МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ ЕЛЕКТРОЕНЕРГЕТИКИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра автоматизації та комп'ютерних систем

С.М. Ткаченко

СИСТЕМНЕ ПРОГРАМУВАННЯ

Методичні рекомендації
до виконання лабораторних робіт
студентами галузі знань 12 Інформаційні технології
спеціальності 123 Комп'ютерна інженерія

Дніпро

НТУ «ДП»

2019

Ткаченко С.М.

Системне програмування. Методичні рекомендації до виконання лабораторних робіт студентами галузі знань 12 Інформаційні технології спеціальності 123 Комп’ютерна інженерія / С.М. Ткаченко; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро: НТУ «ДП», 2019. – 54 с.

Автори:
С.М. Ткаченко, канд. техн. наук, доц. (лаб. роботи 1-6)

Затверджено методичною комісією з галузі знань 12 Інформаційні технології (протокол № 4 від 26.12.2018) за поданням кафедри автоматизації та комп'ютерних систем (протокол № 6 від 28.12.2018).

Подано методичні рекомендації до виконання дипломних проектів студентами зі спеціальності 123 Комп’ютерна інженерія.

Відповідальний за випуск зав. кафедри АКС В.В. Ткачов, д-р техн. наук, проф.

ЗМІСТ

ВСТУП

Методичні вказівки призначені для студентів спеціальності 123 «Комп'ютерна інженерія», що вивчають дисципліну «Системне програмування».

Методичні вказівки включають низку частково взаємопов'язаних робіт, під час виконання яких студенти мають можливість отримати досвід роботи з віртуальною машиною, навчитися встановлювати, конфігурувати та адмініструвати операційні системи лінії Windows NT, засвоїти ряд концептуальних підходів до написання програм, що отримують доступ до даних файлів та їх атрибутів.

При підготовці до виконання лабораторної роботи студент повинен:

– ознайомитися з методичними вказівками;

– повторити лекційний матеріал, пов'язаний з темою лабораторної роботи;

– підготувати відповіді на контрольні питання, які наведені у методичних вказівках наприкінці кожної лабораторної роботи.

Виконавши ці завдання, студент повинен продемонструвати викладачеві роботу на комп'ютері, оформити звіт за результатами даної лабораторної роботи, захистити його і здати викладачу.

Загальні вимоги до виконання лабораторної роботи, що мають забезпечити максимальну оцінку:

– повна відповідність звіту про виконання лабораторної роботи методичним рекомендаціям;

– володіння теоретичним матеріалом про предмет досліджень;

– загальна та професійна грамотність, лаконізм і логічна послідовність викладу матеріалу;

– відповідність оформлення звіту чинним стандартам.

ЛАБОРАТОРНА РОБОТА № 1
ВИВЧЕННЯ НАЛАШТУВАНЬ CMOS BIOS

1.1. Мета лабораторної роботи

Ознайомитись з програмним пакетом VMWare і отримати практичні навики в роботі з настройками BIOS Setup.

1.2. Постановка задачі

У пакеті VMWare створити віртуальну машину з об'ємом оперативної пам'яті у межах 128 - 512 MB, об'ємом жорсткого диску 1 або 2 GB, мережною карткою та дисководом для дискет 3,5 ″. Вивчити меню BIOS Setup, створеної віртуальної машини, та виконати налаштування параметрів останнього згідно завдання за варіантом.

1.3. Порядок виконання роботи

1. Вивчити, користуючись рекомендованою літературою, конспектом лекцій, і методичними вказівками до даної роботи, наступні питання:
   - меню і особливості роботи програмного пакету VMWare;
   - роботу і повідомлення програми POST;
   - поняття CMOS-пам'ять;
   - зміст і настройки меню BIOS Setup.
2. Створити, згідно завданню, власну віртуальну машину у пакеті VMWare.
3. Запустити створену машину на виконання та ввійти до BIOS Setup за допомогою клавіші, вказаної у запрошенні після POST-тесту;
4. Відшукати заздалегідь вивчені пункти меню BIOS Setup, проглянути їх настройки;
5. Законспектувати призначення та доступні настройки знайдених пунктів BIOS Setup;
6. Виконати настройки згідно варіанту.

1.4. Зміст звіту

1. Назва, мета й завдання лабораторної роботи..
2. Короткий конспект по командах меню BIOS SETUP UTILITY.
3. Опис вироблюваних дій в ході виконання роботи згідно варіанту.
4. Висновок про виконану роботу.

1.5. Теоретичні відомості

VMWare – програма-емулятор віртуальної машини від фірми VMWare Inc для операційних систем (ОС) Windows NT, XP, 2003. Вона емулює роботу комп'ютера на рівні команд процесора і на рівні бібліотек ОС Windows 95, 98, ME, NT 4, 2000, XP, Linux, FreeBSD, DOS. Емуляція проводиться від POST-тесту під час включення і завантаження ОС до відключення віртуальної машини віртуальною ОС.

Віртуальні машини VMWare, можуть використовувати майже всі пристрої несучого комп'ютера, включаючи COM і LPT-порти, SCSI, USB, мережеві карти, дисководи, жорсткі диски, CD-ROM. Також підтримуються віртуальні жорсткі, змінні диски і CD-ROM. Віртуальну машину можна зробити видимою у мережі тільки для несучого комп'ютера, для окремої підмережі, для всієї локальної мережі. Несучий комп'ютер при цьому виконує функції роутера.

BIOS – це постійний запам'ятовуючий пристрій (ПЗП) разом із записаною на нього інформацією. ПЗП містить наступні модулі:

- базову систему введення/виведення (Basic Input/Output System – BIOS);
- програму первинного тестування комп'ютера (Power-On Self Test – POST);
- програму первинного завантаження комп'ютера;
- програму перегляду і модифікації CMOS-пам'яті BIOS Setup.

Існують декілька основних фірм, що спеціалізуються на створенні програмного забезпечення BIOS. Принципових відмінностей між BIOS різних фірм не існує, але вони відрізнятись додатковими можливостями і зовнішнім виглядом.

Системна плата комп'ютера містить CMOS-пам'ять, яка побудована на енергозалежній мікросхемі. При виключенні комп'ютера мікросхема CMOS живиться від окремої батареї, що дозволяє постійно зберігати дані про настройку. Ці дані відображають тип підключеного вінчестера, об'єм оперативної пам'яті, поточні дату і час, параметри периферійних портів, інші конфігураційні дані. Для обмеження доступу до комп'ютера в CMOS-пам'яті зберігається пароль. Його можна видалити, тимчасово відключити живлення батареї CMOS. При цьому всі інші настройки комп'ютера втрачаються.

Для читання і редагування настройок, записаних в CMOS-пам'яті, використовується програма BIOS Setup. Як правило, вона зберігається безпосередньо в ПЗП BIOS комп'ютера. BIOS Setup можна запустити при включенні комп'ютера за допомогою клавіші, вказаної у запрошенні, що виводиться на екран відразу після закінчення первинної процедури тестування POST.

1.6. Варіанти завдань

Таблиця 1.1 – Варіанти завдань до лабораторної роботи № 1

№ п/п.	№ завдання зі списку	№ п/п.	№ завдання зі списку
1	1, 3, 6, 11, 14, 18, 20, 25, 29, 32, 35	11	1, 4, 6, 12, 14, 18, 20, 27, 30, 33, 35
2	2, 4, 7, 12, 15, 19, 21, 26, 30, 33, 36	12	2, 5, 7, 13, 15, 19, 21, 28, 31, 34, 36
3	1, 5, 8, 13, 16, 18, 22, 27, 31, 34, 37	13	1, 3, 8, 11, 16, 18, 22, 25, 29, 32, 37
4	2, 3, 9, 11, 17, 19, 23, 28, 29, 32, 38	14	2, 4, 9, 12, 17, 19, 23, 26, 30, 33, 38
5	1, 4, 10, 12, 14, 18, 24, 25, 30, 33, 39	15	1, 5, 10, 13, 14, 18, 24, 27, 31, 34, 39
6	2, 5, 6, 13, 15, 19, 20, 26, 31, 34, 35	16	2, 3, 6, 11, 15, 19, 20, 28, 29, 32, 35
7	1, 3, 7, 11, 16, 18, 21, 27, 29, 32, 36	17	1, 4, 7, 12, 16, 18, 21, 25, 30, 33, 36
8	2, 4, 8, 12, 17, 19, 22, 28, 30, 33, 37	18	2, 5, 8, 13, 17, 19, 22, 26, 31, 34, 37

Продовження таблиці 1.1

№ п/п.	№ завдання зі списку	№ п/п.	№ завдання зі списку
9	1, 5, 9, 13, 14, 18, 23, 25, 31, 34, 38	19	1, 3, 9, 11, 14, 18, 23, 27, 29, 32, 38
10	2, 3, 10, 11, 15, 19, 24, 26, 29, 32, 39	20	2, 4, 10, 12, 15, 19, 24, 28, 30, 33, 39

Список завдань:

1. Встановити системний час;
2. Встановити системну дату;
3. Встановити один жорсткий диск як ведучий;
4. Встановити два жорсткі диски;
5. Встановити один жорсткий диск і два приводи CD;
6. Встановити дисковод A – 720 KB, B – відключити;
7. Встановити дисковод A – 360 KB, B – 1,44 MB;
8. Встановити дисковод A – 2,88 KB, B – 720 KB;
9. Відключити дисководи A і B;
10. Встановити дисковод A – 1,44 MB, B – 1,2 MB;
11. Встановити стартове включення клавіші NumLock;
12. Зняти стартове включення клавіші NumLock;
13. Встановити автовизначення клавіші NumLock;
14. Встановити час повтору символу натиснутої клавіші ¼ с;
15. Встановити час повтору символу натиснутої клавіші ½ с;
16. Встановити час повтору символу натиснутої клавіші ¾ с;
17. Встановити час повтору символу натиснутої клавіші 1с;
18. Підключити екран діагностики при завантаженні системи;
19. Відключити екран діагностики при завантаженні системи;
20. Налагодити COM-порт A на адресу 3F8/IRQ4; COM-порт B відключити
21. Налагодити COM-порт A на адресу 2F8/IRQ3, COM-порт B на 3E8/IRQ4, режим normal;
22. Відключити COM-порт A, підключити COM-порт B у стан auto, режим IrDa;
23. Налагодити COM-порт A у стан auto, а COM-порт B на адресу 2F8/IRQ3, режим FIR;
24. Відключити COM-порти A і B;
25. Налагодити LPT-порт на адресу 378/IRQ5, режим Output only;
26. Налагодити LPT-порт на адресу 278/IRQ7, режим Bi-direct;
27. Налагодити LPT-порт на адресу 3BC/IRQ7, режим ECP;
28. Відключити LPT-порт;
29. Включити максимальне енергозбереження;
30. Включити максимальне використання енергії комп’ютером;
31. Включити ручні настройки параметрів енергоспоживання;
32. Встановити завантаження системи з жорсткого диска;
33. Встановити завантаження системи з дисководу;
34. Встановити завантаження системи з CD-ROM;

35. Вийти з програми BIOS Setup зі збереженням настройок;
36. Вийти з програми BIOS Setup без збереження настройок;
37. Завантажити стандартні настройки;
38. Завантажити попередні настройки;
39. Зберегти внесені зміни.

1.7. Контрольні питання

1. Що таке BIOS? Які його функції? Які модулі входять до складу BIOS?
2. Як визначити об'єм оперативної пам'яті встановленої на комп'ютері?
3. Що відбудеться якщо акумулятор, що живить CMOS-пам'ять вийде з ладу?
4. Які настройки містить вкладка Main?
5. У якому випадку необхідно ініціалізувати диски комп'ютера?

ЛАБОРАТОРНА РОБОТА № 2
ФОРМАТУВАННЯ ЖОРСТКОГО ДИСКА. РОЗМІТКА ТОМІВ

2.1. Мета лабораторної роботи

Вивчити архітектуру жорсткого диска, одержати практичні навики в розбитті його на розділи і форматуванні. Ознайомитися з різновидами файлових систем. Засвоїти основні команди DOS.

2.2. Постановка задачі

На віртуальній машині, створеній у лабораторній роботі №1, виконати розподіл простору жорсткого диску на основний та додатковий розділи, які повинні займати 30% та 70% від усього об'єму відповідно. Основний розділ призначити активним. У додатковому розділі, якщо цього потребує завдання виділити логічні диски та розподілити між ними простір порівну. Відформатувати всі диски, позначивши основний розділ як системний. Виконати умови завдання згідно варіанту.

2.3. Порядок виконання роботи

1. Вивчити, користуючись рекомендованою літературою, конспектом лекцій, і методичними вказівками до роботи, наступні питання:
   - жорсткий диск і його архітектура;
   - файлові системи і їх класифікація;
   - операційна система DOS та її основні команди.
2. Виконати розбиття диска на розділи і його форматування:
   - встановити в BIOS завантаження з системної дискети;
   - завантажитися з системної дискети;
   - за допомогою команди cd d:\ перейти на віртуальний диск d:\, набрати команду help і вивчити за документом допомоги використання утиліт fdisk і format;

- за допомогою утиліти fdisk розділити жорсткий диск відповідно завданню;
- за допомогою утиліти format відформатувати всі виділені диски згідно завданню;
- встановити в BIOS завантаження з диска c:\, вийняти з дисковода системну дискету і переконатися, що завантаження комп'ютера йде нормально.

3. Виконати завдання згідно варіанту.

2.4. Зміст звіту

1. Назва і мета лабораторної роботи.
2. Опис призначення утиліт fdisk и format.
3. Докладний опис послідовності дій при розбиванні жорсткого диска на розділи і його форматуванні.
4. Опис виконаних згідно варіанту команд DOS.
5. Висновок про виконану роботу.

2.5. Теоретичні відомості

Жорсткий диск є блоком з декількох дисків, над поверхнями яких переміщуються зчитуючи головки. Позиціонуються головки по концентричним доріжкам, кожна з яких розділена на сектори. Сектор є мінімальним адресованим блоком даних для диска. Його розмір становить 512 байт.

На нульовій доріжці в нульовому секторі диска, розміщується головний завантажувальний запис (MBR - Master Boot Record). Він включає в себе код початкового завантаження, таблицю розділів (PT - Partition Table) і визначення активного розділу. Таблиця розділів описує тип, активність розділів, початковий і кінцевий номери циліндра розділу. У разі пошкодження MBR завантаження з диска неможливе.

Файлова система визначає формат фізичного зберігання, спосіб організації, іменування даних на носіях інформації. Конкретна файлова система визначає розмір імені файлу, максимально можливий розмір файлу, набір атрибутів файлу. Деякі файлові системи надають додаткові можливості з розмежування доступу або шифрування файлів.

Файлова система зв'язує носій інформації і API (інтерфейс прикладного програмування) для доступу до файлу засобами прикладної програми. При цьому остання не потребує інформації про фізичне розміщення даних файлу та про фізичний тип носія. Все, що знає програма - це ім'я файлу, його розмір і атрибути. Ці дані вона одержує від драйвера файлової системи. Саме файлова система встановлює, де і як буде записаний файл на фізичному носії (наприклад, жорсткому диску).

З погляду операційної системи весь диск є набір кластерів розміром від 512 байт і вище. Драйвери файлової системи організують кластери у файли і каталоги (що реально є файлами, які містять список файлів в цьому каталозі). Ці ж драйвери відстежують, які з кластерів в даний час використовуються, які вільні, які помічені як несправні.

Файлова система не обов'язково безпосередньо пов'язана з фізичним носієм інформації. Існують віртуальні і мережеві файлові системи, які є лише способом доступу до файлів, що знаходяться на видаленому комп'ютері.

Файлова система FAT (скорочення File Allocation Table) – файлова система, що використовується в операційних системах DOS і Windows. Логічний диск, який відформатований в системі FAT, має наступні розділи: завантажувальний сектор, таблиця розміщення файлів, кореневий каталог і, власне, файли.

Для зберігання файлів весь доступний для них простір розбивається на кластери. Таблиця розміщення файлів містить ланцюг номерів кластерів для кожного файлу, невживані кластери помічені нулем.

Існує три версії FAT – FAT12, FAT16 і FAT32. Вони відрізняються кількістю біт, відведених для зберігання номера кластера. FAT12 застосовується в основному для дискет FAT16 – для дисків малого об'єму.

Файлова система NTFS (скорочення New Technology File System) – основна файлова система для сімейства операційних систем Microsoft Windows NT. Вона є журнальною, підтримує систему метаданих і використовує спеціалізовані структури даних для поліпшення надійності і ефективності використання дискового простору. NTFS має вбудовані можливості керування доступом для користувачів і груп користувачів, та квотування дискового простору.

Файлова система NFS (Network File System) – розподілена файлова система, розроблена Sun Microsystems, що дозволяє користувачам звертатися до файлів і каталогів, розташованих на віддалених комп'ютерах, так неначебто ці файли і каталоги є локальними. NFS не залежить від типу комп'ютерів, операційних систем і архітектури мережі.

Файлова система ext2 або друга розширена файлова система – файлова система для ядра Linux, розроблена Rémy Card як заміна extended file system. Є еталоном для тестів продуктивності інших файлових систем. Розвитком ext2 стала журнальна файлова система ext3, що використовується в сучасних версіях Linux. Сімейство ext відрізняється від інших файлових систем мережевою структурою каталогів, тобто один файл або каталог може входити відразу в декілька каталогів у тому числі під різними іменами.

Дискова операційна система MS DOS (далі MS DOS) призначена для обслуговування і програмної підтримки комп'ютерів, що використовують дисководи і жорсткі диски. Вона забезпечує роботу з файлами, каталогами , а також запуск додатків. MSDOS є умовно однозадачною, тобто можливе виконання тільки однієї програми в інтерактивному режимі.

Файлами MS DOS є COMMAND.COM, IO.SYS, MSDOS.SYS (можуть мати і інші імена), скопійовані особливим чином на завантажувальний жорсткий диск або дискету.

Імена пристроїв та дисків в MS DOS:

A: – дисковод A, з нього можна завантажувати систему;

B: – дисковод B, аналогічний дисководу A, зустрічається на старих моделях ПК;

C: і далі за алфавітом – жорсткий диск (диски), або логічні розділи одного фізичного диска;

наступна буква після імені останнього жорсткого диска – дисковод DVD або CD-ROM;

PRN або LPT – паралельний порт, зазвичай використовується принтером або сканером;

COM1, COM2 – послідовні порти, використовуються модемом, com-мишею та іншою периферією.

Команди MS DOS діляться на внутрішні і зовнішні (утиліти). Внутрішні команди виконуються за наявності файлу COMMAND.COM в оперативній пам'яті комп'ютера, а зовнішні входять до складу MSDOS окремими файлами.

Після завантаження MS DOS з'являється запрошення: C:> або A:>, де C: - ім'я диска. Це означає, що комп'ютер готовий до роботи і чекає введення команди. Нижче наведені деякі команди MS DOS для роботи з каталогами і файлами.

Таблиця 2.2 – Команди MS DOS для роботи з каталогами і файлами

Команда	Формат	Опис
dir	dir диск:\шлях\ ім'я_файлу/ параметри	Проглядання каталогу
md	md диск:\ ім'я_каталогу	Створення каталогу
cd	cd диск:\ ім'я_каталогу	Зміна поточного каталогу
rd	rd диск:\ ім'я_каталогу	Видалення каталогу
deltree	deltree ім'я_файлу_ або_ ім'я_каталогу	Видалення каталогу зі всім його вмістом
move	move диск:\ ім'я_каталогу нове_ім'я_каталогу	Перейменування каталогу
path	path диск1:\ім'я_каталогу1 диск2:\ ім'я_каталогу2	Вказівка шляху до каталогу
..		Повернення в попередній каталог
\		Повернення в кореневий каталог
copy	copy диск1:\шлях1\ ім'я_файлу1 /v /a /b диск2:\шлях2\ім'я_файлу2 /v /a /b	Копіювання або приєднання одного або декількох файлів. Файл1 – джерело, а файл 2 – приймач
del	del диск:\шлях\ ім'я_файлу	Видалення файлу
ren	ren диск:\шлях\ ім'я_файлу1 ім'я_файлу2	Перейменування файлу, де ім'я_файлу1 – старе ім'я, а ім'я_файлу2 – нове
move	move /y ім'я_файлу ім'я_каталогу	Переміщення файлу в інший каталог

Продовження таблиці 2.2

Команда	Формат	Опис
type	type диск:\шлях\ ім’я_файлу	Вивод на екран текстового файлу
cls		Очищення екрану
ver		Видача версії операційної системи

Команди обслуговування дисків. Найпоширеніша команда – форматування диска format. Вона виконує розмітку поверхні диска, записує на ньому системну інформацію (завантажувальний сектор, таблицю розміщення файлів і кореневий каталог), а також перевіряє диск на наявність дефектів. За наявності відповідних параметрів визову format також копіює на диск системні файли. Команда format знищує всю інформацію на диску.

Системні файли можуть бути скопійовані на вже відформатований диск командою sys.

Команда diskcopy повністю копіює всю інформацію з однієї дискети на іншу:

Утиліта scandisk призначена для перевірки поверхні диска, файлової системи і каталогів.

Утиліта fdisk – програма для розбиття жорсткого диска на розділи в системі MS DOS, а також для вказання завантажувального розділу. При розбитті диска на розділи програмою FDISK вся інформація знищується.

2.6. Варіанти завдань

Створити структуру файлів відповідно до таблиці 2.1. У файл text_1.txt занести ім’я і прізвище. У файл text_2.txt занести групу і назву лабораторної роботи. Об’єднати файли text_1.txt і text_2.txt у файл subject.txt і розмістити його згідно завданню.

Таблиця 2.1 – Варіанти завдань до лабораторної роботи

№ п/п.	Вигляд дерева каталогів	Завдання
1.	C:\ └Номер за списком └text_1 . txt └Folder_1 E:\ └text_2 . txt	Скопіювати файл subject.txt у корінь диска D:\. Видалити файл text_2.txt. Вивести звіт про виконання останньої команди у текстовий файл text_1.txt. Встановити дату 12/01/2001 поточною. Виконати перевірку диска E:\.

Продовження таблиці 2.1

№ п/п.	Вигляд дерева каталогів	Завдання
2.	C:\ └text_1 . txt D:\ └Folder_1 └.Номер за списком └text_2 . txt	Скопіювати файл subject.txt у корінь диска C:\. Перейменувати файл text_2.txt у text_3.txt. Змінити розширення файлу text_1.txt на *.doc. Встановити час 12:03 поточним. Виконати форматування диска A:\.
3.	C:\ └.Номер за списком └text_1 . txt D:\ └Folder_1 └text_2 . txt	Скопіювати файл subject.txt у корінь диска D:\. Видалити файл text_1.txt. Вивести на екран текстовий файл text_2.txt. Очистити екран. Виконати перевірку диска C:\.
4.	C:\ ├Folder_1 └text_1 . txt └text_2 . txt D:\ └.Номер за списком	Скопіювати файл subject.txt у C:\Folder_1. Переіменувати файл subject.txt в info.txt. Змінити розширення файла text_2.txt на *.doc. Вивести версію операційної системи. Виконати форматування диска D:\.
5.	C:\ └Folder_1 D:\ └.Номер за списком └text_1 . txt E:\ └text_2 . txt	Скопіювати файл subject.txt у корінь диска E:\. Видалити файл text_2.txt. Вивести на екран текстовий файл text_1.txt. Встановити дату 02/11/2005 поточною. Виконати перевірку диска E:\.
6.	C:\ └Номер за списком D:\ ├Folder_1 └text_1 . txt └text_2 . txt	Скопіювати файл subject.txt у корінь диска A:\. Перейменувати файл text_2.txt у text_6.txt. Змінити розширення файлу text_2.txt на *.gif. Встановити час 10:06 поточним. Виконати форматування диска A:\.
7.	A:\ └Номер за списком └text_1 . txt C:\ └Folder_1 └text_2 . txt D:\ └Folder_2	Скопіювати файл subject.txt на диск D:\Folder_2. Видалити файл text_1.txt. Вивести на екран текстовий файл subject.txt. Виконати очищення екрану. Виконати перевірку диска A:\.

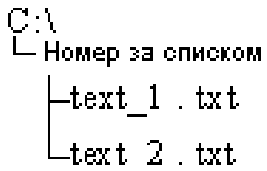
Продовження таблиці 2.1

№ п/п.	Вигляд дерева каталогів	Завдання
8.	B:\ ├Folder_1 └text_1 . txt └Folder_2 └text_2 . txt C:\ └Номер за списком	Скопіювати файл subject.txt у корінь диска D:\. Перейменувати файл text_1.txt у text_8.txt. Змінити розширення файлу text_2.txt на *.avi. Вивести версію операційної системи. Виконати форматування диска B:\.
9.	D:\ ├Номер за списком └Folder_1 └text_1 . txt └ text_2 . txt B:\ └Folder_2	Скопіювати файл subject.txt у корінь диска C:\. Видалити файл text_2.txt. Вивести на екран текстовий файл text_1.txt. Встановити дату 25/03/2009 поточною. Виконати перевірку диска C:\.
10.	C:\ ├Folder_1 └Folder_2 └Номер за списком └text_1 . txt D:\ └Folder_3 └text_2 . txt E:\	Скопіювати файл subject.txt у корінь диска E:\. Перейменувати файл text_2.txt у text_10.txt. Змінити розширення файлу text_1.txt на *.html. Встановити час 11:10 поточним. Виконати форматування диска D:\.
11.	C:\ ├ Номер за списком └text_1 . txt └Folder_1 └text_2 . txt E:\	Скопіювати файл subject.txt у C:\Folder_1 Видалити файл text_1.txt Вивести на екран текстовий файл subject.txt Виконати очищення екрану Виконати перевірку диска E:\
12.	B:\ C:\ ├Folder_1 └text_1 . txt D:\ └Номер за списком └text_2 . txt	Скопіювати файл subject.txt у C:\Folder_1 Перейменувати файл text_1.txt у text_12.txt Змінити розширення файлу text_12.txt на .bmp Вивести версію операційної системи Виконати форматування диска B:\
13.	C:\ └Номер за списком └text_1 . txt D:\ └Folder_1 B:\ └Folder_2 └text_2 . txt	Скопіювати файл subject.txt у B:\Folder_2 Видалити файл text_2.txt Вивести на екран текстовий файл text_2.txt Встановити дату 13/08/1999 поточною Виконати перевірку диска C:\

Продовження таблиці 2.1

№ п/п.	Вигляд дерева каталогів	Завдання
14.	B:\ └Номер за списком └Folder_1 └Folder_2 ├text_1 . txt └text_2 . txt C:\ └Folder_3 E:\	Скопіювати файл subject.txt у корінь диска E:\ Перейменувати файл text_2.txt у text_14.txt Змінити розширення файлу text_12.txt на .arj Встановити час 02:14 поточним Виконати форматування диска B:\
15.	D:\ ├Folder_1 └Folder_2 E:\ └Номер за списком ├text_1 . txt └text_2 . txt	Скопіювати файл subject.txt у D:\Folder_2 Видалити файл text_1.txt Вивести на екран текстовий файл subject.txt Виконати очищення екрану Виконати перевірку диска E:\
16.	B:\ └Номер за списком D:\ └Folder_1 └text_1 . txt E:\ ├Folder_2 └text_2 . txt └Folder_3	Скопіювати файл subject.txt у E:\Folder_3 Перейменувати файл subject.txt у text_16.txt Змінити розширення файлу text_12.txt на .zip Вивести версію операційної системи Виконати форматування диска D:\
17.	D:\ └Folder_1 B:\ └Номер за списком └Folder_2 └text_1 . txt E:\ └text_2 . txt	Скопіювати файл subject.txt у корінь диска E:\ Видалити файл text_2.txt Вивести на екран текстовий файл text.txt Встановити дату 03/01/1997 поточною Виконати перевірку диска B:\
18.	C:\ └Номер за списком D:\ └Folder_1 └text_1 . txt E:\ ├Folder_2 └Folder_3 └text_2 . txt	Скопіювати файл subject.txt у корінь диска C:\ Перейменувати файл text_1.txt у text_18.txt Змінити розширення файлу text_12.txt на .rar Встановити час 01:18 поточним Виконати форматування диска E:\
19.	B:\ C:\ └Folder_1 D:\ └Folder_2 E:\ ├text_1 . txt └text_2 . txt	Скопіювати файл subject.txt у корінь диска B:\ Видалити файл text_1.txt Вивести на екран текстовий файл subject.txt Виконати очищення екрану Виконати перевірку диска E:\

Продовження таблиці 2.1

№ п/п.	Вигляд дерева каталогів	Завдання
20.		Скопіювати файл subject.txt у корінь диска D:\ Перейменувати файл subject.txt у text_20.txt Змінити розширення файлу text_20.txt на .doc Вивести версію операційної системи Виконати форматування диска A:\

2.7. Контрольні питання

1. Яке призначення файлової системи?
2. Охарактеризуйте систему FAT.
3. Для чого необхідні файли AUTOEXEC.BAT і CONFIG.SYS?
4. Як вивести на екран список файлів і каталогів?
5. Як відформатувати дискету?
6. Як перевірити диск на наявність помилок?

ЛАБОРАТОРНА РОБОТА № 3
ВСТАНОВЛЕННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ WINDOWS NT5

3.1. Мета лабораторної роботи

Одержати практичні навики встановлення та адміністрування операційної системи Windows NT.

3.2. Постановка задачі

Створити нову віртуальну машину, установити операційну систему Windows NT. Виконати дії по налагодженню та обслуговуванню системи, які обумовленні завданням на лабораторну роботу.

3.3. Порядок виконання роботи

1. Вивчити, користуючись рекомендованою літературою, конспектом лекцій і методичними вказівками до даної роботи, наступні питання:

- встановлення операційної системи Windows NT;
- настройка операційної системи Windows NT;
- обслуговування операційної системи Windows NT;

2. Виконати підготовчі дії для встановлення операційної системи Windows NT:

- скопіювати образ завантажувального диска Win2000.iso у свій робочий каталог;
- підключити до приводу компакт-дисків віртуальної машини образ диска Win2000.iso зі свого робочого каталогу.

Створити нову віртуальну машину з системою MS-DOS і за допомогою одержаних навиків, встановити операційну систему Windows NT.

4. Виконати завдання згідно варіанту.
5. Виконані кроки описати в звіті.

3.4. Зміст звіту

1. Назва і мета лабораторної роботи.
2. Опис послідовності дій по встановленню та настройці операційної системи Windows NT згідно варіанту.
3. Висновки про виконану роботу.

3.5. Теоретичні відомості

Windows 2000 є представником операційних систем сімейства NT компанії Microsoft. На відміну від серії **Windows 9x(95,98, Millennium),** що не виправдала себе при роботі з мережею та багатокористувачевим режимом, Windows NT має гібридну архітектуру та не намагається налагодити контакт з некоректно працюючою програмою або пристроєм. Це робить систему стабільнішою.

На будь-якому компакт-диску з дистрибутивом Windows 200 Professional знаходиться файл Read1st.txt. Цей файл містить відомості про Windows NT, необхідні для її успішного завершення. Зокрема, у ньому вказане наступне:

- при встановленні Windows NT слід обрати, чи провести оновлення раніше встановленої операційної системи чи виконати встановлення нової копії;
- при оновленні програма встановлення замінює виконувані та бібліотечні файли Windows, але залишає існуючі настройки і додатки. При цьому деякі додатки після оновлення можуть працювати в Windows NT некоректно;
- при встановленні нової копії Windows NT, остання встановлюється у новий каталог. Якщо для існуючої операційної системи оновлення неможливе, наприклад, якщо це система Microsoft Windows 3.1 або OS/2, слід встановити нову копію.

Проводити оновлення операційної системи має сенс, коли на комп'ютері вже встановлена попередня версія Windows і потрібно зберегти наявні додатки і настройки. В інших випадках бажано встановити нову копію.

Можна створювати конфігурацію з подвійним завантаженням Windows NT і іншої сумісної операційної системи. При цьому Windows NT повинна бути встановлена в окремий розділ диска.

Встановлення операційної системи. Якщо жорсткий диск комп'ютера порожній або наявна поточна система не допускає оновлення, слід запустити комп'ютер одним з наступних способів:

- за допомогою завантажувальних дискет;
- за допомогою компакт-диска Windows 2000 Professional, якщо дисковод для компакт-дисків може бути завантажувальним;

– з ОС MS DOS якщо дисковод для компакт-дисків або компакт-диск не є завантажувальними;

– за допомогою мережевого з'єднання.

Для створення завантажувальних дисків використовується чотири 3,5-дюймових гнучких диска ємністю 1,44 Мбайт. Щоб створити завантажувальні диски необхідно з компакт-диска запустити програму MAKEBOOT.EXE, яка знаходиться в папці BOOTDISK.

Встановлення нової копії операційної системи за допомогою завантажувальних дисків потребує наступних кроків:

– при вимкненому комп'ютері вставити перший завантажувальний диск Windows NT в дисковод для гнучких дисків;

– включити комп'ютер;

– слідувати інструкціям на екрані.

Встановлення нової копії операційної системи за допомогою завантажувального компакт-диска потребує наступних кроків:

– запустити на комп'ютері поточну операційну систему і вставити компакт-диск Windows NT Professional в дисковод для компакт-дисків;

– у стартовому меню, що з'явилося, вибрати пункт «Установка Windows NT»;

– виконувати інструкції на екрані.

Встановлення нової копії операційної системи з будь-якого незавантажувального джерела потребує наступних кроків:

– у командну строку Windows 9x ввести команду ШЛЯХ_ДО_ДЖЕРЕЛА_ІНСТАЛЯЦІЇ\i386\winnt32.exe;

– якщо використовується MS-DOS, в її у командну строку ввести команду ШЛЯХ_ДО_ДЖЕРЕЛА_ІНСТАЛЯЦІЇ\i386\winnt.exe ;

– слідувати інструкціям на екрані.

Оновлення операційної системи. Майстер оновлення визначає і встановлює потрібні драйвери або створює звіт про пристрої, що не допускають оновлення. Таким чином, забезпечується гарантія сумісності апаратного і програмного забезпечення з Windows NT. Для оновлення виконуються ті ж кроки, що і для встановлення операційної системи за винятком методу завантажувальних дискет.

Перший запуск Windows NT. Після закінчення встановлення на екрані з'являється вікно входу в Windows NT. Після входу в систему проводиться реєстрація копії Windows, остаточна конфігурація її налагоджень, створюються облікові записи користувачів.

Додаткові параметри установки.

Файлові системи. Перед установкою Windows NT слід визначити, яку файлову систему планується використовувати. Допускається файлова система NTFS або один з різновидів FAT.

Розділи дисків. Створення розділів на жорсткому диску є способом ділення жорсткого диска таким чином, що кожна частина виконує функції

окремого пристрою. Розділи створюються для організації даних, наприклад для їх архівації або для створення конфігурації з подвійним завантаженням. При створенні розділів на диску він ділиться на одну або декілька частин, які можна відформатувати для роботи в таких файлових системах, як FAT або NTFS.

При створенні нової копії Windows NT програма установки автоматично вибирає потрібний розділ диска, якщо при цьому не був вибраний варіант введення додаткових параметрів установки і не були вказані інші параметри. Жорсткий диск може містити до чотирьох розділів.

Налагодження системи. У Windows NT існує єдина папка, в якій зосереджені всі команди, параметри і конфігурації операційної системи. Це системна папка «Панель керування», яка включає наступні утиліти.

Адміністрування/Служби компонентів. Використовуються системними адміністраторами для використання і адміністрування програм COM+ за допомогою графічного інтерфейсу, а також для автоматизації адміністративних завдань за допомогою мов програмування і підготовки сценаріїв. Розробники програмного забезпечення можуть використовувати служби компонентів для настройки стандартних дій компонентів і програм, а також для інтеграції компонентів в програмі COM+.

Адміністрування/Управління комп'ютером. Утиліта призначена для управління локальними або віддаленими комп'ютерами однією об'єднаною службовою програмою робочого столу. Управління комп'ютером об'єднує декілька засобів адміністрування Windows NT в одне дерево консолі, що забезпечує легкий доступ до конкретних властивостей адміністрування комп'ютера.

Адміністрування/Джерела даних (ODBC). Це інтерфейс реєстрації баз даних для забезпечення доступу до них програм, які використовують мову SQL.

Адміністрування/Вікно проглядання подій. Призначене для перегляду і управління журналами системних і програмних подій, а також подій безпеки на комп'ютері. У вікні проглядання подій збираються відомості про несправності устаткування і неполадки програмного забезпечення, а також відображаються події безпеки Windows NT.

Адміністрування/Локальна політика безпеки. Використовується для настройки параметрів безпеки локального комп'ютера. Такими параметрами, крім інших, є політики паролів, облікових записів, аудиту, безпеки IP, визначення привласнення прав користувачам, призначення агентів відновлення зашифрованих даних. Локальні політики безпеки доступні тільки на комп'ютерах з операційною системою Windows NT. Якщо комп'ютер є членом домена, ці параметри можуть бути перевизначені політиками домена.

Адміністрування/Системний монітор. Утиліта призначена для збору і перегляду у реальному часі даних пам'яті, диска, процесора, мережі і інших параметрів у вигляді графіка, гістограми або звіту.

Адміністрування/Служби. Використовується для управління службами комп'ютера, установки дій по відновленню у разі збою служби і створення призначених для користувача імен і описів служб для спрощення їх визначення.

Користувачі і паролі. Утиліта, яка дозволяє реєструвати в системі окремих користувачів і групи користувачів із встановленням їх дозволів і прав. Також служить для створення або зміни паролів локальних облікових записів користувачів. Якщо комп'ютер підключений до мережі, можна додавати облікові записи користувачів мережі в групи локального комп'ютера, а користувачі мережі можуть використовувати свої мережеві паролі для входу в систему. Змінити пароль користувача з мережі не можна.

Мережа і віддалений доступ до мережі. Компонент забезпечує зв'язок локального комп'ютера з Інтернетом, локальною мережею або іншим комп'ютером. Це дозволяє діставати доступ до ресурсів і функціональних можливостей мережі незалежно від методу підключення (локальна мережа, модем, лінія ISDN та інші). Підключення створюються, настроюються і зберігаються в папці «Мережа і віддалений доступ до мережі».

Система. Утиліта, що використовується для виконання наступних дій:

– перегляд і зміна параметрів, що керують використанням пам'яті;

– пошук відомостей та настройка обладнання, властивості служб і про настройку профілів обладнання;

– перегляд відомостей та настройка мережевих підключень і їх профілів.

– Для внесення змін в папці «Система» необхідно увійти до операційної системи на правах адміністратора.

Встановлення обладнання. Щоб пристрій працював правильно з Windows NT, на комп'ютер потрібно завантажити драйвер пристрою, який поставляється виробником пристрою або вже є в Windows NT. Для встановлення пристроїв використовують майстер встановлення обладнання на панелі керування і диспетчер пристроїв. Настройка можлива тільки в сеансі адміністратора. Якщо комп'ютер підключений до мережі, параметри мережевої політики можуть заборонити встановлення.

Обслуговування Windows NT

Дефрагментування диска. Програма дефрагментування об'єднує фрагментовані файли і папки на жорсткому диску комп'ютера, після чого кожен файл або папка тому займає єдиний безперервний простір. В результаті доступ до файлів і папок виконується швидше. Об'єднуючи окремі частини файлів і папок, програма дефрагментування також об'єднує в єдине ціле вільне місце на диску, що робить менш імовірною фрагментацію нових файлів.

Перевірка диска. Виконується періодично з метою виявлення помилок в системі файлів, самому диску а також з метою відновлення загублених ланцюгів кластерів. Інформація в загублених кластерах частіше всього не має значення, але важливо, що під управління файлової системи повертається втрачений об'єм диска.

3.6. Варіанти завдань

Таблиця 3.1. – Варіанти завдань до лабораторної роботи

№ п/п.	Завдання
1.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача зі своїм ім’ям. Виконати дефрагментування диска D:\. Запустити диспетчер пристроїв. Зупинити службу віддаленого управління реєстром. Встановити часовий пояс GTM +03:00. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути відключення дисплея через 10 хвилин. Встановити одну із стандартних екранних заставок.
2.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Виконати перевірку диска C:\. Запустити системний монітор і вивести на нього завантаженість роботи процесора. Запустити службу часу Windows. Відключити автоматичний перехід на літній час. Відобразити приховані папки і файли. Ввімкнути режим очікування через 30 хвилин. Встановити на робочий стіл стандартний фоновий малюнок.
3.	Встановити систему з компакт-диска з середовища MS-DOS вважаючи, що компакт-диск не є завантажувальним. Додати користувача зі своїм ім’ям. Встановити запуск програми перевірки диска C:\ через 10 хвилин. Викликати диспетчер задач. Зупинити службу віддаленого управління реєстром. Додати українську до мов вводу. Встановити, щоб кожна папка відкривалася у новому вікні. Ввімкнути відображення значка управління електроживленням на панелі задач. Визначити частоту оновлення монітора.
4.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача зі своїм ім’ям. Встановити запуск програми очищення диска D:\ через 5 хвилин. Запустити диспетчер пристроїв. Запустити службу часу Windows. Змінити поєднання клавіш для перемикання між мовами на CTRL+SHIFT. Показати приховані папки і файли. Ввімкнути відключення дисплея через 10 хвилин. Встановити розрішення екрану на 1024x768.

Продовження таблиці 3.1

№ п/п.	Завдання
5.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Виконати дефрагментування диска D:\. Запустити системний монітор і вивести на нього завантаженість роботи процесора. Зупинити службу віддаленого управління реєстром. Змінити курсор миші в основному режимі. Вказати, щоб кожна папка відкривалася у новому вікні. Ввімкнути режим очікування через 30 хвилин. Встановити одну із стандартних екранних заставок.
6.	Встановити систему з компакт-диска з середовища MS-DOS вважаючи, що компакт-диск не є завантажувальним. Додати користувача зі своїм ім’ям. Виконати перевірку диска C:\. Викликати диспетчер задач. Запустити службу часу Windows. Встановити часовий пояс GTM +03:00. Відобразити приховані папки і файли. Ввімкнути відображення значка управління електроживленням на панелі задач. Встановити на робочий стіл стандартний фоновий малюнок.
7.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача зі своїм ім’ям. Встановити запуск програми перевірки диска C:\ через 10 хвилин. Запустити диспетчер пристроїв. Зупинити службу віддаленого управління реєстром. Відключити автоматичний перехід на літній час. Вказати, щоб кожна папка відкривалася у новому вікні. Ввімкнути відключення дисплея через 10 хвилин. Визначити частоту оновлення монітора.
8.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Встановити запуск програми очищення диска D:\ через 5 хвилин. Запустити системний монітор і вивести на нього завантаженість роботи процесора. Запустити службу часу Windows. Додати українську до мов вводу. Відобразити приховані папки та файли. Ввімкнути режим очікування через 30 хвилин. Встановити розрішення екрану на 1024x768.

Продовження таблиці 3.1

№ п/п.	Завдання
9.	Встановити систему з компакт-диска з середовища MS-DOS вважаючи, що компакт-диск не є завантажувальним. Додати користувача зі своїм ім’ям. Виконати дефрагментування диска D:\. Викликати диспетчер задач. Зупинити службу віддаленого управління реєстром. Змінити поєднання клавіш для перемикання між мовами на CTRL+SHIFT. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути відображення значка управління електроживленням на панелі задач. Встановити одну із стандартних екранних заставок.
10.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача зі своїм ім’ям. Виконати перевірку диска C:\. Запустити диспетчер пристроїв. Запустити службу часу Windows. Змінити курсор миші в основному режимі. Показати приховані папки і файли. Ввімкнути відключення дисплея через 10 хвилин. Встановити на робочий стіл стандартний фоновий малюнок.
11.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Встановити запуск програми перевірки диска C:\ через 10 мин. Запустити системний монітор і вивести на нього завантаженість роботи процесора. Зупинити службу віддаленого управління реєстром. Встановити часовий пояс GTM +03:00. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути режим очікування через 30 хвилин. Визначити частоту оновлення монітора.
12.	Встановити систему з компакт-диска з середовища MS-DOS (вважати, що компакт-диск не є завантажувальним). Додати користувача зі своїм ім’ям. Встановити запуск програми очищення диска D:\ через 5 хвилин. Викликати диспетчер задач. Запустити службу часу Windows. Відключити автоматичний перехід на літній час. Показати приховані папки і файли. Ввімкнути відображення значка управління електроживленням на панелі задач. Встановити розрішення екрану на 1024x768.

Продовження таблиці 3.1

№ п/п.	Завдання
13.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача зі своїм ім’ям. Виконати дефрагментування диска D:\. Запустити диспетчер пристроїв. Зупинити службу віддаленого управління реєстром. Додати розкладку української мови. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути відключення дисплея через 10 хвилин. Встановити одну із стандартних екранних заставок.
14.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Виконати перевірку диска C:\. Запустити системний монітор і вивести на нього завантаженість роботи процесора. Запустити службу часу Windows. Змінити поєднання клавіш для перемикання між мовами на CTRL+SHIFT. Показати приховані папки і файли. Ввімкнути режим очікування через 30 хвилин. Встановити на робочий стіл стандартний фоновий малюнок.
15.	Встановити систему з компакт-диска з середовища MS-DOS вважаючи, що компакт-диск не є завантажувальним. Додати користувача зі своїм ім’ям. Встановити запуск програми перевірки диска C:\ через 10 хвилин. Викликати диспетчер задач. Зупинити службу віддаленого управління реєстром. Змінити курсор миші в основному режимі. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути відображення значка управління електроживленням на панелі задач. Визначити частоту оновлення монітора.
16.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача зі своїм ім’ям. Встановити запуск програми очищення диска D:\ через 5 хвилин. Запустити диспетчер пристроїв. Запустити службу часу Windows. Встановити часовий пояс GTM +03:00. Показати приховані папки і файли. Ввімкнути відключення дисплея через 10 хвилин. Встановити розрішення екрану на 1024x768.

Продовження таблиці 3.1

№ п/п.	Завдання
17.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Виконати дефрагментування диска D:\ Запустити системний монітор і вивести на нього завантаженість роботи процесора. Зупинити службу віддаленого управління реєстром. Відключити автоматичний перехід на літній час. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути режим очікування через 30 хвилин. Встановити одну із стандартних екранних заставок.
18.	Встановити систему з компакт-диска з середовища MS-DOS вважаючи, що компакт-диск не є завантажувальним. Додати користувача зі своїм ім’ям. Виконати перевірку диска C:\. Викликати диспетчер задач. Запустити службу часу Windows. Додати розкладку української мови. Показати приховані папки і файли. Ввімкнути відображення значка управління електроживленням на панелі задач. Встановити на робочий стіл стандартний фоновий малюнок.
19.	Встановити систему за допомогою завантажувального компакт-диска. Додати користувача з своїм ім’ям. Встановити запуск програми перевірки диска C:\ через 10 хвилин. Запустити диспетчер пристроїв. Зупинити службу віддаленого управління реєстром. Змінити поєднання клавіш для перемикання між мовами на CTRL+SHIFT. Вказати, щоб кожна папка відкривалася в новому вікні. Ввімкнути відключення дисплея через 10 хвилин. Визначити частоту оновлення монітора.
20.	Встановити систему за допомогою завантажувальних дискет. Додати користувача зі своїм ім’ям. Встановити запуск програми очищення диска D:\ через 5 хвилин. Запустити системний монітор і вивести на нього завантаженість роботи процесора. Запустити службу часу Windows. Змінити курсор миші в основному режимі. Показати приховані папки і файли. Ввімкнути режим очікування через 30 хвилин. Встановити розрішення екрану на 1024x768.

3.7. Контрольні питання

1. Які характерні відмінності Windows NT від систем серії Windows 9x?
2. У чому відмінність оновлення від установки нової копії?
3. Опишіть характерні особливості установки з системи MS-DOS якщо компакт-диск не є завантажувальним.
4. Що знаходиться в папці «Адміністрування»?
5. З якими файловими системами працює Windows NT?
6. Навіщо потрібен «Майстер установки і видалення обладнання», як його викликати?

ЛАБОРАТОРНА РОБОТА № 4
НАЛАГОДЖЕННЯ МЕРЕЖІ В ОПЕРАЦІЙНІЙ СИСТЕМІ WINDOWS NT ПРИ РОБОТІ У СЕРЕДОВИЩІ WMWARE

4.1. Мета лабораторної роботи

Одержати практичні навики по налагодженню мережі в операційній системі Windows NT.

4.2. Постановка задачі

Одержати теоретичні і практичні навики по налагодженню однорангової мережі на основі створеної віртуальної машини з операційною системою Windows NT.

Закріпити попередні навики роботи з командною строчкою та пакетними файлами.

4.3. Порядок виконання роботи

1. Вивчити, користуючись рекомендованою літературою, конспектом лекцій, і методичними вказівками до даної роботи, наступні питання:
- загальні відомості про домашні і малі офісні мережі;
- налагодження параметрів мережі в WMWare;
- робота з командним рядком в Windows NT.
2. Визначити за допомогою відомих мережних команд конфігурацію існуючої фізичної мережі.
3. Використовуючи створену в попередній лабораторній роботі віртуальну машину з системою Windows NT і отримані навики, налагодити мережу і виконати завдання згідно варіанту.
4. Виконані кроки описати в звіті.

4.4. Зміст звіту

1. Назва і мета лабораторної роботи.
2. Опис послідовності дій при настройці мережі, а також при роботі з командним рядком в операційній системі Windows NT згідно варіанту.
3. Висновки про виконану роботу.

4.5. Теоретичні відомості

В даній роботі проводиться налагодження однорангової мережі. Однорангова мережа, яку також називають робочою групою, дозволяє комп'ютерам взаємодіяти один з одним безпосередньо і не потребує сервера. Вона застосовується при об'єднанні не більше десяти комп'ютерів. Комп'ютери у робочій групі рівні за можливостями і користуються загальними ресурсами. Кожен користувач вирішує сам, які дані або периферійні пристрої локального комп'ютера можна надати для загального доступу в мережі.

Встановлення віртуальної мережної карти під WM Ware складається з наступних кроків:

- у меню «Команди» вибрати пункт «Правка параметрів віртуального пристрою»;
- у вікні, що з'явилося, додати мережну карту;
- вибрати тип мережі «Сполучено мостом».

Налагодження параметрів мережі. Потребує наступної послідовності дій:

- у Windows NT в меню «Пуск» вибрати пункт «Налагодження/Мережа і віддалений доступ до мережі»;
- активувати утиліту «Створення нового підключення» і слідувати вказанням майстра, відповідаючи на запити.

Змінити настройки вже налагодженої мережі можна в пункті «Підключення по локальній мережі/Властивості/Протокол Інтернету».

Робота з командами Windows NT. Для роботи з командами Windows NT використовується командний рядок операційної системи, дискового менеджера подібного до Total Commander чи Far Manager, або командна оболонка Cmd. Остання може використовуватись для створення і редагування файлів сценаріїв, що дозволяє автоматизувати виконання звичайних завдань. Для виконання в командній оболонці складних сценаріїв також використовується сервер сценаріїв CScript. Сценарії приймають всі команди, доступні з командного рядка.

Повний список команд які підтримує інтерпретатор cmd можна побачити, якщо ввести в його командному рядку *help*. Інформацію про окрему команду можна отримати, якщо ввести в командному рядку *help*, а потім назву потрібної команди, наприклад *help xcopy*.

Для виконання кількох команд можна використовувати наступні конструкції.

Таблиця 4.1 – Правила групування команд

Символ та синтаксис	Визначення
команда1 **&** команда2	Використовується для розділення декількох команд в одному командному рядку. У Cmd.exe виконується перша команда, потім друга команда.

Продовження таблиці 4.1

Символ та синтаксис	Визначення
команда1 **&&** команда2	Запускає команду, що стоїть за символом &&, тільки якщо команда, що стоїть перед цим символом була виконана успішно. У Cmd.exe виконується перша команда. Друга команда виконується, тільки якщо перша була виконана успішно.
команда1 || команда2	Запускає команду, що стоїть за символом ||, тільки якщо команда, що стоїть перед символом || не була виконана. У Cmd.exe виконується перша команда. Друга команда виконується, тільки якщо перша не була виконана (одержаний код помилки перевищує нуль).
(команда1 **&** команда2**)**	Використовується для угрупування або вкладення команд.
команда1 параметр1**,** параметр2	Використовується для розділення параметрів команди.

Відображення і налагодження параметрів протоколу IP для Windows NT здійснюється за допомогою команди **ipconfig**.

Мережні команди системи Windows NT починаються із слова **net**. Вони дозволяють контролювати, настроювати, запускати або зупиняти різні мережні служби, а також змінювати параметри роботи служб, користувачів або груп безпосередньо з командного рядка.

Команда net view служить для розгляду списку доменів, комп'ютерів або загальних ресурсів на даному комп'ютері. Коли команда використовується без параметрів, то відображає список комп'ютерів поточного домена або мережі.

4.6. Варіанти завдань

Використовуючи віртуальний комп'ютер із встановленою у попередній роботі операційною системою Windows NT, задати мережне ім'я комп'ютера відповідно до прізвища, а робочу групу – NGU.

Таблиця 4.2 – Варіанти завдань до лабораторної роботи

№ п/п.	№ завдання зі списку	№ п/п.	№ питання зі списку
1	1, 6, 11, 14, 17	11	1, 6, 12, 15, 18
2	2, 7, 12, 15, 18	12	2, 7, 13, 16, 19
3	3, 8, 13, 16, 19	13	3, 8, 11, 14, 17
4	4, 9, 11, 14, 17	14	4, 9, 12, 15, 18
5	5, 10, 12, 15, 18	15	5, 10, 13, 16, 19
6	1, 6, 13, 16, 19	16	1, 6, 11, 14, 17

Продовження таблиці 4.2

№ п/п.	№ завдання зі списку	№ п/п.	№ питання зі списку
7	2, 7, 11, 14, 17	17	2, 7, 12, 15, 18
8	3, 8, 12, 15, 18	18	3, 8, 13, 16, 19
9	4, 9, 13, 16, 19	19	4, 9, 11, 14, 17
10	5, 10, 11, 14, 17	20	5, 10, 12, 15, 18

Список завдань:

1. Налагодити мережу і відкрити загальний доступ до диска C:\;
2. Налагодити мережу і відкрити загальний доступ до диска D:\;
3. Налагодити мережу і відкрити загальний доступ до дисків C:\ і D:\;
4. Налагодити мережу і відкрити загальний доступ до дисковода A:\;
5. Налагодити мережу і відкрити загальний доступ до дисків C:\ і A:\;
6. Встановити число користувачів для даних ресурсів не більше 1;
7. Встановити число користувачів для даних ресурсів не більше 2;
8. Встановити число користувачів для даних ресурсів не більше 3;
9. Встановити число користувачів для даних ресурсів не більше 4;
10. Встановити число користувачів для даних ресурсів не більше 5;
11. Дозволити доступ до ресурсів: читання;
12. Дозволити доступ до ресурсів: повний доступ;
13. Дозволити доступ до ресурсів: зміни;
14. Дозволити користування ресурсом: всім з робочої групи;
15. Дозволити користування ресурсом: адміністраторам;
16. Дозволити користування ресурсом: адміністраторам і всім з робочої групи;
17. За допомогою командного рядка проглянути список робочих станцій підключених до мережі;
18. За допомогою командного рядка переглянути налагодження мережевого підключення;
19. За допомогою командного рядка протестувати мережеву картку своєї віртуальної машини.

4.7. Контрольні питання

1. Яким чином можна змінити настройки вже налагодженої мережі?
2. Які методи роботи з командами Windows NT існують?
3. Які конструкції використовуються для групування команд?
4. Як визначити конфігурацію існуючої мережі?
5. Чим відрізняються мережні команди Windows NT від інших команд?

ЛАБОРАТОРНА РОБОТА 5
ВИКОРИСТАННЯ ПРОГРАМНИХ ЗАСОБІВ КОПІЮВАННЯ ФАЙЛІВ

5.1 Мета лабораторної роботи

Вивчення можливостей операційної системи з копіювання різними способами.

5.2 Постановка задачі

Дослідити продуктивність програми, що виконують копіювання файлів трьома способами:

- з використанням бібліотеки С;
- з використанням Windows;
- з використанням допоміжної функції Windows – CopyFile.

5.3 Порядок виконання роботи

1. Ознайомитись з теоретичними відомостями за п.5.5 та вивчіть роботу наведених програм, використовуючи довідкову систему Visual Studio або доступний на робочому місці у навчальній лабораторії Інтернет;
2. Використовуючи знання, отримані з курсу програмування створити у середовищі Visual Studio три програми, що виконують копіювання файлів трьома способами:
   - з використанням бібліотеки С;
   - з використанням Windows;
   - з використанням допоміжної функції Windows – CopyFile;
3. Проведіть хронометраж для кожної програми з тривалості копіювання не менше ніж трьох файлів великого розміру (десятки мегабайт). Результати представте у вигляді таблиці. Зробіть висновок стосовно продуктивності способів.

5.4 Зміст звіту

1. Назва, мета й завдання лабораторної роботи.
2. Короткий конспект теоретичних відомостей за п.1.5
3. Таблиці змінних та алгоритми роботи програм з коментарями (для оформлення рекомендується алгоритмів використовувати програму).
4. Алгоритм дій програміста для додавання кнопки і її оброблювача у діалогове вікно.
5. Результати хронометражу програм копіювання файлів.
6. Висновок за виконаною роботою.

5.5 Теоретичні відомості

Копіювання файлів з використанням стандартної бібліотеки С.

Стандартна бібліотека С підтримує об'єкти потоків введення/виведення FILE. Приклад програми наведено нижче.

Програма 5.1. cpC: копіювання файлів з використанням бібліотеки С

```
/* cp файл1 файл2: Копіювати файл1 в файл2. */
#include <stdio.h>
#include <errno.h>
#define BUF_SIZE 256


int main(int argc, char *argv[]) {
 FILE *in_file, *out_file;
 char rec [BUF_SIZE];
 size_t bytes_in, bytes_out;
 if (argc != 3) {
  printf("Використання: cpC файл1 файл2\n");
  return 1;
 }
 in_file = fopen(argv [1], "rb");
 if (in_file == NULL) {
  perror(argv[1]);
  return 2;
 }
 out_file = fopen(argv [2], "wb");
 if (out_file == NULL) {
  perror(argv [2]);
  return 3;
 }
 /* Обробити вхідний файл з одного запису за один раз. */
 while ((bytes_in = fread(rec, 1, BUF_SIZE, in_file)) > 0) {
  bytes_out = fwrite(rec, 1, bytes_in, out_file);
  if (bytes_out != bytes_in) {
   perror("Неусувана помилка запису.");
   return 4;
  }
 }
 fclose (in_file);
 fclose (out_file);
 return 0;
}
```

Приклад 5.1 може слугувати наочною ілюстрацією ряду загальноприйнятих припущень і угод програмування, які не завжди застосовуються в Windows, а саме:

1. Об'єкти відкритих файлів ідентифікуються покажчиками на структури FILE (в UNIX використовуються цілочисельні дескриптори файлів). Вказівником NULL відповідає неіснуючий об'єкт. По суті, покажчики є різновидом дескрипторів об'єктів відкритих файлів.

2. У виклику функції fopen вказується, яким чином має оброблятися файл - як текстовий або як двійковий. У текстових файлах містяться специфічні для кожної системи послідовності символів, що використовуються, наприклад, для позначення кінця рядка. У багатьох системах, включаючи Windows, в процесі виконання операцій введення/виводу кожна з таких послідовностей автоматично перетвориться в нульовий символ, який інтерпретується в мові С як мітка кінця рядка, і навпаки. У нашому прикладі обидва файли відкриваються як виконавчі.

3. Діагностика помилок реалізується за допомогою функції perror, яка, в свою чергу, отримує інформацію щодо природи збою, що виникає при виконанні функції fopen, з глобальної змінної errno. Замість цього можна було б скористатися функцією ferror, що повертає код помилки, асоційований не з системою, а з об'єктом FILE.

4. Функції fread і fwrite повертають кількість оброблених байтів безпосередньо, а не через аргумент, що робить істотний вплив на логіку організації програми. Невід'ємне значення, що повертається говорить про успішне виконання операції читання, тоді як нульове - про спробу читання мітки кінця файлу.

5. Функція fclose може застосовуватися лише до об'єктів типу FILE (аналогічне твердження справедливе і по відношенню до дескрипторів файлів UNIX).

6. Операції введення/виведення здійснюються в синхронному режимі, тобто перш ніж програма зможе виконуватися далі, вона повинна дочекатися завершення операції введення/виводу.

7. Для виведення повідомлень про помилки зручно використовувати входить в бібліотеку З функцію введення/виведення printf, яка навіть буде використана в першому прикладі Windows-програми.

Перевагою реалізації, використовує бібліотеку С, є її переносимість на платформи UNIX, Windows, а також інші системи, які підтримують стандарт ANSI C. Проте, в цьому випадку програми змушені обмежуватися синхронними операціями вводу/виводу, хоча вплив цього обмеження буде дещо ослаблений використанням потоків Windows.

Як і їх еквіваленти в UNIX, програми, засновані на функціях для роботи з файлами, що входять в бібліотеку С, здатні виконувати операції довільного доступу до файлів (з використанням функції fseek або, в разі текстових файлів, функцій fsetpos і fgetpos), але це є вже стелею складності для функцій вводу/виводу стандартної бібліотеки С, вище якого вони піднятися не можуть. Разом з тим, Visual C ++ надає нестандартні розширення, здатні, наприклад, підтримувати блокування файлів. Нарешті, бібліотека С не дозволяє керувати захистом файлів.

Якщо простий синхронний файловий або консольний ввод/вивід – це все, що потрібно, то для написання переносних програм, які будуть виконуватися під керуванням Windows, слід використовувати бібліотеку С.

Копіювання файлів з використанням Windows. У програмі 5.2 вирішується те ж завдання копіювання файлів, але робиться це за допомогою Windows API.

Програма 5.2. cpW: копіювання файлів з використанням Windows

```
/* cpW файл1 файл2: Копіювати файл1 в файл2.  */
    #include <windows.h>
    #include <stdio.h>
    #define BUF_SIZE 256


    int main (int argc, LPTSTR argv []) {
     HANDLE hIn, hOut;
     DWORD nIn, nOut;
     CHAR Buffer [BUF_SIZE];
     if (argc != 3) {
      printf ("Використання: cpW файл1 файл2\n");
      return 1;
     }
     hIn = CreateFile(argv [1], GENERIC_READ, 0, NULL, OPEN_EXISTING, 0,
NULL);
     if (hIn == INVALID_HANDLE_VALUE) {
      printf("Неможливо відкрити вхідний файл. Помилка: %x\n", GetLastError());
      return 2;
     }
     hOut = CreateFile(argv[2], GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
     if (hOut == INVALID_HANDLE_VALUE) {
      printf("Неможливо відкрити вхідний файл. Помилка: %x\n", GetLastError());
      return 3;
     }
     while (ReadFile(hIn, Buffer, BUF_SIZE, &nIn, NULL) && nIn > 0) {
      WriteFile(hOut, Buffer, nIn, &nOut, NULL);
      if (nIn != nOut) {
       printf ("Фатальна помилка запису: %x\n", GetLastError());
       return 4;
      }
     }
     CloseHandle(hIn);
     CloseHandle(hOut);
     return 0;
    }
```

Приклад 5.2 ілюструє особливості програмування в середовищі Windows, а саме:

1. У програму завжди включається файл <windows.h>, в якому містяться всі необхідні визначення функцій і типів даних Windows.

2. Всі об'єкти Windows ідентифікуються змінними типу Handle, причому для більшості об'єктів можна використовувати одну і ту ж загальну функцію CloseHandle.

3. Рекомендується закривати всі раніше відкриті дескриптори, якщо в необхідність в них відпала, щоб звільнити ресурси. У той же час, при завершенні процесів пов'язані з ним дескриптори автоматично закриваються ОС, і якщо не залишається жодного дескриптора, що посилається на який-небудь об'єкт, то ОС знищує цей об'єкт і звільняє відповідні ресурси. (Примітка. Як правило, файли подібним способом не знищуються.)

4. Windows визначає численні символічні константи і прапори. Зазвичай вони мають довгі імена, нерідко пояснюють призначення даного об'єкта. Як типовий приклад можна привести імена INVALID_HANDLE_VALUE і GENERIC_READ.

5. Функції ReadFile і WriteFile повертають булеві значення, а не кількості оброблених байтів, для передачі яких використовуються аргументи функцій. Це певним чином змінює логіку організації роботи циклів. Нульове значення лічильника байтів вказує на спробу читання мітки кінця файлу і не вважається помилкою.

6. Функція GetLastError дозволяє отримувати в будь-якій точці програми коди системних помилок, що подаються значеннями типу DWORD. У програмі 1.2 показано, як організувати виведення генеруюємих Windows текстових повідомлень про помилки.

7. У даному прикладі захист вихідного файлу не забезпечується.

8. Такі функції, як CreateFile, володіють багатим набором додаткових параметрів, але в даному прикладі використані значення за замовчуванням.

Копіювання файлів з використанням допоміжної функції Windows. Для підвищення зручності роботи в Windows передбачено багато допоміжних функцій (convenience functions), які, об'єднуючи в собі кілька інших функцій, забезпечують виконання завдань програмування, що часто зустрічаються. У деяких випадках використання цих функцій може призводити до підвищення продуктивності. У даному випадку, застосування функції CopyFile суттєво спрощується програма копіювання файлів (програма 1.3). Крім того, це позбавляє від необхідності піклуватися про буфер, розмір якого в двох попередніх програмах довільно встановлювався рівним 256.

Програма 5.3.cpCF: копіювання файлів з використанням допоміжної функції Windows:

```
/* cpCF файл1 файл2: Копіювати файл1 в файл2. */
#include <windows.h>
#include <stdio.h>
```

```
int main (int argc, LPTSTR argv []) {
 if (argc != 3) {
  printf ("Використання: cpCF файл1 файл2\n");
  return 1;
 }
 if (!CopyFile(argv[1], argv[2], FALSE)) {
  printf("Помилка при виконанні функції CopyFile: %x\n", GetLastError());
  return 2;
 }
 return 0;
}
```

5.6 Контрольні питання

1. int main(int argc, char *argv[]) – навіщо тут використано аргументи ?
2. Яка перевага у використанні стандартних функцій С порівняно з іншими способами?
3. Для здійснення простого копіювання файлів який шлях кращий – з використанням функцій WinAPI чи допоміжних функцій Windows?
4. Які параметри задаються у функції CreateFile() у прикладі?
5. Чому прикладі 1.2 типи змінних наведені великими літерами?

ЛАБОРАТОРНА РОБОТА 6
ВИКОРИСТАННЯ РОЗШИРЕНИХ ПРОГРАМНИХ ЗАСОБІВ РОБОТИ З ФАЙЛАМИ

6.1 Мета лабораторної роботи

Вивчення можливостей розширених програмних засобів по керуванню файлами.

6.2 Постановка задачі

Виконати наступні вправи з розширеного керування файлами:

- написати програму виведення кількох файлів на стандартний пристрій виводу, модифікувавши функцію CatFile у програмі 5.1 таким чином, щоб при зв'язуванні дескриптора стандартного виведення з консоллю у ній використовувалася не функція WriteFile, а функція WriteConsole;
- написати програму перетворення текстового файла з кодуванням ASCII в Unicode;
- параметри виклику функції CreateFile дозволяють задавати різні характеристики способу доступу до файлу, що може бути використано для підвищення продуктивності програм. Як приклад можна привести параметр FILE_FLAG_SEQUENTIAL_SCAN. Використайте цей прапор у програмі 5.3 і з'ясуйте, чи призведе це до поліпшення показників продуктивності при роботі з

файлами великого розміру. Дослідіть також вплив прапора FILE_FLAG_NO_BUFFERING;

- написати програму виведення на стандартний пристрій змісту поточного каталогу.

6.3 Порядок виконання роботи

1. Ознайомитись з теоретичними відомостями за п.6.5 та вивчіть роботу наведених програм, використовуючи довідкову систему Visual Studio або доступний на робочому місці у навчальній лабораторії Інтернет;
2. Використовуючи знання, отримані з курсу програмування створити у середовищі Visual Studio три програми:
   - програму виведення кількох файлів на стандартний пристрій виводу;
   - написати програму перетворення текстового файла з кодуванням ASCII в Unicode;
   - написати програму виведення на стандартний пристрій змісту поточного каталогу.
3. Модифікуйте функцію CatFile у програмі виведення кількох файлів таким чином, щоб при зв'язуванні дескриптора стандартного виведення з консоллю у ній використовувалася не функція WriteFile, а функція WriteConsole;
4. Параметри виклику функції CreateFile дозволяють задавати різні характеристики способу доступу до файлу, що може бути використано для підвищення продуктивності програм. Як приклад можна привести параметр FILE_FLAG_SEQUENTIAL_SCAN. Використайте цей прапор у функції Asc2Un і з'ясуйте, чи призведе це до поліпшення показників продуктивності при роботі з файлами великого розміру. Дослідіть також вплив прапора FILE_FLAG_NO_BUFFERING;

6.4 Зміст звіту

1. Назва, мета й завдання лабораторної роботи.
2. Короткий конспект теоретичних відомостей за п.6.5
3. Таблиці змінних та алгоритми роботи програм з коментарями (для оформлення рекомендується алгоритмів використовувати програму).
4. Результати хронометражу програм копіювання файлів.
5. Висновок за виконаною роботою.

6.5 Теоретичні відомості

Загальний файл для включення у проект. Призначений для зменшення обсягу кодування у завданнях:

Файл Envirmnt.h

```
//#define UNICODE
#undef UNICODE
#ifdef UNICODE
#define _UNICODE
```

```
#endif
#ifndef UNICODE
#undef _UNICODE
#endif
//#define _STATICLIB

/ * Визначте _STATICLIB, якщо створюєте * /

/ * Або компонуєте статичну бібліотеку. * /
#define LANG_DFLT LANG_ENGLISH
#define SUBLANG_DFLT SUBLANG_ENGLISH_US
```

Виведення на консоль повідомлень і підказок для користувача.
PrintMsg: допоміжні функції виведення на консоль повідомлень і очікування відповіді від користувача.

```
/* PrintMsg.c: ConsolePrompt, PrintStrings, PrintMsg */
#include "Envirmnt.h" /* У цьому файлі встановлюються директиви #define і #undef для UNICODE.*/
#include <windows.h>
#include <stdarg.h>

BOOL PrintStrings (HANDLE hOut, ...)
/* Запис повідомлень у буфер екрана консолі.*/
{
 DWORD MsgLen, Count;
 LPCTSTR pMsg;
 va_list pMsgList; /* Рядок поточного повідомлення.*/
 va_start (pMsgList, hOut); /* Почати обробку повідомлень.*/
 while ((pMsg = va_arg(pMsgList, LPCTSTR)) != NULL) {
  MsgLen = _tcslen(pMsg);
   /* Функція WriteConsole може застосовуватися тільки з дескриптором буфера екрана консолі.*/
  if (!WriteConsole(hOut, pMsg, MsgLen, &Count, NULL)
     /* Функція WriteFile викликається тільки у разі невдалого завершення функції WriteConsole.*/
     && !WriteFile(hOut, pMsg, MsgLen * sizeof (TCHAR), &Count, NULL))
return FALSE;
 }
 va_end(pMsgList);
 return TRUE;
}
```

```
BOOL PrintMsg(HANDLE hOut, LPCTSTR pMsg)
/* Версія PrintStrings для виведення одиночного повідомлення.*/
{
 return PrintStrings(hOut, pMsg, NULL);
}

BOOL ConsolePrompt(LPCTSTR pPromptMsg, LPTSTR pResponse, DWORD
MaxTchar, BOOL Echo)
/* Вивести на консоль підказку для користувача і отримати від нього
відповідь.*/
{
 HANDLE hStdIn, hStdOut;
 DWORD TcharIn, EchoFlag;
 BOOL Success;
 hStdIn = CreateFile(_T("CONIN$"), GENERIC_READ | GENERIC_WRITE,
0, NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
 hStdOut = CreateFile(_T("CONOUT$"), GENERIC_WRITE, 0, NULL,
OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
 EchoFlag = Echo ? ENABLE_ECHO_INPUT : 0;
 Success = SetConsoleMode(hStdIn, ENABLE_LINE_INPUT | EchoFlag |
ENABLE_PROCESSED_INPUT) &&
      SetConsoleMode (hStdOut, ENABLE_WRAP_AT_EOL_OUTPUT |
ENABLE_PROCESSED_OUTPUT) &&
     PrintStrings (hStdOut, pPromptMsg, NULL) &&
     ReadConsole (hStdIn, pResponse, MaxTchar, &TcharIn, NULL);
 if (Success) pResponse [TcharIn – 2] = '\0';
 CloseHandle (hStdIn);
 CloseHandle (hStdOut);
 return Success;
}
```

Зверніть увагу, що при обчисленні повертається функцією значення булевої змінної Success, яке служить індикатором успішності виконання, в програмі використовується так зване "скорочене" обчислення логічних виразів в напрямку зліва направо; тому, як тільки при обчисленні частини виразу, розташованої зліва від будь-якої з операцій логічного "і" (&&), як результат буде отримано значення FALSE, інша частина виразу, розташована праворуч від даної операції, обчислюватися не буде, оскільки результат обчислення за все вираження в цілому виявляється зумовленим. Для отримання більш докладної інформації про можливі помилки можна скористатися функцією GetLastError.

У даній функції повідомлення про помилки виводяться; їх виведеня, якщо це буде необхідно, можна передбачити в визиваючій програмі.

При використанні функції WriteConsole разом з дескриптором, який не є дескриптором консолі, її виконання буде завершено з помилкою. У зв'язку з цим попередній запит властивостей дескриптора не є обов'язковим. Функція скористається консольним режимом лише в тому випадку, якщо зазначений в її виклику дескриптор дійсно пов'язаний з консоллю.

Крім того, функція ReadConsole повертає керуючі символи повернення каретки і переходу на новий рядок, що диктує необхідність вставки додаткових нульових символів після символів повернення каретки в відповідних місцях.

Обробка помилок. Функція FormatMessage перетворює простий номер повідомлення в описову повідомлення, що представляє собою фразу англійською або будь-якому іншому з безлічі можливих мов, і повертає розмір повідомлення.

Нижче представлена корисна функція ReportError, призначена для обробки помилок, за своїми можливостями вона аналогічна функції perror, що входить до складу бібліотеки С. Функція ReportError передає у вихідний буфер повідомлення у вигляді, який визначається першим аргументом і, або припиняє виконання з кодом виходу помилково, або здійснює звичайне повернення керування, в залежності від значення другого аргументу. Третій аргумент визначає, чи повинні відображатися системні повідомлення про помилки.

Зверніть увагу на аргументи функції FormatMessage. В якості одного з параметрів використовується значення, яке повертається функцією GetLastError, а на необхідність генерації повідомлення системою вказує прапор. Сгенерироване повідомлення залишиться в буфері, що виділяється функцією, а відповідна адреса повертається в параметрі. Є також інші параметри, для яких вказані значення за замовчуванням. Мова повідомлень може бути задана як під час компіляції, так і під час виконання.

У наведеній нижче програмі вводиться заголовок EvryThng.h. Як випливає з самого його назви, цей файл включає у себе файли <windows.h>, Envirmnt.h і <stdarg.h>. Крім того, у ньому описані такі використні функції, як PrintMsg, PrintStrings і ReportError.

Функція Report Error, призначена для виведення повідомлень про помилки при виконанні системних викликів

```
#include "EvryThng.h"

VOID ReportError(LPCTSTR UserMessage, DWORD ExitCode, BOOL
PrintErrorMsg)
/* Універсальна функція для виведення повідомлень про системні
помилки.*/
{
 DWORD eMsgLen, LastErr = GetLastError();
```

```
LPTSTR lpvSysMsg;
HANDLE hStdErr = GetStdHandle(STD_ERROR_HANDLE);
PrintMsg(hStdErr, UserMessage);
if (PrintErrorMsg) {
 eMsgLen = FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM, NULL, LastErr,
MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPTSTR)&lpvSysMsg, 0, NULL);
 PrintStrings (hStdErr, _T("\n"), lpvSysMsg, _T("\n"), NULL);
 /* Звільнити блок пам'яті, що містить повідомлення про помилку. */
 HeapFree(GetProcessHeap(), 0, lpvSysMsg); /* См. гл. 5. */
}
if (ExitCode > 0) ExitProcess (ExitCode);
else return;
}
```

Виведення кількох файлів на стандартний пристрій виведення.

Програма 6.1. cat: виведення декількох файлів на стандартний пристрій виводу

```
#include "EvryThng.h"
#define BUF_SIZE 0x200

static VOID CatFile(HANDLE, HANDLE);
int _tmain(int argc, LPTSTR argv[]) {
 HANDLE hInFile, hStdIn = GetStdHandle(STD_INPUT_HANDLE);
 HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
 BOOL DashS;
 int iArg, iFirstFile;
 /* Змінна DashS буде встановлена тільки в разі завдання параметра "-s" у
командному рядку. */
 /* iFirstFile — індекс першого вхідного файлу у списку argv[]. */
 iFirstFile = Options(argc, argv, _T("s"), &DashS, NULL);
 if (iFirstFile == argc) { /* Відсутність вхідних файлів у списку аргументів.*/
  /* Використовувати стандартний пристрій введення. */
  CatFile(hStdIn, hStdOut);
  return 0;
 }
 /* Обробити кожен вхідний файл. */
 for (iArg = iFirstFile; iArg < argc; iArg++) {
    hInFile = CreateFile(argv [iArg], GENERIC_READ, 0, NULL,
OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
```

```
    if (hInFile == INVALID_HANDLE_VALUE && !DashS) ReportError
(_T("Cat: помилка при відкритті файлу "), 1, TRUE);
     CatFile (hInFile, hStdOut);
     CloseHandle (hInFile);
    }
    return 0;
   }

   /* Функція, що виконує всю роботу:
   /* читає вхідні дані і копіює їх на стандартні пристрої виведення.*/
   static VOID CatFile(HANDLE hInFile, HANDLE hOutFile) {
    DWORD nIn, nOut;
    BYTE Buffer [BUF_SIZE];
    while (ReadFile(hInFile, Buffer, BUF_SIZE, &nIn, NULL) && (nIn != 0) &&
WriteFile(hOutFile, Buffer, nIn, &nOut, NULL));
    return;
   }
```

Перетворення символів з ASCII у Unicode

Програма 6.2. atou: перетворення файлу з висновком повідомлень про помилки

```
   #include "EvryThng.h"
   #define BUF_SIZE 256

   BOOL Asc2Un(LPCTSTR, LPCTSTR, BOOL);
   int _tmain(int argc, LPTSTR argv[]) {
    DWORD LocFileIn, LocFileOut;
    BOOL DashI = FALSE;
    TCHAR YNResp[3] = _T("y");
    /* Отримати параметри командного рядка і індекс вхідного файлу. */
    LocFileIn = Options(argc, argv, _T("i"), &DashI, NULL);
    LocFileOut = LocFileIn + 1;
    if (DashI) { /* Чи існує вихідний файл?*/
     /* Узагальнена версія функції access, що здійснює перевірку існування
файлу.*/
     if (_taccess(argv[LocFileOut], 0) == 0) {
      _tprintf(_T("Перезаписати існуючий файл? [y/n]"));
      _tscanf(_T ("%s"), &YNResp);
      if (lstrcmp(CharLower(YNResp), YES) != 0) ReportError(_T("Відмова від
перезапису "), 4, FALSE);
     }
```

```
}
/* Ця функція побудована на основі функції CopyFile. */
Asc2Un(argv[LocFileIn], argv [LocFileOut], FALSE);
return 0;
}

BOOL Asc2Un(LPCTSTR fIn, LPCTSTR fOut, BOOL bFailIfExists)
/* Функція копіювання файлів з перетворенням з ASCII у Unicode. Функція побудована на основі функції CopyFile. */
{
HANDLE hIn, hOut;
DWORD dwOut, nIn, nOut, iCopy;
CHAR aBuffer[BUF_SIZE];
WCHAR uBuffer [BUF_SIZE];
BOOL WriteOK = TRUE;
hIn = CreateFile(fin, GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
/* Визначити поведінку функції CreateFile, якщо вихідний файл вже існує.*/
dwOut = bFailIfExists ? CREATE_NEW : CREATE_ALWAYS;
hOut = CreateFile(fOut, GENERIC_WRITE, 0, NULL, dwOut, FILE_ATTRIBUTE_NORMAL, NULL);
while (ReadFile(hIn, aBuffer, BUF_SIZE, &nIn, NULL) && nIn > 0 && WriteOK) {
 for (iCopy = 0; iCopy < nIn; iCopy++)
  /* Перетворити кожен символ.*/
  uBuffer[iCopy] = (WCHAR)aBuffer [iCopy];
 WriteOK = WriteFile(hOut, uBuffer, 2 * nIn, &nOut, NULL);
}
CloseHandle(hIn);
CloseHandle(hOut);
return WriteOK;
}
```

Друк поточного каталогу

Програма 6.3. pwd: друк поточного каталогу

```
#include "EvryThng.h"
#define DIRNAME_LEN MAX_PATH + 2

int _tmain(int argc, LPTSTR argv[]) {
```

```
    TCHAR pwdBuffer [DIRNAME_LEN];
    DWORD LenCurDir;
    LenCurDir = GetCurrentDirectory(DIRNAME_LEN, pwdBuffer);
    if (LenCurDir == 0) ReportError(_T("Не вдається отримати шлях."), 1,
TRUE);
    if (LenCurDir > DIRNAME_LEN) ReportError(_T("Занадто довгий шлях."),
2, FALSE);
    PrintMsg(GetStdHandle(STD_OUTPUT_HANDLE), pwdBuffer);
    return 0;
   }
```

6.6 Контрольні питання

1. Розкажіть, як працює програма у файлі Envirmnt.h?
2. Як використовується функція WriteConsole?
3. Для чого у функції ReportError () використано GetStdHandle?
4. Як працює функція CatFile() у прикладі?
5. Як працює строка dwOut = bFailIfExists ? CREATE_NEW : CREATE_ALWAYS; у прикладі?

ЛАБОРАТОРНА РОБОТА 7
ВДОСКОНАЛЕНІ ЗАСОБИ ДЛЯ РОБОТИ З ФАЙЛАМИ, КАТАЛОГАМИ ТА ЗНАЙОМСТВО З РЕЄСТРОМ

7.1 Мета лабораторної роботи

Вивчення можливостей програмних засобів для роботи з файлами, каталогами, реєстром.

7.2 Постановка задачі

Виконати наступні вправи:

- удоскональте програму 7.1 таким чином, щоб в виведений список включалися також поточний (".") і батьківський (".."). Крім того, додайте опції, що дозволяють поряд з датою і часом останньої зміни відображати дату і час створення файлу, а також дату і час останнього доступу до нього.
- напишіть програму, яка реалізує команду rm, що дозволяє видаляти файли, змінивши для цього функцію ProcessItem в програмі 7.1;
- удоскональте програму 7.2 таким чином, щоб новий час створення файлу можна було вказувати в командному рядку. Мітки дати і часу мають формат MMddhhmm [yy], де MM - місяці, dd - дні, hh - години, mm - хвилини, yy - року. Двох цифр для позначення року буде недостатньо, тому передбачте для зазначення року чотири розряду;

- скомпілюйте, запустіть та вивчіть роботу програми 7.3;
- напишіть програму, яка блокує заданий файл і утримує його в блокованому стані протягом тривалого часу. Скориставшись будь-яким текстовим редактором, спробуйте отримати доступ до файлу (використовуйте текстовий файл) в період дії блокування. Програма повинна пропонувати користувачеві задати вид блокування для текстового файлу.

7.3 Порядок виконання роботи

1. Ознайомитись з теоретичними відомостями за п.7.5 та вивчіть роботу наведених програм, використовуючи довідкову систему Visual Studio або доступний на робочому місці у навчальній лабораторії Інтернет;
2. Використовуючи знання, отримані з курсу програмування створити у середовищі Visual Studio п’ять програми, згідно п.7.3.

7.4 Зміст звіту

1. Назва, мета й завдання лабораторної роботи.
2. Короткий конспект теоретичних відомостей за п.7.5
3. Таблиці змінних та алгоритми роботи програм з коментарями (для оформлення рекомендується алгоритмів використовувати програму).
4. Описання контрольних прикладів та результати роботи написаних програм.
5. Висновок за виконаною роботою.

7.5 Теоретичні відомості

Виведення списку файлів і обхід дерева каталогів. Коректна робота представленої програми можлива лише з використанням відносних повних шляхів файлів. Програма 7.1:

```
#include "EvryThng.h"
BOOL TraverseDirectory(LPCTSTR, DWORD, LPBOOL);
DWORD FileType(LPWIN32_FIND_DATA);
BOOL ProcessItem(LPWIN32_FIND_DATA, DWORD, LPBOOL);

int _tmain(int argc, LPTSTR argv[]) {
 BOOL Flags [MAX_OPTIONS], ok = TRUE;
 TCHAR PathName [MAX_PATH +1], CurrPath [MAX_PATH + 1];
 LPTSTR pSlash, pFileName;
 int i, FileIndex;
 FileIndex = Options(argc, argv, _T("R1"), &Flags[0], &Flags[1], NULL);
 I* " Розібрати "шаблон пошуку" на "батьківську частину"і ім'я файлу. */
 GetCurrentDirectory(MAX_PATH, CurrPath); / * Зберегти поточний шлях доступу. */
```

```
if (argc < FileIndex +1) /* Шлях доступу не вказано. Використовувати поточний каталог. */
 ok = TraverseDirectory(_T("*"), MAX_OPTIONS, Flags);
else for (i = FileIndex; i < argc; i++) {
 /* Обробити всі шляхи, зазначені в командному рядку.*/
 ok = TraverseDirectory(pFileName, MAX_OPTIONS, Flags) && ok;
 SetCurrentDirectory(CurrPath);
 /* Відновити каталог.*/
 }
 return ok ? 0 : 1;
}

static BOOL TraverseDirectory(LPCTSTR PathName, DWORD NumFlags, LPBOOL Flags)
/* Обхід дерева каталогів; виконати функцію ProcessItem для кожного випадку збігу.*/
/* PathName: відносне або абсолютне ім'я каталогу, що проглядається.*/
{
 HANDLE SearchHandle;
 WIN32_FIND_DATA FindData;
 BOOL Recursive = Flags[0];
 DWORD FType, iPass;
 TCHAR CurrPath[MAX_PATH + 1];
 GetCurrentDirectory(MAX_PATH, CurrPath);
 for (iPass = 1; iPass <= 2; iPass++) {
  /* Прохід 1: виведення списку файлів. * /
  /* Прохід 2: обхід дерева каталогів (якщо задана опція -R). */
  SearchHandle = FindFirstFile(PathName, &FindData);
  do {
   FType = FileType(&FindData);
   /* Файл чи каталог?*/
   if (iPass == 1) /* Вивести ім'я і атрибути файлу. */
    ProcessItem(&FindData, MAX_OPTIONS, Flags);
   if (FType == TYPE_DIR && iPass == 2 && Recursive) {
    /* Обробити підкаталог. */
    _tprintf(_T ("\n%s\\%s:"), CurrPath, FindData.cFileName);
    /* Підготовка до обходу каталогу.*/
    SetCurrentDirectory(FindData.cFileName);
    TraverseDirectory(_T("*"), NumFlags, Flags);
    /* Рекурсивний виклик.*/
```

```
        SetCurrentDirectory(_T(".."));
       }
      } while (FindNextFile(SearchHandle, &FindData));
      FindClose (SearchHandle);
     }
     return TRUE;
    }

    static BOOL ProcessItem(LPWIN32_FIND_DATA pFileData, DWORD
NumFlags, LPBOOL Flags)
    /* Виводить список атрибутів файлу або каталогу.*/
    {
     const TCHAR FileTypeChar[] = {' ', 'd'};
     DWORD FType = FileType(pFileData);
     BOOL Long = Flags[1];
     SYSTEMTIME LastWrite;
     if (FType != TYPE_FILE && FType != TYPE_DIR) return FALSE;
     _tprintf(_T ("\n"));
     if (Long) { /* Чи зазначено в командному рядку параметр "-1"? */
      _tprintf(_T("%c"), FileTypeChar[FType – 1]);
      _tprintf(_T("%10d"), pFileData->nFileSizeLow);
      FileTimeToSystemTime(&(pFileData->ftLastWriteTime), &LastWrite);
        _tprintf(_T(" %02d/%02d/%04d %02d:%02d:%02d"), LastWrite.wMonth,
LastWrite.wDay, LastWrite.wYear, LastWrite.wHour, LastWrite.wMinute,
LastWrite.wSecond);
     }
     _tprintf(_T(" %s"), pFileData->cFileName);
     return TRUE;
    }

    static DWORD FileType(LPWIN32_FIND_DATA pFileData)
    /* Підтримувані типи файлів - TYPE_FILE: файл; TYPE_DIR: каталог;
TYPE_DOT: каталоги. або ..*/
    {
     BOOL IsDir;
     DWORD FType;
     FType = TYPE_FILE;
     IsDir = (pFileData->dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
!= 0;
```

```
 if (IsDir) if (lstrcmp(pFileData->cFileName, _T(".")) == 0 || lstrcmp(pFileData-
>cFileName, _T("..")) == 0) FType = TYPE_DOT;
  else FType = TYPE_DIR;
  return FType;
 }
```

Установка міток дати і часу файлу.

Програма реалізує UNIX-команду touch, призначену для зміни коду захисту файлів і оновлення міток часу до поточних значень системного часу. Необхідно розширити можливості функції touch таким чином, щоб нові значення міток часу можна було вказувати в параметрах командного рядка. Програма 7.2:

```
 #include "EvryThng.h"
 int _tmain(int argc, LPTSTR argv[]) {
  SYSTEMTIME SysTime;
  FILETIME NewFileTime;
  LPFILETIME pAccessTime = NULL, pModifyTime = NULL;
  HANDLE hFile;
  BOOL Flags[MAX_OPTIONS], SetAccessTime, SetModTime, CreateNew;
  DWORD CreateFlag;
  int i, FileIndex;
  FileIndex = Options(argc, argv, _T("amc"), &Flags[0], &Flags[1], &Flags[2],
NULL);
  SetAccessTime = !Flags[0];
  SetModTime = !Flags[1];
  CreateNew = !Flags[2];
  CreateFlag = CreateNew ? OPEN_ALWAYS : OPEN_EXISTING;
  for (i = FileIndex; i < argc; i++) {
   hFile = CreateFile(argv[i], GENERIC_READ | GENERIC_WRITE, 0, NULL,
CreateFlag, FILE_ATTRIBUTE_NORMAL, NULL);
   GetSystemTime(&SysTime);
   SystemTimeToFileTime(&SysTime, &NewFileTime);
   if (SetAccessTime) pAccessTime = &NewFileTime;
   if (SetModTime) pModifyTime = &NewFileTime;
   SetFileTime(hFile, NULL, pAccessTime, pModifyTime);
   CloseHandle(hFile);
  }
  return 0;
 }
```

Виведення списку розділів і вмісту реєстру. Програма 7.3:

```
/* lsReg [параметри] підрозділ */
#include "EvryThng.h"
BOOL TraverseRegistry(HKEY, LPTSTR, LPTSTR, LPBOOL);
BOOL DisplayPair(LPTSTR, DWORD, LPBYTE, DWORD, LPBOOL);
BOOL DisplaySubKey (LPTSTR, LPTSTR, PFILETIME, LPBOOL);


int _tmain(int argc, LPTSTR argv[]) {
 BOOL Flags[2], ok = TRUE;
 TCHAR KeyName[MAX_PATH + 1];
 LPTSTR pScan;
 DWORD i, KeyIndex;
 HKEY hKey, hNextKey;
 /* Таблиця зумовлених імен та дескрипторів розділів.*/
 LPTSTR PreDefKeyNames[] = {
  _T("HKEY_LOCAL_MACHINE"), _T("HKEY_CLASSES_ROOT"),
  _T("HKEY CURRENT USER"), _T ("HKEY CURRENT CONFIG"), NULL
 };
 HKEY PreDefKeys[] = {
                HKEY_LOCAL_MACHINE,          HKEY_CLASSES_ROOT,
HKEY_CURRENT_USER, HKEY_CURRENT_CONFIG
 };
 KeyIndex = Options(argc, argv, _T("Rl"), &Flags[0], &Flags[1], NULL);
 /* "Розібрати" шаблон пошуку на "розділ" і "підрозділ".*/
 /* Відновити розділ.*/
 pScan = argv[KeyIndex];
 for (i = 0; *pScan != _T('\\') && *pScan != _T('\0'); pScan++, i++) KeyName
[i] = *pScan;
 KeyName[i] = _T('\0');
 if (*pScan == _T('\\')) pScan++;
 /* Перетворити визначене ім'я розділу в відповідний HKEY.*/
 for (i = 0; PreDefKeyNames [i] != NULL && _tcscmp(PreDefKeyNames[i],
KeyName) != 0; i++);
 hKey = PreDefKeys[i];
 RegOpenKeyEx(hKey, pScan, 0, KEY_READ, &hNextKey);
 hKey = hNextKey;
 ok = TraverseRegistry(hKey, argv[KeyIndex], NULL, Flags);
 return ok ? 0 : 1;
}
```

```
BOOL TraverseRegistry(HKEY hKey, LPTSTR FullKeyName, LPTSTR SubKey, LPBOOL Flags)
/* Здійснити обхід розділів і підрозділів реєстру, якщо заданий параметр -R.*/
{
 HKEY hSubK;
 BOOL Recursive = Flags[0];
 LONG Result;
 DWORD ValType, Index, NumSubKs, SubKNameLen, ValNameLen, ValLen;
 DWORD MaxSubKLen, NumVals, MaxValNameLen, MaxValLen;
 FILETIME LastWriteTime;
 LPTSTR SubKName, ValName;
 LPBYTE Val;
 TCHAR FullSubKName[MAX_PATH + 1];
 /* Відкрити дескриптор розділу.*/
 RegOpenKeyEx(hKey, SubKey, 0, KEY_READ, &hSubK);
 /* Визначити максимальний обсяг інформації щодо розділу і розподілити пам'ять.*/
 RegQueryInfoKey(hSubK, NULL, NULL, NULL, &NumSubKs, &MaxSubKLen, NULL, &NumVals, &MaxValNameLen, &MaxValLen, NULL, &LastWriteTime);
 SubKName = malloc (MaxSubKLen+1); /* Розмір без урахування завершального нульового символу.*/
 ValName = malloc(MaxValNameLen+1); /* Врахувати нульовий символ.*/
 Val = malloc(MaxValLen); /* Розмір в байтах.*/
 /* Перший прохід: пари "ім'я-значення". */
 for (Index = 0; Index < NumVals; Index++) {
  ValNameLen = MaxValNameLen + 1; /* Встановлюється кожен раз!*/
  ValLen = MaxValLen + 1;
  RegEnumValue(hSubK, Index, ValName, &ValNameLen, NULL, &ValType, Val, &ValLen);
  DisplayPair(ValName, ValType, Val, ValLen, Flags);
 }
 /* Другий прохід: підрозділи.*/
 for (Index = 0; Index < NumSubKs; Index++) {
  SubKNameLen = MaxSubKLen + 1;
  RegEnumKeyEx(hSubK, Index, SubKName, &SubKNameLen, NULL, NULL, NULL, &LastWriteTime);
  DisplaySubKey(FullKName, SubKName, &LastWriteTime, Flags);
```

```
 if (Recursive) {
  _stprintf(FullSubKName, _T("%s\\%s"), FullKName, SubKName);
  TraverseRegistry(hSubK, FullSubKName, SubKName, Flags);
 }
}
_tprintf(_T("\n"));
free(SubKName);
free(ValName);
free(Val);
RegCloseKey(hSubK);
return TRUE;
}
```

BOOL DisplayPair(LPTSTR ValueName, DWORD ValueType, LPBYTE Value, DWORD ValueLen, LPBOOL Flags)

```
/* Функція, що відображає пари "ім'я-значення".*/
{
LPBYTE pV = Value;
DWORD i;
_tprintf(_T("\nValue: %s = "), ValueName);
switch (ValueType) {
case REG_FULL_RESOURCE_DESCRIPTOR: /* 9: опис обладнання. */
case REG_BINARY: /* 3: Будь-які двійкові дані.*/
 for (i = 0; i < ValueLen; i++, pV++) _tprintf (_T (" %x"), *pV);
 break;
case REG_DWORD: /* 4: 32-бітове число.*/
 _tprintf(_T ("%x"), (DWORD)*Value);
 break;
```

case REG_MULTI SZ: /*7: масив рядків, що завершуються нульовим символом.*/

```
case REG_SZ: /* 1: рядок, що завершується нульовим символом.*/
 _tprintf(_T("%s"), (LPTSTR)Value);
 break;
/* … Кілька інших типів … */
}
return TRUE;
}
```

```
BOOL DisplaySubKey(LPTSTR KeyName, LPTSTR SubKeyName,
PFILETIME pLastWrite, LPBOOL Flags) {
    BOOL Long = Flags[1];
    SYSTEMTIME SysLastWrite;
    _tprintf(_T("\nSubkey: %s"), KeyName);
    if (_tcslen(SubKeyName) > 0) _tprintf (_T ("\\%s "), SubKeyName);
    if (Long) {
     FileTimeToSystemTime(pLastWrite, &SysLastWrite);
     _tprintf(_T("%02d/%02d/%04d  %02d:%02d:%02d"), SysLastWrite.wMonth,
SysLastWrite.wDay, SysLastWrite.wYear, SysLastWrite.wHour,
SysLastWrite.wMinute, SysLastWrite.wSecond);
    }
    return TRUE;
   }
```

7.6 Контрольні питання

1. Як Ви вдосконалили програму 7.1, щоб виконати завдання?
2. Які засоби Ви використали, щоб програмно видаляти файли?
3. Як Ви вдосконалили програму 7.2, щоб виконати завдання?
4. Що виконує програма 7.3 і які засоби при цьому використовує?
5. Які засоби реалізації і які види блокування використані у Вашій програмі блокування файлів?

ПЕРЕЛІК ПОСИЛАНЬ

1) Шеховцов В.А. Операційні системи. – К.: Видавнича група BHV, 2005. – 576 с.

2) Дж.М. Харт Системное программирование в среде Windows. – М.: Издательский дом "Вильямс", 2005. – 587 с.

3) Таненбаум Э., Вудхалл А. Операционные системы: Разработка и реализация. – М. – С.Пб.: Питер, 2006. – 576 с.

Ткаченко Сергій Миколайович

СИСТЕМНЕ ПРОГРАМУВАННЯ

Методичні рекомендації
до виконання лабораторних робіт
студентами галузі знань 12 Інформаційні технології
спеціальності 123 Комп'ютерна інженерія

Видано в редакції автора

Підписано до друку __.__.2019. Формат 30х42/4.
Папір офсетний. Ризографія. Ум. друк. арк. __.
Обл.-вид. арк. __. Тираж 20 прим. Зам. № .

Національний технічний університет
"Дніпровська політехніка"
49005, м. Дніпропетровськ, просп. Д.Яворницького, 19