p 2873
  PID TTY          TIME CMD
 2873 pts/17   00:00:00 bash
```

```
bee@home:~$ ps -p 2873 -O ppid
  PID  PPID S TTY          TIME COMMAND
 2873  2796 S pts/17   00:00:00 bash
```

```
bee@home:~$ ps -p 2873 -o pid,ppid,cmd
  PID  PPID CMD
 2873  2796 bash
```

Приклад:

```
bee@home:~$ ps -p 2796 -o pid,ppid,cmd
  PID  PPID CMD
 2796  1248 /usr/lib/gnome-terminal/gnome-terminal-server
bee@home:~$ ps -p 1248 -o pid,ppid,cmd
  PID  PPID CMD
 1248  1222 /sbin/upstart --user
bee@home:~$ ps -p 1222 -o pid,ppid,cmd
  PID  PPID CMD
 1222   893 lightdm --session-child 12 15
bee@home:~$ ps -p 893 -o pid,ppid,cmd
  PID  PPID CMD
  893     1 /usr/sbin/lightdm
bee@home:~$ ps -p 1 -o pid,ppid,cmd
  PID  PPID CMD
    1     0 /sbin/init splash
```

1.4. Побудуйте дерево процесів, які визначено у попередньому пункті.

```
bee@home:~$ ps -p 1,893,1222,1248,2796,2873 -f --forest
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 19:45 ?        00:00:01 /sbin/init splash
root       893     1  0 19:46 ?        00:00:00 /usr/sbin/lightdm
root      1222   893  0 19:46 ?        00:00:00  \_ lightdm --session-child 12 15
bee       1248  1222  0 19:46 ?        00:00:00      \_ /sbin/upstart --user
bee       2796  1248  0 19:53 ?        00:00:14          \_ /usr/lib/gnome-terminal/gnome-terminal-server
bee       2873  2796  0 20:03 pts/17   00:00:00              \_ bash
bee@home:~$ ps -p 1,893,1222,1248,2796,2873 -l --forest
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 29959 -      ?        00:00:01 systemd
4 S     0   893     1  0  80   0 - 87627 -      ?        00:00:00 lightdm
4 S     0  1222   893  0  80   0 - 58092 -      ?        00:00:00  \_ lightdm
4 S  1000  1248  1222  0  80   0 - 11995 poll_s ?        00:00:00      \_ upstart
0 S  1000  2796  1248  0  80   0 - 163789 poll_s ?       00:00:14          \_ gnome-terminal-
0 S  1000  2873  2796  0  80   0 -  6060 wait   pts/17   00:00:00              \_ bash
```

Додаємо порожній рядок у файл звіту (важливо для автоматизованої перевірки результатів лабораторної роботи) і виводимо на екран та у файл результат побудови дерева:

```
bee@home:~$ echo -e >> result5
```

```
bee@home:~$ ps -p 1,893,1222,1248,2796,2873 -l --forest | tee -a result
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 29959 -      ?        00:00:01 systemd
4 S     0   893     1  0  80   0 - 87627 -      ?        00:00:00 lightdm
4 S     0  1222   893  0  80   0 - 58092 -      ?        00:00:00  \_ lightdm
4 S  1000  1248  1222  0  80   0 - 11995 poll_s ?        00:00:00      \_ upstart
0 S  1000  2796  1248  0  80   0 - 163789 poll_s ?       00:00:14          \_ gnome-terminal-
0 S  1000  2873  2796  0  80   0 -  6060 wait   pts/17   00:00:00              \_ bash
```

Як видно із дерева, батьком наших процесів є процес номер один systemd, який запускається із файлу /sbin/init.

Примітка: завдання 4-6 потрібно виконувати підряд без перерви, оскільки файл procsN використовується для перевірки результатів.

1.5. Перегляньте список процесів вашого користувача:

```
bee@home:~$ ps -lf -u bee
```

1.6. Виведіть список процесів вашого користувача у вигляді дерева (команда pstree).

1.6.1. Виведіть список процесів вашого користувача у вигляді дерева:

```
bee@home:~$ pstree bee
VBoxClient---VBoxClient---{SHCLIP}

VBoxClient---VBoxClient---{MONITOR}

VBoxClient---VBoxClient---{MONITOR}

VBoxClient---VBoxClient-+-{dndHGCM}
                        `-{dndX11}

upstart-+-at-spi-bus-laun-+-dbus-daemon
                          `-{dconf worker}
```

1.6.2. Для того, щоб знайти будь-який процес, використовуйте конвеєр:

```
bee@home:~$ pstree bee | less
```

В інтерактивному режимі програми less надрукуйте:

```
/pstree
```

і натисніть клавішу Enter. Вийдіть з програми натисканням клавіші q.

1.7. Виведіть на екран та у файл результатів інформацію про процеси вашого користувача (поля S, PID, UID, параметр сортування, CMD), сортовані відповідно до завдання (параметр сортування взяти із табл. 5 відповідно з останньою цифрою номеру варіанту). Необхідно вивести (n+2) перші рядки лістингу, де n – остання цифра в номері вашого варіанту.

1.7.1. Спершу додаємо порожній рядок у файл звіту:

```
bee@home:~$ echo -e >> result5
```

1.7.2. Потім виводимо на екран та записуємо результати сортування процесів:

```
bee@home:~$ ps -o s,pid,uid,pmem,cmd -u bee --sort -pmem | head -n 7 | tee -a result5
S   PID   UID %MEM CMD
R  1513  1000  9.9 compiz
S  1718  1000  5.3 /usr/bin/gnome-software --gapplication-service
S  1708  1000  3.0 /usr/lib/evolution/evolution-calendar-factory
S  1806  1000  2.6 /usr/lib/evolution/evolution-calendar-factory-subprocess --factory
aserver/Subprocess/Backend/Calendar/1708/3
S  1684  1000  2.5 nautilus -n
S  1797  1000  2.5 /usr/lib/evolution/evolution-calendar-factory-subprocess --factory
dataserver/Subprocess/Backend/Calendar/1708/2
```

2. Керування фоновими процесами та наданням процесам пріоритету

Bash володіє корисною властивістю виконувати завдання у фоновому режимі, для цього після виконуваної команди потрібно поставити символ амперсанд – «&». Для виконання завдання використовується команда yes [option], яка виводить на термінал по замовчуванню літеру «y» або рядок, який вказано як опцію. Вивід цієї команди буде перенаправлено у спеціальний файл-поглинач /dev/null.

2.1. Запустити у фоні виконання 3 завдань із вказаними пріоритетами:

2.2.1. Запустіть команду, яка буде працювати «вічно»:

```
bee@home:~$ yes "fg" >/dev/null
```

2.2.2. Зупиніть виконання команди, натиснувши поєднання клавіш Ctrl + Z – призупинення виконання процесу.

```
^Z
[1]+  Stopped                 yes "fg" > /dev/null
```

2.2.3. Продовжте виконання цієї команди в фоновому режимі:

```
bee@home:~$ bg
[1]+ yes "fg" > /dev/null &
```

З’явиться напис на зразок наступного: [1] (в квадратних дужках вказано номер завдання).

2.2.4. Перегляньте пріоритет процесу

```
bee@home:~$ ps -l -C yes
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 R  1000  2964  2784 81  80   0 -  2198 -      pts/5    00:01:12 yes
```

2.2.5. Відкрийте директорію, яка містить інформацію про роботу щойно створеного процесу:

```
bee@home:/proc/2964/fd$ ls -l
total 0
lrwx------ 1 bee bee 64 бер  5 21:28 0 -> /dev/pts/5
l-wx------ 1 bee bee 64 бер  5 21:28 1 -> /dev/null
lrwx------ 1 bee bee 64 бер  5 21:02 2 -> /dev/pts/5
```

Як видно із результатів команди, дескриптор вихідного потоку 1 перенаправлено у файл-поглинач.

2.2.6. Переконайтеся за допомогою команди jobs, що в фоні працює одна задача

```
bee@home:~$ jobs
[1]+  Running                 yes "fg" > /dev/null &
```

2.2.7. Запустіть ще одну задачу з пріоритетом 0:

```
bee@home:~$ yes "bg" >/dev/null&
[2] 2986
```

2.2.8. За допомогою команди *nice* запускаємо ще один фоновий процес:

```
bee@home:~$ nice yes "nice" > /dev/null&
[3] 2997
```

2.2. Змінити пріоритети завдань та вивести інформацію про них у resultN.

2.2.1. Змініть пріоритет процесу на 10+n (сповільнить виконання команди), n – остання цифра номеру варіанту:

```
bee@home:~$ renice -n 19 2964
2964 (process ID) old priority 0, new priority 19
```

```
bee@home:~$ ps -l -C yes
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 R  1000  2964  2784 92  99  19 -  2198 -      pts/5    00:04:42 yes
```

2.2.2. Переглянемо пріоритети процесів, запущених за допомогою команди yes.

```
bee@home:~$ ps -C yes -lf
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 R bee       2964  2784 46  99  19 -  2198 -      21:00 pts/5    00:07:16 yes fg
0 R bee       2986  2784 94  80   0 -  2198 -      21:08 pts/5    00:07:34 yes bg
0 R bee       2997  2784  2  90  10 -  2198 -      21:11 pts/5    00:00:06 yes nice
```

2.2.3. Збільште пріоритет процесу yes bg (від імені root) до –(10+ n), де n – остання цифра номеру варіанту

```
bee@home:~$ sudo renice -n -19 2986
2986 (process ID) old priority 0, new priority -19
```

2.2.4. Виведіть на екран та у файл результатів інформацію про процеси, породжені за допомогою команди yes.

Спершу додаємо порожній рядок у файл звіту:

```
bee@home:~$ echo -n >> result5
```

```
bee@home:~$ ps -C yes -lf | tee -a result
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 R bee       2964  2784 33  99  19 -  2198 -      21:00 pts/5    00:07:18 yes fg
0 R bee       2986  2784 93  61 -19 -  2198 -      21:08 pts/5    00:13:16 yes bg
0 R bee       2997  2784  3  90  10 -  2198 -      21:11 pts/5    00:00:21 yes nice
```

Зверніть увагу, що процес, який із вищим пріоритетом, має більший показник використання процесорного часу.

3. Команда top: моніторинг процесів та надсилання сигналів

3.1. Налаштувати команду top і визначити процеси, які використали найбільше часу ЦП.

3.1.1. Відкрийте команду top. Налаштуйте команду:

- на оновлення результатів кожні 5 секунд (опція s);
- кількість процесів у виведенні – n+2 (опція n), де n – остання цифра номеру варіанту;
- команда повинна бути записана у повному форматі (опція c);

➢ стовпчики для відображення – PID, USER, PRI, NI, S, %CPU, %MEM, TIME+, COMMAND (опція f);

➢ процеси сортовано за TIME+ (опція f-s).

```
top - 10:12:07 up 20 min,  1 user,  load average: 0,27, 0,17, 0,17
Tasks: 196 total,   1 running, 195 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3,7 us,  1,0 sy,  0,0 ni, 95,0 id,  0,3 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem :  2048456 total,   466556 free,   779544 used,   802356 buff/cache
KiB Swap:   784380 total,   784380 free,        0 used.  1070136 avail Mem

  PID USER      PR  NI S %CPU %MEM     TIME+ COMMAND
 2986 bee        1 -19 R 89,7  0,0  57:35.92 yes bg
 2964 bee       39  19 R  0,0  0,0   7:18.87 yes fg
 1561 bee       20   0 S  6,9 11,2   4:22.65 compiz
  945 root      20   0 S  1,1  8,0   1:08.25 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth
 2997 bee       30  10 R  0,0  0,0   0:26.07 yes nice
 1337 bee       20   0 S  0,0  0,1   0:12.43 /usr/bin/VBoxClient --draganddrop
 1986 bee       20   0 S  0,0  1,8   0:11.41 /usr/lib/gnome-terminal/gnome-terminal-server
```

3.1.2. Зробіть скріншот екрану.

Оскільки наші завдання не працюють із пам’яттю, відсоток її використання відповідно нульовий, але процесорного часу вони споживають багато.

3.2. Знищити процес “yes bg”, який займає найбільше процесорного часу, виконавши в інтерактивному режимі утиліти top команду *kill (опція* k).

Необхідно буде ввести PID процесу (можна подивитися в команді top), який треба «вбити», та натиснути Enter.

Зверніть увагу, що користувач має право надсилати сингали виключно своїм процесам. Для надсилання сигналів будь-якому процесу потрібні привілеї адміністратора.

3.3. Змініть пріоритет процесу “yes nice” на 10+n (опція r). В команді top можна уповільнювати свої процеси. Для пришвидшення їх виконання, або для впливу на процеси інших користувачів потрібно запускати top із правами адміністратора.

Виконайте вихід з команди top клавішею q. Оскільки управління терміналом знову повернулося до оболонки, вона отримала і обробила повідомлення про знищення процесу “yes bg”.

```
[2]-  Terminated              yes "bg" > /dev/null
```

3.4. Виведіть на екран та у файл результатів інформацію про процеси, породжені за допомогою команди yes.

Спершу додаємо порожній рядок у файл звіту:

```
bee@home:~$ echo -e '\n' >> result5
```

```
bee@home:~$ ps -lf -C yes | tee -a
```

4. Надсилання сигналів

4.1. Зупинити процес “yes fg”, надіславши йому сигнал SIGSTOP (19).

4.1.1. З'ясуйте PID та номери решти завдань, результат виведіть на екран:

```
bee@home:~$ jobs -l | tee -a result
[1]-  2964 Running                 yes "fg" > /dev/null &
[3]+  2997 Running                 nice yes "nice" > /dev/null &
```

4.1.2. Зупинимо процес “yes fg”, надіславши йому сигнал SIGSTOP (19), який не може бути проігнорованим.

```
bee@home:~$ kill -19 2964
bee@home:~$ ps -fl
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S bee       2364  1986  0  80   0 -  6060 wait   19:41 pts/5    00:00:00 bash
0 S bee       2784  2364  0  80   0 -  6071 wait   20:34 pts/5    00:00:00 bash
0 T bee       2964  2784  8  99  19 -  2198 signal 21:00 pts/5    00:09:48 yes fg
0 R bee       2997  2784 18  90  10 -  2198 -      21:11 pts/5    00:18:48 yes nice
0 R bee       3382  2784  0  80   0 -  9716 -      22:54 pts/5    00:00:00 ps -fl

[1]+  Stopped                 yes "fg" > /dev/null
```

4.2. Якщо тепер ми надішлемо призупиненому процесу сигнал на завершення роботи, процес не зможе його прийняти, доки не вийде із стану «T», тобто знову буде запущений. Але, якщо надіслати сигнал SIGKILL (9), який призначений планувальнику, процес буде знищено.

```
bee@home:~$ kill -9 2964
bee@home:~$ ps -fl
F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
0 S bee       2364  1986  0  80   0 -  6060 wait   19:41 pts/5    00:00:00 bash
0 S bee       2784  2364  0  80   0 -  6071 wait   20:34 pts/5    00:00:00 bash
0 R bee       2997  2784 23  90  10 -  2198 -      21:11 pts/5    00:25:55 yes nice
0 R bee       3408  2784  0  80   0 -  9716 -      23:02 pts/5    00:00:00 ps -fl
[1]+  Killed                  yes "fg" > /dev/null
```

4.3.Необхідно знищити останній фоновий процес, скориставшись його номером:

```
bee@home:~$ kill %3
```

, де %N – номер завдання

Переконайтеся, що в фоні тепер не виконується жодної задачі

```
bee@home:~$ jobs
bee@home:~$
```

Процеси можна також знищувати:

- командою kill, використовуючи PID (надсилає сигнал «ввічливого» завершення програми SIGTERM(15)): *$ kill PID;*
- командою killall. Дана команда може зупинити процес (набір процесів) за його іменем:*$ killall yes.*

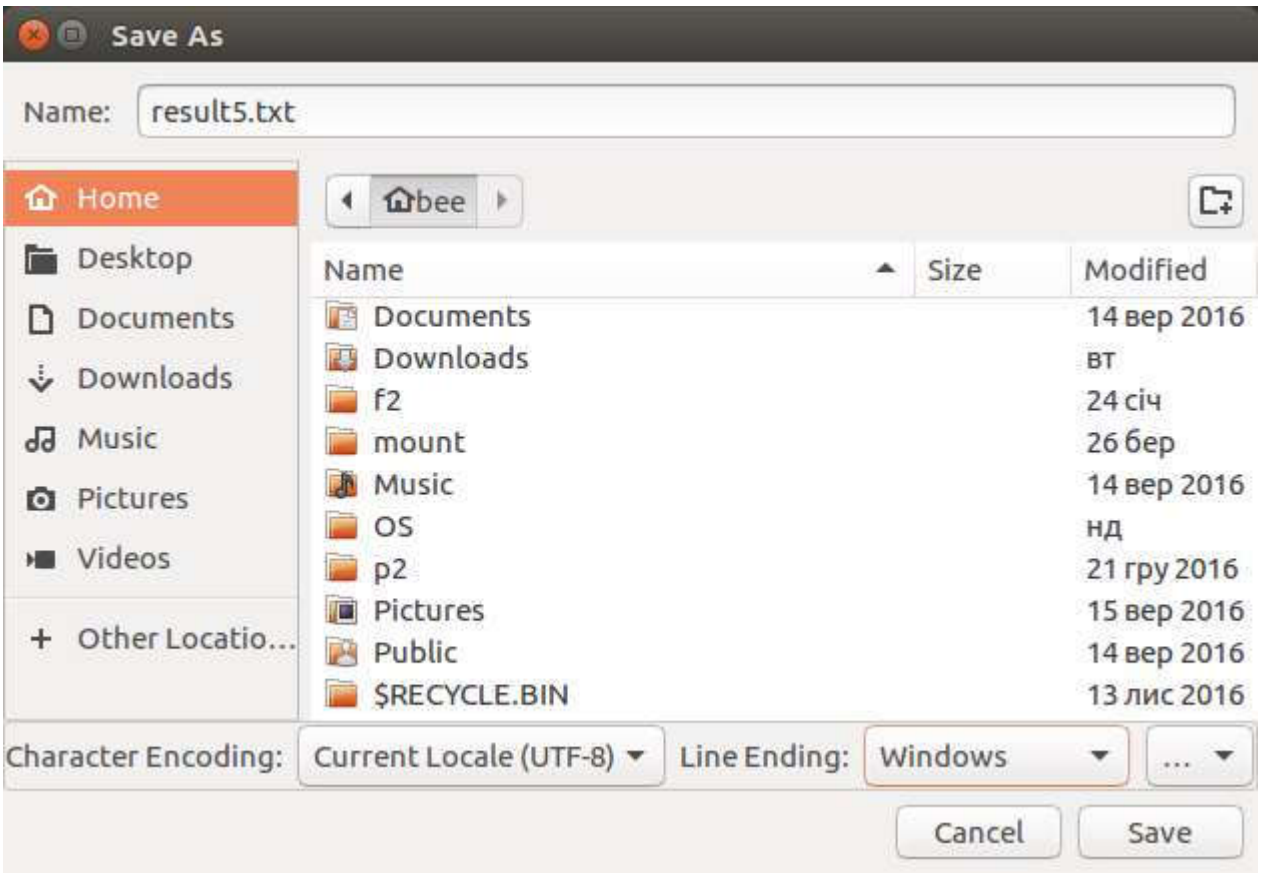
4.4. За допомогою клавіш Ctrl + D (закриття вхідного потоку) завершите дочірній bash.

5. Підготовка файлів результатів resultN.txt, procN.txt

5.1. Відкриємо файли результатів resultN, procN у текстовому редакторі gedit.

```
bee@home:~$ gedit result5 proc5&
[1] 3369
```

5.2. Збережемо файли у форматі txt в кодуванні utf-8 із закінченням рядків як у Windows.



Вимоги до оформлення звіту

1. Електронний звіт про лабораторну роботу повинен мати назву N.doc та містити:

1) титульний лист;

2) скріншот до п.3.1;

3) висновки з виконання лабораторної роботи.

До захисту друкуються пункти 1, 2 електронного звіту.

2. Файл результату resultN.txt (utf-8, Windows) має складатися із 5 структурних елементів, які відокремлені порожнім рядком:

1) завдання 1.1,

2) завдання 1.4,

3) завдання 1.7,

4) завдання 2.2,

5) завдання 3.4.

3. В файлі procN.txt (utf-8, Windows) міститься результат виконання завдання 1.2.

4. На GoogleDrive викладаються файли resultN.txt, procN.txt, N.doc.

Додаткові практичні завдання

1. Запустити процес з конкретним пріоритетом.

2. Змінити пріоритет конкретного процесу.

3. Вивести інформацію про працюючі процеси.

4. Запустити процес у фоновому режимі, вивести процес із нього.

5. Передати вихідний потік іншій програмі.

6. Записати вихідний потік процесу в файл.

7. Послати сигнал на припинення процесу за PID, за іменем, за номером завдання.

8. Вивести інформацію про працюючі процеси.

9. Знищити процес (надсилати сигнали).

10. Вивести інформацію про працюючі процеси з різною кількістю стовпців і процесів.

11. Вивести динамічну інформацію про процеси, сортувати цю інформацію за одним стовпчиком, прибрати і додати стовпці, змінити пріоритет процесу, знищити процес.

12. Пояснити виведення команд ps, top (перші 3 рядки).

Лабораторна робота №7-8
Створення і завершення процесу в ОС Windows та реалізація потоків в ОС Windows

- Робота виконується на основній машині із ОС Windows 7.
- Використовується програма Oracle VM VirtualBox 5.0.2, файл твіку реєстру lab11.reg, iso-образ гостьової ОС.

1. Встановлення Oracle VM VirtualBox

Дистрибутив Oracle VM VirtualBox (інсталяційний файл) можна знайти
за адресою: https://www.virtualbox.org//wiki/Downloads

Виберіть дистрибутив, що відповідає операційній системі хоста. Наприклад, VirtualBox-5.0.2-102096-Win.exe, розміру 114,2 МБ для Windows hosts x86/amd64 та відповідні гостьові доповнення Oracle_VM_VirtualBox_Extension_Pack-5.0.2-102096.vbox-extpack.
Після запуску дистрибутива дотримуйтесь інструкцій.

Рисунок 1. Діалогове вікно запуску дистрибутива

При першому запуску VirtualBox повинно з'явитися вікно (рис. 2)

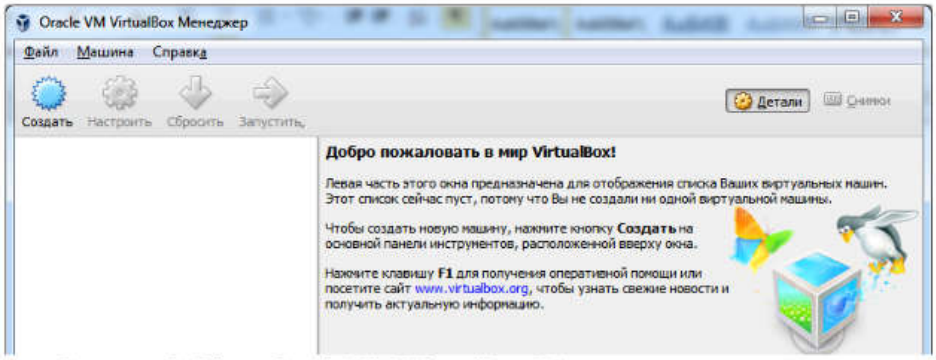


Рисунок 2. Вікно Oracle VM VirtualBox Manager при першому запуску

Панель ліворуч призначена для відображення списку встановлених віртуальних машин. Оскільки віртуальні машини не встановлені, то список порожній. Права панель призначена для відображення властивостей обраної віртуальної машини. Оскільки віртуальні машини не встановлені, то в панелі відображається вітальне повідомлення.

2. Створення віртуальної машини

Створення ВМ виконується на прикладі ОС Windows 10. Натиснувши клавішу "Створити", з'явиться діалогове вікно. Дотримуйтесь підказок у діалоговому вікні. «Детальний режим» створення ВМ. За замовчуванням користувач створює ВМ в «детальному режимі»:

- спершу вказуємо Ім’я та тип ОС;
- обираємо розмір оперативної пам'яті для ВМ;
- обираємо формат віртуального жорсткого диску (ВЖД).

Ім'я ВМ буде пізніше відображатися в списку ВМ, також воно буде використовуватися для імені файлу налаштувань ВМ. Тому корисніше використовувати інформативні імена. Оберіть із списку операційних систем тип встановлюваної ОС. Якщо ви хочете встановити щось інше, чого немає в списку, виберіть "Other". Версія вибирається користувачем із запропонованого списку і повинна точно відповідати наявному дистрибутиву ОС.

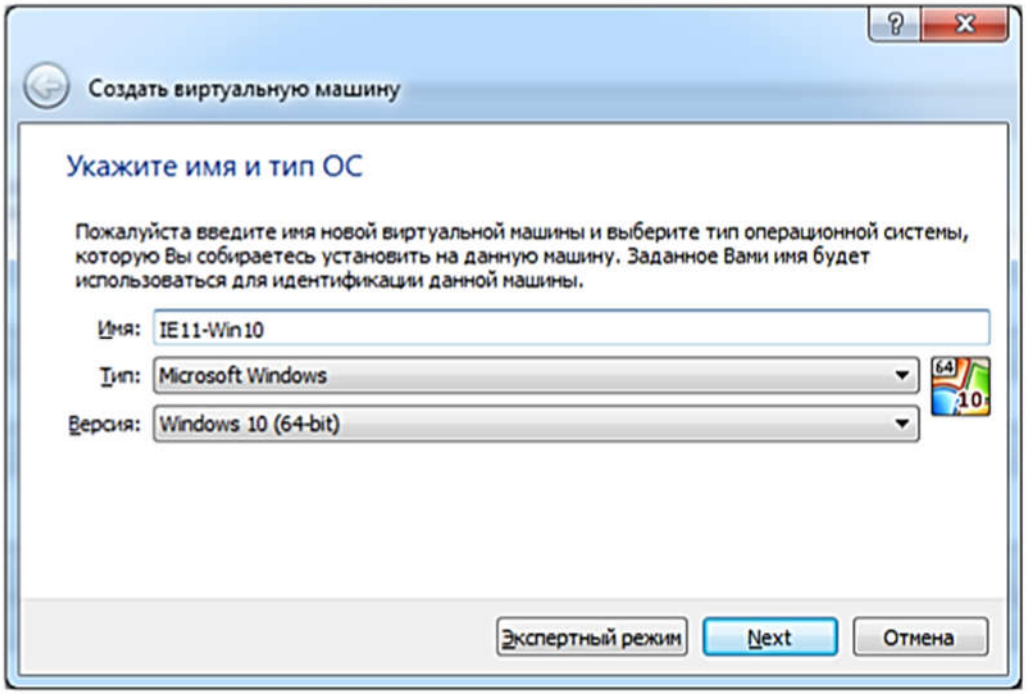


Рисунок 3. Діалогове вікно створення віртуальної машини

У наступному вікні вказуємо об'єм пам'яті, що виділений для ВМ. Він буде не доступний для хоста (рис. 4).

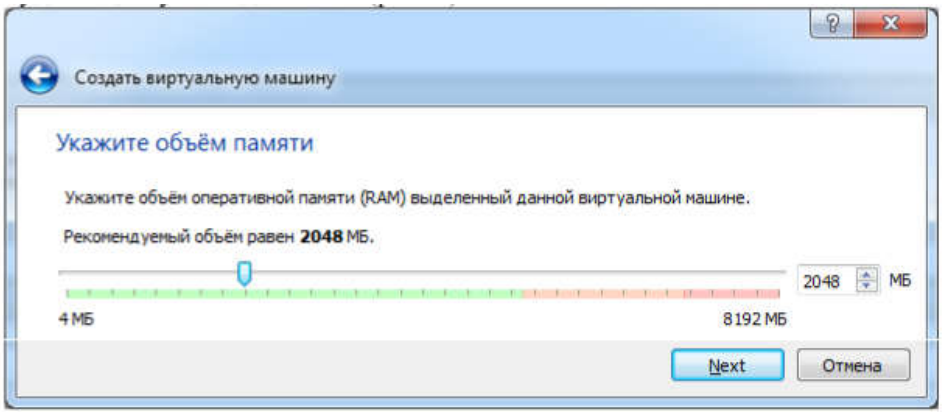


Рисунок 4. Вікно для вибору розміру пам'яті

Наступний крок – вибір віртуального жорсткого диску. При цьому можна використовувати існуючий ВЖД для раніше створеної

ВМ (рис. 5).

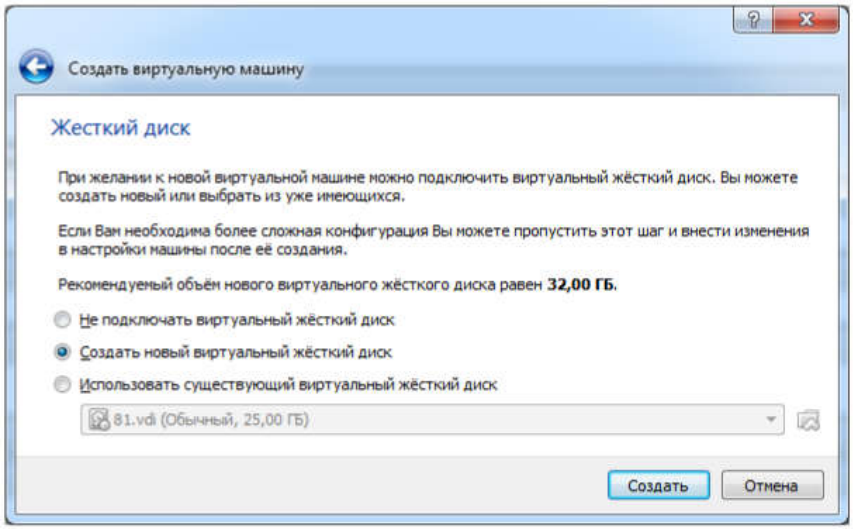


Рисунок 5. Вікно для підключення жорсткого диску для ВМ

Тип ВЖД залишаємо VDI – це рідний формат для VirtualBox (рис. 6).

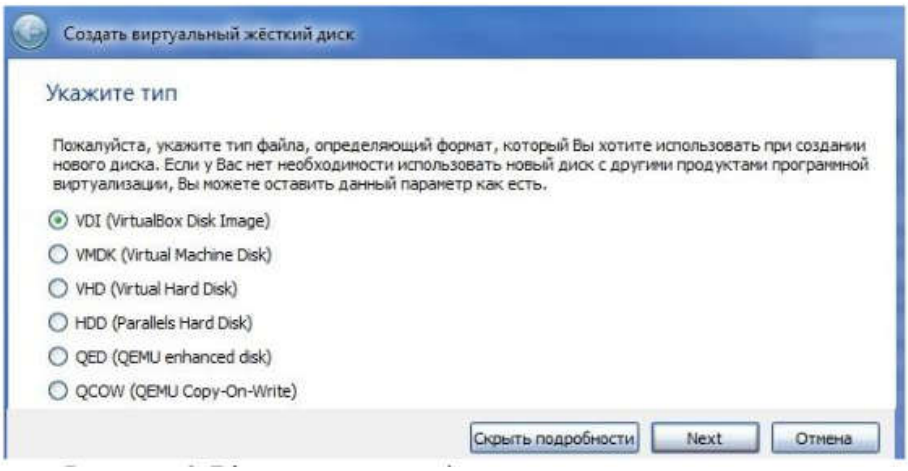


Рисунок 6. Вікно створення віртуального жорсткого диска

Після кліка по кнопці "Next" наступне вікно запропонує вибрати формат зберігання ВЖД.

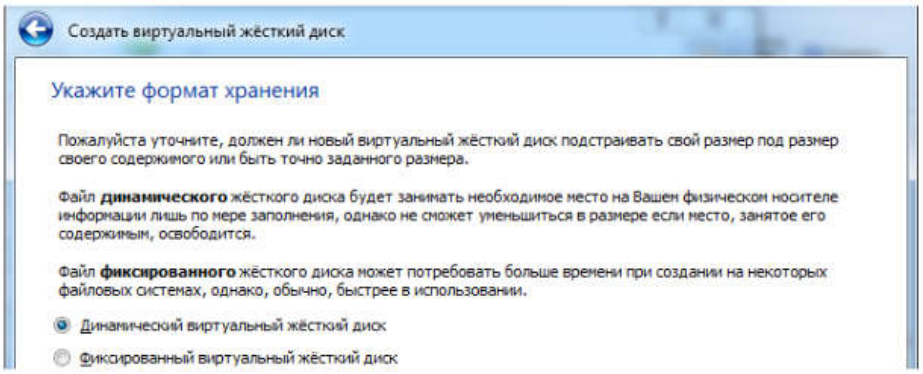


Рисунок 7. Вікно вибору формату зберігання

Після підтвердження вибору нова віртуальна машина з вказаними параметрами буде створена. «Експертний режим» створення ВМ

«Експертний режим» створення ВМ дозволяє користувачу вказати згадані вище параметри віртуальної машини в одному вікні (рис. 8).

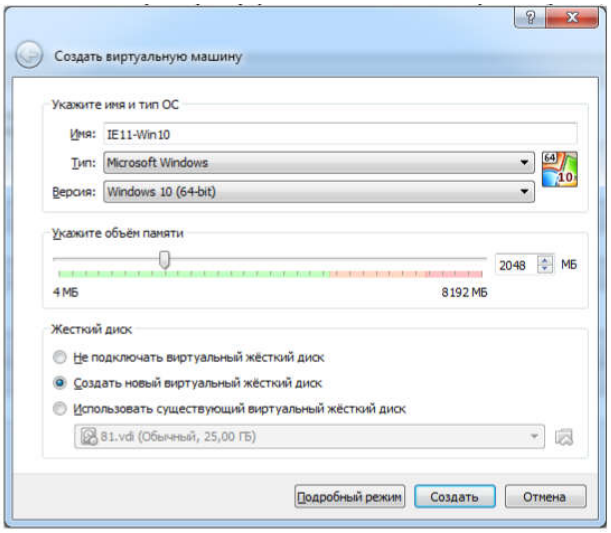


Рисунок 8. Вікно «експертного режиму» створення ВМ

«Експертний режим» також доступний при створенні віртуального жорсткого диску і дозволяє вказати назву ВЖД, його розташування у файловій системі основної ОС, розмір, тип та формат зберігання (рис. 9).

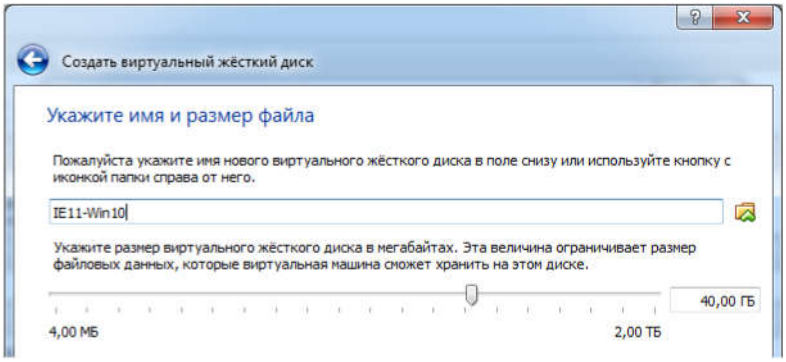


Рисунок 9. Вікно «експертного режиму» створення ВМ

За замовчуванням, диск ВМ буде розташовуватися в папці C:\Users\%username%\VirtualBox VMs\, де «%username%» – ім'я облікового запису користувача в Windows. В інших ОС все буде трохи відрізнятися. Запропонований обсяг диска ВМ залишаємо без змін або змінюємо у випадку, якщо необхідно заощадити або виділити додаткове місце.

3. Запуск віртуальної машини і встановлення гостьової ОС

Після створення нової ВМ на лівій панелі відкритого вікна "Oracle VM VirtualBox Менеджер" з'явиться відповідний запис. Машина вже готова, для запуску ВМ залишилося підключити образ

завантажувального диска до приводу ВМ або вказати, що ми будемо використовувати фізичний привід оптичних дисків, якщо інсталяційний диск у вас вже є на окремому оптичному носії. Є наступні методи запуску віртуальної машини: – двічі клікнути мишею на її назві в списку вікна менеджера ВМ; – вибрати її в списку вікна менеджера і натиснути кнопку "Запустити". Ця команда відкриє нове вікно з ВМ, в якому з'явиться вікно "Виберіть завантажувальний диск" для завантаження операційної системи віртуальної машини (рис. 10).

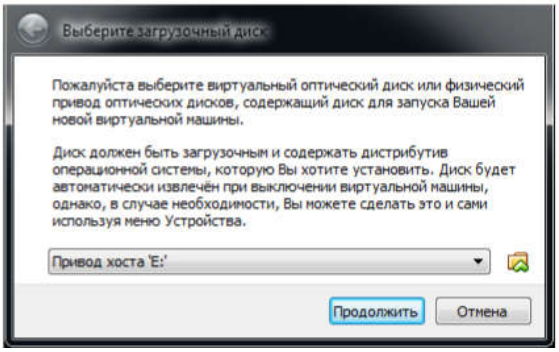


Рисунок 10. Вікно вибору завантажувального диска

Далі все буде точно так само, як при звичайному встановленні операційної системи.

4. Налаштовування віртуальної машини

Кожну віртуальну машину можна налаштувати з урахуванням особливостей її використання. Кліком по кнопці "Налаштування" відкриваємо вікно "Ім’я_ВМ – Налаштування" (рис. 11).

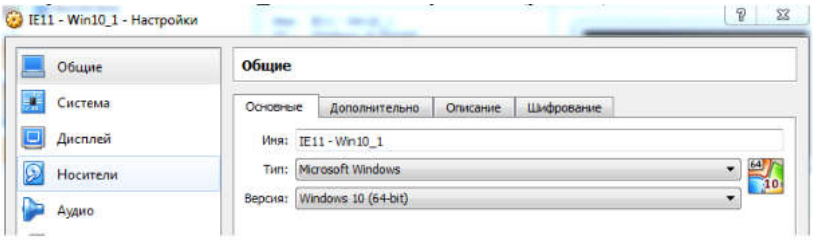


Рисунок 11. Вікно для зміни параметрів ВМ

Ліва панель нагадує диспетчер пристроїв. Права панель містить набори вкладок, що відповідають активному пункту лівої панелі. У нижній частині правої панелі – інтерактивна довідка. Розберемо призначення основних вкладок.

- Вкладка "Загальне. Основне" містить значення основних параметрів нашої віртуальної машини.

• Вкладка "Загальне. Додатково" містить такі параметри (рис. 12):

○ «Папка для знімків» містить значення шляхів для знімків ВМ. Знімки (snapshots) ВМ – це файлові знімки стану, даних диска і конфігурації ВМ у певний момент часу. На одну ВМ можна створити кілька знімків, які містять відмінні один від одного налаштування і встановлені додатки.

○ «Загальний буфер обміну» і "drag'n'drop" може приймати чотири значення: «вимкнено», «тільки з гостьової ОС в основну», «тільки з основної ОС в гостьову», «двонаправлений», які визначають, як буде працювати буфер обміну та drag'n'drop між Вашою host-системою і віртуальною машиною.

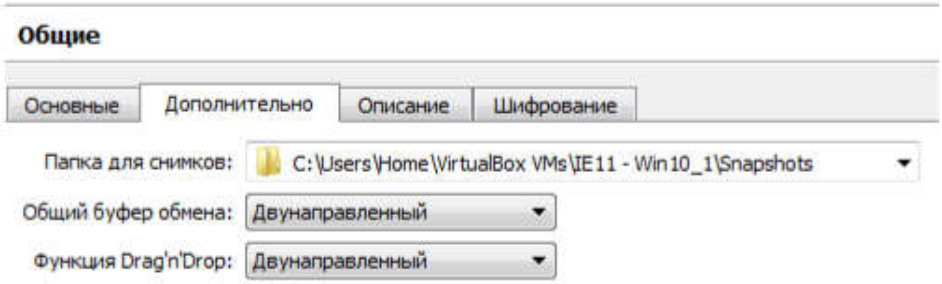


Рисунок 12. Вікно Загальне для зміни параметрів ВМ

Буфер обміну та функція drag'n'drop можуть працювати по-різному на різних ОС (залежить від ОС хоста та гостя). Одні ОС здатні обмінюватися тільки текстовими чи графічними даними (копіювання і вставка із текстового чи графічного редактора, знімки екранів), інші можуть копіювати і передавати об’єкти файлової системи. Робота із загальним буфером обміну (clipboard) буде доступна тільки в тому випадку, якщо встановлено Гостьові доповнення VirtualBox в гостьовій операційній системі.

• Вкладка "Система. Материнська плата" містить інформацію (рис. 13):

○ про розмір оперативної пам'яті;

○ про порядок завантаження;

○ про набор мікросхем ВМ, що використовується;

○ про значення інших параметрів, що описані в інтерактивному меню.

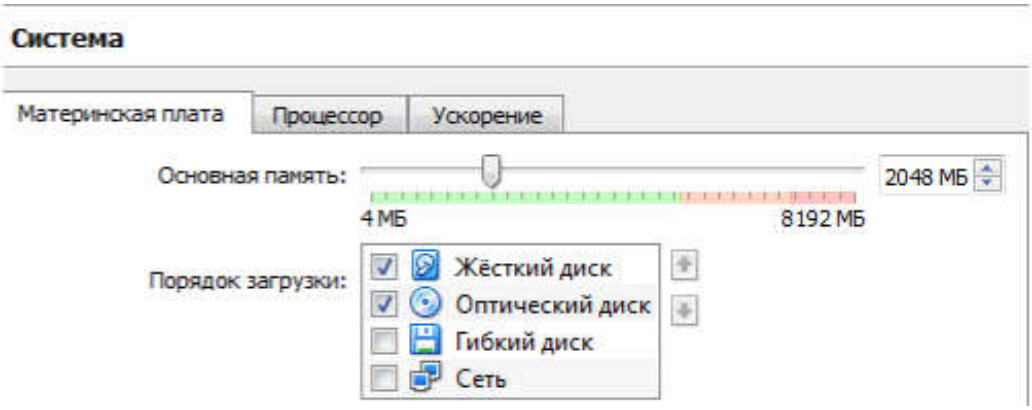


Рисунок 13. Вікно Система для зміни параметрів ВМ

- Вкладка "Система. Процесор" містить інформацію про кількість процесорів, доступних ВМ і деякі режими їх роботи (опис режимів в інтерактивній підказці).
- Вкладка "Система. Прискорення" містить інформацію про апаратну підтримку віртуалізації AMD-V або VT-x.
- Вкладка "Дисплей. Віддалений дисплей" дозволяє включити режим роботи ВМ як серверу віддаленого робочого столу (RDP).
- Вкладка "Носії" відображає образи віртуальних дисків і приводи хоста. Підключимо iso-образ диску Гостьових доповнень (рис. 14).

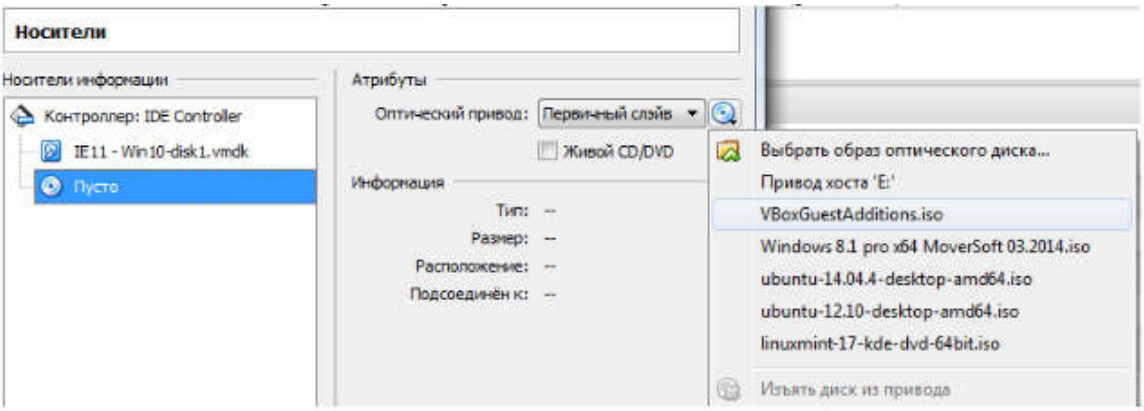


Рисунок 14. Вибір образу диску

- Вкладка "Мережа. Адаптер 1" відображає наступну інформацію:
  - ознака включення мережного адаптера;
  - тип підключення;

  Додатково:
  - "Ім'я" контролера, що використовується.
  - "Нерозбірливий режим" визначає політику режиму даного віртуального мережевого адаптера, якщо він підключений до внутрішньої мережі, віртуального адаптера або мережного мосту.

○ Підключення кабелю.

Налаштовуємо мережу через NAT, не змінюючи додаткові параметри (рис. 15).

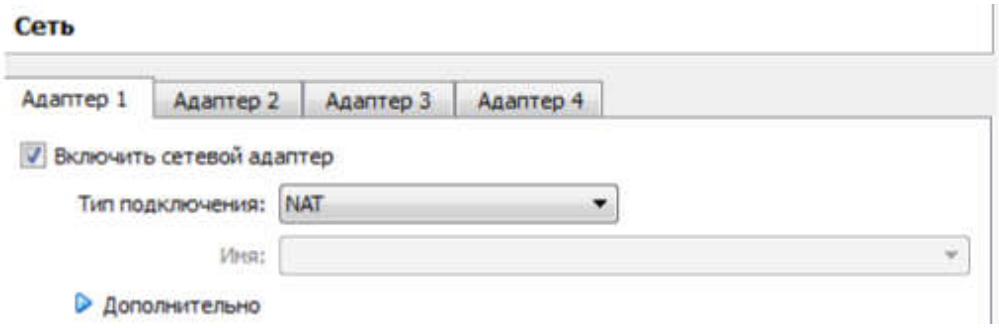


Рисунок 15. Налаштування мережі через NAT

• Вибір "USB" на лівій панелі дозволить підключити USB-пристрої, підключені до хосту.

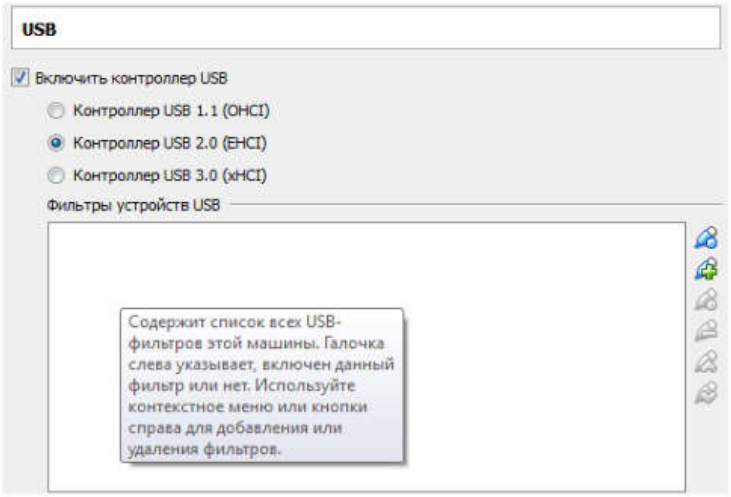


Рисунок 16. Налаштування підключення USB

• Вибір "Спільні папки" на лівій панелі дозволить підключити папки хоста з регульованими параметрами доступу (повний доступ чи тільки для читання) та можливість авто-підключення.

Для створення спільної папки необхідно на цій вкладці обрати папку у файловій системі основної ОС (рис. 16), перезавантажити ВМ, для деяких версій VM VirtualBox у командному рядку гостьової ОС необхідно виконати команду net use x: \\vboxsvr\назва спільної папки.

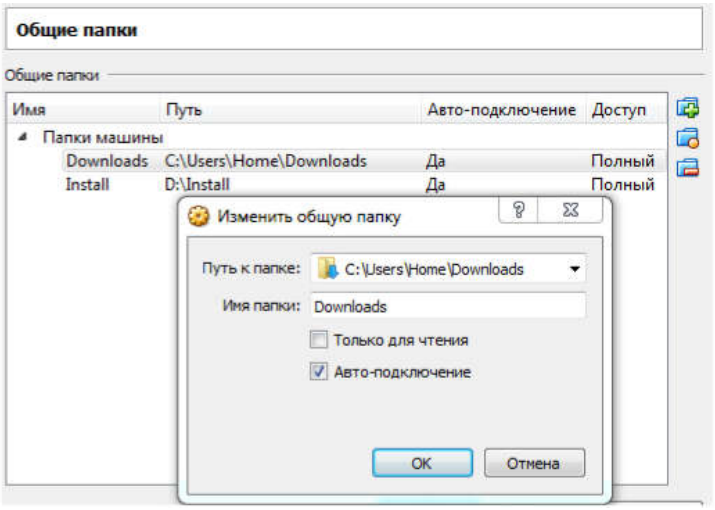


Рисунок 17. Налаштування спільних тек

Зверніть увагу, що відображення спільних тек (рис. 18) може бути доступне тільки після встановлення гостьових доповнень.

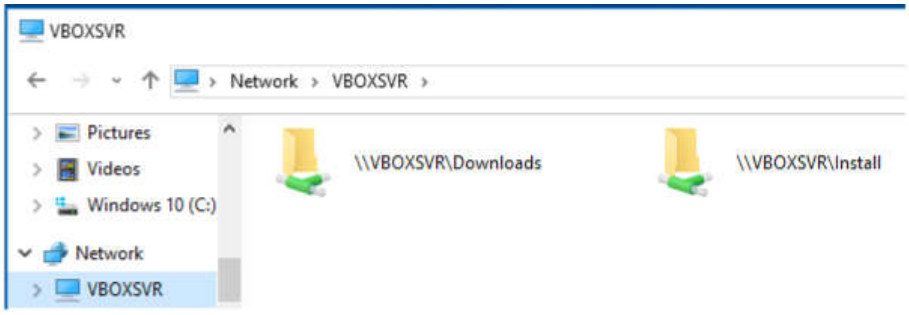


Рисунок 18. Відображення спільних тек у гостьовій ОС Windows 10

Після проведення налаштувань зробіть знімок екрану менеджера ВМ, щоб видно було параметри створеної ВМ (див. приклад оформлення результатів роботи).

5. Встановлення гостьових доповнень

При запущеній гостьовій ОС оберіть Пристрої – Підключити образ Доповнень гостьової ОС (рис. 19).

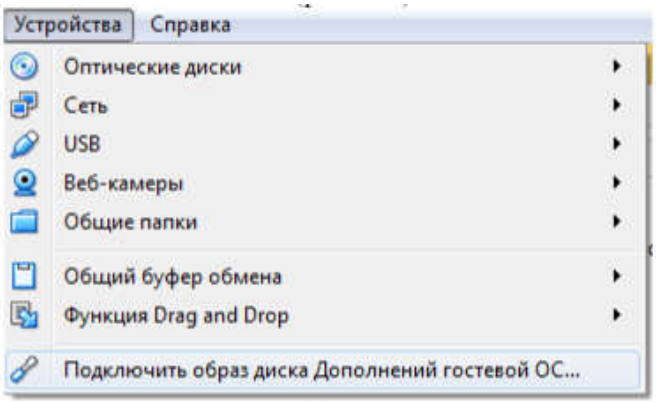


Рисунок 19. Підключення образу Доповнень гостьової ОС

У Провіднику знаходимо відповідний образ та обираємо до

виконання файл, який відповідає архітектурі процесору (рис. 20).

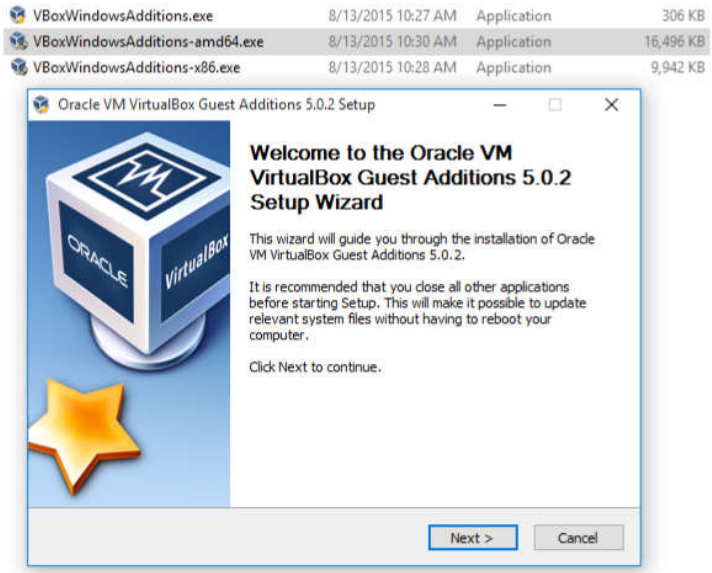


Рисунок 20. Встановлення Доповнень гостьової ОС

Дотримуйтесь інструкції із встановлення. Перезавантажте ОС.

6. Дослідження властивостей ВМ

Перевіримо, як працює ВМ. Результати перевірки записуємо у табл. 1 (див. приклад оформлення результатів). Переглянути параметри основної ОС можна за посиланням «Панель керування – Система і безпека – Система» або «Комп'ютер – Властивості». Переглянути параметри віртуальних ЖД можна у Менеджері віртуальних носіїв («Файл – Менеджер віртуальних носіїв»).

7. Дослідження розміщення файлів ВМ та їх призначення

7.1. Файли віртуального ЖД, файли конфігурації.

Основні файли віртуального середовища знаходяться у каталогах C:\Users\%username%\VirtualBox VMs\VM_name, де «%username%» – ім'я облікового запису користувача в Windows, VM_name – назва ВМ.

Вміст каталогу VM_name (рис. 21):

- VM_name.vbox – файл конфігурації (Virtual Box Machine Definion);
- VM_name.vbox-prev – файл-бекап налаштувань ОС;
- папка Logs із логами роботи ВМ;
- папка Snapshots зі знімками стану ВМ;
- VM_name.vdi – файл віртуального ЖД (Virtual Box Image).

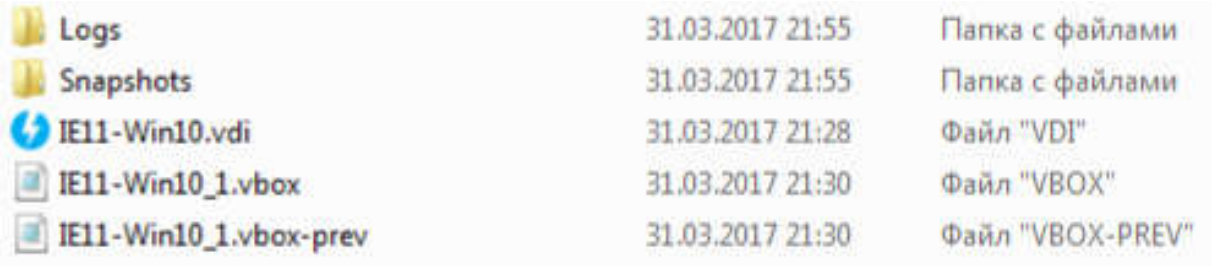


Рисунок 21. Основні файли віртуального середовища ВМ

Файли віртуального ЖД – це віртуальне подання жорсткого диску, яке складається із повного вмісту та структури даних жорсткого диску, що використовується гостьовою ОС віртуальної машини. Файли конфігурації (vbox та .vbox-prev) – файли конфігурації Oracle VM VirtualBox, що містять у собі такі налаштування ВМ як назва машини, тип ОС, кількість системної пам'яті і т.д.

7.2. Робота із знімками стану ВМ.

7.2.1 Створимо знімок стану ВМ. Після створення знімку стану у каталозі Snapshots програма створила два файли із розширеннями .vdi (virtual disk image) та .sav (memory dump). Файл .vdi містить різницю між попереднім та поточним станом файлової системи ВМ.

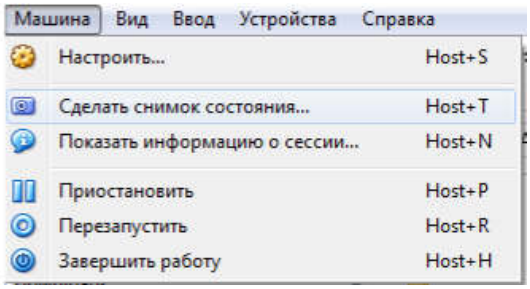


Рисунок 22. Знімок стану ВМ

7.2.2. Внесемо зміни у систему, використавши твік реєстру lab11.reg. Оскільки це зміни у реєстр, отримуємо попередження (рис. 23), яке ігноруємо.

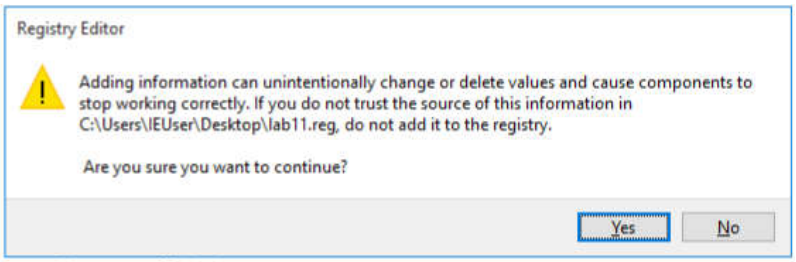


Рисунок 23. Попередження

7.2.3. Перезавантажимося. При завантаженні ОС стандартною оболонкою користувача призначено cmd (рис. 24).

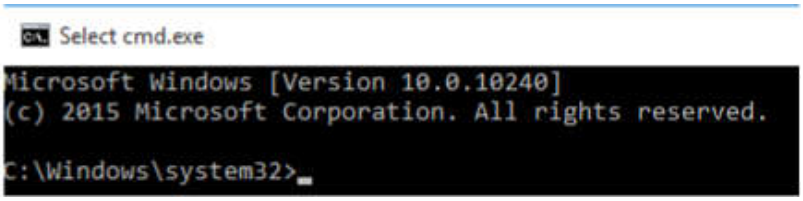


Рисунок 24. Оболонка користувача

7.2.4. Щоб відноситися до останнього робочого знімку, вимкнемо ВМ. Переглянемо доступні знімки ВМ (пункт «Знімки» у менеджері ВМ відповідної віртуальної машини (рис. 25)).

Зі списку доступних знімків ВМ обираємо необхідний знімок (у нас це Знімок 4) – обираємо відновлення знімку.

7.2.5. Виконаємо відновлення із останнього знімку («Восстановить снимок»), зберігати поточний знімок не потрібно.

7.2.6. Після успішного відновлення запускаємо ВМ із відновленого знімку («Запустити»).

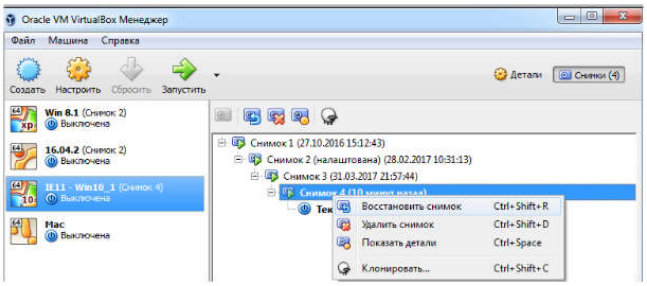


Рисунок 25. Менеджер віртуальних машин

7.3. Збереження та відновлення стану ВМ. При включеній ВМ виконуємо «Файл – Закрити – Зберегти стан машини». Система видасть повідомлення (рис. 26). Обираємо саме «Зберегти стан машини».

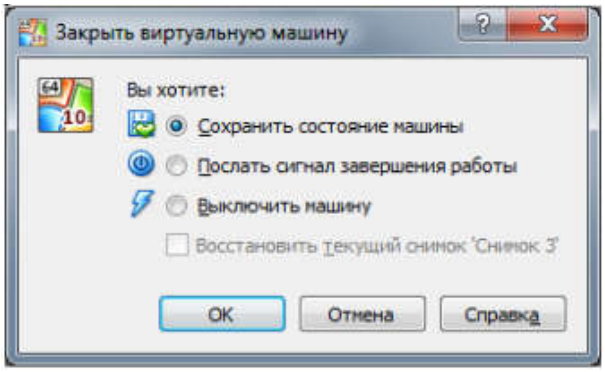


Рисунок 26. Збереження стану ВМ

При збереженні стану ВМ у папці Snapshots змінено файл .vdi та створено новий дамп пам’яті .sav. Стан ВМ у менеджері ВМ позначено як «Збережена» (рис. 27).

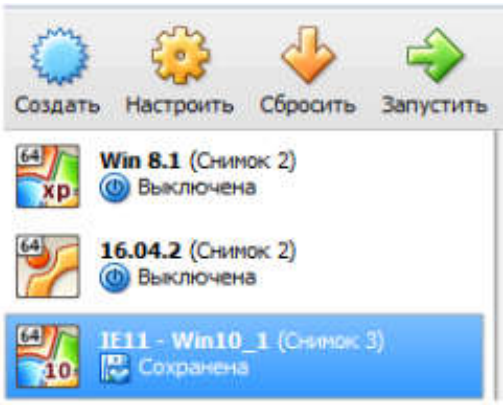


Рисунок 27. Стан ВМ у Менеджері ВМ

При відновленні стану («Запустити») останній файл .sav зникає і робота ВМ відновлюється із того місця, яке було збережено. 7.4. Створення файлу експорту ВМ. Перед експортом конфігурації віртуальну машину необхідно вимкнути. Експорт проводиться у вікні Менеджера ВМ (рис. 28).

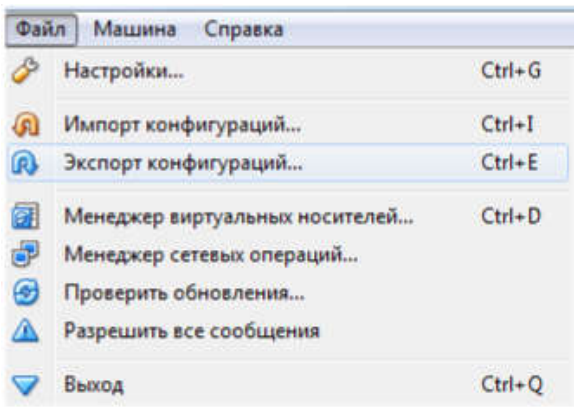


Рисунок 28. Експорту конфігурації ВМ

Обираємо необхідну ВМ (рис. 29) та формат і шлях зберігання файлу (рис. 30).

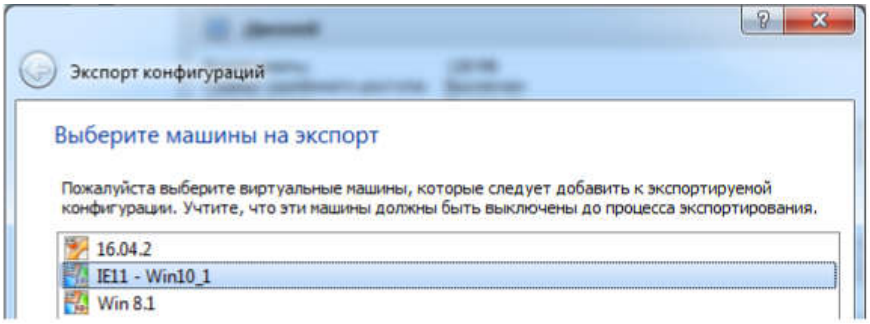


Рисунок 29. Вибір ВМ

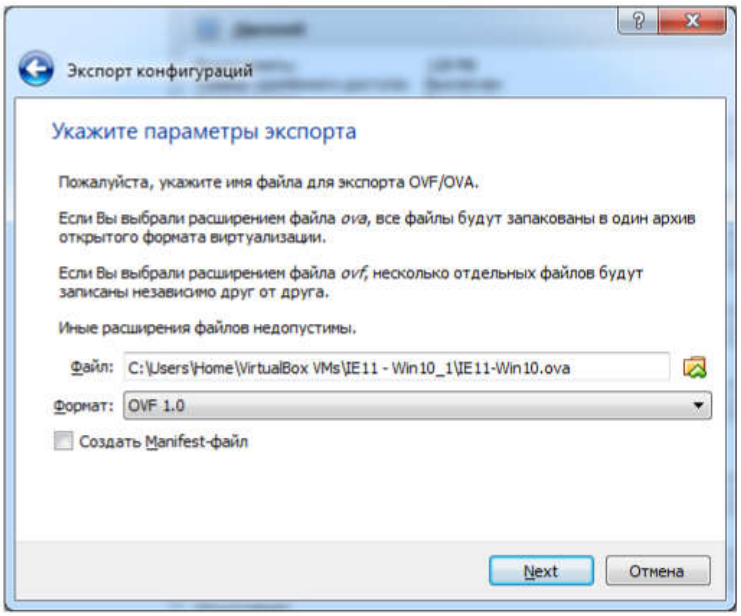


Рисунок 30. Вибір параметрів експорту

За необхідності можна вказати додаткові дані (параметри експорту), які будуть додані до конфігурації. Після підтвердження з'явиться вікно зі станом виконання експорту (рис. 31), який при потребі можна відмінити.

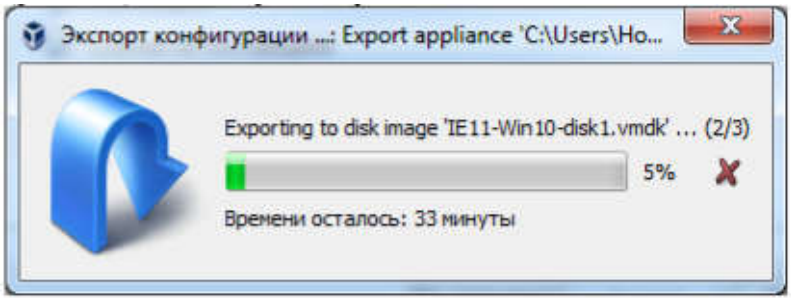


Рисунок 31. Процес експорту конфігурації

Результатом виконання є файл експорту (рис. 32), який можна використовувати для перенесення нашої ВМ на інші комп'ютери.

Рисунок 32. Результуючий файл експорту ВМ

Приклад оформлення результатів

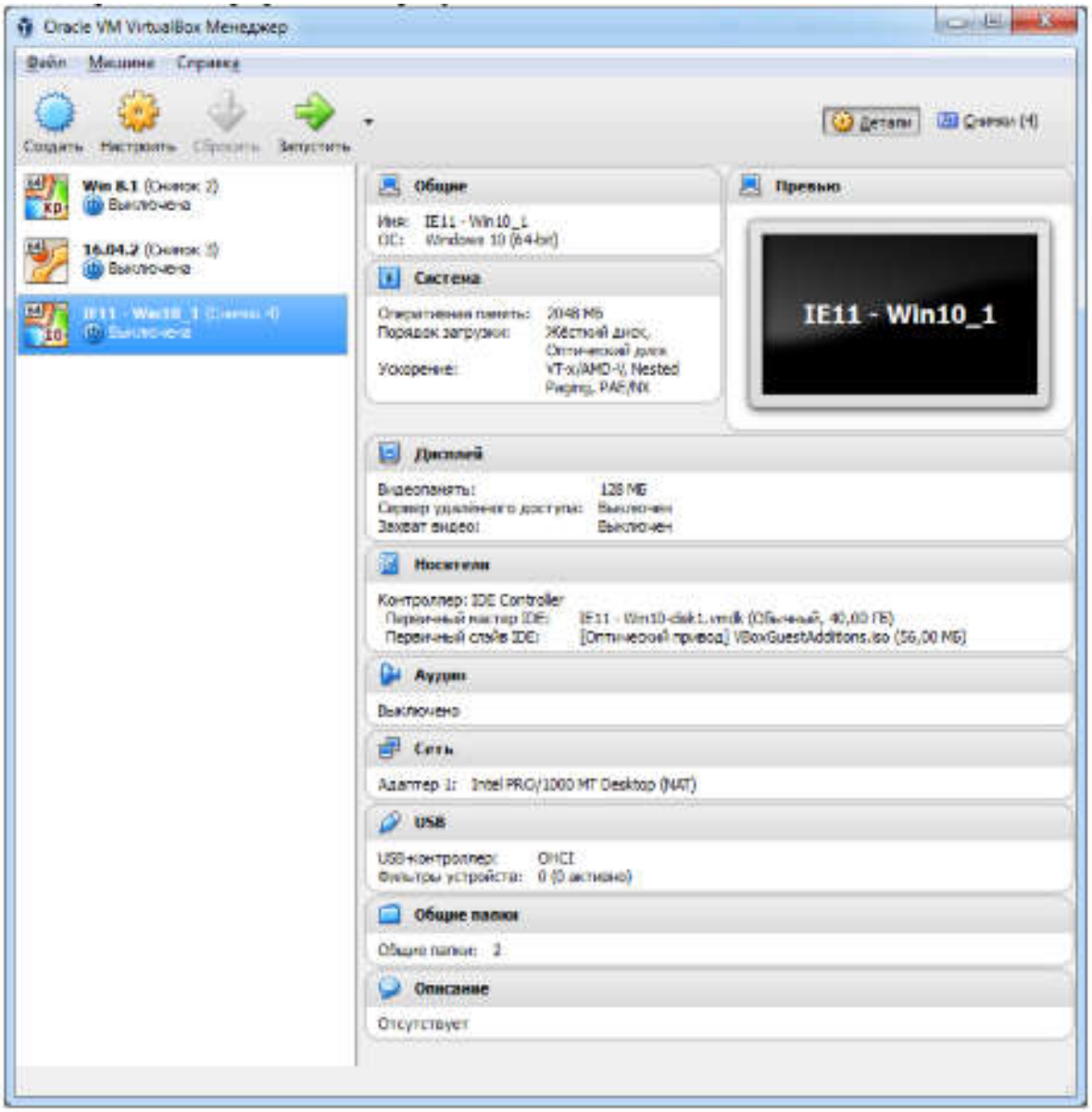
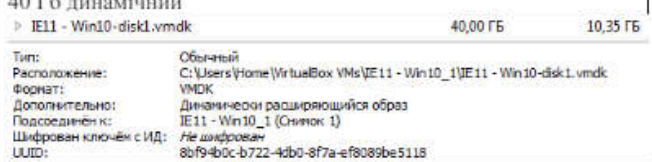


Рисунок 34. Знімок головного екрану ВМ IE11-Win10_1

Таблиця 1. Параметри основної та гостьової ОС

Oracle VM VirtualBox 5.0.2		
параметри	*основна ОС*	*гостьова ОС*
версія ОС	Windows 7 Максимальная 64-разрядная операционная система	Windows 10 64 біт
об'єм ОП	Установленная память (ОЗУ): 8,00 ГБ (7,88 ГБ доступно)	2048 б
об'єм ЖД та тип		40 Гб динамічний 
двонаправ-лений буфер	працює в гостьову із буфером (текстом, скріншотами); з файлами не працює	працює в основну із буфером (текстом, скріншотами); з файлами не працює
drag'n'drop	не працює	не працює
мережа		NAT, працює
USB		працює
спільні теки		працюють

ЗМІСТОВИЙ МОДУЛЬ 3. УПРАВЛІННЯ ПАМ'ЯТТЮ ТА РОБОТА ПРИКЛАДНИХ ПРОГРАМ

Лабораторна робота №9. Принципи функціонування прикладних програм в середовищі ОС Windows.

Мета роботи – отримати досвід застосування системних утиліт для моніторингу використання пам'яті обчислювальної системи та окремих додатків.

- Робота виконується: на основній або віртуальній машині із ОС Windows.
- Використовуються утиліти: консоль Монітор продуктивності (Performance Monitor), Диспетчер завдань (Task Manager), Монітор ресурсів (Recource Monitor).

Теоретичні відомості

Оцінювання використання фізичної та віртуальної пам'яті у Windows

Недостатність пам'яті зазвичай викликається недостатнім обсягом ОЗУ, витоком пам'яті або параметром управління пам'яттю /3GB, який використовується у 32-бітних системах. Якщо параметр /3GB не є джерелом проблеми, використовують відповідні лічильники для діагностики можливого вузького місця, пов'язаного з пам'яттю. Вони подані у табл. 1. Пам'ять\% використання виділеної пам'яті – показує частку використання віртуальної пам'яті.

Пам'ять\Обмін сторінок/сек – це число сторінок, прочитаних із диска або записаних на диск. Ця величина є сумою величин Введення сторінок/сек і Виведення сторінок/сек і включає сторінковий обмін (підкачування) системної кеш-пам'яті для доступу до файлів. Крім того, сюди включається сторінковий обмін для некешованих файлів, які безпосередньо відображаються у пам'ять.

Таблиця 1. Основні лічильники пам'яті

№	Об'єкт \ лічильник рекомендоване граничне значення	Коментар
1.	Memory\% Committed Bytes In Use (Память\ %використання виділеної пам'яті) **80**	якщо це число перевищує 80 відсотків, це вказує на недостатній обсяг пам'яті. Очевидним рішенням у цьому випадку є додавання

		пам'яті.
2.	Paging File \% Usage **75**	надто високе використання файлу підкачки – необхідність збільшення пам’яті, або збільшення файлу підкачки
3.	Memory\Pages/sec (Обмін сторінок/сек) **20**	перевищення порогу обміну сторінок може бути викликане великим стопінгом, тому необхідно збільшити пам’ять
4.	Memory\Input Pages/sec (введення сторінок /сек) **5**	при регулярних помилках - необхідно збільшити пам’ять

Пам'ять\Введення сторінок/сек – це кількість сторінок, зчитаних із диска при вирішенні посилань на сторінки, які відсутні в пам'яті в момент обробки посилання. Помилка сторінки (Page fault) виникає, коли процес посилається на сторінку віртуальної пам'яті, яка не перебуває у робочому наборі процесу в оперативній пам'яті. Якщо сторінка перебуває в кеші, помилка відсутності сторінки може бути оброблена без звернення до диску, якщо цієї сторінки в кеші немає, необхідно виконати її введення.

Пам'ять\Виведення сторінок/сек – це лічильник запису сторінок на диск, який виконується для звільнення місця в оперативній пам'яті. Сторінки записуються на диск тільки в тому випадку, якщо вони були змінені в оперативній пам'яті. Високе значення виведення сторінок може свідчити про нестачу оперативної пам'яті.

Для створення лічильників та оцінювання роботи пам’яті використовується системна утиліта Журнали продуктивності (perfmon.exe).

Запуск нового процесу

Коли створюється новий процес, зазвичай в пам’яті відсутні дані, які з ним пов’язані. Спроба виконання коду такого процесу спричиняє помилку відсутності сторінки (Page fault), яка призводить до блокування процесу, введення необхідної сторінки у оперативну пам'ять, відновлення роботи процесу. Далі, коли процес звертається до даних, знову відбувається Page fault, який спричиняє введення відповідної сторінки. Для роботи процесу потрібна також наявність сторінок стеку, кучі, необхідних динамічних бібліотек. Щоб розмістити ці сторінки в ОП, необхідно виконати їх введення. Тому при запуску нового процесу виникнення Page fault і введення сторінок є нормальними явищами. Показником нестачі пам’яті у цьому випадку є інтенсивне виведення сторінок (витіснення одних сторінок із пам’яті для розміщення інших). В загальному випадку процес обміну сторінок повинен майже повністю складатися із введення сторінок і стабілізуватися після деякого часу від запуску процесу. Якщо введення є постійним і перевищує рекомендоване значення, необхідно

збільшити пам'ять, або проблема може бути у процесі, який ми запустили.

Структура та стан фізичної пам'яті у Windows

Пам'ять, встановлена на комп'ютері під керуванням Windows, починаючи із Windows 7, поділяється на такі категорії (табл.2).

Таблиця 2. Структура фізичної пам'яті

Виділення пам'яті	Опис
Зарезервовано обладнанням Hardware Reserved	Пам'ять, зарезервована для використання BIOS і деякими драйверами інших периферійних пристроїв
Використовується In Use	Пам'ять, яка використовується процесами, драйверами або операційною системою
Змінено Modified	Пам'ять, вміст якої має бути переміщений на диск перед використанням за іншим призначенням
Зарезервовано (Очікується) Standby	Пам'ять, яка містить невикористовувані кешовані дані і код
Вільно Free	Пам'ять, яка не містить ніяких важливих даних. Саме вона буде використовуватися у першу чергу, якщо процесам, драйверам або операційній системі буде потрібно більший обсяг пам'яті

Секція «Зарезервована обладнанням»

Це пам'ять, виділена на потреби підключеного обладнання, яку воно використовує для взаємодії з операційною системою. Зарезервована для обладнання пам'ять заблокована і недоступна Диспетчеру пам'яті.

Обсяг пам'яті, виділеної обладнанню залежить від конкретної конфігурації системи і в деяких випадках може досягати декількох сотень мегабайт. До компонентів, що впливають на обсяг зарезервованої пам'яті, відносяться:

- BIOS;
- компоненти материнської плати – наприклад, вдосконалений програмований контролер переривань введення/виведення (APIC);
- звукові карти та інші пристрої, які здійснюють введення/виведення із відображенням на пам'ять;
- шина PCI Express (PCIe); відеокарти; різні набори мікросхем; флешнакопичувачі.

Деякі користувачі скаржаться, що в їх системах для обладнання зарезервовано ненормально багато пам'яті. Багато хто відзначає, що оновлення версії BIOS дозволяє вирішити цю проблему.

Секція «Використовується»

Це пам'ять, що використовується системою, драйверами і запущеними процесами. Кількість використовуваної пам'яті розраховується, як значення

«Всього» за вирахуванням суми показників «Змінено», «Очікування» і «Вільно».

Секція «Змінено»

Це змінена, але не задіяна пам'ять (сторінки пам’яті, які були змінені і потребують запису на диск). Фактично вона не використовується, але може бути в будь-який момент задіяна, якщо знову знадобиться. Якщо пам'ять не використовується досить давно, дані переносяться у файл підкачки, а пам'ять переходить у категорію «Очікування».

Секція «Очікування»

Це сторінки пам'яті, видалені з робочих наборів, але, як і раніше з ними пов'язані. Іншими словами, категорія «Очікування» – це фактично кеш. Для сторінок пам'яті в цій категорії присвоюється пріоритет від 0 до 7 (максимум). Сторінки, пов'язані з високопріоритетними процесами, отримують максимальний пріоритет. Наприклад, спільно використовувані процеси мають високий пріоритет, тому їх сторінкам присвоюється найвищий пріоритет у категорії «Очікування».

Якщо процесу потрібні дані зі сторінки, яка очікує, Диспетчер пам'яті відразу ж повертає цю сторінку в робочий набір. Проте, всі сторінки в категорії «Очікування» доступні для запису даних від інших процесів: коли процесу потрібна додаткова пам'ять, а вільної пам'яті недостатньо, Диспетчер пам'яті вибирає сторінку, яка очікує, з найменшим пріоритетом, ініціює її та виділяє на запит процесу.

Секція «Вільно»

Сторінки пам'яті, ще не виділені жодному процесу або звільнилися після завершення процесу. У цій секції відображується як ще не задіяна, так і вже звільнена пам'ять, але, насправді, ще не задіяна пам'ять відноситься до іншої категорії – «Нульові сторінки» (Zero Page), яка так називається тому, що ці сторінки ініційовані нульовим значенням і готові для використання. У таблиці 3 даються визначення станів фізичної пам'яті, встановленої на комп'ютері з Windows.

Таблиця 3. Стани фізичної пам’яті

Стан пам'яті	Опис
Доступно Available	Об'єм пам'яті (включаючи зарезервовану і вільну пам'ять), доступний для використання процесами, драйверами і операційною системою.
Кешовано Cached	Об'єм пам'яті (включаючи зарезервовану і змінену пам'ять), що містить кешовані дані і код для швидкого доступу з боку процесів, драйверів і операційної системи.

Всього Total	Обсяг фізичної пам'яті, доступної операційній системі, драйверам пристроїв і процесам (розмір встановленої ОП з вирахуванням пам’яті, що зарезервована обладнанням)
Встановлено Installed	Обсяг фізичної пам'яті, встановленої на комп'ютері.

Моніторинг використання пам’яті окремими процесами

Для оцінювання використання фізичної та віртуальної пам’яті окремими процесами в утилітах Диспетчер задач та Монітор ресурсів застосовуються такі показники (табл. 4).

Таблиця 4. Показники використання пам’яті процесами

Показник	Опис
Робочий набір Working set (WS)	обсяг пам'яті в приватному робочому наборі плюс обсяг пам'яті, що використовується процесом, яку можна використати спільно з іншими процесами
Приватний робочий набір Privite WS	показує конкретний обсяг використовуваної процесом пам'яті, який даний процес не може використати спільно з іншими процесами
Спільний робочий набір Sahred WS	показує обсяг пам'яті, який даний процес може використати спільно з іншими процесами
Виділена пам’ять	обсяг віртуальної пам'яті, що виділена процесу
Вивантажуваний пул Paged pull	обсяг сторінкової віртуальної пам'яті ядра, виділеної ядром або драйверами процесу, яку можна переписати на інший носій, наприклад жорсткий диск
Невивантажуваний пул Non-paged pull	обсяг пам'яті ядра, виділеної ядром або драйверами процесу, яку не можна переписати на інший носій
Помилки сторінок Page faults	число помилок сторінок, що виникли з моменту запуску процесу. Для усунення деяких помилок сторінок потрібно отримати вміст сторінки з диска – жорсткі помилки, а інші можна усунути без звернення до диска – м’які помилки.

Hard page faults/c	Помилки, які потребують читання вмісту сторінки із диску

Завдання на лабораторну роботу

Завдання 1. Моніторинг та оцінка використання фізичної і віртуальної пам’яті через Диспетчер завдань та Монітор ресурсів

1.1. Зробити 2 знімка стану пам’яті за допомогою утиліт Монітор ресурсів та Диспетчер завдань до виконання навантаження на систему. 1.2. Оцінка використання пам’яті через Монітор продуктивності

1.2.1. Створити групу лічильників

Відсоткові значення:

- «Пам'ять\% використання виділеної пам'яті»;
- «Файл підкачки\% використання».

Числові значення:

- Memory\ Pages/sec (Обмін сторінок/сек);
- Memory\Input Pages/sec (введення сторінок /сек).

1.2.2. Запустити завдання, що використовують великий обсяг ресурсів комп'ютера (наприклад, графічний редактор, запуск декількох віртуальних машин).

1.2.3. Оцінити ступінь використання пам’яті під час створення відповідного навантаження (заповнити таблицю 5).

1.2.4. Зробити висновк– порівняти отримані значення із рекомендованими, наведеними у таблиці 1.

1.3. Зробити 2 знімка стану пам’яті за допомогою утиліт Монітор ресурсів та Диспетчер завдань після навантаження.

Таблиця 5. Результати завдання 1

№	Об'єкт \ лічильник рекомендоване граничне значення	Максимальні значення	Тривалість перевищення, с
Навантаження на систему:			
1.	Memory\% Committed Bytes In Use 80		
2.	Paging File \% Usage 75		
3.	Memory\ Pages/sec 20		
4.	Memory\Input Pages/sec 5		

Завдання 2. Моніторинг використання пам’яті процесами через Диспетчер завдань та Монітор ресурсів

2.1. Налаштувати параметри Диспетчера завдань: меню "Вид" – "Вибрати стовпці" – поставити галочки на полях, які відповідають характеристикам пам’яті (п. 1-4, 6-8) табл. 6.

2.2. Налаштувати параметри Монітору ресурсів: Пам'ять – "Вибрати

стовпці" (п. 1-5) табл. 6.

2.3. Запустити два процеси на вибір, які виконують однакові завдання (наприклад, два текстові редактори: Notepad і Notepad++, два браузери із однаковою кількістю вкладок (Opera, Mozilla)).

2.4. Встановити у Моніторі ресурсів фільтрацію за запущеними у попередньому пункті завданнями та зробити скріншот. Зробити скріншот даних цих завдань у Диспетчері завдань. За результатами виконання завдання зробити висновок, який процес споживає більше ресурсів пам’яті.

Таблиця 6. Результатів завдання 2

№	Показник
1.	виділено пам’яті (TM, RM)
2.	робочий набір (TM ,RM)
3.	приватний набір (TM, RM)
4.	спільний набір (TM, RM)
5.	кількість «жорстких» помилок сторінок (RM)
6.	вивантажувальний пул (TM)
7.	невивантажувальний пул (TM)
8.	кількість помилок сторінок (TM)

Питання для самоперевірки

1. Які граничні значення лічильників вказують на можливі негаразди в роботі системи?

2. Чому при запуску нового процесу високе значення помилок відсутності сторінок та введення сторінок є нормальним явищем?

3. Які лічильники використовуються для опису пам’яті у Диспетчері завдань?

4. Як за показниками Диспетчера завдань визначити розмір та ступінь використання файлу підкачки?

5. Які лічильники використовуються для опису пам’яті у Моніторі ресурсів?

6. На які 5 частин поділена фізична пам'ять?

7. 4 стани фізичної пам'яті.

Лабораторна робота № 10. Створення діалогового вікна прикладної програми засобами Windows API

Мета роботи – вивчення основних можливостей командного рядка ОС Windows та утиліт для роботи з протоколом TCP/IP.

- Робота виконується: на основній або віртуальній машині.

- Вивчаються команди Netstat, Ipconfig, Ping, Tracert, Pathping

Теоретичні відомості

Основні поняття протоколу TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) є найпопулярнішим мережевим протоколом, що слугує основою глобальної мережі Інтернет. Запропоновані ним засоби маршрутизації забезпечують максимальну гнучкість функціонування локальних мереж підприємств. У сучасних ОС сімейства Windows TCP/IP встановлюється автоматично.

У мережах TCP/IP кожному клієнту повинна бути призначена відповідна IP-адреса, формат якої залежить від протоколу: в IPv4 IP-адреса має довжину 4 байти, а у версії IPv6 – 16 байт. Крім того, клієнту може знадобитися служба імен або алгоритм розв'язання імен.

DNS (Domain Name System) – ієрархічно розподілена база даних, що містить співставлення доменних імен, зокрема з IP-адресами мережі. DNS дозволяє знаходити комп'ютери і служби зі зрозумілих імен, а також переглядати інші відомості з бази даних.

DHCP (Dynamic Host Configuration Protocol) – протокол динамічної конфігурації вузлів мережі, що забезпечує динамічний розподіл IP-адрес і інших параметрів конфігурації між клієнтами мережі, підтримує безпечну, надійну і просту конфігурацію мережі TCP/IP, перешкоджає виникненню конфліктів IP-адрес і допомагає зберігати використання IP-адрес клієнтів. Протокол DHCP використовує клієнт-серверну модель, у якій сервер DHCP здійснює централізоване управління IP-адресами мережі.

ICMP (Internet Control Message Protocol) – це обов'язковий керуючий протокол у наборі протоколів TCP/IP, який повідомляє про помилки і забезпечує зв'язок між вузлами мережі.

UDP (User Datagram Protocol) – протокол користувацьких датаграм, один із ключових елементів TCP/IP. З UDP комп'ютерні програми можуть посилати повідомлення (у даному випадку названі датаграмами) іншим хостам по IPмережі без необхідності попереднього повідомлення для установки спеціальних каналів передачі або шляхів даних.

NetBIOS використовується для забезпечення можливостей взаємодії станцій у невеликих мережах: призначення станції мережевого імені, за яким вона буде доступна в мережі, пошук станції в мережі за її іменем, під'єднання до ресурсів інших станцій і обміну даними між ними, отримання списку мережевих станцій, які підключені до сегменту мережі, та багато іншого, що визначається терміном "мережева взаємодія". NetBIOS через TCP/IP не підтримує IPv6. Для цілей взаємодії в локальному сегменті мережі без участі опорних серверів

(DNS/WINS) з використанням IPv4/IPv6 розроблено протокол LLMNR.

Службові програми і утиліти TCP/IP забезпечують підключення до різних сучасних мереж. При цьому, щоб використовувати ці утиліти, на комп'ютері має бути встановлена підтримка протоколу TCP/IP. До числа підтримуваних цим протоколом службових команд і утиліт відносяться: Finger, Ping, Ftp, Rcp, Hostname, Rexec, Ipconfig, Route, Lpq, Rsh, Lpr, Tftp, Nbtstat, Tracert, Netstat, Getmac, а також набір команд із приставкою Net. Додаткові відомості про запуск служб TCP/IP з командного рядка знаходяться в розділі Net start.

Деякі команди для роботи з протоколом TCP/IP

Команда **Ipconfig**

Служить для відображення всіх поточних параметрів мережі TCP/IP та оновлення параметрів DHCP і DNS. При виклику команди ipconfig без параметрів виводиться тільки IP-адреса, маска підмережі і основний шлюз для кожного мережевого адаптера.

Синтаксис:

ipconfig [/all] [/renew [адаптер]] [/release [адаптер]] [/flushdns] [/displaydns] [/registerdns] [/showclassid адаптер] [/setclassid адаптер [код_класа]]

Команда **Netstat**

Відображення активних підключень TCP, портів, що прослуховуються комп'ютером, статистики Ethernet, таблиці маршрутизації IP, статистики IPv4 (для протоколів IP, ICMP, TCP і UDP) і IPv6 (для протоколів IPv6, ICMPv6, TCP через IPv6 і UDP через IPv6). Викликана без параметрів, команда netstat відображає підключення TCP.

Синтаксис:

netstat [-a] [-e] [-n] [-o] [-p протокол] [-r] [-s] [інтервал]

Командою netstat можна користуватися для відображення статистики протоколу і поточних TCP/IP-з'єднань. Команда netstat -a виводить відомості про всі підключення, а команда netstat -r відображає таблицю маршрутизації і відомості про активні зв'язки. Команда netstat -o відображає коди процесів, що дозволяє переглянути власника порту для кожного підключення. Команда netstat -e виводить статистику інтерфейсу Ethernet, а команда netstat -s відображає статистику протоколів. При використанні команди netstat -n адреси і номери портів не перетворюються на імена.

Команда **Ping** За допомогою відправлення повідомлень із «ехо-запитом» по протоколу ICMP перевіряє з'єднання на рівні протоколу IP із іншим комп'ютером, що підтримує TCP/IP. Після кожної передачі виводиться відповідне повідомлення з «ехо-відповіддю». Ping – це основна TCP/IP-команда, що використовується для усунення неполадки в з'єднанні, перевірки можливості доступу та дозволу імен. Команда ping, запущена без параметрів, виводить

довідку.

Синтаксис:

ping [-t] [-a] [-n лічильник] [-l розмір] [-f] [-i TTL] [-v тип] [-r лічильник] [-s лічильник] [(-j список_вузлів |-k список_вузлів)] [-w інтервал] [імя_кінцевого_комп'ютера]

Команда **Tracert**

Визначає шлях до точки призначення за допомогою відправки в точку призначення «ехо-повідомлення» (за замовчуванням 3 пакети) протоколу Control Message Protocol (ICMP) з постійним збільшенням значення терміну життя (Time to Live, TTL) на 1. Виведений шлях – це список найближчих інтерфейсів маршрутизаторів, що знаходяться на шляху між вузлом-джерелом і точкою призначення. Ближній інтерфейс представляє собою інтерфейс маршрутизатора, який є найближчим до вузла відправника на шляху. Викликана без параметрів, команда tracert виводить довідку.

Синтаксис:

tracert [-d] [-h максимальне_число_переходів] [-j список_вузлів] [-w інтервал] [ім'я_кінцевого_комп'ютера]

Для збільшення швидкодії команди слід використовувати параметр –d.

Команда **Pathping**

Це одна з найбільш корисних нових команд діагностики TCP/IP. Вона об'єднує функціональність Ping і Tracert. Надає інформацію про латентності (затримки передачі даних) мережі та втрати даних на проміжних вузлах між 66 вихідним пунктом і пунктом призначення. Команда pathping протягом деякого періоду часу відправляє численні повідомлення з «ехо-запитом» кожному маршрутизатору, що знаходиться між вихідним пунктом і пунктом призначення, а потім на підставі пакетів, отриманих від кожного з них, обчислює результати.

Оскільки pathping показує коефіцієнт втрати пакетів для кожного маршрутизатора або зв'язку, можна визначити маршрутизатори або субмережі, що мають проблеми з мережею. Команда pathping виконує еквівалентну команді tracert дію, ідентифікуючи маршрутизатори, що знаходяться на шляху. Потім вона періодично протягом заданого часу обмінюється пакетами з усіма маршрутизаторами і на підставі числа пакетів, отриманих від кожного з них, обробляє статистику. Викликана без параметрів, команда pathping виводить довідку.

Синтаксис:

pathping [-n] [-h максимальне_число_переходів] [-g список_вузлів] [-p період] [-q число_запитів] [-w інтервал] [-T] [-R] [ім'я_кінцевого_комп'ютера]

За замовчуванням дозволяється виконувати не більше 30 стрибків, а стандартний час очікування дорівнює 3 секундам. Період за замовчуванням

дорівнює 250 мілісекундам, а кількість запитів кожного маршрутизатора – 100. Для збільшення швидкодії команди слід використовувати параметр –n

Завдання на лабораторну роботу

1. Використовуючи команду ipconfig, виведіть повну конфігурацію TCP/IP для всіх адаптерів. Завдання потрібно виконати на комп'ютері в аудиторії, де проводиться заняття, та на Вашому персональному (домашньому) комп'ютері, підключеному до Інтернет. Результати виконання завдання відобразити в табл.1.

Таблиця 1. Результати виконання команди ipconfig

Дані	Вміст даних 1	Вміст даних 2
імя вузла TCP/IP		
тип з’єднання		
опис		
NetBIOS		
MAC-адреса мережевої		
плати		
(фізична адреса)		
IP-адреса (IPv4)		
маска підмережі		
шлюз за замовчуванням		

2. Використовуючи команду netstat виконайте Завдання 2 згідно свого варіанту. Завдання дивіться в табл. 2. Номер варіанта визначається порядковим номером у журналі. У звіт із лабораторної роботи розмістіть інформацію про виклик команди та результати її виконання.

3. Для виконання завдання 3 потрібно із таблиці 2 за номером варіанту визначити адресу, з якою буде виконуватися завдання. Необхідно самостійно обрати схожий за тематикою варіанта іноземний сайт.

3.1. За допомогою інтернет-сервісів (наприклад, https://2ip.ua/ua/services/information-service/site-location) визначте реальне місцезнаходження сайтів та їх IP-адреси. Внесіть їх до таблиці результатів.

3.2. Використовуючи команду ping, перевірте з'єднання на рівні протоколу IP із зазначеними сайтами. У звіт із лабораторної роботи розмістіть інформацію про виклик команди та результати її виконання. Проаналізуйте отримані результати, порівнявши параметри з'єднання цими вузлами на рівні протоколу IP (втрати, середній час прийому-передачі).

3.3. Використовуючи команду tracert визначте шлях до точок призначення. У якості точок призначення вкажіть адреси, використані в попередньому завданні. У звіт із лабораторної роботи розмістіть інформацію про виклик команди та результати її виконання. Проаналізуйте дані, отримані в результаті використання команд tracert, та зробіть висновки щодо порівняння шляхів до

точок призначення (кількість стрибків до точки призначення, час очікування відповіді).

3.4. Використовуючи команду pathping отримайте інформацію про латентності мережі та втрати даних на проміжних вузлах між вихідним пунктом і пунктом призначення. У якості пунктів призначення вкажіть адреси, використані в попередньому завданні. У звіт із лабораторної роботи розмістіть інформацію про виклик команди та результати її виконання. Проаналізуйте дані, отримані в результаті використання команд, та зробіть висновки щодо порівняння латентності мережі та втрати даних на проміжних вузлах між вихідним пунктом і пунктами призначення.

Таблиця 2. Варіанти завдань

№	Завдання
1.	Завдання 2. Виведіть всі активні підключення TCP Завдання 3. www.meteoprog.ua
2.	Завдання 2. Виведіть статистики Ethernet Завдання 3. www.pravda.com.ua
3.	Завдання 2. Виведіть активні підключення TCP із відображенням адрес і номерів портів у числовому форматі Завдання 3. www.google.com.ua
4.	Завдання 2. Виведіть активні підключення TCP і PID для кожного підключення. Завдання 3. uk.wikipedia.org
5.	Завдання 2. Виведіть підключення по протоколу TCP Завдання 3. novaposhta.ua
6.	Завдання 2. Виведіть статистику по протоколу UDP Завдання 3. www.youtube.com
7.	Завдання 2. Виведіть статистику по протоколу ICMP Завдання 3. www.facebook.com
8.	Завдання 2. Виведіть статистику по протоколу IP Завдання 3. www.nbuv.gov.ua
9.	Завдання 2. Виведіть статистику по протоколу TCPv6 Завдання 3. mnau.edu.ua
10.	Завдання 2. Виведіть статистику по протоколу UDPv6 Завдання 3. itua.info
11.	Завдання 2. Виведіть статистику по протоколу ICMPv6 Завдання 3. prom.ua
12.	Завдання 2. Виведіть статистику по протоколу IPv6 Завдання 3. fex.net
13.	Завдання 2. Виведіть вміст таблиці маршрутизації IP Завдання 3. www.mon.gov.ua
14.	Завдання 2. Виведіть статистики Ethernet Завдання 3. medical-wiki.in.ua
15.	Завдання 2. Виведіть всі активні підключення TCP Завдання 3. www.netacad.kiev.ua
16.	Завдання 2. Виведіть підключення для протоколу TCP

	Завдання 3. kvartal95.com
17.	Завдання 2. Виведіть статистику по протоколу UDP Завдання 3. ababahalamaha.com.ua
18.	Завдання 2. Виведіть статистику по протоколу ICMP Завдання 3. e-commerce.com.ua
19.	Завдання 2. Виведіть статистику по протоколу IP Завдання 3. privatbank.ua
20.	Завдання 2. Виведіть статистику по протоколу TCPv6 Завдання 3. mega.nz
21.	Завдання 2. Виведіть статистику по протоколу UDPv6 Завдання 3. kancboom.com.ua
22.	Завдання 2. Виведіть статистику по протоколу ICMPv6 Завдання 3. www.ukr.net
23.	Завдання 2. Виведіть статистику по протоколу IPv6 Завдання 3. www.bigmir.net
24.	Завдання 2. Виведіть активні підключення TCP і PID для кожного підключення Завдання 3. www.kyivstar.ua
25.	Завдання 2. Виведіть статистику по протоколу IP Завдання 3. meta.ua

Питання для самоперевірки

1. Що виконує команда Netstat.
2. Що виконує команда Ping.
3. Що виконує команда Pathping.
4. Що виконує команда Tracert.
5. Що виконує команда Ipconfig.
6. Що виконує команда Route.

ЗМІСТОВНИЙ МОДУЛЬ 4. СИСТЕМНЕ ПРОГРАМУВАННЯ ВВЕДЕННЯ/ВИВЕДЕННЯ ТА ФАЙЛОВИХ СИСТЕМ

Лабораторна робота №11-12

. Побудова програми з інтерфейсом у виді піктограми на лінійці задач та принципи написання зберігача екрану для ОС Windows

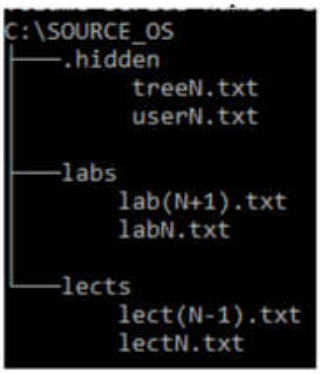
• Робота виконується: на основній або віртуальній машині із ОС Windows 10 на розділі у ФС NTFS.

• Використовуються системні утиліти: командний рядок (cmd), DiskPart та консоль Керування дисками (Disk Management).

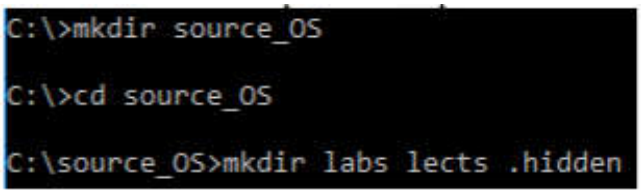
1. Створення об’єктів ФС в командному рядку

1.1. На розділі із файловою системою NTFS створити дерево каталогів source_OS з відповідним вмістом, де N – номер варіанту:

- каталог .hidden – прихований;
- файли labN.txt, lab(N+1).txt наповнити рядками labN, lab(N+1)
- userN.txt наповнити іменем поточного користувача,
- treeN.txt – деревом source_OS.



1.1.1. Створюємо дерево каталогів, скориставшись командою mkdir.



Перевірка:

```
C:\source_OS>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\source_OS

04/02/2017  07:15 AM    <DIR>          .
04/02/2017  07:15 AM    <DIR>          ..
04/02/2017  07:15 AM    <DIR>          .hidden
04/02/2017  07:15 AM    <DIR>          labs
04/02/2017  07:15 AM    <DIR>          lects
               0 File(s)              0 bytes
               5 Dir(s)   9,485,156,352 bytes free
```

1.1.2. Надамо .hidden властивості прихованого каталогу, скориставшись командою attrib.

Для відображення прихованих об’єктів ФС командою dir необхідно використовувати ключ /AH.

```
c:\source_OS>attrib +H .hidden

C:\source_OS>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\source_OS

04/02/2017  07:15 AM    <DIR>          .
04/02/2017  07:15 AM    <DIR>          ..
04/02/2017  07:15 AM    <DIR>          labs
04/02/2017  07:15 AM    <DIR>          lects
               0 File(s)              0 bytes
               4 Dir(s)   9,295,446,016 bytes free

C:\source_OS>dir /AH
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\source_OS

04/02/2017  07:15 AM    <DIR>          .hidden
               0 File(s)              0 bytes
               1 Dir(s)   9,142,206,464 bytes free
```

1.1.3. Створимо файли в labs, наповнивши їх відповідним вмістом:

```
c:\source_OS>echo labN > labs\labN.txt

c:\source_OS>echo lab(N+1) > labs\lab(N+1).txt
```

Перевірка:

- перегляд вмісту каталогу:

```
c:\source_OS>dir labs
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of c:\source_OS\labs

04/21/2017  02:48 AM    <DIR>          .
04/21/2017  02:48 AM    <DIR>          ..
04/21/2017  02:53 AM                12 lab(N+1).txt
04/21/2017  02:52 AM                 8 labN.txt
               2 File(s)             20 bytes
               2 Dir(s)  26,244,009,984 bytes free
```

➢ перегляд вмісту файлів:

```
c:\source_OS>type labN.txt
The system cannot find the file specified.

c:\source_OS>type labs\labN.txt
labN

c:\source_OS>type labs\lab(N+1).txt
lab(N+1)
```

1.1.4. Створимо пусті файли в lects:

```
c:\source_OS>copy con lects\lectN.txt
^Z
        1 file(s) copied.

c:\source_OS>copy con lects\lect(N-1).txt
^Z
        1 file(s) copied.
```

Перевірка:

```
c:\source_OS>dir lects
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of c:\source_OS\lects

04/21/2017  02:58 AM    <DIR>          .
04/21/2017  02:58 AM    <DIR>          ..
04/21/2017  02:58 AM                 0 lect(N-1).txt
04/21/2017  02:57 AM                 0 lectN.txt
               2 File(s)              0 bytes
               2 Dir(s)  26,244,534,272 bytes free
```

1.1.5. Переглянемо дерево каталогу OS, виконавши команду tree:

```
C:\source_OS>tree . /f
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\SOURCE_OS
├───labs
│       lab(N+1)
│       labN
│
└───lects
        lect(N-1)
        lectN
```

1.1.6. Створимо файл .hidden\userN.txt, наповнивши його іменем поточного користувача:

```
C:\source_OS>echo %USERNAME% > .hidden\userN.txt
```

Перевірка:

```
C:\source_OS>type .hidden\userN.txt
IEUser
```

https://technet.microsoft.com/en-gb/library/bb490954.aspx - використання змінних середовища в cmd.

1.1.7. Створимо файл .hidden\treeN.txt, наповнивши його деревом каталогу OS:

```
C:\source_OS>tree /f /a . > .hidden\treeN.txt
```

Перевірка:

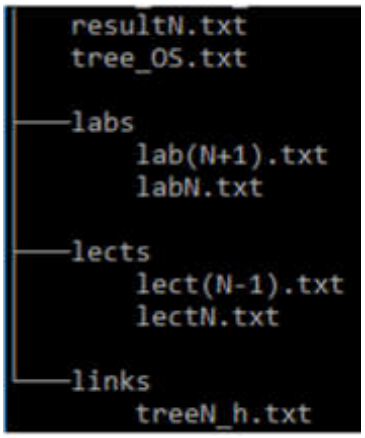
```
C:\source_OS>type .hidden\treeN.txt
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\SOURCE_OS
+---labs
|       lab(N+1).txt
|       labN.txt
|
\---lects
        lect(N-1).txt
        lectN.txt
```

Зауважимо, що ми можемо входити до прихованого каталогу через cmd та переглядати його вміст, якщо знаємо назву.

```
C:\source_OS>cd .hidden

C:\source_OS\.hidden>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\source_OS\.hidden

04/22/2017  09:15 AM               215 treeN.txt
04/22/2017  09:13 AM                 9 userN.txt
               2 File(s)            224 bytes
               0 Dir(s)  25,076,666,368 bytes free
```

1.2. В домашній директорії користувача створити директорію OS з відповідним вмістом:

- labs, lects – посилання (junction та symlinkd) на відповідні каталоги;
- treeN_h.txt – жорстке посилання на файл source_OS/.hidden/treeN.txt,
- tree_OS.txt містить дерево каталогів OS,
- resultN.txt містить результати виконання лабораторної роботи.



1.2.1. Створюємо дерево каталогів OS, скориставшись командою mkdir.

```
C:\>mkdir OS OS\links
```

Перевірка:

```
C:\>tree OS /f
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\OS
└───links
```

1.2.2. Створюємо символічний зв'язок на каталог lects та junction на каталог labs, скориставшись командою mkdir.

```
C:\OS>mklink /d lects ..\source_OS\lects
symbolic link created for lects <<===>> ..\source_OS\lects

C:\OS>mklink /j labs C:\source_OS\labs
Junction created for labs <<===>> C:\source_OS\labs
```

Перевірка:

```
C:\OS>tree /f .
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\OS
├───labs
│       lab(N+1).txt
│       labN.txt
│
├───lects
│       lect(N-1).txt
│       lectN.txt
│
└───links
```

```
C:\OS>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\OS

04/21/2017  04:19 AM    <DIR>          .
04/21/2017  04:19 AM    <DIR>          ..
04/21/2017  04:15 AM    <JUNCTION>     labs [C:\source_OS\labs]
04/21/2017  04:14 AM    <SYMLINKD>     lects [..\source_OS\lects]
04/21/2017  04:11 AM    <DIR>          links
               0 File(s)              0 bytes
               5 Dir(s)  26,440,814,592 bytes free
```

1.2.3. Створюємо файл результату OS\resultN.txt та заповнюємо його наступним:

- вміст файлу source_OS\.hidden\userN.txt,
- № варіанту;
- 2 порожні рядки;
- вміст OS\labs\labN.txt;
- вміст OS\labs\lab(N+1).txt:

```
c:\OS>type ..\source_OS\.hidden\userN.txt > resultN.txt

C:\OS>echo N >> resultN.txt
```

Для додавання двох порожніх рядків використовуємо команду echo:

```
c:\OS>echo. >> resultN.txt && echo. >> resultN.txt
```

Розміщуємо вміст відповідних файлів:

```
C:\OS>type labs\labN.txt >> resultN.txt

C:\OS>type labs\lab(N+1).txt >> resultN.txt
```

Перевірка:

```
IEUser
N


labN
lab(N+1)
```

1.2.4. Створюємо файл links\treeN_h.txt – жорстке посилання на source_OS\.hidden\treeN.txt

```
C:\OS>mklink /h links\treeN_h.txt ..\source_OS\.hidden\ treeN.txt
Hardlink created for links\treeN_h.txt <<===>> ..\source_OS\.hidden\ treeN.txt
```

Перевірка (перегляд жорстких зв’язків файлу):

```
C:\OS>fsutil hardlink list  links\treeN_h.txt
\OS\links\treeN_h.txt
\source_OS\.hidden\ treeN.txt
```

Перегляд вмісту файлу treeN_h.txt:

```
C:\OS>type links\treeN_h.txt
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\SOURCE_OS
+---labs
|       lab(N+1).txt
|       labN.txt
|
\---lects
        lect(N-1).txt
        lectN.txt
```

1.2.5. Оформлюємо файл результату resultN.txt:

- дописуємо 2 порожні рядки;
- дописуємо список файлів, пов’язаних жорстким зв’язком (рез-т п.1.2.4);
- дописуємо 2 порожні рядки.

```
c:\OS>echo. >> resultN.txt && echo. >> resultN.txt
c:\OS>fsutil hardlink list links\treeN_h.txt >> resultN.txt
```

1.2.6. Створюємо файл OS\treeN_OS.txt, перенаправивши у нього результат виведення дерева OS:

```
C:\OS>tree /f /a . > tree_OS.txt
```

Перевірка:

- перегляд дерева:

```
C:\OS>type tree_OS.txt
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\OS
|   resultN.txt
|   tree_OS.txt
|
+---labs
|       lab(N+1).txt
|       labN.txt
|
+---lects
|       lect(N-1).txt
|       lectN.txt
|
\---links
        treeN_h.txt
```

➢ перегляд файлових об’єктів:

```
C:\OS>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\OS

04/21/2017  04:50 AM    <DIR>          .
04/21/2017  04:50 AM    <DIR>          ..
04/21/2017  04:15 AM    <JUNCTION>     labs [C:\source_OS\labs]
04/21/2017  04:14 AM    <SYMLINKD>     lects [..\source_OS\lects]
04/21/2017  04:53 AM    <DIR>          links
04/21/2017  04:47 AM                93 resultN.txt
04/21/2017  04:51 AM               307 tree_OS.txt
               2 File(s)            400 bytes
               5 Dir(s)  26,475,184,128 bytes free
```

1.3. Створити точки майбутнього монтування розділів – каталоги mount\NTFS_N, mount\FAT_N, N – номер варіанту.

1.3.1. Створюємо каталоги, скориставшись командою mkdir:

```
C:\>mkdir mount mount\NTFS_N mount\FAT_N
```

Перевірка:

```
C:\>tree mount
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\MOUNT
├───FAT_N
└───NTFS_N
```

2. Створення, форматування та монтування розділів віртуального жорсткого диску (ВЖД) за допомогою утиліти Diskpart

2.1. Створити ВЖД із MBR об’ємом 1 Гб, розбити його на 2 рівні розділи ємністю 512 Мб.

2.1.1. Запускаємо утиліту Diskpart. Оцінюємо вільне місце на жорстких дисках комп’ютера.

```
DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online           40 GB  1024 KB
```

Команда list disk показує список дисків у системі: маємо 1 диск розміром 40 Гб, який містить 1024 КБ нерозміченого простору, базовий (не має помітки динамічний), із MBR (поле GPT не відмічено). Зверніть увагу, що диски та томи нумеруються із 0, а розділи – із 1.

2.1.2. Створюємо ВЖД, вказавши його місцезнаходження та розмір.

```
DISKPART> create vdisk file="c:\vdisk.vhd" maximum=1024

  100 percent completed

DiskPart successfully created the virtual disk file.
```

Переглянемо інформацію про ВЖД:

```
DISKPART> list vdisk

  VDisk ###  Disk ###  State                Type       File
  ---------  --------  -------------------  ---------  ----
* VDisk 0    Disk ---  Added                Unknown     c:\vdisk.vhd
```

2.1.3. Приєднаємо щойно створений ВЖД:

```
DISKPART> attach vdisk

  100 percent completed

DiskPart successfully attached the virtual disk file.
```

Якщо необхідно приєднати вже існуючий (раніше створений ВЖД), необхідно спочатку його вибрати командою select, потім виконати attach.

```
DISKPART> select vdisk file="c:\vdisk.vhd"

DiskPart successfully selected the virtual disk file.

DISKPART> attach vdisk

  100 percent completed

DiskPart successfully attached the virtual disk file.
```

Перевірка: ВЖД з’являється в списку дисків, він не розмічений (Free=maximum):

```
DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online           40 GB  1024 KB
* Disk 1    Online         1024 MB  1024 MB
```

2.1.4. Створимо на ВЖД розмітку MBR

```
DISKPART> convert mbr

DiskPart successfully converted the selected disk to MBR format.
```

2.2. Відформатувати перший первинний розділ у NTFS (розмір кластеру за замовчуванням, форматування швидке, мітка NTFS_N), другий – FAT32 (розмір кластеру 2048 Б, , мітка FAT_N).

2.2.1. Створимо єдиний первинний розділ на ВЖД:

```
DISKPART> create partition primary

DiskPart succeeded in creating the specified partition.
```

Перевірка: розділ з’явився у списку томів, він не має ФС (Raw):

```
DISKPART> list volume

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
  Volume 0     D   VBOXADDITIO  CDFS   CD-ROM        55 MB  Healthy
  Volume 1     C   Windows 10   NTFS   Partition     39 GB  Healthy    System
* Volume 2                      RAW    Partition   1022 MB  Healthy
```

Розділ, який має фокус, позначено зірочкою. 2.2.2. Проведемо його швидке форматування (QUICK) у ФС NTFS з міткою NTFS_N, де N – № варіанту/

```
DISKPART> format FS=NTFS label=NTFS_N QUICK

  100 percent completed

DiskPart successfully formatted the volume.
```

Перевірка: ВЖД розмічений (Free=0), базовий (не динамічний), із MBR (не містить помітки GPT):

```
DISKPART> list disk

  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    Online           40 GB  1024 KB
* Disk 1    Online         1024 MB      0 B
```

Розділ – первинний:

```
DISKPART> list partition

  Partition ###  Type              Size     Offset
  -------------  ----------------  -------  -------
* Partition 1    Primary           1022 MB    64 KB
```

Том має відповідну мітку та ФС:

```
DISKPART> list volume

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
  Volume 0     D   VBOXADDITIO  CDFS   CD-ROM        55 MB  Healthy
  Volume 1     C   Windows 10   NTFS   Partition     39 GB  Healthy    System
* Volume 2         NTFS_N       NTFS   Partition   1022 MB  Healthy
```

Перегляд властивостей ВЖД:

```
DISKPART> detail vdisk

Device type ID: 2 (VHD)
Vendor ID: {EC984AEC-A0F9-47E9-901F-71415A66345B} (Microsoft Corporation)
State: Attached not open
Virtual size: 1024 MB
Physical size: 1024 MB
Filename: c:\vdisk.vhd
Is Child: No
Parent Filename:
Associated disk#: 1
```

Перегляд властивостей ФС (ФС – NTFS, розмір кластеру – 4096; можливе форматування в інші ФС):

```
DISKPART> filesystem

Current File System

  Type                : NTFS
  Allocation Unit Size : 4096
  Flags : 00000000

File Systems Supported for Formatting

  Type                : NTFS (Default)
  Allocation Unit Sizes: 512, 1024, 2048, 4096 (Default), 8192, 16K, 32K, 64K

  Type                : FAT
  Allocation Unit Sizes: 16K (Default), 32K, 64K

  Type                : FAT32
  Allocation Unit Sizes: 512, 1024, 2048, 4096 (Default), 8192
```

2.2.3. Створення другого первинного розділу та форматування у FAT32

2.2.3.1. Стискаємо том, звільнюючи місце під новий розділ розміром 512 МБ (вказуємо у команді), та створюємо новий розділ:

```
DISKPART> shrink desired 512

DiskPart successfully shrunk the volume by:  512 MB
```

```
DISKPART> create partition primary

DiskPart succeeded in creating the specified partition.
```

2.2.3.2. Форматуємо створений розділ у ФС FAT32, мітка FAT_N, розмір кластеру 2048Б:

```
DISKPART> format FS=FAT32 label=FAT_N Unit=2048

  100 percent completed

DiskPart successfully formatted the volume.
```

Перевірка:

```
DISKPART> list volume

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
  Volume 0     D   VBOXADDITIO  CDFS   CD-ROM       55 MB  Healthy
  Volume 1     C   Windows 10   NTFS   Partition    39 GB  Healthy    System
  Volume 2         NTFS_N       NTFS   Partition   509 MB  Healthy
  Volume 3         FAT_N        FAT32  Partition   511 MB  Healthy
```

```
DISKPART> filesystem

Current File System

  Type                 : FAT32
  Allocation Unit Size : 2048
  Flags : 00000000

File Systems Supported for Formatting

  Type                 : NTFS (Recommended)
  Allocation Unit Sizes: 512, 1024, 2048, 4096 (Default), 8192, 16K, 32K, 64K

  Type                 : FAT
  Allocation Unit Sizes: 8192 (Default), 16K, 32K, 64K

  Type                 : FAT32 (Default)
  Allocation Unit Sizes: 512, 1024, 2048 (Default), 4096
```

2.3. Монтування томів ВЖД до літер, каталогів-точок монтування.

2.3.1. Підмонтуємо тома ВЖД до літер (FAT32 – F, NTFS – N):

```
DISKPART> assign letter=F

DiskPart successfully assigned the drive letter or mount point.

DISKPART> select partition 1

Partition 1 is now the selected partition.

DISKPART> assign letter=N

DiskPart successfully assigned the drive letter or mount point.
```

Відмітимо, що операція монтування застосовується до тому, який

має фокус: спочатку монтується том FAT_N, потім обираємо попередній розділ і монтуємо його.

2.3.2. Підмонтуємо тома ВЖД до раніше створених відповідних каталогів монтування:

```
DISKPART> assign mount=C:\mount\NTFS_N

DiskPart successfully assigned the drive letter or mount point.

DISKPART> select partition 2

Partition 2 is now the selected partition.

DISKPART> assign mount=C:\mount\FAT_N

DiskPart successfully assigned the drive letter or mount point.
```

Стежимо за тим, який розділ має фокус.

Перевірка:

```
DISKPART> list volume

  Volume ###  Ltr  Label        Fs     Type        Size     Status     Info
  ----------  ---  -----------  -----  ----------  -------  ---------  --------
  Volume 0     D   VBOXADDITIO  CDFS   CD-ROM       55 MB  Healthy
  Volume 1     C   Windows 10   NTFS   Partition    39 GB  Healthy    System
  Volume 2     N   NTFS_N       NTFS   Partition   509 MB  Healthy
    C:\mount\NTFS_N\
* Volume 3     F   FAT_N        FAT32  Partition   511 MB  Healthy
    C:\mount\FAT_N\
```

2.3.3. Видаляємо літерні позначення розділів:

```
DISKPART> remove letter=F:

DiskPart successfully removed the drive letter or mount point.
```

```
DISKPART> select partition 1

Partition 1 is now the selected partition.

DISKPART> remove letter=N:

DiskPart successfully removed the drive letter or mount point.
```

Повинні залишитися розділи, підмонтовані до каталогів.

2.3.4. Командою exit вийдемо із Diskpart.

2.4. Запустивши команний рядок від імені адміністратора, налаштуємо перенаправлення виведення списку томів утиліти Diskpart у файл результатів, попередньо додавший до нього два порожні рядки:

```
C:\mount\NTFS_N\OS>echo. >>resultN.txt && echo. >> resultN.txt
```

```
c:\>diskpart >> OS\resultN.txt
list volume
^Z
```

Перевірка: файл результатів повинен містити запис:

Microsoft DiskPart version 10.0.10240

Copyright (C) 1999-2013 Microsoft Corporation.

On computer: IE11WIN10

DISKPART>

Volume ### Ltr Label Fs Type Size Status Info

--- --- --- --- --- --- --- ---

Volume 0 D VBOXADDITIO CDFS CD-ROM 55 MB Healthy

Volume 1 C Windows 10 NTFS Partition 39 GB Healthy System

Volume 2 NTFS_N NTFS Partition 509 MB Healthy

C:\mount\NTFS_N\

Volume 3 FAT_N FAT32 Partition 511 MB Healthy

C:\mount\FAT_N\

3. Переміщення об’єктів ФС на монтовані розділи та перевірка властивостей зв’язків в командному рядку

3.1. Перемістити каталог OS на щойно створений розділ NTFS_N.

3.1.1. Переглянемо властивості каталогів, куди відбулося монтування. Відмітимо, що вони мають тип junction і містять посилання на унікальний ідентифікатор розділу (UID – рядок у фігурних дужках).

```
c:\>cd mount

c:\mount>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of c:\mount

04/21/2017  04:55 AM    <DIR>          .
04/21/2017  04:55 AM    <DIR>          ..
04/21/2017  04:55 AM    <JUNCTION>     FAT_N [\??\Volume{bed36d47-0000-0000-0000-f01f00000000}\]
04/21/2017  04:55 AM    <JUNCTION>     NTFS_N [\??\Volume{bed36d47-0000-0000-0000-010000000000}\]
               0 File(s)              0 bytes
               4 Dir(s)  25,084,305,408 bytes free
```

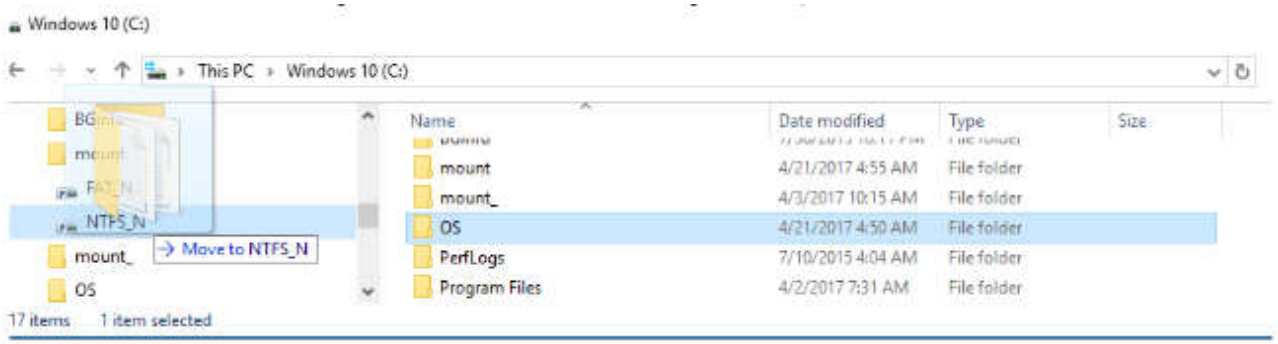
3.1.2. Виконаємо команду mountvol без параметрів:

```
c:\mount>mountvol
Creates, deletes, or lists a volume mount point.
```

Її результатом є виведення довідки про команду та відображення розділів, ідентифікованих за UID, із відповідними точками монтування:

```
Possible values for VolumeName along with current mount points are:

    \\?\Volume{81b4ec29-0000-0000-0000-100000000000}\
        C:\

    \\?\Volume{bed36d47-0000-0000-0000-010000000000}\
        C:\mount\NTFS_N\

    \\?\Volume{bed36d47-0000-0000-0000-f01f00000000}\
        C:\mount\FAT_N\

    \\?\Volume{70b132b0-378d-11e5-9bc2-806e6f6e6963}\
        D:\
```

3.1.3. Перемістимо каталог OS на розділ NTFS_N (у Провіднику, оскільки командний рядок не дозволяє це зробити):



3.2. Відновлення зв’язків (junction, symlinkd) між каталогами.

Структура ФС на монтованому розділі у каталозі OS повинна мати вигляд:

```
<JUNCTION>     labs [\\?\Volume{81b4ec29-0000-0000-0000-100000000000}\source_OS\labs]
<SYMLINKD>     lects [C:\source_OS\lects]
<DIR>          links
         1,852 resultN.txt
           262 tree_OS
```

3.2.1. Перевіряємо властивості посилань labs, lects – вони втрачені, зараз це просто каталоги, без зв’язку із відповідними каталогами source_OS

```
C:\mount\NTFS_N\OS>dir
 Volume in drive C is Windows 10
 Volume Serial Number is 1A20-A6C9

 Directory of C:\mount\NTFS_N\OS

04/22/2017  06:51 AM    <DIR>          .
04/22/2017  06:51 AM    <DIR>          ..
04/21/2017  02:48 AM    <DIR>          labs
04/21/2017  02:58 AM    <DIR>          lects
04/22/2017  06:51 AM    <DIR>          links
04/22/2017  06:36 AM             1,380 resultN.txt
04/21/2017  04:51 AM               307 tree_OS.txt
               2 File(s)          1,687 bytes
               5 Dir(s)     517,115,904 bytes free
```

```
C:\mount\NTFS_N\OS>tree /f
Folder PATH listing for volume Windows 10
Volume serial number is 000000EF 1A20:A6C9
C:.
    resultN.txt
    tree_OS.txt

├───labs
├───lects
└───links
        treeN_h.txt
```

3.2.2. Видалимо «зламані» зв’язки labs, lects.

```
C:\mount\NTFS_N\OS>rmdir labs lects
```

3.2.3. Створимо новий зв’язок labs, скориставшись командами mountvol та mklink: для створення junction labs використаємо UID розділу, який дізнаємося із mountvol. Це дасть нам змогу, навіть при зміні літерного позначення розділу, отримати доступ до його даних за UID.

```
C:\mount\NTFS_N\OS>mountvol /

    \\?\Volume{81b4ec29-0000-0000-0000-100000000000}\
        C:\
```

```
C:\mount\NTFS_N\OS>mklink /j labs \\?\Volume{81b4ec29-0000-0000-0000-100000000000}\source_OS\labs
Junction created for labs <<===>> \\?\Volume{81b4ec29-0000-0000-0000-100000000000}\source_OS\labs
```

Зауважимо, що для створення такого типу зв’язку можна

скористатися командою mountvol і монтувати каталог \\?\Volume{81b4ec29-0000-0000-0000-100000000000}\source_OS\labs в попередньо створений каталог C:\mount\NTFS_N\OS\labs.

Результат виконання команди буде той же, що і в попередньому пункті, різниця – для виконання mountvol папка монтування (labs) повинна існувати, а mklink створює відповідну папку і видає помилку при її наявності.

3.2.4. Створимо новий зв’язок lects, скориставшись командою mklink і абсолютним шляхом, який включає літерне позначення розділу. Зазначимо, що при монтуванні розділу на іншу літеру чи у каталог, доступ до його даних буде втрачено.

```
C:\mount\NTFS_N\OS>mklink /d lects c:\source_OS\lects
symbolic link created for lects <<===>> c:\source_OS\lects
```

Перевірка:

```
C:\mount\NTFS_N\OS>tree /f .
Folder PATH listing for volume Windows 10
Volume serial number is 1A20-A6C9
C:\MOUNT\NTFS_N\OS
    resultN.txt
    tree_OS.txt

├───labs
        lab(N+1).txt
        labN.txt

├───lects
        lect(N-1).txt
        lectN.txt

└───links
        treeN_h.txt
```

3.3. Перевірка наявності жорсткого зв’язку між файлами.

3.3.1. Файл OS\links\treeN_h.txt був пов'язаний жорстким зв’язком із файлом source_OS\.hidden\treeN.txt, який містив дерево каталогів source_OS. Перевіримо, чи зберігся зв'язок при переміщенні файлів на інший розділ. Оскільки дані файли належать різним розділам і різним ФС – зв'язок втрачено.

```
C:\mount\NTFS_N\OS>fsutil hardlink list links\treeN_h.txt
\OS\links\treeN_h.txt
```

3.3.2. Додамо до файлу OS\links\treeN_h.txt ім’я свого користувача:

```
echo %USERNAME% >> links\resultN.txt
```

3.4. Оформлення файлу результатів 3.4.1. Розмістимо у файл результатів виведення команди dir каталогу OS, попередньо додавши до нього два порожні рядки:

```
C:\mount\NTFS_N\OS>echo. >>resultN.txt && echo. >> resultN.txt
```

```
C:\mount\NTFS_N\OS>dir >> resultN.txt
```

3.4.2. Проведемо порівняння файлів OS\links\treeN_h.txt та source_OS\.hidden\treeN.txt, результат якого запишемо у файл результатів, попередньо додавши два порожні рядки:

```
C:\mount\NTFS_N\OS>echo. >>resultN.txt && echo. >> resultN.txt
C:\mount\NTFS_N\OS>fc links\treeN_h.txt c:\source_OS\.hidden\treeN.txt >>resultN.txt
```

Приклад оформлення результатів Файл результату resultN.txt (utf-8, Windows) має складатися із 6 структурних елементів, які відокремлені двома порожніми рядками:

1) 3 елементи – завдання 1,

2) 1 елемент – завдання 2,

3) 2 елементи – завдання 3.

ЗМІСТОВНИЙ МОДУЛЬ 5. ТЕНДЕНЦІЇ РОЗВИТКУ СУЧАСНИХ ОПЕРАЦІЙНИХ СИСТЕМ ТА НИЗЬКОРІВНЕВЕ ПРОГРАМУВАННЯ

Лабораторна робота №13.

Отримання інформації про ОС та встановлене обладнання

Мета роботи – одержання практичних навичок роботи з утилітою Autoruns, консоллю Перегляд подій, редактором реєстру regedit для налаштовування автоматичного запуску додатків та сервісів при завантаженні ОС Windows.

- Робота виконується на основній або віртуальній машині із ОС Windows.
- Використовується програма Autoruns, системні утиліти: Редактор реєстру (regedit.exe), консоль Перегляд подій (eventvwr.exe).

Теоретичні відомості

Завантаження ОС Windows та автоматичний запуск додатків Завантаження операційної Windows системи є складним процесом і складається із декількох етапів. Розглянемо його на прикладі BIOS, MBR для ОС Windows 7.

1. Виконання коду BIOS та MBR (UEFI,GPT) для тестування обладнання, визначення пристрою завантаження, визначення активного розділу, запуск менеджера завантаження Windows.

2. Менеджер завантаження (файл bootmgr.exe) зчитує дані конфігурації системи, які зберігаються у файлі BCD (Boot Configuration Data). При наявності декількох записів у файлі BCD буде відображено меню вибору операційної системи. Файл BCD знаходиться у папці Boot активного розділу.

3. Після вибору системи запускаються модуль завантаження операційної системи Winload.exe, компоненти ядра Ntoskrnl.exe і Hal.dll, системні служби і інші компоненти – цей етап супроводжується виведенням анімованого екрану з логотипом Windows.

4. Завантажується процес winlogon.exe, який керує входом користувачів в систему. Якщо на комп'ютері є всього один обліковий запис, не захищений паролем, вхід буде виконаний автоматично. В іншому випадку система буде очікувати вибору імені користувача і введення пароля.

5. У процесі входу в систему запускаються елементи автозавантаження, які прописані в реєстрі Windows і папці Автозавантаження. Завантаження ОС Windows триває до повного завантаження робочого столу, тобто до припинення активності процесів, що беруть участь у завантаженні. Для налаштування автоматичного запуску (автозавантаження) додатків, фонових сервісів і скриптів при завантаженні ОС Windows (крок 5), як правило, використовуються:

- розділи реєстру Run;
- папка «Автозавантаження»;

- сервіси операційної системи;
- завдання планувальника і скрипти групової політики, що виконуються при вході користувача в систему.

Програми з перших трьох пунктів цього списку можна побачити в утиліті msconfig.exe. Зауважимо, що, починаючи з Windows 8.1, частина функцій msconfig.exe для швидкого управління програмами в автозавантаженні винесено в Диспетчер завдань.

Для гнучкого налаштування запуску сервісів системи можна скористатися консоллю services.msc. Для керування завданнями планувальника та скриптами групової політики призначені консолі taskschd.msc та gpedit.msc відповідно. Найбільш повну картину автозавантаження дає утиліта Autoruns.

Можливості утиліти Autoruns

Для того, щоб дізнатися про всі процеси, які були запущені разом з вашою системою, вам допоможе утиліта Autoruns від Sysinternals, останню версію якої можна знайти за адресою https://technet.microsoft.com/uk-ua/sysinternals.

Програма Autoruns від Марка Русиновича і Брайса Когсуелла допомагає перевіряти максимальну кількість розміщень автозапуску на наявність програм, налаштованих на запуск в процесі завантаження або входу в систему, на відміну від будь-яких інших програм моніторингу автозапуску. Ця програма абсолютно безкоштовна і до однієї з її переваг можна віднести те, що всі програми відображаються у тому порядку, в якому операційна система обробляє їх. Із даною програмою можна працювати як під 32-розрядними, так і під 64-розрядними операційними системами Windows.

Налаштування запуску програм автоматично при старті ОС

Налаштувати запуск програм автоматично при старті ОС можна декількома способами.

1. Запуск додатків при вході користувача в систему

Якщо програма може бути запущена при вході користувача в систему, вона відобразиться на вкладці «Logon» в Autoruns. При цьому є наступні варіанти розміщення програми в автозапуску: 1.1. Запис про її автозапуск прописаний у реєстрі:

- HKCU\Software\Microsoft\ CurrentVersion\Run;
- HKCU\Software\Microsoft\CurrentVersion\RunOnce;
- HKLM\Software\Microsoft\CurrentVersion\Run;
- HKLM\Software\Microsoft\Windows\CurrentVersion\ RunOnce.

Додатково (цих розділів може не бути у вас в реєстрі), подивіться наступні місця:

- HKLMSoftware\Wow6432Node\Microsoft\Windows\CurrentVersion\Run;
- HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

Once;

- HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run;
- HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run.

У відповідній гілці реєстру Windows створюється запис про програму, що завантажується. В якості назви він має назву програми, в якості значення – запис шляху до виконуваного файлу з опціями. При «вимкненні» програми із автозавантаження через Autoruns у відповідній гілці Autoruns створює папку AutorunsDisabled, куди переміщує запис про відповідну програму. Таким чином, при потребі можливе швидке відновлення програми у автозапуску.

Якщо виконати «видалення» відповідного запису, то він назавжди видалиться із реєстру. Для налаштування автозавантаження додатків у ОС Windows традиційно використовується утиліта Конфігурація системи (msconfig.exe). Починаючи з Windows 8.1 частина функцій msconfig.exe для швидкого управління програмами в автозавантаженні винесено в Диспетчер завдань (рис. 1).

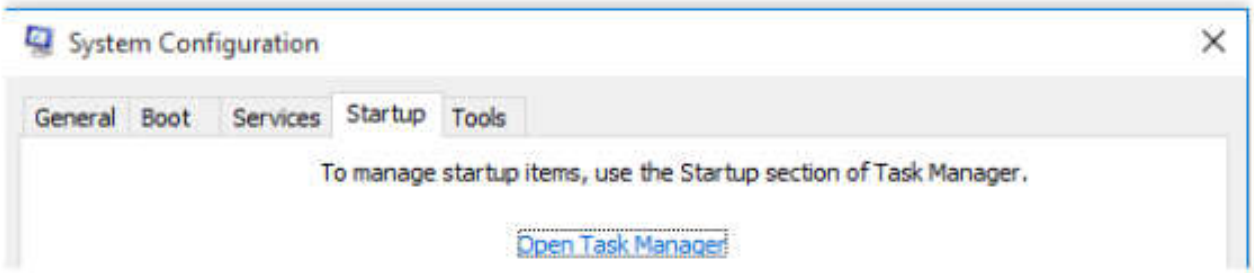


Рис. 1а. Вкладка Автозавантаження утиліти msconfig

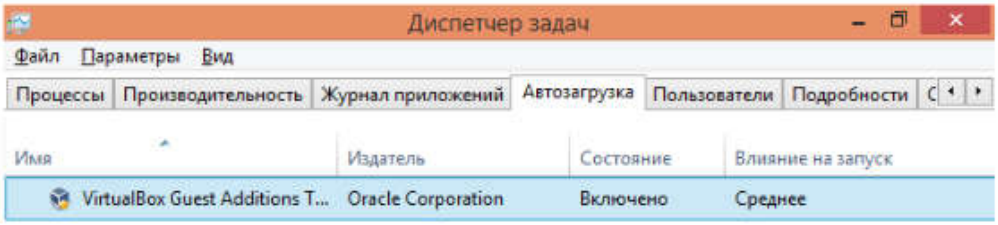


Рис. 1б. Вкладка Автозавантаження Диспетчера завдань Windows 8.1

1.2. Для автозапуску програми при завантаженні ОС в одній із папок Startup в файловій системі можна розмістити її ярлик:

- «%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup»;
- «C:\ProgramData\Microsoft\Windows\StartMenu\Programs\Startup» (щоб потрапити в цю папку, можна ввести у вікно «Виконати» системне посилання на папку автозавантаження shell:startup).

При «вимкненні» програми із автозавантаження через Autoruns програма створює у відповідній гілці папку AutorunsDisabled, куди переміщує запис про

відповідну програму. Таким чином, при потребі можливе швидке відновлення програми у автозапуску. Якщо виконати «видалення» відповідного запису, то він назавжди видалиться із Startup.

2. Запуск додатків через Планувальник завдань

Крім додатків, що запускаються автоматично з операційною системою, ви можете переглянути всі завдання, що призначені на запуск Планувальником завдань. Якщо включити записи Microsoft і Windows, то виявиться, що в системі заплановано дуже багато завдань. Лише частина із них має відношення до процесу завантаження ОС. Завдання, для яких вказані тригери «При запуску» та «При вході в систему», стосуються процесу завантаження ОС і будуть виконані при настанні відповідної події.

3.Налаштування запуску сервісів та драйверів

Записи про сервісні процеси (системні та окремих програм) та драйвери можна відредагувати на вкладках Services та Drivers програми Autoruns відповідно. Записи про сервіси також можна відредагувати безпосередньо в реєстрі за адресою: HKLM\SYSTEM\CurrentControlSet\services\. Для кожного сервісу є параметр «Start» типу «REG_DWORD». Він може приймати таке значення:

1. Низькорівневі драйвери, наприклад, драйвери дисків, які завантажуються на самому ранньому етапі завантаження – завантаження ядра;
2. Драйвери, які завантажуються після ініціалізації ядра ОС;
3. Драйвери та послуги, які повинні бути завантажені диспетчером управління службами (дорівнює параметру – «Авто») ;
4. Драйвери та служби, що запускаються диспетчером управління службами тільки в разі отримання явної інструкції на завантаження (дорівнює параметру – «Вручну»);
5. Драйвери та служби, які не завантажуються (дорівнює параметру – «Відключено»).

Якщо для сервісу вказати параметр запуску «Автоматичний (відкладений запуск)», то запуск сервісу відбудеться не відразу. При цьому значення параметру Start у реєстрі буде «0x02», але з’явиться новий параметр, який вказує саме на затримку запуску (DelayedAutostart=”0x01”).

При «вимкненні» сервісу із автозавантаження через Autoruns у відповідному розділі створюється параметр AutorunsDisabled, значення якого зберігає рівень, з яким раніше запускався сервіс. Таким чином, при потребі, можливе швидке відновлення автозапуску сервісу з відповідним рівнем запуску.

Якщо виконати «видалення» відповідного запису через Autoruns, то він назавжди видалиться із реєстру. Налаштування сервісів (служб) ОС Windows Всі служби ОС Windows можна умовно розділити на три групи:

- служби, які не можна відключати;
- служби, які можна відключити практично на будь-якому комп'ютері, тому що в більшості випадків вони не потрібні;
- служби, які можна відключити на домашньому комп'ютері або ноутбуці (без мережі).

Консоль Перегляд подій

У системі Windows для перегляду системних журналів використовується консоль **Перегляд подій** (Event Viewer – eventvwr.exe). «Перегляд подій» входить до складу консолі Управління комп'ютером (Computer Management). З його допомогою можна переглядати Журнал системи (System), що містить записи про події, які реєструються системними компонентами Windows. Журнал зберігає події операційної системи або її компонентів, наприклад, невдачі при запусках служб або ініціалізації драйверів, загальносистемні повідомлення та інші повідомлення, що відносяться до системи в цілому. За замовчуванням він розміщується в %SystemRoot%\System32\Winevt\Logs\System.Evtx

У журналах реєструються 5 типів подій:

Помилка (Error) – подія реєструється у випадку виникнення серйозної події (такої, як втрата даних або функціональних можливостей). Подія даного типу буде зареєстрована, якщо неможливо завантажити який-небудь сервіс у ході запуску системи.

Попередження (Warning) – подія не серйозна, але може привести до виникнення проблем у майбутньому. Наприклад, якщо недостатньо дискового простору, то в журнал буде занесене попередження.

Повідомлення (Information) – значима подія, яка свідчить про успішне завершення операції прикладною програмою, драйвером або сервісом. Така подія може, наприклад, зареєструвати мережний драйвер, що успішно завантажився.

Аудит успіхів (Success Audit) – подія, відповідна успішно завершеній дії, пов'язаній із підтримкою безпеки системи. Прикладом такої події є успішна спроба входу користувача в систему.

Аудит відмов (Failure Audit) – подія, відповідна невдало завершеній дії, пов'язаній із підтримкою безпеки системи. Наприклад, така подія буде зареєстрована, якщо спроба доступу користувачем до мережного диска закінчилася невдачею.

Маніпуляції із автозавантаженням додатків та сервісів впливають на час завантаження системи. Для оцінки тривалості завантаження ОС використовується Журнал додатків та служб Microsoft (Просмотр событий – Журнал приложений и служб Microsoft – Windows – Diagnostics – Performance – журнал «Работает»). Запис про подію із кодом 100 (100 – це pid процесу

завантаження ОС) дає змогу оцінити швидкість завантаження ОС.

Завдання на лабораторну роботу

1. В ОС своєї віртуальної машини зробити копію реєстру (гілок HKEY_LOCAL_MACHINE\SOFTWARE\,HKEY_LOCAL_MACHINE\SYSTEM\ ).

2. За допомогою консолі Перегляд подій (eventvwr.exe) визначити час завантаження ОС. Розмістити отримане значення у звіт (таблиця 1).

3. За допомогою утиліти Autoruns (запуск від імені адміністратора) налаштувати додатки, які не є системними (приховати записи Windows):

3.1. налаштувати додатки, які запускаються при старті ОС (Logon):

3.1.1. створити запис автозавантаження для Autoruns від імені адміністратора через папку Автозавантаження;

3.1.2. встановити CCleaner http://www.piriform.com/ccleaner та створити запис автозавантаження для CCleaner через реєстр (HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run);

3.1.3. встановити Skype та відключити його автозавантаження через Autoruns. https://www.skype.com/en/download-skype/skype-for-windows/downloading/

При встановленні програми вона прописується на автозавантаження у реєстр. Потрібно відключити її автозавантаження через Autoruns;

3.1.4. за потреби відключити із автозавантаження інші програми;

3.1.5. перезавантажити систему. Визначити час завантаження системи і записати його у таблицю 1.

3.2. дослідити і налаштувати сервіси, які є сервісами додатків (не ОС) (Services):

3.2.1. для SkypeUpdate через реєстр встановити запуск сервісу вручну (параметр Start=3); зробити експорт відповідної гілки реєстру в файл SkypeUpdate.reg;

3.2.2. за потреби відключити із автозавантаження інші сервіси;

3.3. на вкладці Драйвери видалити драйвери, які відображаються жовтим кольором;

3.4. перезавантажити систему. Визначити час завантаження системи і записати його у таблицю 1.

4. Провести налаштування запуску сервісів ОС Windows через Autoruns:

4.1. відповідно до варіанта налаштувати на автоматичний запуск сервіс із таблиці 2;

4.2. відповідно до варіанта налаштувати на запуск вручну сервіс із таблиці 3;

4.3. відповідно до варіанта відключити сервіс із таблиці 4.

Якщо відповідний сервіс відсутній, оберіть довільний із відповідної таблиці.

4.4. Виконати експорт гілок реєстру, які відповідають завданням 4.1-4.3.

4.5. Перезавантажити систему. Визначити час завантаження системи і записати його у таблицю 1.

5. За допомогою консолі Перегляд подій (eventvwr.exe) проаналізувати:

5.1. як змінився час завантаження системи;

5.2. чи виникли в системі події (критичні, помилки, попередження), пов'язані із налаштуваннями сервісів (Просмотр событий – Журналы Windows – Система). Зробити знімок екрану. Якщо так, то виправити відповідні налаштування.

6. При увімкнених записах Windows та Microsoft зберегти через Autoruns записи про налаштування автозапуску у файл №варіанту.txt.

7. Доповнити файл №варіанта.txt записами із файлів п.4.4 (reg-файлів).

8. Заповнити таблицю 1 та оформити звіт.

Таблиця 1. Результати автоналаштувань додатків та сервісів

Версія ОС	
Час завантаження системи	до правок _____ після змін: 1 (програми) _____ 2 (сервіси програм та драйвери)_____ 3 (сервіси ОС)_____
Додатки, які запускаються при старті ОС	додано через папку: додано через реєстр: в имкнено через Autoruns:
Сервіси додатків	змінено через запис реєстру:
Сервіси ОС	
автоматичний запуск	
запуск вручну	
відключено сервіс	
Помилки, попередження (консоль Перегляд подій)	
відсутні/наявні	

Питання для самоперевірки

1. Процес завантаження ОС Windows.
2. Рівні завантаження сервісів та драйверів.
3. Які тригери в Планувальнику завдань пов'язані із завантаженням ОС?
4. Як у реєстрі відображається запис про програми із вкладки Logon при їх включенні та відключенні.
5. Що таке запуск програми "вручну"?
6. Чим запуск програми «автоматично» відрізняється від запуску «автоматично відкладено». Яким значенням у реєстрі відповідають ці рівні запуску.

7. Які можна дати рекомендації щодо включення та відключення автозавантаження програм та сервісів?
8. Можливості та інтерфейс програми Autoruns.
9. Призначення сервісів ОС Windows (відповідно до номеру варіанта)

Лабораторна робота №14-15
Лінійні алгоритми та алгоритми розгалуження з використанням вставок на мові Assembler та циклічні алгоритми та операції над масивами з використанням вставок на мові Assembler.

- Робота виконується на основній або віртуальній машині із ОС Windows 10.
- Використовуються системні утиліти: консоль Монітор продуктивності (Performance Monitor), Диспетчер завдань (Task Manager).

1. Дослідження режимів роботи процесора

1.1. Запуск Монітора продуктивності Відкрийте «Монітор продуктивності», одночасно натиснувши клавіші «WIN» (на ній зображений прапор-логотип MS Windows) + «R», ввівши в вікні «Виконати» рядок perfmon.exe і натиснувши Enter для підтвердження введення.

Альтернативний спосіб запуску програми полягає в наступному:

- натисніть Пуск|Виконати,
- наберіть у вікні perfmon.exe (або просто perfmon),
- натисніть Enter для введення.

1.2. Створіть Нову групу збирачів даних Lab2 (Продуктивність – Групи збирачів даних – Особливий – Створити – Група збирачів даних), наповнення якої будете виконуватись вручну (рис.1а). Для спостереження за режимами роботи обчислювальної системи та обробкою переривань будемо використовувати журнали даних на основі лічильників продуктивності (рис.1б).

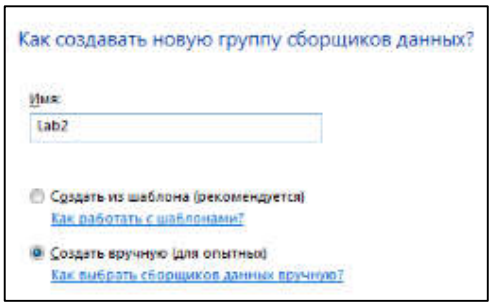


Рис. 1а. Створення групи збирачів даних (вибір імені та типу)

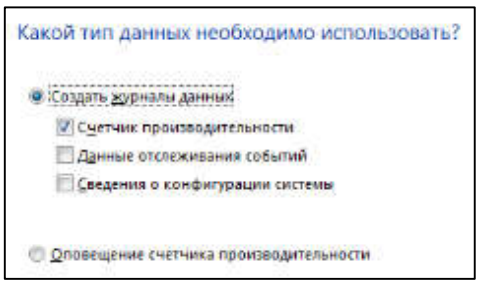


Рис. 1б. Створення групи збирачів даних (вибір типу даних)

1.3. Створіть збирач даних для дослідження режимів функціонування ОС.

➢ Щоб вказати, які лічильники ми будемо використовувати, можна продовжити процедуру створення нової групи збирачів даних, натиснувши кнопку Далі.

➢ На формі вибору (рис. 2) обираємо лічильники:

• Processor:% Processor Time(_Total), Processor: % Idle Time(_Total),

• Processor: %Privileged Time(_Total), Processor: %User Time(_Total),

• Process\Idle:%Privileged Time, Process\Idle:% User Time.

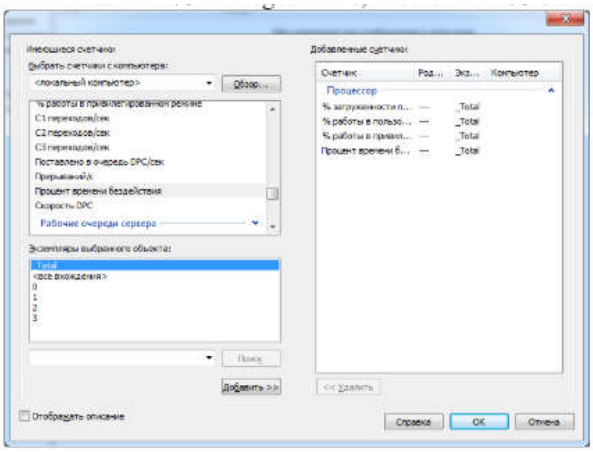


Рис. 2. Редагування лічильників

➢ Підтверджуємо завершення створення групи лічильників,

натиснувши Готово.

➢ Змінимо властивості створеної групи лічильників Lab2: маємо змогу відредагувати місце зберігання журналів продуктивності та умову зупинки спостереження – встановлюємо 1 хвилину (рис. 3). За замовчуванням місце зберігання журналів продуктивності %systemdrive%\PerfLogs\Admin\назва групи.

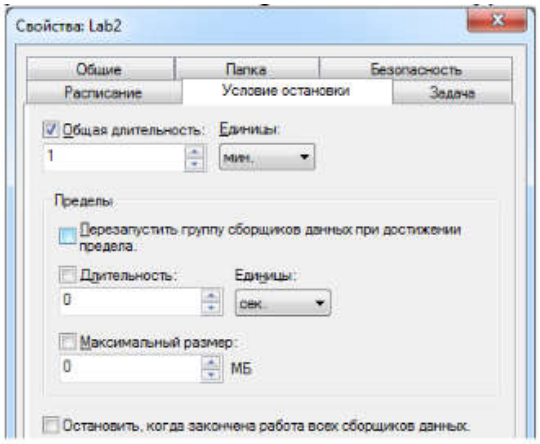


Рис. 3. Вкладинка «Умова зупинки» вікна «Властивості»

Змінимо властивості створеного набору збирачів даних: вказуємо інтервал вибірки лічильника 5 сек (рис. 4) – вказує на частоту зняття показників; на вкладниці Файл вказуємо ім’я журналу та режим запису із заміною (рис. 5) – кожного разу при запуску журналу файл буде переписуватися.

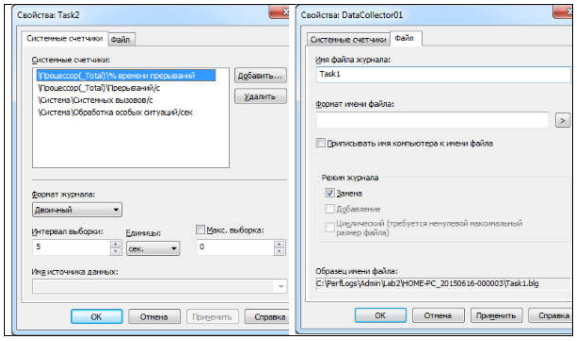


Рис. 4. Вкладинки «Системні лічильники» та «Файл» вікна «Властивості»

2. Спостереження за обробкою переривань в ОС

➢ Створюємо новий збирач даних (рис. 5).

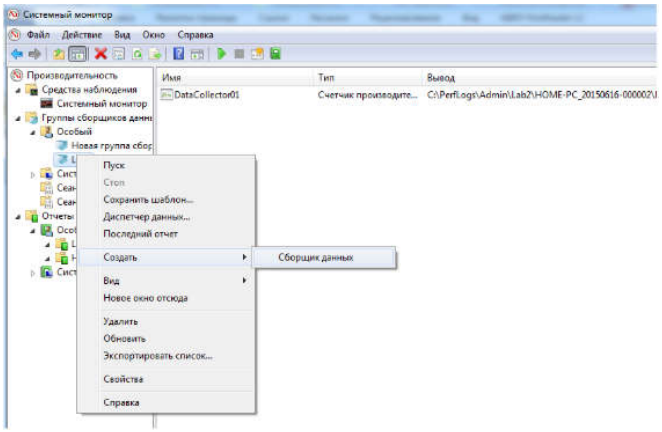


Рис. 5. Створення нового збирача даних

➢ Надаємо йому інформативну назву та обираємо відповідний тип.

➢ Додаємо лічильники:

• Processor: Interrupts /sec(_Total), Processor: %Interrupt Time(_Total),

• System: System Calls/sec, System: Exceptions/ Sec.

➢ Змінюємо властивості збирача даних, вказавши інтервал вибірки 5 секунд, режим ведення журналу «із заміною».

3. Збір даних при відсутності взаємодії із системою

3.1. Закрийте всі зайві вікна (крім Perfomance monitor) і проведіть збір даних по двох групах створених лічильників, коли ви не виконуєте ніяких дій із системою.

3.2. Виконавши «Продуктивність – Групи збирачів даних – Особливий – Lab2 – Пуск», система побудує 2 набори лічильників, протягом хвилини, фіксуючи відповідні показники. Під час збору показників біля пункту Нова група збирачів даних з’явиться знак виконання запису, який зникне, коли запис буде завершено.

3.3. Щоб переглянути зібрані дані можна виконати перегляд Останнього звіту або перейти до пункту «Продуктивність – Звіти – Особливий – Нова га група збирачів даних».

3.4. Відформатуйте отримані графіки, клацнувши правою кнопкою миші на рисунку і обравши пункт «Властивості».

Рекомендовані такі масштаби для лічильників:

– набір 1 – масштаб 1;

– набір 2: % часу переривань та Обробка особливих ситуацій –

масштаб 1;

Переривань/с та Системних викликів/с – обрати такий масштаб, щоб графік відповідного лічильника розміщувався в діапазоні 0-100. Рекомендовано вибрати такі кольори графіків, або змінити їх стиль і ширину, щоб було видно відповідності графіків та лічильників.

3.5. Після форматування графіків їх необхідно зберегти у вигляді рисунків (формат gif) та у вигляді даних (двійковий файл blg).

3.6. Після виконання цього завдання повинно бути 2 відформатованих рисунка gif та 2 файли .blg.

4. Збір даних при роботі з обраним процесом

4.1. Запустіть збір статистики.

4.2. Попрацюйте деякий час із процесом, який відповідає вашому варіанту (табл.5).

4.3. Коли запис у журнал закінчиться, перегляньте 2 створені графіки, відредагуйте їх (як у п. 3.4) та збережіть у вигляді рисунків (формат gif) та у вигляді даних (двійковий файл blg).

4.4. Після виконання цього завдання повинно бути 2 відформатованих рисунка та 2 файли .blg. 28

5. Створення електронного звіту

5.1. Дослідження відсотку роботи процесору у режимах ядра та користувача.

5.1.1. На одній сторінці розмістіть 2 рисунки із графіками лічильників таблиці 1, отриманих в п. 3 та 4.

5.1.2. На основі рисунків потрібно заповнити відповідними даними таблиці 3 та 4 (середнє та максимальне значення лічильника будується автоматично (рис.1)).

5.1.3. Зробіть висновок, в якому режимі (користувача чи ядра) здебільшого працював процесор, чи залежить це від процесів, які на ньому виконувалися? Які дії досліджуваний вами процес виконував у режимі користувача, а що переадресовував на виконання ОС? Чи не перевищує показник Processor:%Processor Time (_Total) рекомендованого максимального значення?

5.2. Дослідження роботи операційної системи по обробці переривань

5.2.1. На одній сторінці розмістіть 2 рисунки із графіками лічильників таблиці 2, отриманих в п. 3 та 4.

5.2.2. На основі рисунків потрібно заповнити відповідними даними таблиці 3 та 4.

5.2.3. Зробіть висновки, чи не перевищує показник Processor: %Interrupt Time (_Total) рекомендованого максимального значення

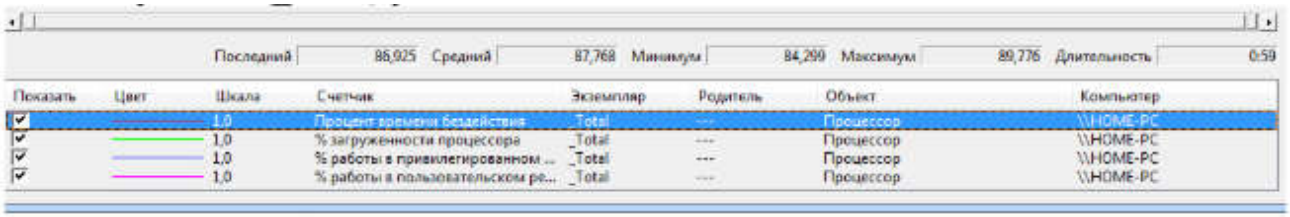


Рис. 6. Середнє та максимальне значення лічильника

Приклад оформлення результатів

В якості програми для взаємодії обрано IDE

1. Дослідження відсотків роботи процесору у режимах ядра та користувача

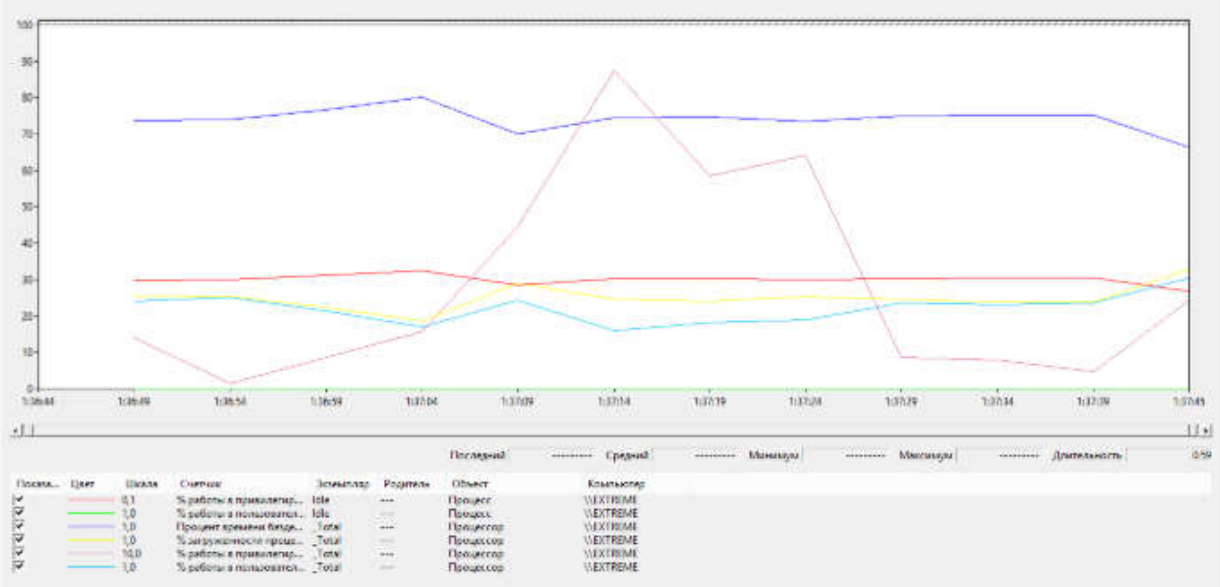


Рис. 7. Робота процесора у режимах ядра та користувача (Бездіяльність системи)

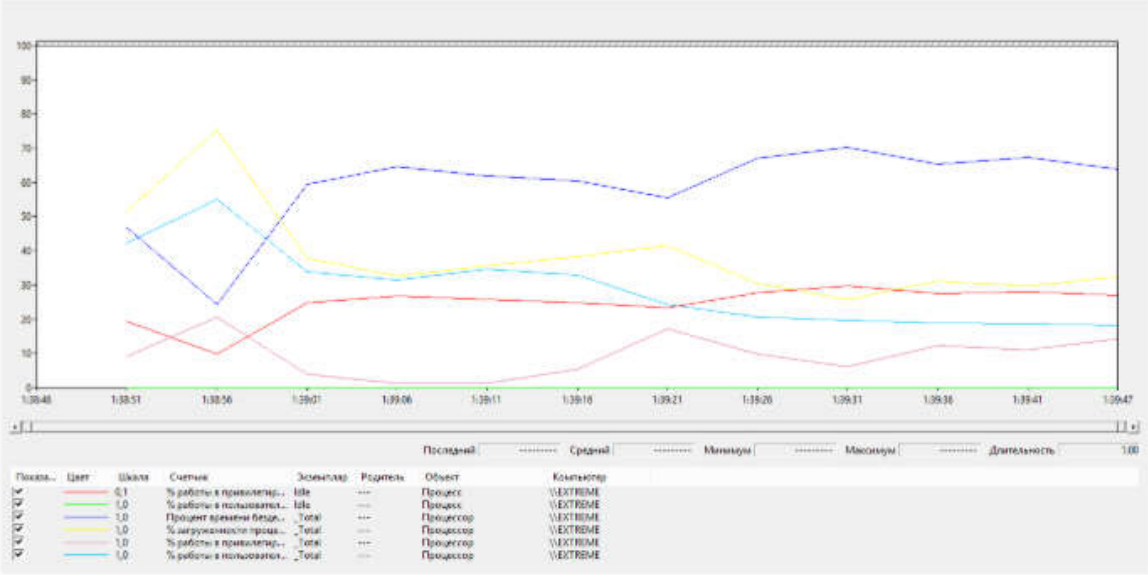


Рис. 8. Робота процесора у режимах ядра та користувача (Робота у IDE)

2. Дослідження роботи операційної системи з обробки переривань

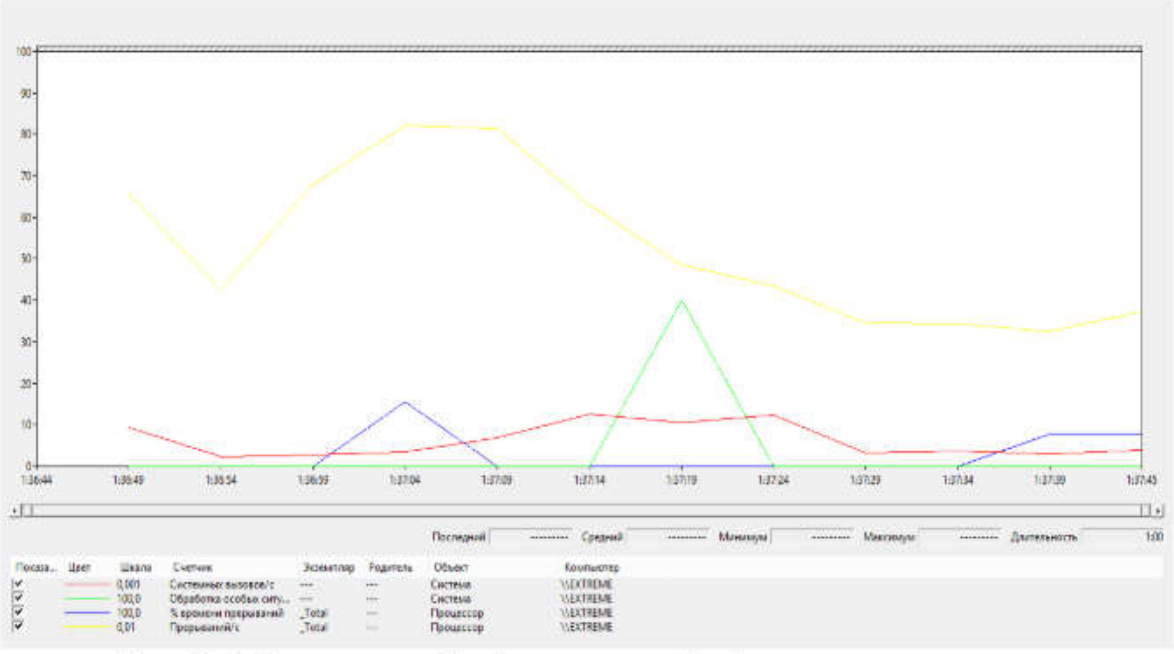


Рис. 9. Робота операційної системи з обробки переривань (Бездіяльність системи)

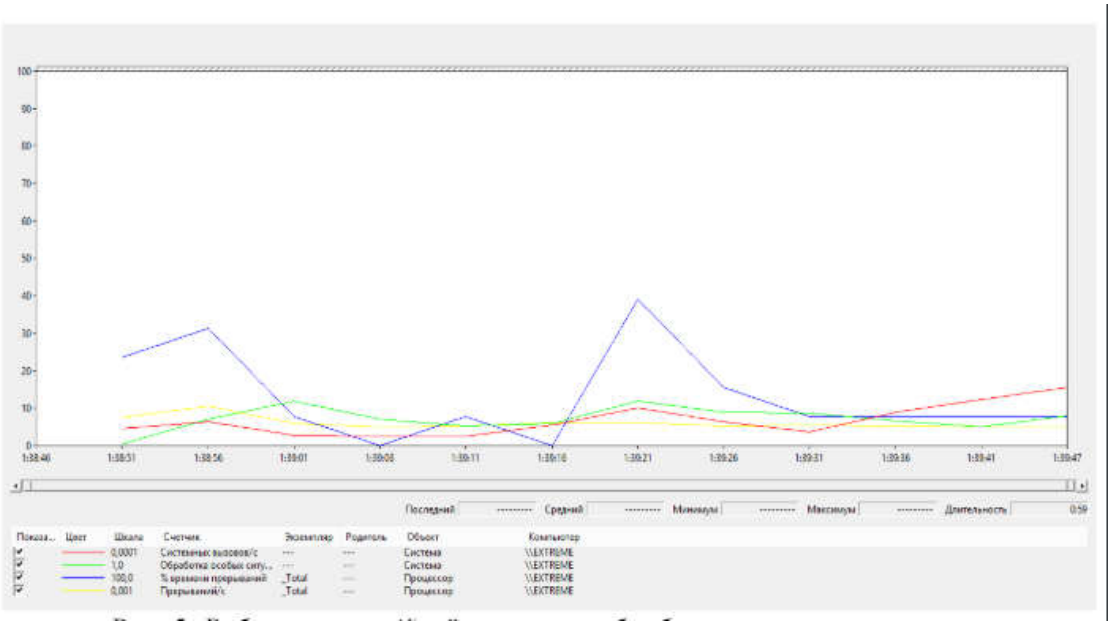


Рис. 10. Робота операційної системи з обробки переривань (Робота у IDE)

3. Середнє та максимальне значення лічильників.

Таблиця 1. Середнє значення лічильників

Лічильник	Середнє значення лічильника	
	відсутність взаємодії, %	взаємодія з обраним процесом, %
Processor:% Processor Time (_Total)	24,988	38,529
Processor: % Idle Time (_Total)	74,048	58,928
Processor: %Privileged Time (_Total)	2,833	9,323
Processor: %User Time (_Total)	22,154	29,206
Process\Idle: %Privileged Time	300,048	245,885
Process\Idle:% User Time	0,000	0,000
Processor: Interrupts /sec (_Total)	5276,109	6065,058
Processor: %Interrupt Time (_Total)	0,026	0,130
System: System Calls/sec	6190,994	67695,264
System: Exceptions/ Sec	0,033	7,254

Таблиця 2. Максимальне значення лічильників

Лічильник	Максимальне значення лічильника		Тривалість перевищення, с
	відсутність взаємодії, %	взаємодія з обраним процесом, %	
Processor:% Processor Time (_Total)	32,831	75,318	0
Processor: %Interrupt Time (_Total)	0,156	0,391	0

4. Висновки до таблиць

В якому режимі (користувача чи ядра) здебільшого працював процесор? Чи залежить це від процесів, які на ньому виконувалися?

Процесор більше працював у режимі користувача. Режим роботи системи залежить від типу виконуваних процесів, адже при роботі в IDE менеджер вікон оброблює, наприклад, input-операції від користувача, або, навпаки, виконує якусь операцію (наприклад, створює проект) у режимі ядра, а під час виконання процесу Idle процесор взагалі постійно перебуває у режимі ядра.

Які дії досліджуваний вами процес виконував у режимі користувача, а що переадресовував на виконання ОС?

При роботі з IDE переадресація йшла на менеджер вікон для обробки input-операцій від користувача, і при цьому, дії, наприклад, створення проекту, переадресовувалися на виконання ОС, оскільки дії виконуються над файловою системою. У режимі користувача виконується перевірка синтаксису введеного тексту, зміна режимів відображення, надання користувачу відповідних функцій, відображення структури проекту тощо.

Чи не перевищують показники Processor:%Processor Time(_Total), Processor: %Interrupt Time(_Total) рекомендованого максимального значення?

Обидва показники не перевищили максимального значення, що вказує на те, що із відповідним навантаженням процесор справляється.

ПИТАННЯ ДЛЯ ПІДСУМКОВОГО КОНТРОЛЮ « ОПЕРАЦІЙЕНІ СИСТЕМИ ТА СИСТЕМНЕ ПРОГРАМУВАННЯ»

1. Призначення операційної системи як проміжної ланки між користувачем і апаратним забезпеченням
2. Основні функції операційних систем та їх роль у керуванні ресурсами
3. Класифікація операційних систем за сферою застосування
4. Порівняння однокористувацьких і багатокористувацьких ОС
5. Етапи розвитку операційних систем від монолітних до мікроядерних
6. Архітектура сучасних ОС та склад її основних компонентів
7. Переваги мікроядерної архітектури над монолітною
8. Функції ядра ОС та механізми взаємодії з апаратним забезпеченням
9. Поняття процесу як одиниці виконання програми
10. Життєвий цикл процесу та його основні стани
11. Методи планування процесів у сучасних ОС
12. Пріоритети процесів та вплив їх зміни на продуктивність системи
13. Механізми створення та завершення процесів у ОС Linux
14. Засоби моніторингу процесів у командному середовищі Linux
15. Потоки як легковагові процеси та їх переваги
16. Основні моделі багатопоточного програмування
17. Синхронізація потоків за допомогою семафорів і м’ютексів
18. Проблема взаємоблокування та методи її уникнення
19. Організація обміну сигналами між процесами в ОС
20. Системні виклики, пов’язані з керуванням сигналами у Linux
21. Методи перенаправлення потоків введення/виведення в Linux
22. Порівняльна характеристика процесів і потоків у Windows
23. Створення потоків у Windows API
24. Взаємодія процесів через канали та сокети
25. Механізми міжпроцесної взаємодії у Windows і Linux
26. Основи керування фізичною пам’яттю в ОС
27. Організація сегментованої та сторінкової пам’яті
28. Переваги та недоліки сторінкової організації пам’яті
29. Принцип роботи механізму віртуальної пам’яті
30. Алгоритми заміщення сторінок у ОС
31. Вплив продуктивності підсистеми пам’яті на роботу ОС
32. Принципи роботи стеку й купи в прикладних програмах
33. Робота прикладних програм у середовищі Windows
34. Створення графічних інтерфейсів засобами Windows API
35. Обробка подій у віконних застосунках Windows

36. Функціонування драйверів введення/виведення
37. Рівні організації підсистеми введення/виведення в ОС
38. Методи буферизації введення/виведення
39. Принципи побудови файлових систем
40. Структура каталогів і блоків у файлових системах
41. Порівняння файлових систем NTFS, ext4 та FAT32
42. Механізми журналювання у файлових системах
43. Забезпечення безпеки даних у файлових системах
44. Особливості роботи систем з графічним інтерфейсом у Linux і Windows
45. Створення піктограм та елементів інтерфейсу у Windows
46. Принципи розроблення зберігача екрану в ОС Windows
47. Основні тенденції розвитку сучасних операційних систем
48. Роль відкритого програмного забезпечення у розвитку ОС
49. Розвиток мобільних операційних систем Android та iOS
50. Хмарні операційні системи та їх призначення
51. Особливості безпеки у сучасних ОС
52. Методи виявлення та усунення уразливостей операційних систем
53. Порівняння механізмів безпеки Windows і Linux
54. Інструменти отримання інформації про апаратне забезпечення
55. Методи оптимізації продуктивності ОС
56. Використання командного рядка Linux для роботи з файлами
57. Особливості роботи з Perl-скриптами у Linux
58. Використання механізмів перенаправлення потоків для автоматизації
59. Основи вставок на мові Assembler у високорівневі програми
60. Організація циклічних алгоритмів та роботи з масивами із використанням вставок Assembler

СПИСОК РЕКОМЕНДОВАНИХ ТА ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура системного програмного забезпечення : підручник / Л. О. Левченко, Н. Г. Кучук, Ю. А. Тарнавський, В. П. Колумбет. Київ : КПІ ім. Ігоря Сікорського, 2022. 497 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/d932fa9d-e1b7-4275-9dea-77b3af0c2c66/content

2. Бондаренко М.Ф., Качко О.Г. Операційні системи: Навчальний посібник. Х.: Компанія СМІТ, 2021. – 432 с.

3. Гагарін О.О., Левченко Л.О. Практикум низькорівневого програмування. Основи розробки асемблерних програм у середовищі MS DOS: навч. посіб. К.: НТУУ «КПІ». 2020. 352 с.

4. Задерейко О. В., Гура В. І., Толокнов А. А. Операційні системи : навчально-методичний посібник. Одеса : Фенікс, 2023. 298 с.

5. Задерейко О. В., Зіноватна С. Л., Толокнов А. А. Операційні системи : навчальний посібник. Одеса : Фенікс, 2022. 140 с.

6. Зайцев В.Г. Операційні системи: [Електронний ресурс]: навч. посіб. для студ. спеціальності 123 «Комп’ютерна інженерія» / В. Г. Зайцев, І. П. Дробязко; КПІ ім. Ігоря Сікорського. Електронні текстові дані (1 файл: 3 Мбайт). Київ: КПІ ім. Ігоря Сікорського, 2019. 240 с.

7. Зілінський Ю. В., Перекрест А. Л., Юдіна А. Л. Системне програмування. Програмування на ассеблері: навч. посібник. Кременчук : Кременчуцький національний університет ім. Михайла Остроградського, 2023. 258 с. URL: https://document.kdu.edu.ua/monogr/2023_105.pdf

8. Козак Л. І., Костюк І. В., Стасевич С. П. Основи програмування : навчальний посібник. Львів : Новий Світ-2000, 2020. 328 с.

9. Левченко Л. О., Тарнавський Ю. А. Операційні системи : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2023. 256 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/e8b6f0e6-9381-4aeb-b699-dfccc92edddb/content

10. Мосіюк О. О., Федорчук А. Л. Операційні системи та системне програмування : навчально-методичний посібник. Житомир : ЖДУ ім. І. Франка, 2022. 76 с. URL: http://eprints.zu.edu.ua/33751/1/OS_ost_Feb_04.pdf

11. Системне програмування. Програмування на асемблері: комп'ютерний практикум : навч. посіб. для студ. освітньої програми «Комп'ютерні системи та мережі» спеціальності 123 «Комп'ютерна інженерія» / КПІ ім. Ігоря Сікорського; уклад. В. М. Порєв. Київ : КПІ ім. Ігоря Сікорського, 2022. 146 с. https://ela.kpi.ua/server/api/core/bitstreams/11d2c8c6-6b6d-4c2f-ab16-09a5d4b18fd6/content

12. Сумець О. М. Проєктування операційних систем : підручник. Київ :

Університет «КРОК», 2021. 322 с. https://library.krok.edu.ua/media/library/category/pidruchniki/sumets_0002.pdf

13. Системне програмування та операційні системи. Метод. вказівки до викон. лаб. робіт для студ. напрямків підготов. 6.050103 „Програмна інженерія" та „Комп'ютерні науки" / Уклад : О.О. Гагарін, Л.О. Левченко. К.: НТУУ «КПІ». 2021. 141 с.

14. Харченко В.П., Знаковська Є.А, Бородін В.А. Операційні системи та системи програмування. Київ: НАУ, 2020.

15. Федотова-Півень І. М. Операційні системи: навчальний посібник. [за ред. В. М. Рудницького] / І. М. Федотова-Півень, І. В. Миронець, О. Б. Півень, С. В. Сисоєнко, Т. В. Миронюк; Черкаський державний технологічний університет. Харків : ТОВ «ДІСА ПЛЮС», 2019. 216 с.

16. Kusswurm Daniel. Modern X86 Assembly Language Programming / Daniel Kusswurm. Apress, 2019. 604 p.

17. Яковина В.С. Операційні системи. Конспект лекцій. Національний Університет «Львівська політехніка», Львів, 2019. 128 с.

18. Сумець О. М. Проєктування операційних систем: підручник. Київ: Університет «КРОК», 2021. 32 с.

19. Linux Foundation. Linux Documentation Project. URL: https://www.kernel.org/doc/html/latest/

20. Microsoft Learn. Windows System Programming Documentation. URL: https://learn.microsoft.com/en-us/windows/win32/

21. FreeBSD Project. Official Documentation. URL: https://docs.freebsd.org/

Навчальне видання

Операційні системи та системне програмування

Методичні рекомендації

Укладачі:

Тищенко Світлана Іванівна
Пархоменко Олександр Юрійович
Ємельянов Святослав Ігорович
Кучмійова Тетяна Сергіївна
Жебко Олександр Олегович
Волосюк Юрій Вікторович
Коломієць Андрій Миколайович

Формат 60x84 1/16. Ум. друк. арк. 9,00.
Наклад 50 прим. Зам. № _____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету
54020, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб'єкта видавничої справи ДК № 4490 від 20.02.2013 р.