Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод адміністрування драйверів та служб на локальному хості під керуванням Windows

(тема)

Виконав:

студент II курсу, групи СПм-20-3

Шебанов Є.О.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доцент Голубничий Д.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

В.о. зав. кафедри ЕОМ _____ Волк М.О.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____”_____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Шебанову Євгенію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Метод адміністрування драйверів та служб на локальному хості під керуванням Windows

затверджена наказом по університету від “ 24 ” березня 2022 р. № 413 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2022 р.

3. Вхідні дані до роботи 1) Операційна система – Windows;
2) Об'єкт моніторингу операцій вводу-виводу – диспетчер управління службами та диспетчер Plug and Play;
3) Мова програмування – асемблер

4. Перелік питань, що потрібно опрацювати у роботі
1) Аналіз принципів побудови підсистеми вводу/виводу Windows
2) Програмні методи управління службами і драйверами
3) Метод адміністрування службами й драйверами
4) Програмна модель адміністрування драйверів та сервісів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (слайдів)

Слайд-презентація – 15 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд способів побудови драйверів та служб	29.03.22 – 04.04.22	
	на вузлах комп'ютерної мережі		
2	Вибір та обґрунтування методики дослідження	05.04.22 – 12.04.22	
3	Вибір інструментальних засобів	13.04.22 – 19.04.22	
4	Розробка моделі драйверів та служб	20.04.22 – 26.04.22	
5	Проведення експериментів	27.04.22 – 04.05.22	
6	Оформлення матеріалів кваліфікаційної роботи	05.05.22 – 10.05.22	
7	Подання кваліфікаційної роботи керівникові та	11.05.22 – 12.05.22	
	її попередній захист		
8	Подання кваліфікаційної роботи на	13.05.22 – 17.05.22	
	рецензування		

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доцент Голубничий Д.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 123 с., 24 рис., 6 табл., 2 дод., 11 джерел.

ОПЕРАЦІЙНА СИСТЕМА, СЕРВІС, ДРАЙВЕР, ДИСПЕТЧЕР, АДМІНІСТРУВАННЯ.

Метою кваліфікаційної роботи є покращення процесу адміністрування сервісів та драйверів на локальному вузлі під керівництвом операційної системи Windows за рахунок створення програми-монітору за їх схованими від користувача операціями вводу/виводу.

Об’єктом дослідження є процеси адміністрування сервісів та драйверів довільного хосту локальної мережі.

Предметом дослідження є підсистема вводу/виводу операційної системи Windows.

У ході виконання кваліфікаційної роботи були розглянуті принципи побудови підсистеми вводу/виводу Windows, проаналізована архітектура операційної системи Windows. Задля полегшення розуміння підсистеми вводу/виводу було створено 4 розділи, які описують певні компоненти цієї системи. Основна увага приділяється інструментальним засобам управління службами, їх порівнянню з існуючими на ринку програмного забезпечення. Основною частиною роботи є створення спеціального системного програмного забезпечення, яке отримало назву "Управління сервісами Windows". Основною мовою програмування обрана мова асемблеру.

ABSTRACT

Master’s thesis: 123 pages, 24 figures, 6 tables, 2 appendices, 11 sources.

OPERATING SYSTEM, SERVICE, DRIVER, DISPATCHER, ADMINISTRATION.

The major goal of this thesis is to improve the process of administration of services and drivers on the local node under the guidance of the Windows operating system by creating a monitor program for their hidden from the user I/O operations.

The object of the study is the processes of administration of services and drivers of any local network host.

The subject of the study is the I/O subsystem of the Windows operating system.

In order to work the principles of construction of the Windows I/O subsystem were considered, the architecture of the Windows operating system was analyzed. To facilitate understanding of the I/O subsystem, 4 sections have been created that describe certain components of this system. The focus is on service management tools, comparing them to existing software on the market. The main part of the work is the creation of special system software, which is called "Windows Service Management". The main programming language is the assembly language.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1 АНАЛІЗ ПРИНЦИПІВ ПОБУДОВИ ПІДСИСТЕМИ ВВОДУ/ВИВОДУ WINDOWS 11
1.1 Аналіз архітектури ОС Windows 11
1.2 Підсистема вводу/виводу ОС Windows 15
1.3 Системне адміністрування сервісів та драйверів 18
2 ПРОГРАМНІ МЕТОДИ УПРАВЛІННЯ СЛУЖБАМИ І ДРАЙВЕРАМИ 22
2.1 Програмний застосунок Microsoft System Center Operations Manager Enterprise 22
2.2 Програмний застосунок ServiceUtility 24
2.3 Програмний застосунок ServiceView 26
2.4 Програмний застосунок Windows Services Controller 27
3 МЕТОД АДМІНІСТРУВАННЯ СЛУЖБАМИ Й ДРАЙВЕРАМИ 30
3.1 Адміністрування служб з боку користувача 30
3.2 Адміністрування служб з боку операційної системи 34
3.2.1 Програми управління службами, що поставляють із Windows 36
3.2.2 Функції API по управлінню службами в SCP 43
3.3 Облікові записи служб 49
4 ПРОГРАМНА МОДЕЛЬ АДМІНІСТРУВАННЯ ДРАЙВЕРІВ ТА СЕРВІСІВ 53
4.1 Загальна характеристика програмної моделі 53
4.2 Загальна характеристика програмної моделі 54
4.3 Вхідні дані, робота програми і результати її виконання 59
4.4 Рекомендації користувачеві 63

ВИСНОВКИ 71
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 72
ДОДАТОК А Графічний матеріал кваліфікаційної роботи 73
ДОДАТОК Б Лістинги основних файлів програмної моделі 82
Б.1 Лістинг головного файлу проекту main.asm 82
Б.2 Лістинг файлу опису та функцій main.inc до проекту 122

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система

ПЗ – програмне забезпечення

ACPI – удосконалений інтерфейс керування конфігурацією й енергоспоживанням (англ., Advanced Configuration Power-management Interface)

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

DDK – набір із засобів розробки (англ., Driver Development Kit)

DPC – відкладений виклик процедури (англ., Deferred Procedure Call)

EFS – шифрована файлова система (англ., Encrypted File System)

HAL – рівень абстрагування від обладнання (англ., Hardware Abstraction Layer)

IRP – пакет запиту вводу/виводу (англ., I/O Request Packet)

ISR – процедура обслуговування переривання (англ., Interrupt Service Routine)

MDL – список дескрипторів пам'яті (англ., Memory Descriptor List)

PnP – підключи та працюй (англ.,Plug-and-Play)

SDK – комплект засобів розробки (англ., Software Development Kit)

SCM – диспетчера управління сервісом (англ.,Service Control Manager)

SCP – програми управління сервісом (англ., Service Control Program)

VDD – драйвери віртуальних пристроїв (англ., Virtual Driver Device)

WDM – модель драйверів Windows (англ., Windows Driver Model)

WMI – інструменти управління Windows (англ., Windows Management Instrumentation)

ВСТУП

Сьогодні інформаційні технології все більше проникають у наше життя. Сучасний рівень апаратних засобів, які використовуються в автоматизації, давно "розмив" чітку границю між комп'ютерами верхнього рівня й контролерами з погляду системного програмного забезпечення. Апаратні ресурси сучасних контролерів дозволяють використовувати не тільки оптимізовані до мінімальних апаратних вимог операційні системи, наприклад Windows CE .NET, але й стандартні ОС класу Windows або Linux.

В той же час, з розповсюдженням комп'ютерних систем виникає необхідність їх захисту від посягань зловмисників. Однак, крім безпосереднього обмеження доступу, інколи необхідно контролювати, як працює той або інший апаратний або програмний драйвер, які дії виконуються встановленими службами, які ресурсі задіяні для цього і т.ін.

В України, як і в цілому світі, домінуюче положення серед стандартних операційних систем (ОС) у сфері автоматизації займають ОС Windows компанії Microsoft. Тому, в якості предмета кваліфікаційної роботи була обрана саме ця ОС. Використання стандартної універсальної ОС Windows дозволяє максимально застосовувати досвід програмування, накопичений фахівцями, які звикли працювати зі знайомим набором API (Application Programming Interface – інтерфейс прикладного програмування). У числі розв'язуваних програмістами завдань, крім властиво написання прикладної програми, може виникнути необхідність у написанні драйверів, наприклад, якщо за якимись причинами драйвери під ОС Windows не поставляються. Наприклад, плати вводу/виводу у форматі MicroPC фірми Octagon Systems або Fastwel, або, якщо мова йде про плату власної розробки. При цьому, часто використовуються драйвері, які не відповідають в повній мірі апаратному обладнанню. Результатом роботи таких драйверів є помилки, збої та відкази ОС, аж до повної втрати нею працездатності.

Для уникнення цієї ситуації як правило застосовуються засоби аудиту або моніторингу ОС. Але іноді рівень деталізації звітів системи аудиту не достатній для поставлених задач. Наприклад, необхідно написати програму, яка б записувала у файл виклики всіх функцій, які є цікавими, конкретної програми, і так само всіх вхідних і вихідним аргументів цих функцій. Крім того, ця програма могла б змінити або доповнити функціональність певних функцій а, отже, вплинути на логіку роботи конкретної програми.

Незважаючи на велику кількість технічної літератури, присвяченої програмуванню в ОС Windows, проблеми написання драйверів практично не висвітлюються. Безумовно, це питання досить специфічне для програмістів загального плану, але для фахівців, що працюють у сфері автоматизації виробництва, ця тема викликає значну зацікавленість. Навіть якщо програмістові не прийдеться безпосередньо займатися написанням драйверів, більш глибоке розуміння функціонування цієї підсистеми ОС Windows, дасть більше чітке уявлення про можливості функціонування ОС у цілому. Це дозволяє реально оцінювати тимчасові можливості стандартних драйверів і при необхідності одержання кращих тимчасових показників проектованої системи в цілому написати свій, можливо менш функціональний, але більш швидкий драйвер.

Тому, основна мета кваліфікаційної роботи полягає в покращенні процесу адміністрування сервісів та драйверів на локальному вузлі під керівництвом операційної системи Windows за рахунок створення програми-монітору за їх схованими від користувача операціями вводу/виводу.

1 АНАЛІЗ ПРИНЦИПІВ ПОБУДОВИ ПІДСИСТЕМИ ВВОДУ/ВИВОДУ WINDOWS

1.1 Аналіз архітектури ОС Windows

В Windows існує чітке розмежування двох областей в оперативній пам'яті і режимах процесора для виконавчого коду:

- область виконавчого коду у непривілейованому режимі роботи процесора (режимі користувача) для застосунків користувача та частини компонентів ОС;
- область виконавчого коду операційної системи у привілейованому режимі процесора (режимі ядра).

Під областю виконавчого коду треба розуміти області завантаження (діапазон адрес) в оперативній пам'яті обчислювальної системи. Якщо розглядати 32-розрядну ОС Windows, то всім застосункам доступно до 4 Гбайт (232) лінійні адресні простори. Найчастіше в системі встановлений істотно менший обсяг фізичної пам'яті, але для працюючих програм це непомітно. Спеціальні системні механізми забезпечують можливість віртуальної присутності 4 Гбайт пам'яті в системі. Розподіл 4 Гбайт віртуального адресного простору між застосунками користувача та системними програмами здійснюється нарівно: перші 2 Гбайт користувачеві, інші – системний адресний простір.

Виконавчий код у режимі користувача має обмеження на доступ до системних ресурсів, зокрема, на прямий доступ до обладнання. Це пов'язане з бажанням забезпечити більш стійке функціонування системи при наявності помилок у програмах користувачів. Треба враховувати, що Windows проектувалася як багатозадачна та багатокористувачева система. Тому, крах одного застосунка не повинен приводити до краху ОС і, отже, до краху інших застосунків користувача, запущених на виконання в цій системі. Застосунки

ОС і інші програми, що виконуються в режимі ядра, мають повний доступ до всіх ресурсів системи. Спрощена схема архітектури Windows [1] наведена на рисунку 1.1.

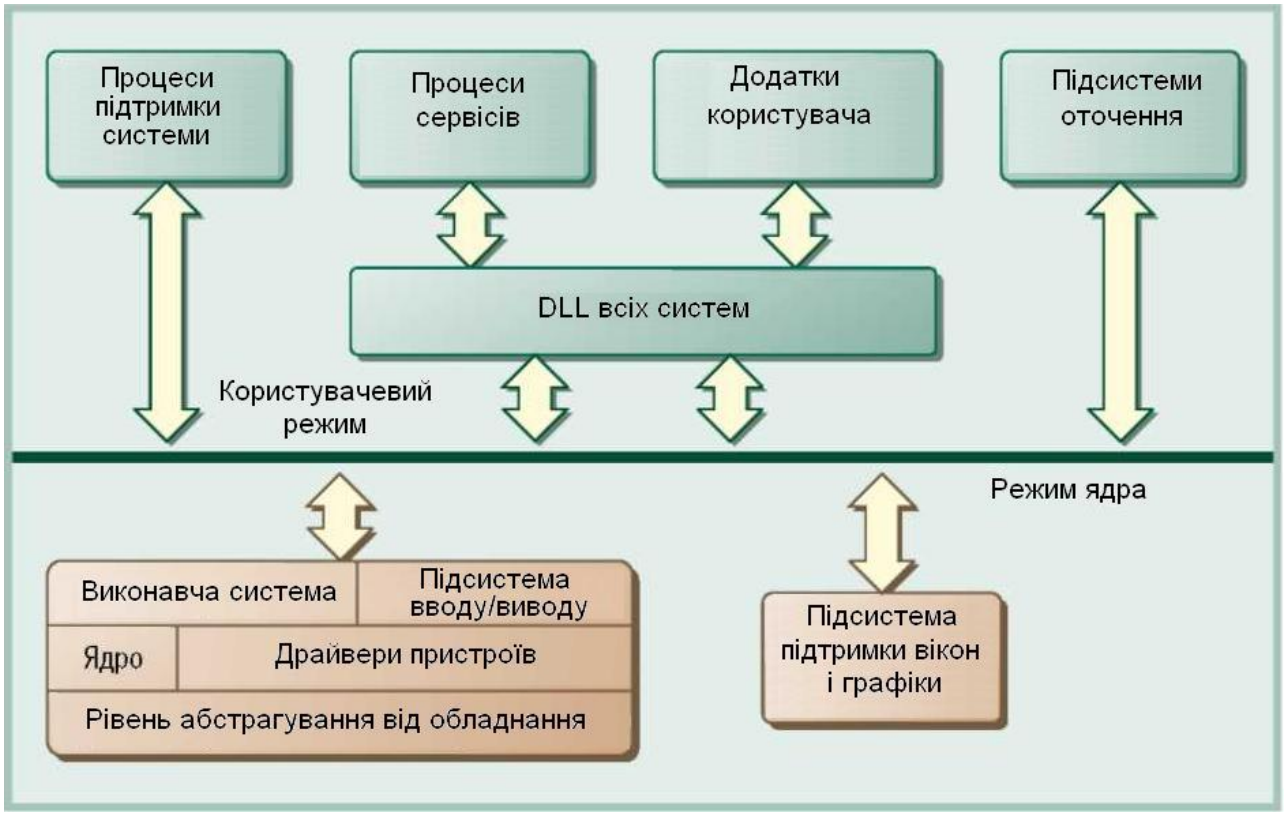


Рисунок 1.1 – Спрощена схема архітектури Windows

У режимі користувача функціонують не тільки прикладні програми користувача, але й частина процесів самої ОС. До компонентів операційної системи, що працюють в режимі користувача, відносяться:

- деякі процеси підтримки системи, наприклад процес обробки входу в систему (Winlogon);
- процеси Windows-сервісів. У вигляді сервісів оформлені як деякі системні сервіси (наприклад, Task Scheduler), так і окремі компоненти прикладних програм, наприклад Microsoft SQL Server, а також деякі драйвери;
- застосунки користувача. На сучасний момент вони можуть бути

таких типів: Win32, Win64 (в 64-бітово версії ОС), POSIX;

- підсистеми оточення. Це частина операційної системи (програмні оболонки), що надає застосункам користувача певний для конкретної підсистеми набір функцій. Windows забезпечує роботу із трьома підсистемами оточення: Win32, POSIX і OS/2.

На сьогодні в Windows 10 за замовчуванням не поставляються інші підсистеми оточення.

До компонентів ОС, які працюють у режимі ядра, відносяться:

- виконавча система, що забезпечує базові сервіси при керуванні пам'яттю, процесами й потоками, вводом/виводом і т.ін.;

- ядро, яке містить узагальнений набір функцій ОС, що приховує розходження між апаратними платформами (на різних етапах розвитку ОС NT підтримувалися не тільки процесори Intel, але й MIPS, Alpha AXP, Motorola PowerPC). Ядро надає процедури/функції й базові об'єкти, які застосовуються виконавчою системою й драйверами для реалізації структур і функцій більш високого рівня. До таких функцій відносяться планування потоків, диспетчеризація переривань, синхронізація процесів тощо;

- підсистема вводу/виводу;

- драйвери пристроїв;

- рівень абстрагування від обладнання (Hardware Abstraction Layer, HAL) - набір низькорівневих функцій (близько 92), що забезпечує стандартний інтерфейс взаємодії з апаратно-залежними елементами для функцій, які викликаються компонентами ядра, драйверами та виконавчою системою, що дозволяє абстрагуватися від того, на якій конкретно елементній базі (чипі контролера переривання, контролера ПДП) реалізоване виконання доступу до шини, таймеру тощо;

- підсистема підтримки вікон і графіки.

Драйвери пристроїв в Windows для підтримки перенесення не звертаються до обладнання прямо, а використають функції, надані HAL. Умовно драйвери пристроїв режиму ядра діляться на наступні основні

категорії:

- драйвери файлової системи (наприклад мережні редиректори й сервери). Не варто розуміти буквально, що мова йде тільки про файлову систему ОС. Насправді багато фізичних пристроїв (наприклад COM-порти) представляються в системі як файли, і звертання до них здійснюється за допомогою виклику функцій, як до звичайних файлів, але зі специфічними параметрами. Далі вже драйвери файлової системи, що одержали запит на ввід-висновок, визначають, про який пристрій мова йде, і викликають відповідному фізичному пристрою драйвери наступного рівня;
- драйвери з підтримкою Plug-and-Play (PnP) і ACPI (advanced configuration power-management interface - удосконалений інтерфейс керування конфігурацією й енергоспоживанням);
- драйвери, що не підтримують специфікації PnP і ACPI (наприклад драйвери протоколів TCP/IP, IPX/SPX і т.д.), які розширюють функціональність системи, надаючи доступ з режиму користувача до системних сервісам і драйверам режиму ядра.

У свою чергу, у кожній з категорій є групи драйверів, які розрізняються залежно від моделі пристрою і місця драйверів у ланцюжку обробки запиту на обслуговування операцій вводу/виводу.

На сьогодні для розробки драйверів використовується набір із засобів розробки – Driver Development Kit (DDK), який дозволяє програмістам створювати драйвери для пристроїв за певною технологією або для певної платформи (програмної або програмно-апаратної). Назва походить від більш загального терміна SDK (Software Development Kit), яким позначають комплекти для розробки програм взагалі, не тільки драйверів.

Створення драйвера можливо і без використання DDK, проте DDK містить засоби, що спрощують розробку драйвера (наприклад, готові приклади і шаблони коду), що забезпечують сумісність драйвера з операційною системою, а також установку і тестування драйвера.

Модель DDK передбачає існування трьох типів драйверів:

- драйвер шини;
- функціональний драйвер;
- драйвер фільтра.

У рамках узагальнення поняття пристрою в Windows існує поняття класу пристроїв. Введення цього рівня абстрагування супроводжується неминучою появою типу драйверів, відповідальних за обслуговування пристроїв одного класу (наприклад, DVD-ROM), і драйверів, відповідальних за рішення того або іншого рівня взаємодії з конкретним обладнання. У рамках цього розподілу існують драйвери:

- класів пристроїв;
- порти-драйвери;
- мініпорт-драйвери.

Драйвери пристроїв в ОС Windows можуть працювати як у режимі ядра, так і в режимі користувача. До останнього відносяться:

- драйвери віртуальних пристроїв (VDD);
- драйвери принтерів.

Найважливішим компонентом виконавчої системи, відповідальним за зв'язок із пристроями, є підсистема вводу/виводу.

1.2 Підсистема вводу/виводу ОС Windows

Підсистема вводу/виводу відповідає за обробку запитів вводу/виводу, виконує наступні завдання:

- забезпечення продуктивності операцій вводу/виводу;
- можливість використання асинхронного вводу/виводу;
- підтримку декількох файлових систем;
- модульну архітектуру, з можливістю додавання нових файлових систем і пристроїв;
- підтримку PnP для пристроїв;
- надання розширених можливостей, наприклад кешування;

- захист спільно використовуваних ресурсів.

Функціональність підсистеми вводу/виводу полягає в:

- стандартних засобах безпеки й іменування пристроїв для захисту поділюваних пристроїв;
- високопродуктивному асинхронному пакетному вводу/виводу для підтримки масштабування;
- сервісах для написання драйверів пристроїв на високорівневій мові програмування;
- підтримці багаторівневої моделі й розширюваності для додавання драйверів, що модифікують поводження інших драйверів або пристроїв без внесення змін у них;
- динамічному завантаженні та вивантаженні драйверів пристроїв;
- підтримці PnP, завдяки якій система знаходить і встановлює драйвери для нового обладнання, а потім виділяє їм потрібні апаратні ресурси;
- керуванні електроживленням;
- підтримці множини файлових систем;
- підтримці Windows Management Instrumentation (WMI) і засобів діагностики.

Підсистема вводу/виводу має модульну структуру (як і всі інші компоненти Windows) і складається з наступних компонентів (рисунок 1.2):

- диспетчер вводу/виводу;
- програмний інтерфейс застосунків вводу/виводу (I/O API);
- драйвери файлових систем;
- диспетчер PnP;
- диспетчер електроживлення;
- інші драйвери (наприклад: драйвер дисків, клавіатури);
- реєстр.

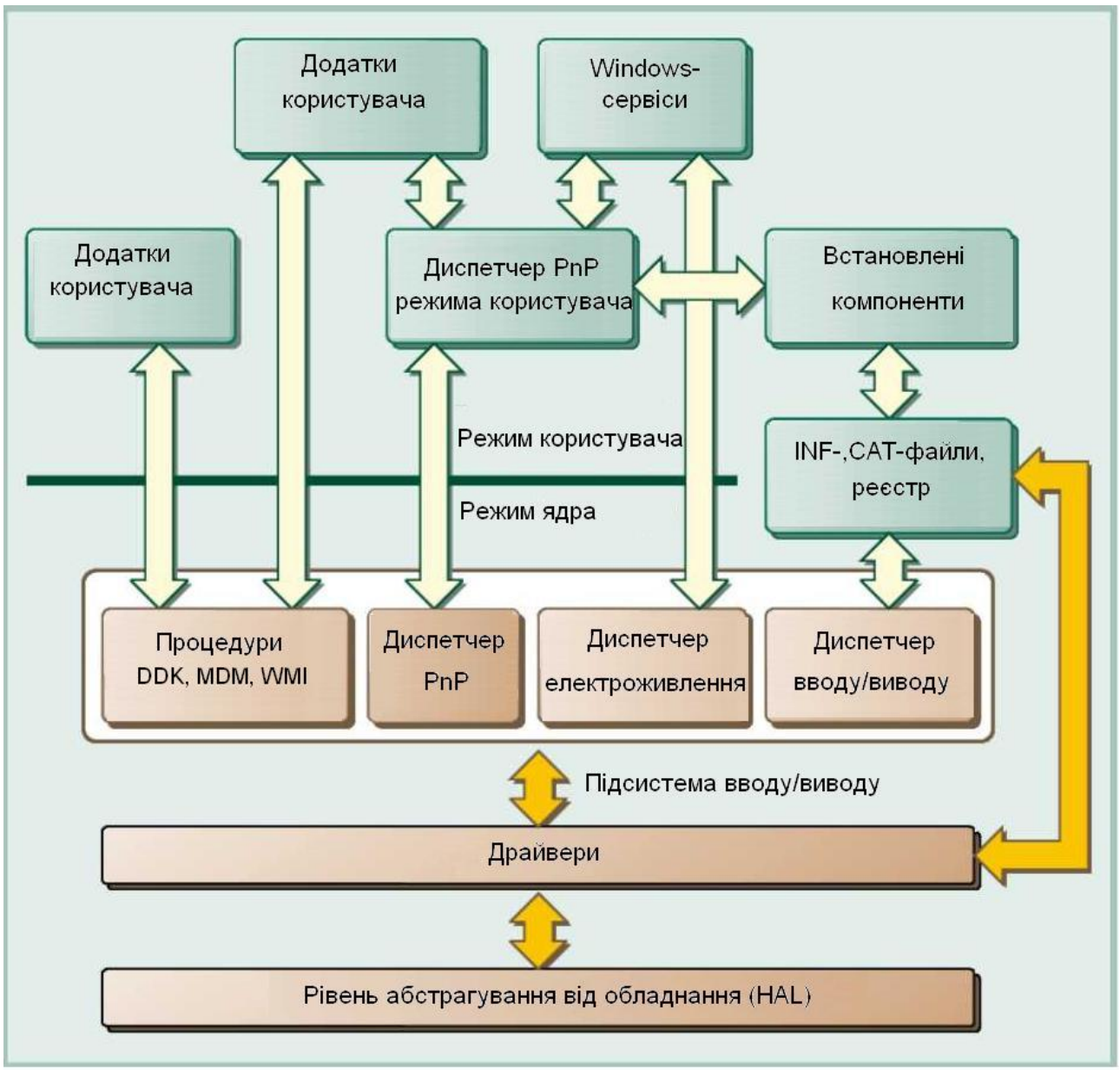


Рисунок 1.2 – Компоненти підсистеми вводу/виводу

Висновок. Важливими моментами забезпечення функціональності є можливості динамічного завантаження і вивантаження драйверів, узагальнений вид формованих структур запитів на ввід/вивід і диспетчеризація. Одним з інструментів реєстрації, запуску, зупинки і вивантаження драйверів служить механізм керування сервісами, який буде розглянутий в наступному розділі.

1.3 Системне адміністрування сервісів та драйверів

Засоби мережного і системного адміністрування ніколи не займали домінуючих позицій у комп’ютерних інформаційних системах. Традиційна допоміжна роль, яка їм відводилась, призвела до того, що структура і функції програмного забезпечення (ПЗ) даного класу виявилися в прямій залежності від архітектури обчислюваних систем і еволюціонували разом із ними.

Тим часом комп’ютерні користувачі інформаційних технологій поступово приходять до усвідомлення значущості того впливу, який засоби адміністрування роблять на бізнес-процеси. Цей факт, з одного боку, змушує більш строго ставитися до формулювання критеріїв, котрим повинне відповідати керуюче ПЗ, а з іншого, дозволяє припустити, що вже незабаром йому може бути відведене більш почесне місце в структурі інформаційної системи.

Еволюція концепцій адміністрування відбувалася не тільки уздовж архітектурної осі, але й у просторі тих об'єктів, що поступово втягувалися в "сферу компетенції" засобів управління.

Функціональна область управління, що відносяться до цієї сфери, чітко визначена в специфікаціях ISO [7]:

- управління ресурсами (врахування, контроль використання ресурсів, виставляння рахунків за використані ресурси й обмеження доступу до них);
- управління конфігурацією, спрямоване на забезпечення надійного й ефективного функціонування всіх компонентів інформаційної системи;
- контроль продуктивності (збір і аналіз інформації про роботу окремих ресурсів, прогнозування ступеня задоволення потреб користувачів/застосунків, заходи для збільшення продуктивності);
- захист даних (управління доступом користувачів до ресурсів, забезпечення цілісності даних і управління їхнім шифруванням);
- рішення проблемних ситуацій (діагностика, локалізація й усунення несправностей, реєстрація помилок, тестування).

Сьогодні на ринку присутні сотні продуктів, що дозволяють вирішувати ті або інші задачі системного адміністрування. В основі своєрідної піраміди знаходяться базові платформи управління - CA-Unicenter TNG компанії Computer Associates, HP OpenView і Tivoli Enterprise виробництва IBM. Названі системи покривають усю функціональну область, обумовлену стандартом ISO. Окремі пробіли в цьому спектрі згодом усуваються шляхом власних розробок, складанням партнерських угод або придбання інших фірм.

Самостійну групу складають сімейства керуючих застосунків, що охоплюють лише частину областей системного адміністрування. Прикладами програм цієї категорії можуть служити пакети PATROL компанії BMC Software і EcoSYSTEMS виробництва Compuware.

Нарешті, множина фірм сьогодні пропонує так називані "крапкові" продукти, орієнтовані на рішення конкретних вузьких задач (аналіз подій, генерація звітів про продуктивність, управління IP-адресами і т.і.).

Основний результат тривалого розвитку галузі системного адміністрування полягає в тому, що з функціональної точки зору основні платформи управління на сьогодні досить схожі одна на одну. Розходження між ними криються в сфері структурного виконання і, крім того, пов'язані з тими вихідними цілями, що ставилися на початкових етапах розробки. Так, корені сімейства OpenView лежать у сфері мережного адміністрування (відповідні функції вважаються найбільш сильними в порівнянні з конкуруючими пакетами), засоби ж системного адміністрування були додані пізніше. Навпроти, пакет Tivoli Enterprise (тодіще Tivoli Management Environment) спочатку був націлений на рішення проблем системного адміністрування; те ж справедливо й у відношенні ПЗ CA-Unicenter.

Архітектурні аспекти виходять на перший план, коли виникає потреба в інтеграції різноманітних продуктів, наприклад точкових рішень із базовою платформою адміністрування. Про актуальність цієї задачі свідчить поява окремого інтегруючого сегмента програмного забезпечення, що сьогодні

пропонується такими фірмами, як CACI Products, Ganymede Software, Make Systems, Managed Object Solutions, NetReality.

Серед численних категорій користувачів ПЗ системного адміністрування усе більшу вагу набирають фірми середнього розміру, що у своєму розвитку вже переросли найпростіші задачі керування настільними комп'ютерами, об'єднаними в локальну мережу, але ще не готові - ні в психологічному, ні у фінансовому плані - розгорнути в собі складні і дорогі керуючі платформи. Очевидно, користувачам цієї групи потрібні не точкові продукти, у надлишку присутні сьогодні на ринку, а закінчені рішення, що мають обмежену функціональність, інтуїтивний Web-інтерфейс і прийнятну ціну. Ніша, що тут утворилася, поступово починає заповнюватися: як приклад можна назвати пакет FrontLine Manager, що випускається компанією Manage Com.

Останнім часом у закордонній літературі усе активніше обговорюється концепція динамічного адміністрування. Її поява відповідає загальній тенденції у світі мережного і системного адміністрування - переносу акцентів із контролю за окремими ресурсами або їхніми групами, із керування робочими характеристиками ІС на максимальне задоволення запитів кінцевих споживачів інформаційних технологій.

Такий підхід припускає насамперед наявність засобів аналізу поводження користувачів, у ході якого повинні бути виявлені як їхні преваги, так і проблеми, що виникають у повсякденній роботі. Результати, отримані на цьому етапі, повинні служити відправною точкою для так називаного активного керування взаємодією між основними об'єктами адміністрування - користувачами, застосунками і мережею. Термін "активне" розуміє постійне відстеження характеру роботи застосунків користувачів і оперативне втручання в цей процес у тому випадку, коли рівень сервісу, одержуваний користувачем, не відповідає очікуваному. Для найбільше адекватного реагування на виникаючі проблеми прихильники концепції активного адміністрування закликають використовувати аналітичні засоби підтримки

прийняття рішень.

Проте перед тим, як справа дійде до конкретних дій, системний адміністратор повинен мати формальний опис тих послуг, що IT-відділ зобов'язаний надати конкретному користувачу, і мати у своєму розпорядженні надійні засоби виміру параметрів сервісів у реальному часі. Один із варіантів цього сценарію поступово оформився в самостійну концепцію керування IT-послугами.

2 ПРОГРАМНІ МЕТОДИ УПРАВЛІННЯ СЛУЖБАМИ І ДРАЙВЕРАМИ

2.1 Програмний застосунок Microsoft System Center Operations Manager Enterprise

System Center Operations Manager можна рекомендувати як гарне рішення для управління сервісами Windows (рисунок 2.1), а також всеосяжного моніторингу і підготовки звітів, яке дозволяє підвищити ефективність при надійнішому управлінні IT-середовищем.

Рисунок 2.1 – Логотип System Center Operations Manager

Microsoft System Center Operations Manager допомагає системним адміністраторам перетворити інформацію про роботу системи в конкретні дії за допомогою нових пакетів управління (management pack), призначених для управління різними типами серверних застосунків. Дані пакети дозволяють користувачам створювати докладні звіти про роботу системи, скорочувати час для ухвалення рішення, віддалено управляти своїми серверами і використовувати переваги управління застосунками через Internet. Пакети управління в своєму складі містять не тільки описи проблем, що часто зустрічаються, але і варіанти їх рішення і пропонуються як для продуктів Microsoft, так і для інших.

Microsoft System Center Operations Manager містить вбудовані засоби безпеки – наприклад, функцію взаємної аутентифікації серверів і агентів, допомагає компанії захистити свої системи управління від несанкціонованого

використання.

Microsoft System Center Operations Manager передбачає також зв'язок через Internet з іншими системами управління, і включає нову консоль, яка відображає всю необхідну інформацію про стан системи, її працездатності і надає доступ до вбудованого серверу звітів (рисунок 2.2).

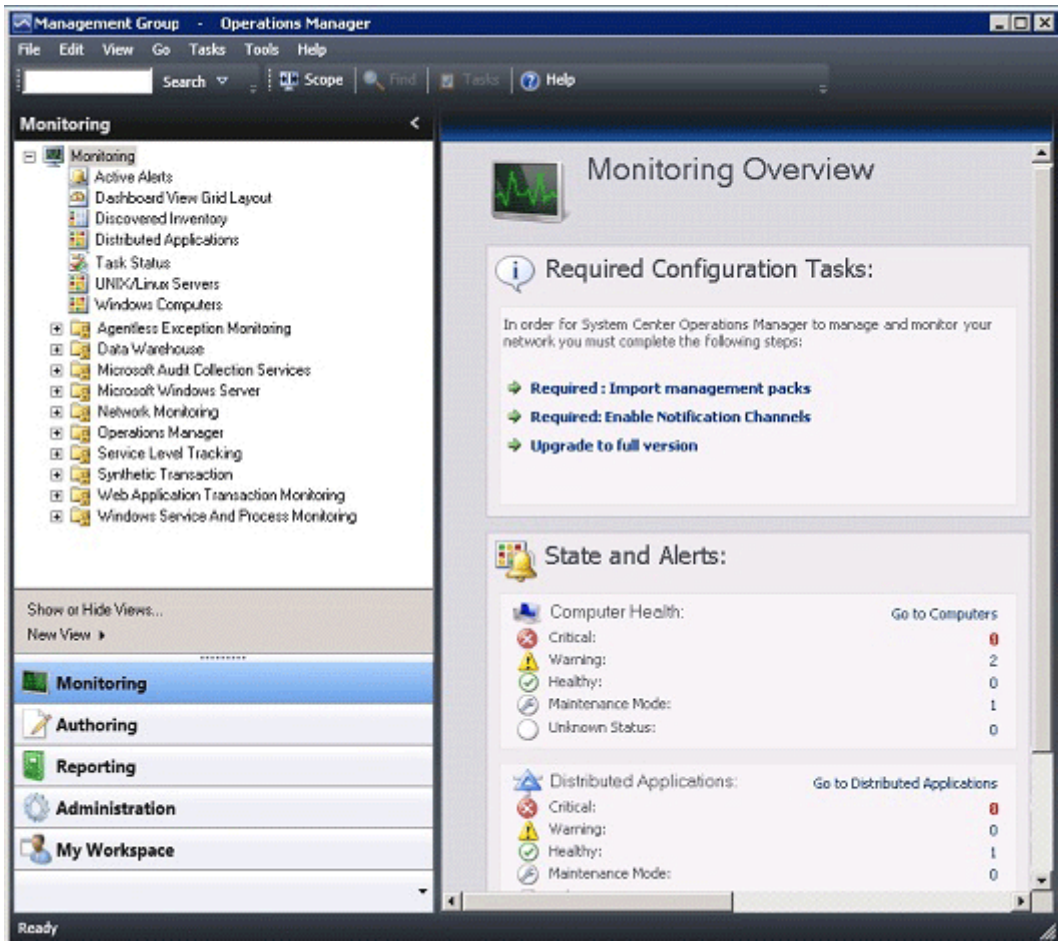


Рисунок 2.2 – Головне вікно System Center Operations Manager

Основні функції застосунку полягають в:

- вичерпаному управлінні сервісами. System Center Operations Manager забезпечує відмінне управління службами, його легко настроювати і розширювати в міру необхідності для підвищення рівня обслуговування в масштабах компанії;

- кращої інтеграції з Windows. System Center Operations Manager –

кращий вибір для систем Windows, оскільки цей продукт створений з урахуванням досвіду команд розробників серверних, клієнтських і прикладних рішень Microsoft. Керівництво, що міститься в продукті, методики і реалізовані в ньому функції дозволяють значно підвищити ефективність управління;

- якісно новому рівні управління. System Center Operations Manager автоматизує задачі, що повторюються, забезпечує повні звіти і моніторинг, підвищуючи ефективність і оптимізуючи управління IT - середовищем.

System Center Operations Manager забезпечує можливість контролю IT - служб, надійно масштабується на всю компанію і мережеву інфраструктуру. Він надає відомості про застосунки і операційні системи Microsoft, необхідні для усунення проблем експлуатації. Вимоги до hardware наведені в таблиці 2.1.

Таблиця 2.1 – Апаратні вимоги Microsoft System Center Operations Manager

Характеристика	Мінімальні вимоги	Рекомендовані вимоги
Тактова частота процесора, ГГц	1,8	2,8 і вище
Об'єм оперативної пам'яті, ГГб	1	4 і вище
Об'їм простори на жорсткому диску, ГГб	10	50

Висновок. По своїх функціональних можливостях даний програмний продукт достатньо універсальний, виконує безліч функцій різного призначення. Його недолік полягає в універсалізмі і достатньо високих вимогах до апаратного забезпечення.

2.2 Програмний застосунок ServiceUtility

Програма ServiceUtility призначена для використання як починаючими користувачами, так і досвідченими: для роботи, аналізу, стеження за

службами і драйверами. Набір інструментів, наданий програмою, дозволить вирішувати проблеми при спілкуванні із службами. Крім того, дозволяє відстежувати знов встановлені служби, а також всі зміни з вже встановленими службами (рисунок 2.3).

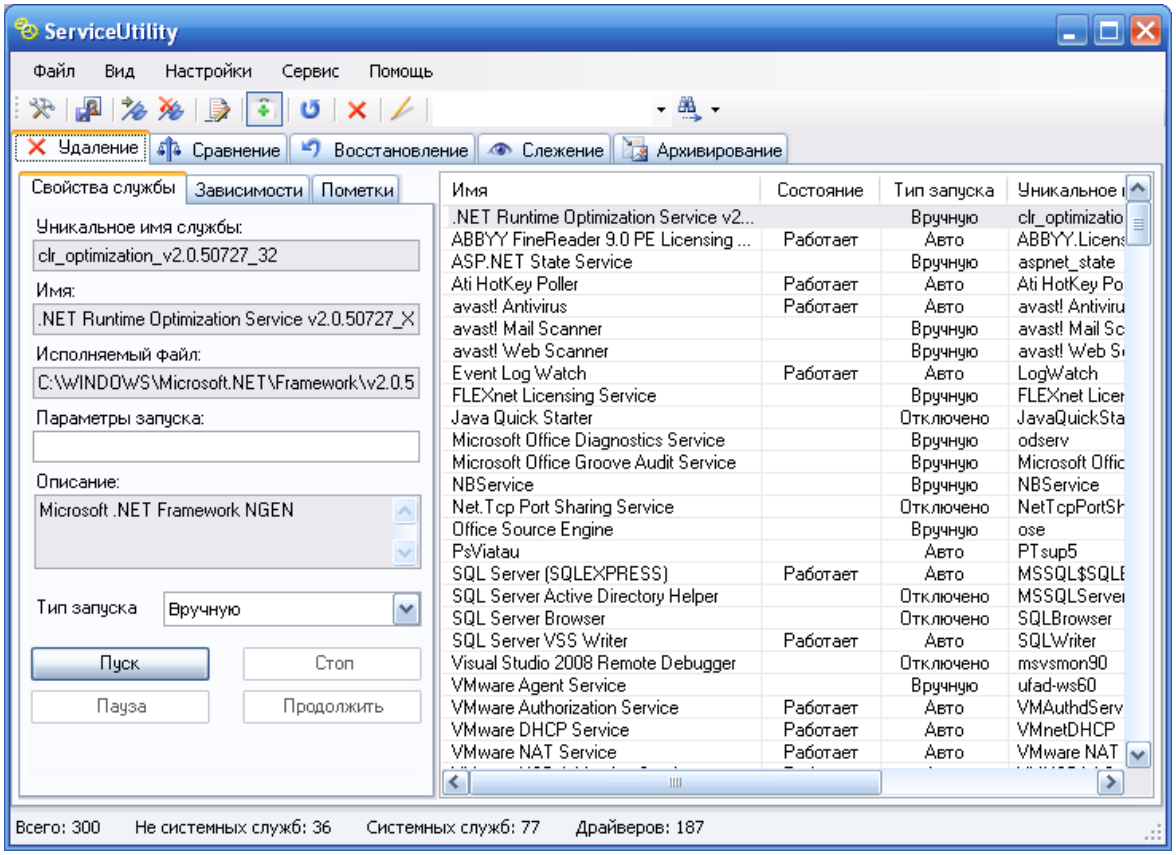


Рисунок 2.3 – Головне вікно ServiceUtility

Основні можливості програми:

- підтримка всіх основних можливостей стандартної програми "Служби" це перегляд, зміна властивостей, перегляд залежностей;
- видалення служби одним натисненням кнопки;
- окреме кольорове підсвічування і сортування по наступних групах: не системні сервіси, системні сервіси і драйвера;
- створення точки образу (зберігаються у файл всі параметри служб і драйверів на даний момент);
- функція порівняння допоможе знайти відмінності в двох знімках (на

основі файлів образу або між знімком і поточними службами і драйверами) такі як: стан, тип запуску, ім'я, шлях, опис, нові служби і драйвера, видалені служби і драйвера. Всі відмінності виділяються кольором;

- на основі файлу точки образу можна завжди повернути стан служб і драйверів на момент створення знімка. Відновлюються наступні значення: стан і тип запуску;
- якщо потрібно постійно бути в курсі, що відбувається із службами, то є можливість постійного стеження і у разі яких-небудь змін з'явиться інформаційне вікно з докладним описом про виниклу подію. Так само ведеться докладний протокол останніх повідомлень;
- для деяких адміністраторів є можливість підключати службу.

Для роботи програми необхідно, щоб на комп'ютері була встановлена платформа Microsoft .NET Framework. Об'єм дистрибутива складає 1405 Кб. Є можливість русифікації призначеного для користувача інтерфейсу програми.

Висновок. Достатньо функціональна і проста програма, яка не вимоглива до ресурсів самого комп'ютера. Основний недолік – не можливо її використовувати для моніторингу за драйверами і службами видаленого комп'ютера.

2.3 Програмний застосунок ServiceView

Програма компанії Native Computer Systems (NCS) Z-tools.he ServiceView служить для розширення можливостей управління службами Windows. Управляти службами (драйверами пристроїв) можна або із списку, або, обравши необхідну службу (драйвер пристрою) в дереві. Останній спосіб надає багатші можливості. Проте, використовуючи список, можна оперувати групами служб (позначаючи необхідні об'єкти відповідно до певного критерію, як, наприклад, по місцю розташування файлу запуску, стану і інше). Для цього передбачене сортування, активувати яке можна,

клацнувши по відповідному заголовку (рисунок 2.4).

Так само програма характеризується відмінним інтерфейсом по управлінню сервісами і драйверами пристроїв. Дозволяється групова зміна типу запуску і поточного стану служби, сортування по таких критеріях, як шлях до виконавчого файлу, можливість модифікації деяких параметрів служби, не вдаючись до безпосередньої модифікації системного реєстру.

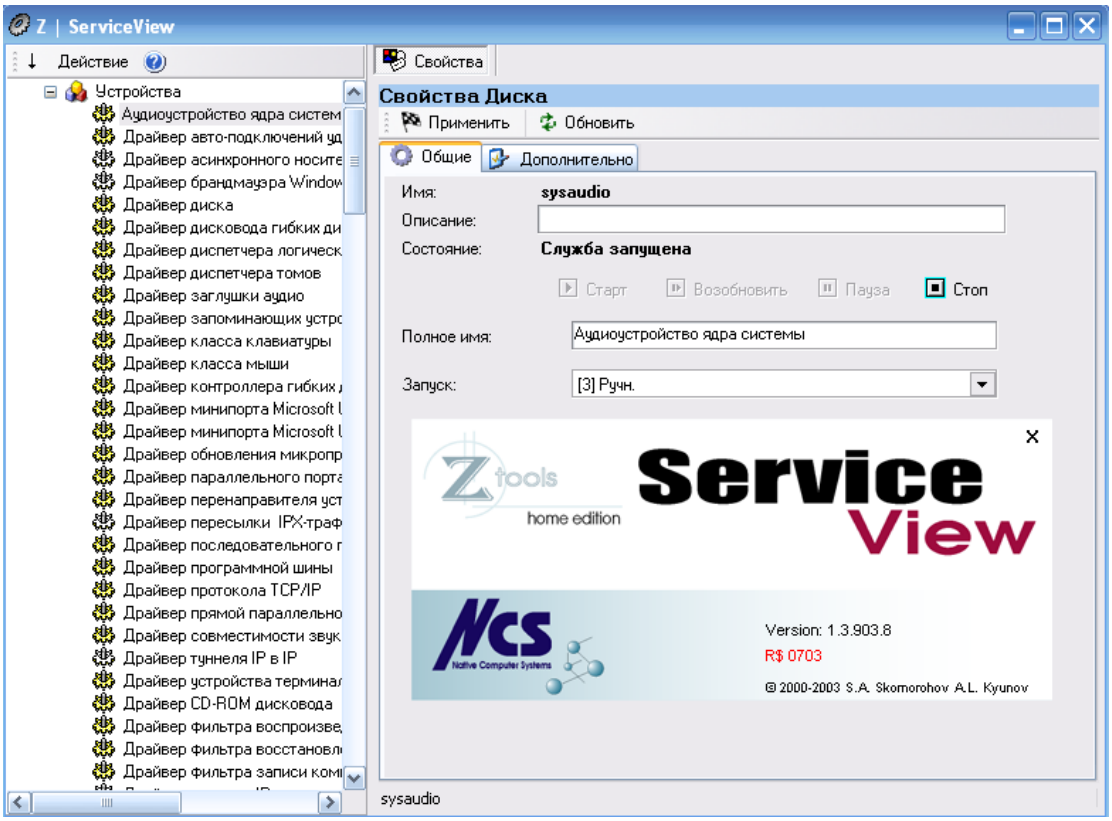


Рисунок 2.4 – Головне вікно ServiceView

Висновок. Займає дистрибутив програми біля 1 Мб, підтримуються різні інтерфейси програми. Недолік – орієнтація тільки на 32-розрядні версії ОС Windows.

2.4 Програмний застосунок Windows Services Controller

Утиліта розробки компанії ALXR призначена для роботи із Службами в

операційній системі. Windows Services Controller візьме на себе створення, моніторинг, настройку, управління, збереження і відновлення конфігурації всіх служб в ОС Windows (рисунок 2.5).

Windows Services Controller – програма дозволяє виконати тонку настройку всіх служб в системах Windows. У реальному часі показує поточний стан служби і тип її запуску. На відміну від аналогів показує час, коли служба була створена, а також має нагоду виконати настройку служб-драйверів (драйвер CD|DVD, миші тощо), змінювати тип служби, створювати нову або видаляти існуючу, змінювати приховані параметри служби, виробляти пошук по імені служби (опису модуля) служби.

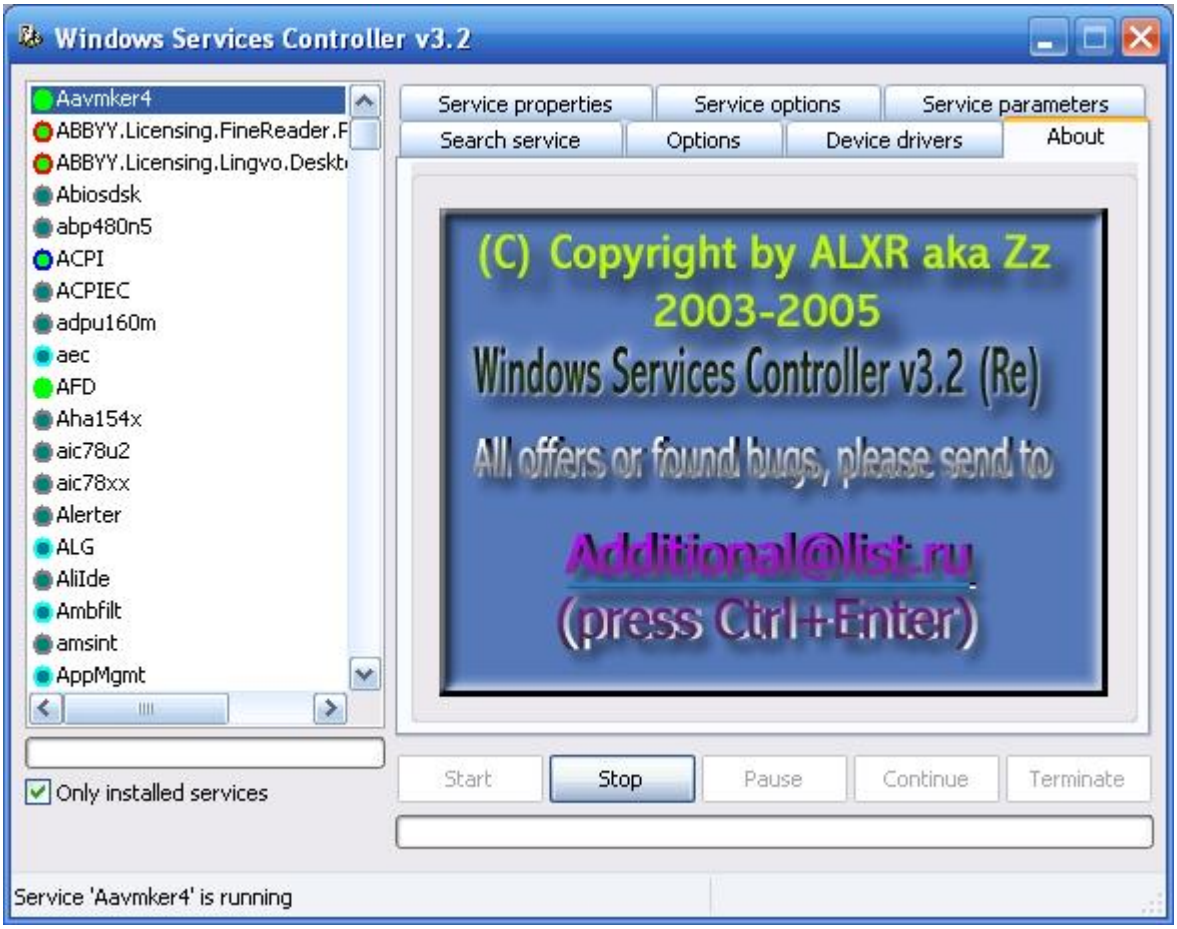


Рисунок 2.5 – Головне вікно Windows Services Controller

Також є можливість збереження, а потім відновлення поточної

конфігурації служб. Таке буває необхідне для досвідчених користувачів системи Windows, які кожного разу настроюють служби при установці системи. В той же час програма має стандартну нагоду настройки конфігурації і стану служб, такі як запуск/остановка/приостановка служб, зміна опису, імені служби, що виводиться, тип запуску і виконуваного модуля служби.

Також в програму вбудована можливість огляду всіх завантажених на даний момент драйверів з висновком адреси, куди завантажений драйвер, а також описом драйвера. Це дозволяє вести діагностику появи "синіх екранів смерті" в системі, методом пошуку драйвера, завантаженого в ділянку пам'яті, в якому відбувся збій. Працює Windows Services Controller без інсталяції. Основною перевагою при цьому є достатньо малий (всього 174 Кбайт) розмір дистрибутива, проте не має русифікованого інтерфейсу. Недолік – орієнтація тільки на 32-розрядні версії ОС Windows.

3 МЕТОД АДМІНІСТРУВАННЯ СЛУЖБАМИ Й ДРАЙВЕРАМИ

3.1 Адміністрування служб з боку користувача

Практично в кожній операційній системі є механізм, який запускає при її завантаженні процеси, що надають сервіси, не пов'язані з інтерактивними діями користувача. В Windows такі процеси називаються службами (сервісами, services), або Windows-сервісами, оскільки при взаємодії із системою вони покладаються на Windows API. Прикладом Windows-сервісу може служити Web-сервер, оскільки він повинен запускатися разом із системою та працювати незалежно від того, чи зареєструвався в системі який-небудь користувач. Windows-сервіси складаються із трьох основних складових (рисунок 3.1):

- застосунка сервісу (service application);
- програми управління сервісом (service control program, SCP);
- диспетчера управління сервісом (service control manager, SCM).

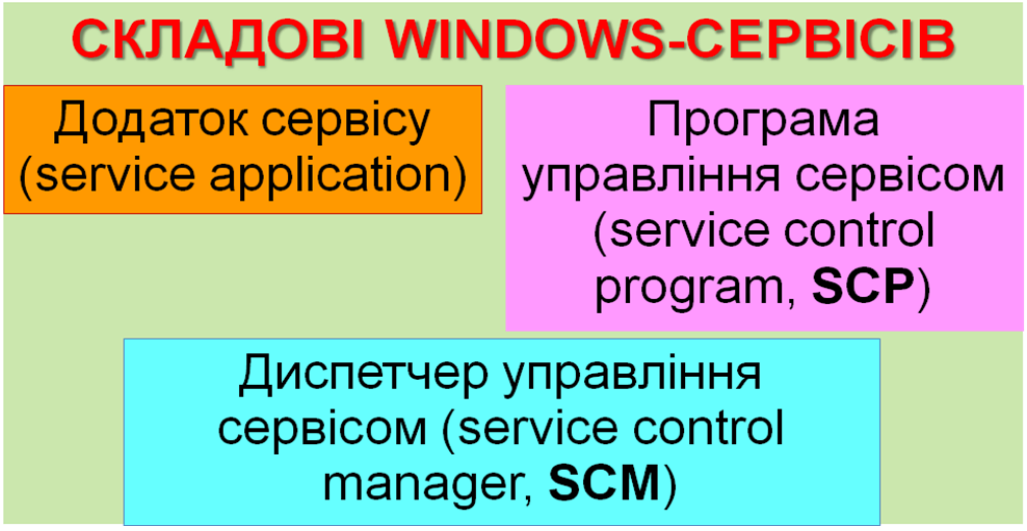


Рисунок 3.1 – Основні складові Windows-сервісів

Архітектура Windows, що опирається на базову технологію NT, припускає наявність у складі операційної системи певної кількості сервісів, або служб — спеціальних програм, які, стартуючи в момент запуску Windows, забезпечують виконання спеціалізованих системних завдань. Тому сервіси Windows іноді називають також системними сервісами. Безумовно, далеко не всі сервіси, що настроєні по замовчуванню у процесі установки Windows, потрібні кінцевому користувачеві, особливо, якщо він використовує комп'ютер у якості настільної обчислювальної системи або робочої станції. Управління системними сервісами, які використовує у своїй роботі Windows, можливе через реєстр, однак розроблювачі передбачили для цих цілей куди більш зручний інструмент - програму Настройки системи.

Для того щоб запустити програму Настройка системи, необхідно в Головному меню Windows натиснути на кнопку Пуск (Start), вибрати в ньому пункт Виконати (Run). У вікні, що відкрилося, набрати команду msconfig і клацнути мишею на кнопці ОК. З'явиться вікно, показане на рисунку 3.2.

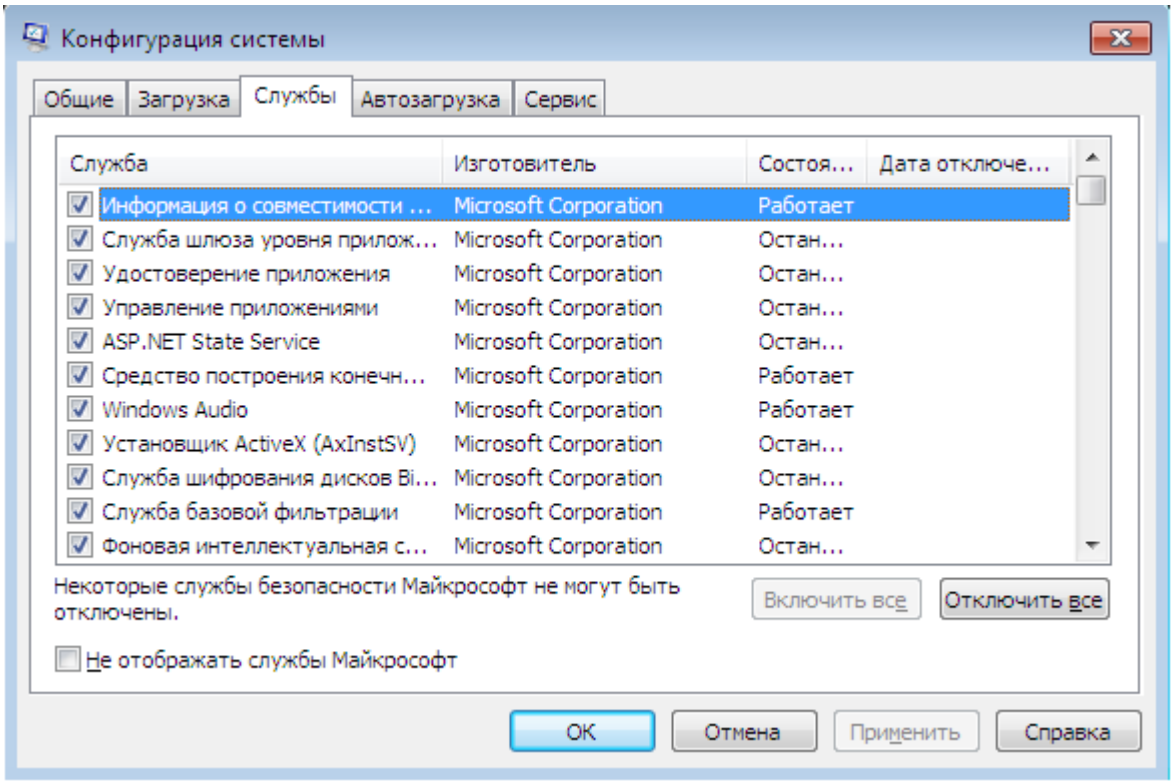


Рисунок 3.2 – Вікно програми Настройки системи

Всі зміни, внесені в конфігурацію операційної системи за допомогою даного застосунка, будуть автоматично внесені до реєстру й інших системних файлів.

Сервіси Windows є системними процесами, які здійснюють підтримку обладнання й програмного забезпечення на низькому, близькому до апаратного, рівні. Для того щоб гнучко провести настройку списку сервісів, які виконуються на комп'ютері, бажано входити в систему з повноваженнями адміністратора.

Для надання користувачеві графічного інтерфейсу по управлінню сервісами служить консоль Служби (рисунок 3.3).

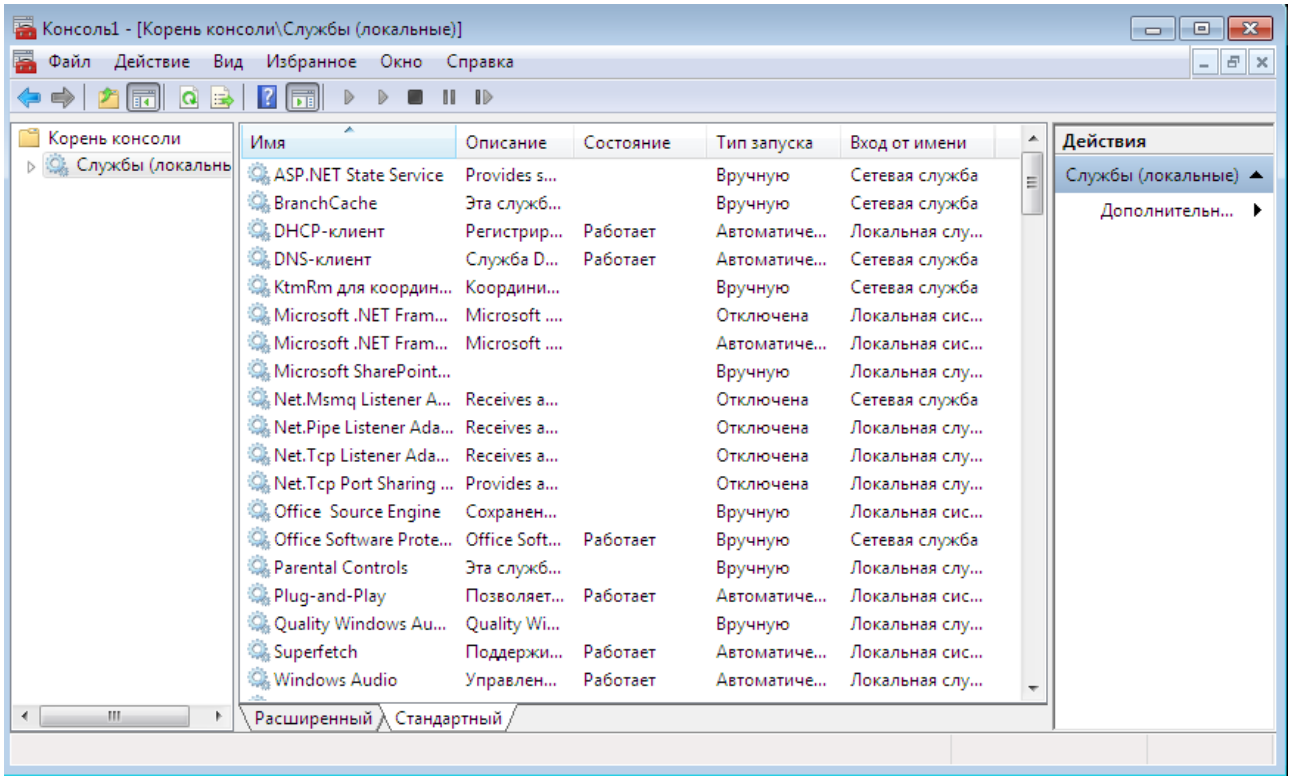


Рисунок 3.3 – Вікно консолі MMC з відкритим оснащенням Служби

Доступ до консолі Служби можна здійснити різними способами:

- з використанням консолі управління (Пуск - Виконати - mmc);
- відкрити контекстне вікно ярлика на робочому столі Мій комп'ютер →Управління → Управління комп'ютером → Сервіси.

У панелі відомостей консолі перераховані всі наявні в складі Windows системні сервіси, причому інформація про їх представлена у вигляді декількох тематичних стовпчиків:

- ім'я (Name);
- опис (Description);
- тип запуску (Startup type) (автоматично або вручну);
- стан (Status);
- вхід від імені (Log on as).

Останній стовпчик демонструє назву облікового запису, від імені якого був запущений сервіс. Це дозволяє визначити, з якими правами він виконується в системі.

Алгоритм роботи з консоллю Служби в цілому простий:

- щоб переглянути відомості про сервіс в панелі інформації, клацнути на неї щигликом миші;
- щоб призупинити виконання сервісу, виділити необхідний сервіс в панелі відомостей, після чого виконати команди Дія → Пауза (Action → Pause) або клацнути мишею на кнопці Припинення сервісу в інструментальній панелі консолі;
- щоб відновити виконання сервісу, виділити необхідний сервіс в панелі відомостей, після чого виконати команди Дія → Продовжити (Action → Resume) або клацнути мишею на кнопці Запуск сервісу в інструментальній панелі консолі;
- щоб припинити роботу сервісу, виділити необхідний сервіс в панелі відомостей, після чого виконати команди Дія → Стіп (Action → Stop) або клацнути мишею на кнопці Зупинка сервісу в інструментальній панелі консолі;
- щоб запустити знову сервіс, виділити необхідний сервіс в панелі відомостей, після чого виконати команди Дія → Запустити знову (Action → Restart) або клацнути мишею на кнопці Перезапуск сервісу в інструментальній панелі консолі.

Кожен сервіс характеризується властивостями, показаними на рисунку 3.4.

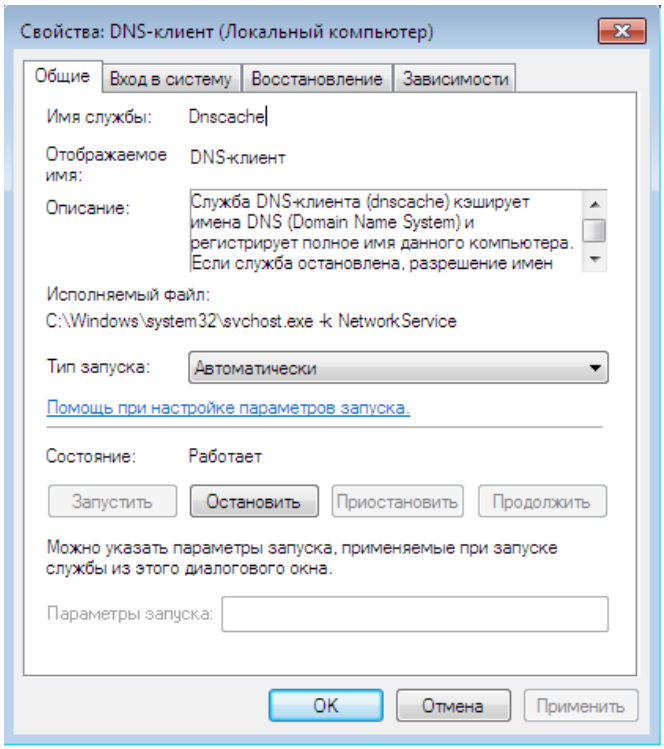


Рисунок 3.4 – Вікно Властивостей сервісу (приклад)

К ним відносяться:

- загальні відомості про сервіс (ім'я, виведене ім'я, опис, виконавчий файл із ключами запуску, тип запуску, стан) (закладка Загальні);
- запуск сервісу від імені облікового запису користувача (закладка Вхід у систему);
- настроювання правил обробки помилок при збої сервісу (закладка Відновлення);
- перегляд служб, пов'язаних з даним сервісом (закладка Залежності).

3.2 Адміністрування служб з боку операційної системи

Робота служб забезпечується взаємодією компонентів трьох типів:

- диспетчер управління сервісами (Service Control Manager, SCM). Всі системи Windows поставляються з компонентом, названим диспетчером управління сервісами (SCM, вимовляється "скам"). Цей компонент зберігається у файлі Scrvices.exe. Він автоматично викликається при запуску ОС і переривається при виході з неї. SCM має привілею системи й забезпечує уніфікований і безпечний метод управління застосунками-сервісами. SCM відповідає за взаємодію з різними сервісами, указуючи їм, коли запуститися, зупинитися, тимчасово перервати й продовжити роботу тощо;

- застосунок-сервіс. Сервіс – це просто застосунок, що містить інфраструктуру для взаємодії з SCM, що посилає службі команди запуску, зупинки, паузи, продовження або вимикання. Сервіс також викликає спеціальні функції, що передають SCM відомості про її стан;

- програма управління сервісом (Service Control Program, SCP). Це застосунок, який має інтерфейс користувача, що дозволяє йому запустити, зупинити, призупинити, продовжити роботу й виконати інші функції управління сервісами, установленими на машині. Програма управління сервісом викликає спеціальні функції Windows, що дозволяють їй спілкуватися з SCM.

На рисунку 3.5 показано, як ці компоненти взаємодіють. SCP-застосунки взаємодіють зі сервісами не прямо, а через SCM. Така архітектура робить віддалене адміністрування прозорим для SCP і застосунків-служб. Можна реалізувати архітектуру й протокол, які дозволять SCP-застосункам звертатися прямо до застосунків-служб, але код для забезпечення такої взаємодії прийдеться писати програмістові.

SCM – це та із трьох компонентів, що не прийдеться створювати самим. Microsoft реалізувала й поставляє SCM з усіма версіями Windows. Користувач повинен реалізовувати сервіс й програми управління сервісами.

Надалі спершу будуть розглянуті питання проектування й реалізації сервісу, а далі - програми управління сервісами.

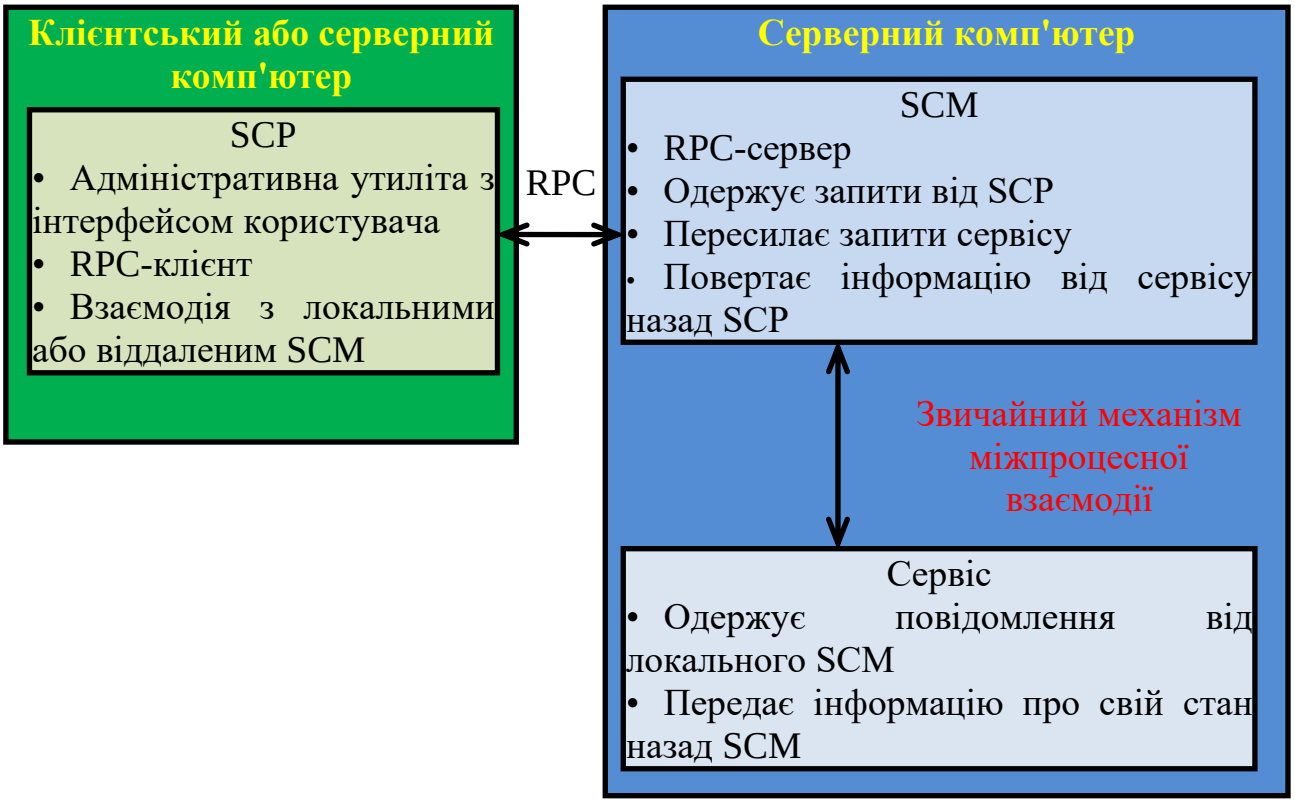


Рисунок 3.5 – Структурна схема обміну інформацією зі сервісами Windows

3.2.1 Програми управління службами, що поставляють із Windows

Відомості про встановлені сервіси в операційній системі зберігаються в базі даних SCM, що розташовується в наступному підрозділі реєстру:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`

Ніколи неможна звертатися прямо до цього підрозділу. Виконувати потрібно лише через SCP, що повинна звертатися до функцій Windows, які маніпулюють базою даних сервісів. Безпосередня зміна вмісту цього розділу приведе до непередбачених результатів.

Коли встановлюється який-небудь програмний продукт, що включає сервіс, програма установки цього продукту і є SCP, що додає відомості про цей сервіс в базу даних SCM. Менеджер зберігає кожну характеристику як параметр у розділі, створеному для даного сервісу. Приклад такого розділу реєстру показаний на рисунку 3.6.

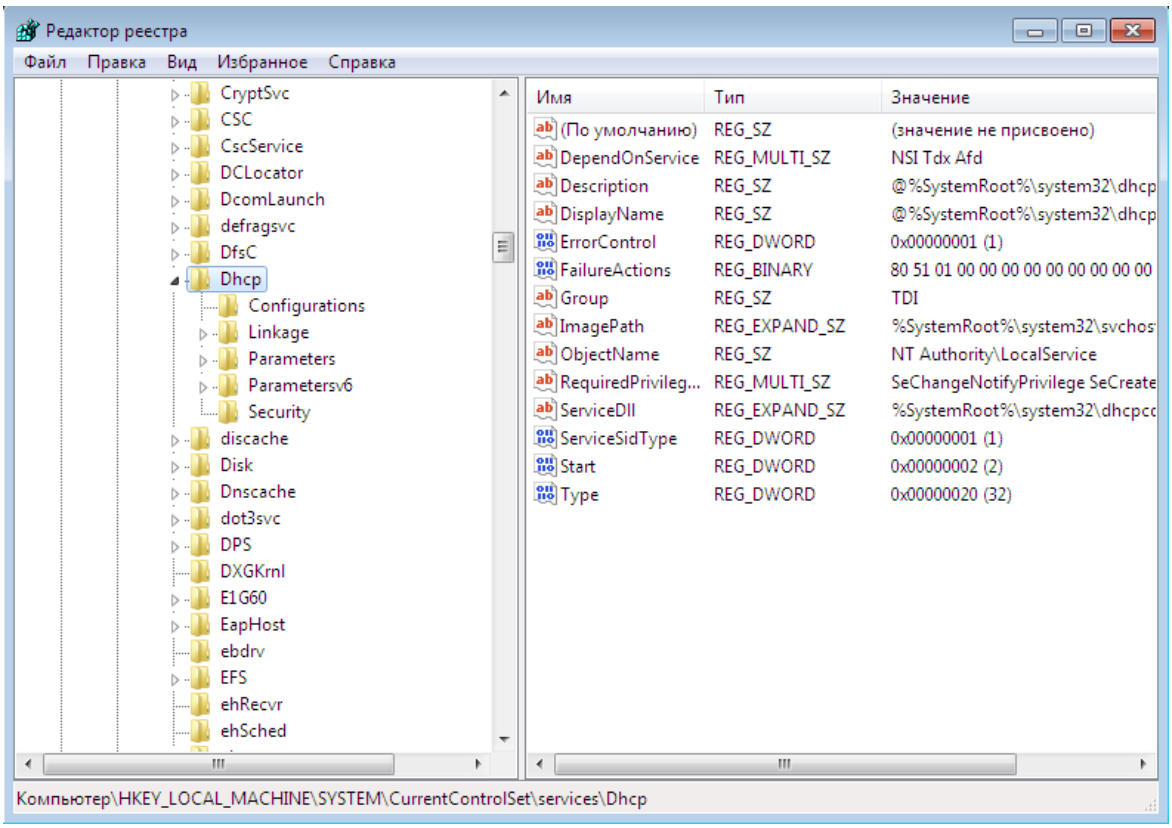


Рисунок 3.6 – Приклад розділу реєстру для сервісу

У таблиці 3.1 перераховані характеристики сервісу, багато з яких застосовуються і до драйверів пристроїв.

Таблиця 3.1 – Параметри реєстру для сервісів і драйверів

Параметр	Значення	Опис
1	2	3
Group Tag	Ім'я групи	Драйвер або сервіс стартує при ініціалізації цієї групи
	Номер тега	Указує порядковий номер у послідовності ініціалізації групи. Цей параметр не застосовується до сервісів
DependOn Group	Ім'я групи	Драйвер або сервіс не завантажується, доки не буде завантажені драйвер або сервіс із зазначеної групи

Продовження таблиці 3.1

1	2	3
Start	SERVICE_BOOT_START (0)	Ntcdr або Osloader попередньо завантажують драйвер, щоб під час завантаження він перебував у пам'яті. Подібні драйвери ініціалізуются безпосередньо перед драйверами "типу" SERVICE_SYSTEM_START
	SERVICE_SYSTEM_START (1)	Драйвер завантажується й ініціалізуются після ініціалізації драйверів "типу" SERVICE_BOOT_START
	SERVICE_AUTO_START (2)	Драйвер або сервіс запускається SCM після запуску SCM-процесу Services.exe
	SERVICE_DEMAND_START (3)	Драйвер або сервіс запускається SCM на вимогу
	SERVICE_DISABLED (4)	Драйвер або сервіс не завантажується й не ініціалізуются
Depend OnService	Ім'я сервісу	Сервіс не завантажується, доки не буде завантажений зазначений сервіс. Цей параметр не застосуємо до драйверів пристроїв, крім тих, у яких тип запуску визначений як SERVICE_AUTO_START
Object-Name	Звичайно Local System, але може бути ім'ям облікового запису, наприклад .\Administrator	Указує обліковий запис, під яким буде працювати сервіс. Якщо ObjectName не визначений, використовується обліковий запис Local System. Цей параметр не застосовується до драйверів пристроїв
Description	Опис сервісу	Рядок довжиною до 32767 байтів

Продовження таблиці 3.1

1	2	3
ErrorControl	SERVICE_ERROR_IGNORE (0)	Диспетчер введення-виведення ігнорує повертаємий драйвером або сервісом код помилок. Попередження не виводяться й не реєструються
	SERVICE_ERROR_NORMAL (1)	Якщо драйвер або сервіс повідомляє про помилку, виводиться попередження
	SERVICE_ERROR_SEVERE (2)	Якщо драйвер або сервіс повідомляє про помилку й останню вдалу конфігурацію що не використовується, завантажується саме ця конфігурація. Якщо вона вже використовується, завантаження триває
	SERVICE_ERROR_CRITICAL (3)	Якщо драйвер або сервіс повідомляє про помилку й останню вдалу конфігурацію що не використовується, завантажується саме ця конфігурація.
Security	Дескриптор захисту	Містить дескриптор захисту, що визначає, хто й з якими правами може одержувати доступ до об'єкта сервісу, створеному SCM
ImagePath	Шлях до виконує файлу, що, сервісу або драйвера	Якщо ImagePath не визначений, диспетчер вводу/виводу шукає драйвери в Windows\System32\Drivers, а SCM використовує Windows-функції для пошуку образу за значенням змінної оточення PATH
DisplayName	Ім'я сервісу	Ім'я сервісу, показуване сервісним застосунком на екрані

Продовження таблиці 3.1

1	2	3
Type	SERVICE_KERNEL_ DRIVER (1)	Драйвер пристрою
	SERVICE_FILE_SYS TEM_ DRIVER (2)	Драйвер файлової системи режиму ядра
	SERVICE_ADAPTER (4)	Застаріле значення
	SERVICE_RECOGNI ZER_ DRIVER (8)	Драйвер, що розпізнає файлову систему
	SERVICE_WIN32_O WN_ PROCESS (16)	Сервіс виконується у власному процесі, призначеному тільки для нього
	SERVICE_WIN32_SH ARE_ PROCESS (32)	Сервіс виконується в процесі, який використовується декількома сервісами
	SERVICE_INTERAC TIVE_PROCESS (256)	Сервіс може виводити на консоль вікна й приймати введення від користувача
Failure Actions	Опис дій SCM при несподіваному завершенні процесу сервісу	Дії при збої включають перезапуск процесу сервісу, перезавантаження системи й запуск зазначеної програми. Цей параметр не застосується до драйверів
FailureCom-mand	Командний рядок програми	SCM зчитує цей параметр, тільки якщо в FailureActions зазначена необхідність запуску якої-небудь програми при збої сервісу. Цей параметр не застосуємо до драйверів

Завдання компонента Services.exe (рис.2.4):

- запуск сервісу. Адміністратор запускає сервіс, вибираючи його зі списку й натискаючи кнопку Start на панелі інструментів. Запускатися

можуть тільки сервіси з видом запуску Automatic (Автоматично) і Manual (Вручну); якщо в якості Startup Type встановлене Disabled (Заблокована), сервіс не буде запустися. Блокування сервісу корисне при виявленні проблем з комп'ютером;

- зупинка сервісу. Адміністратор зупиняє сервіс, вибираючи його зі списку й натискаючи кнопку Stop на панелі інструментів. Деякі сервіси не дозволяють себе зупинити, після того як вони запустилися. Так, сервіс реєстрації подій Event Log зупиняється тільки при вимиканні комп'ютера;

- припинення й продовження роботи сервісу. Адміністратор припиняє сервіс, вибираючи його зі списку й натискаючи кнопку Pause на панелі інструментів. Деякі сервіси не дозволяють себе призупинити. Крім того, немає чіткого визначення "припинення" ("паузи") у роботі сервісу. Для одного сервісу пауза може означати, що сервіс не може приймати клієнтські запити, поки не закінчить обробку запитів. Для іншого - що сервіс взагалі більше не виконує ніяких дій. Припинені сервіси можна запустити, натиснувши кнопку Start на панелі інструментів;

- перезапуск сервісу. Адміністратор перезапускає сервіс, вибираючи працюючий або припинений сервіс зі списку й натискаючи кнопку Restart на панелі інструментів. Ця функція служить лише для зручності й дуже корисна при налагоджуванні сервісу.

Перераховані в цьому списку функції покривають 99% можливостей, які компонент Services надає адміністраторам, але їх можна використати й для перенастройки сервісу.

Крім модульного компонента Services, з Windows поставляється SCP-утиліта командного рядка Net.exe (рисунок 3.7).

Область її застосування обмежена - вона дозволяє управляти тільки сервісами, що працюють на локальній машині. За допомогою Net.exe також можна запустити, зупинити, призупинити й продовжити виконання служб, вводячи наступні команди:

```
NET START ім'я_сервісу
NET STOP ім'я_сервісу
```

```
NET PAUSE ім'я_сервісу
NET CONTINUE ім'я_сервісу
```

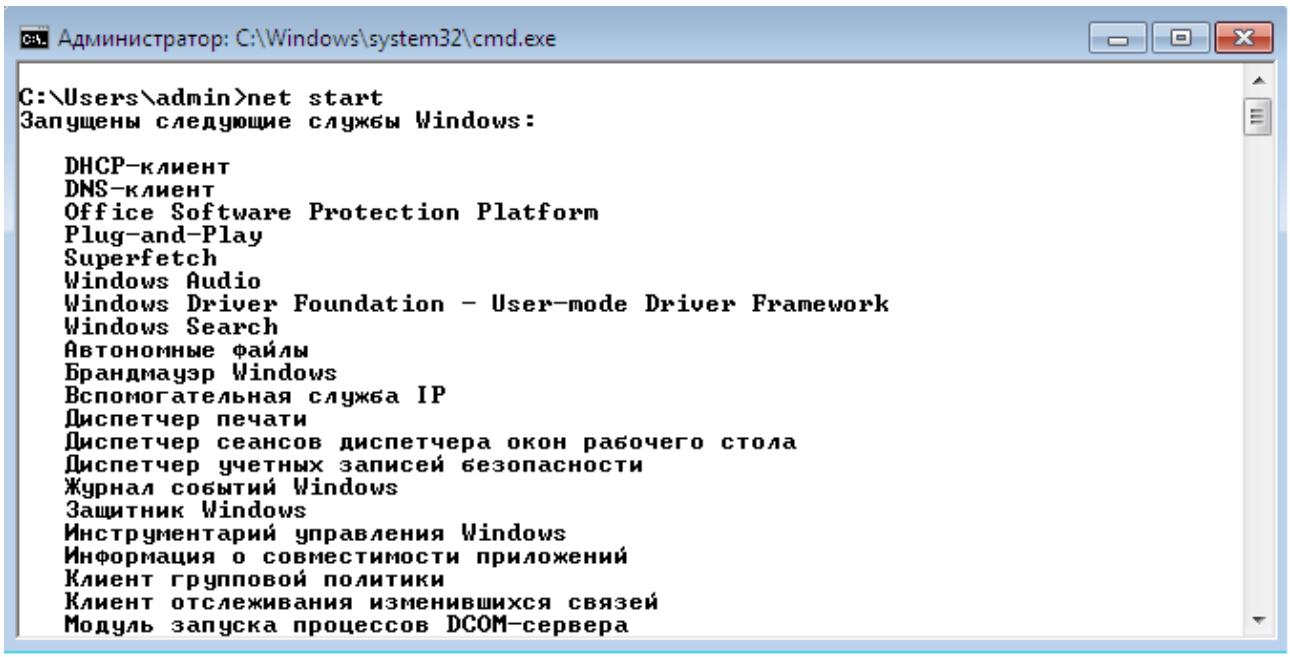


Рисунок 3.7 – SCP-утиліта командного рядка Net.exe

Net.exe дозволяє також вивести список працюючих на локальній машині служб командою без вказівки імені сервісу:

NET START

Ця утиліта досить зручна для налагодження, тому що звертання до неї можна помістити в будь-який командний файл або інший файл сценарію.

Із програмної точки зору подивитися перелік сервісів можна так (лістинг 3.1).

Лістинг 1.1 – Формування переліку сервісів (файл service.c)

```
#include "stdafx.h"
#include "windows.h"
#include "iostream.h"
#define SIZE_BUF 4096
void main()
{     ENUM_SERVICE_STATUS Status[SIZE_BUF];
      DWORD Size = sizeof(Status);
      DWORD Needed = 0;
      DWORD Return = 0;
      DWORD Handle = 0;
      SC_HANDLE Manager;
      Manager = OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS);
      if (Manager!=NULL)
      {
```

```
          if
(EnumServicesStatus(Manager,SERVICE_WIN32,SERVICE_ACTIVE,

     (LPENUM_SERVICE_STATUS)&Status,Size,&Needed,&Return,&Handle
))
          {    for (unsigned int x=0; x < Return; x++)
                    cout << Status[x].lpServiceName << endl;
          }
          else cout << "Error Open Manager " << endl;
     }
     else cout << "Error enum Services" << endl;
}
```

Перша функція, що викликається - це OpenSCManager. Ця функція встановлює зв'язок з менеджером SCM, відповідальним за роботу із сервісами:

```
SC_HANDLE OpenSCManager(
     LPCTSTR lpMachineName,   // ім'я комп'ютера
     LPCTSTR lpDatabaseName,  // ім'я менеджера
     DWORD dwDesiredAccess);  // тип доступу
```

Як видно деякі значення можна встановлювати в NULL.

Друга функція - EnumServicesStatus - перебирає всі сервіси й заносить їх у буфер. Формат функції наступний:

```
BOOL EnumServicesStatus (
SC_HANDLE hSCManager,    // хендл менеджера
DWORD     dwServiceType, // тип сервісу
DWORD     dwServiceState,     // стан сервісу
LPENUM_SERVICE_STATUS    lpServices,    // буфер
DWORD     cbBufSize,     // розмір буфера
LPDWORD   pcbBytesNeeded,
LPDWORD   lpServicesReturned,
LPDWORD   lpResumeHandle);
```

3.2.2 Функції API по управлінню службами в SCP

Структура застосунків-служб Windows. Розглянемо додаткову інфраструктуру, що перетворює серверний застосунок у сервіс, завдяки чому серверним застосунком можна управляти віддалено. Спочатку архітектура служб Microsoft здається складної для розуміння. Складність пояснюється тим, що кожен процес сервісу завжди має хоча б два потоки й потоки ці повинні взаємодіяти. Так що доводиться мати справу із синхронізацією потоків і обміном інформацією між ними.

Проектуючи сервіс, насамперед, потрібно звернути увагу на три види функцій:

- основна функція процесу — стандартна функція main або _tWinMain. У сервісах вона запускає процес у цілому й потім викликає спеціальні функції, які зв'язують процес із SCM локальної системи. Тепер SCM управляє основним потоком для власних потреб. Код знову одержить управління, тільки коли всі сервіси з виконавчого файлу припинять роботу;
- функція ServiceMain сервісу — це функція, яку потрібно реалізувати для кожного сервісу виконавчого файлу. Щоб запустити сервіс, SCM породжує в процесі потік, що виконує функцію ServiceMain. Коли управління в потоці повертається з ServiceMain, SCM вирішує, що сервіс зупинена. Ця функція не обов'язково повинна називатися ServiceMain — можна назвати її як завгодно;
- функція HandlerEx сервісу — кожний сервіс повинен мати пов'язану з ним функцію HandlerEx. SCM взаємодіє з сервісом, викликаючи його функцію HandlerEx. Її код виконується в основному потоці процесу. HandlerEx або виконує потрібні дії, або, застосовуючи ту або іншу форму міжпотокової взаємодії, повинна взаємодіяти з тією частиною потоку, керованого SCM, що виконує функцію ServiceMain сервісу. Кожний сервіс може мати власну HandlerEx або кілька служб (об'єднаних одним виконавчим файлом) а також можуть спільно використати єдину HandlerEx. Один з параметрів HandlerEx указує сервіс, з яким бажає взаємодіяти SCM. Ця функція не обов'язково повинна називатися HandlerEx — можна назвати її як завгодно.

Основна функція процесу. Коли адміністратор хоче запустити сервіс, SCM визначає, чи запущений виконавчий файл, який містить цей сервіс. Якщо ні, SCM його запускає. Основний потік процесу відповідає за ініціалізацію процесу в цілому (ініціалізацію конкретного сервісу повинна виконувати функція ServiceMain цього сервісу).

Після ініціалізації процесу його основна функція повинна зв'язатися з

SCM, що тепер управляє процесом. Для цього основна функція процесу повинна насамперед виділити й ініціалізувати масив структур SERVICE_TABLE_ENTRY:

```
typedef struct _SERVICE_TABLE_ENTRY {
PTSTR lpServlceName; // Внутрішнє ім'я сервісу
LPSERVICE_MAIN_FUNCTION lpServiceProc; // Функція ServiceMain
сервісу
} SERVICE_TABLE_ENTRY, *LPSERVICE_TABLE_ENTRY;
```

Перший елемент цієї структури вказує внутрішнє, програмне ім'я сервісу, а другий — адресу функції зворотного виклику сервісу SetviceMain. Якщо виконавчий файл містить тільки один сервіс, масив структур SERVICE_TABLE_ENTRY повинен ініціалізуватися так:

```
SERVICE_TABLE_ENTRY ServiceTable[] = {
{ TEXT("ServiceName1"), ServiceMain1},
{ HULL, NULL } // Указує кінець масиву
};
```

Якщо виконавчий файл містить три сервіси, масив треба ініціалізувати так:

```
SERVICE_TABLE_ENTRY ServlceTable[] = {
{ TEXT("ServiceName1"), ServiceMain1},
{ TEXT("ServiceName2"), ServiceMain2},
{ TEXT("ServiceName3"), ServiceMain3},
{ HULL, NULL } // Указує кінець масиву
};
```

Обидва елементи в останній структурі масиву повинні бути встановлені в NULL, щоб указати кінець масиву. Тепер процес приєднує себе до SCM, викликаючи функцію StartServiceCtrlDispatcher:

```
BOOL StartServiceCtrlDispatcher (CONST
SERVICE_TABLE_ENTRY pServiceTable);
```

Викликаючи цю функцію й передаючи їй адресу масиву таблиць служб виконавчий процес указує, які сервіси в ньому втримуються. Тепер SCM може визначити, які сервіси він повинен запустити, послідовно переглядаючи даний масив. Коли він знаходить сервіс, для нього створюється потік і починається виконання функції ServiceMain цього сервісу (адреса цієї функції визначиться з масиву SERVICE_TABLE_ENTRY).

StartServiceCtrlDispatcher не повертає управління, доки не припинять роботу всі сервіси, що утримуються в виконавчому модулі. Поки працює хоч

один сервіс, SCM контролює виконання основного потоку процесу. Як правило цей потік просто перебуває в стані очікування, не витрачаючи процесорного часу. Якщо адміністратор намагається запустити інший сервіс, реалізований в тім же виконавчому модулі, SCM не породжує нового екземпляра виконавчого модуля, а зв'язується з основним потоком існуючого виконавчого модуля, і знову переглядає список служб, цього разу в пошуках сервісу, який потрібно запустити. Знайшовши його, він породжує новий потік, який виконує функцію ServiceMaiп відповідного сервісу.

Система сама відслідковує, які сервіси виконуються в запущеному процесі. Коли який-небудь сервіс закінчує роботу (зазвичай при поверненні з ServiceMain), система перевіряє, чи залишилися ще працюючі сервіси. Повернення з StartServiceCtrlDispatcher відбувається, тільки коли не залишається працюючих служб. Потім програма повинна звільнити ресурси всього процесу, після чого варто виконати повернення з основної функції процесу, при якому система припинить його роботу. Ресурси повинні бути звільнені протягом 30 секунд, інакше SCM знищить процес.

Функція ServiceMain сервісу. Кожен сервіс в виконавчому файлі повинен мати свою функцію ServiceMain:

```
VOID WINAPI ServiceMain ( DWORD dwArgc, PTSTR* pszArgv);
```

SCM запускає сервіс, створюючи новий потік, що починає своє виконання з виклику ServiceMain. Йому можна дати будь-яке ім'я. Ім'я не має значення, тому що передається адреса цієї функції елементом lpServiceProc структури SERVICE_TABLE_ENTRY. Однак не можна мати дві однойменні функції ServiceMain в одному виконавчому файлі. Якщо це відбудеться, компілятор або компонувальник видадуть помилку, коли спробують зібрати свій проект.

В ServiceMain передаються два параметри. Вони створюють механізм, що дозволяє адміністраторові запустити сервіс із вказівкою параметрів командного рядка за допомогою функції StartService, хоча не рекомендується їх застосовувати. Ніж передавати необхідні для настройки сервісу параметри

в ServiceMain, краще зберігати й читати їх з підрозділу реєстру (Ім'я_сервісу варто замінити на дійсне ім'я сервісу).

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Ім'я_сервіс
у\Parameters
```

З багатьма сервісами поставляються клієнтські застосунки, що дозволяють адміністраторові їх настроювати. Ці клієнтські застосунки просто зберігають відповідні параметри в зазначеному підрозділі реєстру, і сервіс зчитує їх при запуску.

Якщо параметри конфігурації сервісу міняються в процесі його роботи, можливі три варіанти реакції на це з боку сервісу:

- сервіс ігнорує зміни конфігураційних параметрів до наступного запуску. Цей найпростіший варіант застосовується в багатьох існуючих на сьогодні сервісах;

- службі можна явно вказати, що їй треба себе перенастроїти. Це робить SCP за допомогою функції ServiceControl, передаючи їй значення SERVICE_CONTROL_PARAMCHANGE;

- сервіс може викликати функцію RegNotifyChangeKeyValue, щоб одержувати повідомлення, коли зовнішні застосунки змінюють його параметри в реєстрі. Це дозволяє службі перенастроюватися "на лету".

Перше, що повинен зробити ServiceMain, — це вказати SCM адресу функції зворотного виклику HandlerEx сервісу. Це робить функція RegisterServiceCtrlHandlerEx:

```
SERVICE_STATUS_HANDLE RegisterServiceCtrlHandlerEx
PCTSTR pszServiceName, // Внутрішнє ім'я сервісу
LPHANDLER_FUNCTION_EX pfnHandler,//Функція HandlerEx сервісу
PVOID pvContext); // Обумовлене користувачем значення
```

Перший параметр указує сервіс, для якого встановлюється функція HandlerEx, а другий — це адреса HandlerEx. Параметр pszServiceName повинен збігатися з ім'ям, зазначеним у масиві SERVICE_TABLE_ENTRY, який був переданий функції StartServiceCtrlDispatcher. Параметр pvContext — це обумовлене користувачем значення, що передається HandlerEx.

Функція RegisterServiceCtrlHandlerEx повертає дескриптор SERVICE_STATUS_HANDLE — унікальний ідентифікатор сервісу, що

використовується SCM. Вся подальша взаємодія сервісу з SCM здійснюється з використанням цього дескриптору, а не тексту, що визначає внутрішнє ім'я сервісу.

Після повернення з RegisterServiceCtrlHandlerEx потік ServiceMain повинен відразу повідомити SCM, що сервіс продовжує ініціалізуватися. Це робить функція SetServiceStatus:

```
BOOL SetServiceStatus ( SERVICE_STATUS_HANDLE hService,
LPSERVICE_STATUS pServiceStatus);
```

якій потрібно передати дескриптор, що ідентифікує сервіс (повернутий функцією RegisterServiceCtrlHandlerEx), і адресу структури SERVICE_STATUS:

```
typedef struct _SERVICE_STATUS
{DWORD dwServiceType; // тип модуля, що виконує
DWORD dwCurrentState; //поточний стан сервісу
DWORD dwControlsAccepted; //сигнали,сприймані сервісом
DWORD dwWin32ExitCode; // видача кодів помилок
DWORD dwServiceSpecificExitCode; // видача кодів помилок
DWORD dwCheckPoint; //повідомлення про роботу
DWORD dwWaitHint; } SERVICE_STATUS, *LPSERVICE_STATUS;
```

Структура SERVICE_STATUS складається із семи елементів, які відображають поточний стан сервісу. Ці наведені нижче елементи потрібно правильно встановити, перш ніж структура буде передана функції SetServiceStatus.

Функція HandlerEx сервісу. Кожен сервіс, що утримується у виконавчому модулі, повинен мати пов'язану з ним функцію HandlerEx:

```
DWORD WINAPI HandlerEx (
DWORD dwControl, // указує запитувану операцію
DWORD dwEventType,
PVOI0 pvEventData,
PVOID pvContext);
```

Більшу частини часу основний потік процесу перебуває в стані очікування під час звернення до функції StartServiceCtrlDispatcher. Коли SCP потрібне управління сервісом, SCM зв'язується з основним потоком процесу, у результаті чого цей потік розблоковується й викликає функцію HandlerEx відповідного сервісу.

Крім того, SCM повідомляє функції HandlerEx сервісу про пристрій, профіль обладнання і події, пов'язані з живленням комп'ютера. Ці

повідомлення дозволяють службі перенастроїти саму себе так, щоб адекватно реагувати на зміни в системі.

Оскільки HandlerEx виконується основним потоком, а сервіс — іншим, може виникнути необхідність взаємодії HandlerEx з потоком сервісу. Стандартного способу такої взаємодії немає. Обраний метод буде залежати від способу реалізації сервісу.

Можна організувати чергу викликів асинхронних процедур (APC), відправляти стан завершення вводу/виводу, повідомлення у вікно або діяти інакше.

3.3 Облікові записи служб

Контекст захисту сервісу дуже важливий для розроблювачів і системних адміністраторів, оскільки він визначає, до яких ресурсів отримає доступ цей сервіс. Більшість сервісів виконується в контексті захисту облікового запису локальної системи, якщо системним адміністратором або програмою установки не зазначене інше. В інтерфейсі користувача назва облікового запису локальної системи показується як Local System або SYSTEM. В Windows уведені два різновиди облікового запису локальної системи:

- локальний сервіс (local service);
- мережний сервіс (network service).

У порівнянні з обліковим записом локальної системи нові облікові записи мають менші права, і будь-який убудований в Windows сервіс, що не вимагає всіх можливостей облікового запису локальної системи, виконується під відповідним альтернативним обліковим записом.

Обліковий запис локальної системи Local System. Під цим обліковим записом виконуються базові компоненти режиму користувача, включаючи:

- диспетчер сеансів (\Windows\System32\Smss.exe);
- процес підсистеми Windows (Csrss.exe);

- підсистема локальної аутентифікації (\Windows\System32\LSASS.exe);
- процес входу до системи Winlogon (\Windows\System32\Winlogon.exe).

З позиції захисту, обліковий запис Local System має виняткові можливості – більші, ніж будь-який інший локальний або доменний обліковий запис. Далі наводяться його характеристики:

- його власник є членом групи локальних адміністраторів. У таблиці 3.2 перераховані групи, яким призначається обліковий запис локальної системи;
- він надає право на завдання практично будь-яких привілеїв (навіть таких, які зазвичай не призначаються обліковому запису локального адміністратора);
- він надає право на повний доступ до більшості файлів і розділів реєстру. Навіть якщо якісь об'єкти не дозволяють повний доступ, процеси під цим обліковим записом можуть скористатися привілеєм захоплення об'єкта у володіння (take-ownership privilege) і тим самим одержати потрібний вид доступу.

Таблиця 3.2 – Членство облікового запису сервісу в групах

Local System	Network Service	Local Service
Everyone	Everyone	Everyone
Authenticated Users	Authenticated Users	Authenticated Users
Administrators	Users	Users
	Local	Local
	Network Service	Local Service
	Service	Service

Процеси, що працюють під обліковим записом Local System, застосовують профіль користувача за замовчуванням (HKU\.DEFAULT).

Тому їм недоступна конфігураційна інформація, що зберігається в профілях користувачів, зіставлених з іншими обліковими записами.

Якщо дана система входить у домен Windows, обліковий запис Local System включає ідентифікатор захисту (SID) для комп'ютера, на якому виконується сервісний процес. Тому сервіс, що працює під обліковим записом Local System, буде автоматично автентифікований на інших машинах у тім же лісі.

Якщо тільки обліковому запису комп'ютера спеціально не призначені права доступу (до загальних мережних ресурсів, іменованим каналам тощо), процес може одержувати доступ до мережних ресурсів, що дозволяють так звані null-сеанси, тобто з'єднання, що не вимагають відповідних посвідчень захисту. Можливо вказувати загальні ресурси й канали, що дозволяють null-сеанси на конкретному комп'ютері, у параметрах NullSessionPipes і NullSessionShares розділу реєстру HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters.

Обліковий запис Network Service. Цей обліковий запис призначений для сервісів, яким потрібно автентифікуватися на інших комп'ютерах у мережі по обліковому записі комп'ютера, як це робиться у випадку облікового запису Local System. Але не їм потрібне членство в адміністративних групах або привілеї, призначувані обліковим записом Local System. Оскільки обліковий запис Network Service не відноситься до групи адміністраторів, виконувані під нею сервіси за замовчуванням одержують доступ до набагато меншої кількості розділів реєстру, а також каталогів і файлів у файловій системі, ніж обліковий запис Local System. Більш того, призначення меншого числа привілеїв обмежує можливості скомпрометованого процесу мережного сервісу.

Процеси, виконувані під обліковим записом Network Service, використають його профіль. Він завантажується в розділ реєстру HKU\S-1-5-20, а його файли й каталоги перебувають в \Documents and Settings\Network-Service. В Windows під обліковим записом Network Service виконується DNS-

клієнт, відповідальний за дозвіл DNS-імен і пошук контролерів домена.

Обліковий запис Local Service. Цей обліковий запис практично ідентичний Network Service з тією відмінністю, що дозволяє звертатися лише до тих мережних ресурсів, які дозволяють анонімний доступ. У таблиці 3.2 показано, що він належить до тих же груп (якщо не вважати групи Network Service і Local Service). Профіль, що використовується процесами, виконуваними під обліковим записом Local Service, завантажується в розділ реєстру HKU\S-l-5-19 і зберігається в \Documents and Settings\LocalService.

4 ПРОГРАМНА МОДЕЛЬ АДМІНІСТРУВАННЯ ДРАЙВЕРІВ ТА СЕРВІСІВ

4.1 Загальна характеристика програмної моделі

Інструментом рішення поставленого завдання є мова програмування. Вибір мови був зупинений на ассемблері. Він є одним з найбільш потужних мов низького рівня, відповідає всім вимогам ефективного програмування. Програмний код відрізняється невеликим обсягом виконавчого файлу.

Проект будується за модульним принципом, а також складається з наступних файлів:

- головного файлу проекту – main.asm. Його основне призначення полягає в створенні головного вікна застосунку та обробки повідомлень;
- файлу services.asm, який містить функції для роботи із сервісами;
- файлу servicedlg.asm, який призначений для обробки дій користувача з діалоговим вікном "Властивості сервісу";
- файлу controlservicedlg.asm, який містить процедури діалогового вікна "Керування сервісом";
- файлу subserventdlg.asm, функції якого використовуються для обробки діалогових вікон прав доступу та з'єднання з вилученим комп'ютером;
- файлу memory.asm, в якому наводяться функції для роботи з пам'яттю з метою її виділення, звільнення;
- файлу log.asm, який містить функції для роботи з помилками й повідомленнями;
- файлу about.asm, який не тільки дозволяє переглядати вміст діалогового вікна "Про програму".

Крім того, в проекті присутні відповідні inc-файли з прототипами використаних функцій та констант, а також файл ресурсу – main.rc. Для

облегшення процесів компіляції та компонування exe-файлу використовується файл MakeFile.bat, який дозволяє провести компіляцію ресурсів та asm-файлів за допомогою таких команд:

```
Rc.exe /v main.rc
ML.EXE /c /coff /Cp /nologo main.asm
ML.EXE /c /coff /Cp /nologo memory.asm
ML.EXE /c /coff /Cp /nologo log.asm
ML.EXE /c /coff /Cp /nologo services.asm
ML.EXE /c /coff /Cp /nologo servicedlg.asm
ML.EXE /c /coff /Cp /nologo controlservicedlg.asm
ML.EXE /c /coff /Cp /nologo subserventdlg.asm
ML.EXE /c /coff /Cp /nologo about.asm
LINK.EXE /SUBSYSTEM:WINDOWS /RELEASE /OUT:SrvCtrl.exe main.res
main.obj memory.obj log.obj services.obj servicedlg.obj
controlservicedlg.obj subserventdlg.obj about.obj
```

Результатом виконання MakeFile.bat буде виконавчий файл SrvCtrl.exe.

Допоміжна директорія \..\Resources\ містить файли курсору, іконки та bitmap з піктограмами панелі інструментів.

4.2 Загальна характеристика програмної моделі

Загалом проект складається з 86 функцій, які розташовуються в окремих файлах (таблиця 4.1)

Таблиця 4.1 – Розміщення та призначення функцій проекту

№	Назва функції	Призначення
1	2	3
Файл: main.asm		
1	WinMain	Головна функція застосунка
2	MainWndProc	Віконна процедура головного вікна застосунка
3	OnCreate	Обробка події WM_CREATE головного вікна
4	OnSize	Обробка події WM_SIZE головного вікна
5	GetMinMaxinfo	Заповнення структури щодо розмірів вікна
6	OnLButtonDown	Оброблювач натискання лівої кнопки миші
7	OnLButtonDown	Оброблювач віджатої лівої кнопки миші

Продовження таблиці 4.1

1	2	3
Файл: main.asm		
8	OnMouseMove	Оброблювач переміщення миші (робота splitter)
9	OnNotify	Повідомлення від дочірніх контролів
10	UpdateServControls	Обновити стан кнопок керування сервісом
11	CreateToolBar	Функція створення головного toolbar
12	SetBmpColor()	Залити фон bitmap кольором кнопки
13	SetToolBarButton	Замінити кнопку toolbar
14	CreateTreeView	Функція створення дерева
15	FreeResources	Звільнення ресурсів
16	SCMConnectTo	З'єднання з вилученим комп'ютером
17	ExportDataToFile	Експортувати дані
18	Exit	Вийти із програми
19	OpenSCMAccess	Відкрити діалог Необхідні права доступу до SCM
20	OpenSCMConnection	Відкрити з'єднання з SCM
21	CloseSCMConnection	Закрити з'єднання з SCM
22	LockSCM	Заблокувати базу SCM
23	UnLockSCM	Розблокувати базу SCM
24	OpenServAccess	Відкрити діалог Необхідні права доступу
25	OpenServSecurity	Відкрити діалог Безпека
26	AddServRecord	Додавати новий сервіс у базу SCM
27	EditServRecord	Редагування властивостей сервісу
28	DeleteServRecord	Видалити сервіс із бази SCM
29	ServControl	Управління сервісом
30	ServControlDlg	Послати сервісу керуючий код
31	RefreshData	Обновити дані
32	OpenHelp	Допомога
33	OpenAboutDlg	Про програму

Продовження таблиці 4.1

1	2	3
34	ExportListToFile	Експортувати дані з listview control у файл
35	ServiceListProc	Віконна процедура listview списку сервісів
Файл: about.asm		
36	AboutDialogFunc	Процедура діалогового вікна Про Програму
37	PrepareUrlParam	Підготовляє структуру для роботи з URL
38	ClearUrlParam	Очищає блок параметрів URL
Файл: about.asm		
39	SetHyperLink	Змінює елемент тексту для функціонування в якості URL
40	MailToWndProc	Віконна процедура URL
Файл: controlservicedlg.asm		
41	ServiceControlDialogFunc	Процедура діалогового вікна Керування сервісом
42	CloseDialog	Звільнити ресурси й закрити діалог
43	UpdateStatusControls	Обновити статусну інформацію
Файл: log.asm		
44	GetLastErrorString	Заповнити буфер рядком опису помилки
45	LogMessage	Вставити рядок статусу у вікно повідомлень
Файл: memory.asm		
46	malloc	Виділити пам'ять із купи процесу
47	realloc	Змінити розмір пам'яті(без перенесення даних!)
48	mfree	Звільнити пам'ять
Файл: servicedlg.asm		
49	ServicePropertiesDialogFunc	Процедура діалогового вікна Властивості сервісу
50	ServicePropertiesClose	Звільнити ресурси й закрити діалог

Продовження таблиці 4.1

1	2	3
51	ServGeneralDialogFunc	Процедура закладки загальні
Файл: servicedlg.asm		
52	ServLogonDialogFunc	Процедура закладки обліковий запис
53	ServRecoveryDialogFunc	Процедура закладки відновлення
54	ServDependenciesDialogFunc	Процедура закладки залежності
55	NumEditWndProc	Віконна процедура EditBox'a для уведення тільки цифр
56	FillComboBox	Заповнити combobox
57	GetRecovereAction-FromComboBox	Одержати прапор
58	FillDependenciesList	Заповнити список залежностей
59	SetFailureAction	Заповнити combobox і пов'язані з ним controls даннями про збій
60	GetFailureAction	Заповнити combobox і пов'язані з ним controls даннями про збій
61	LoadData	Завантажити дані й заповнити controls
62	SaveData	Зберегти дані
63	GetGeneralData	Заповнити буфер загальними даними сервісу
64	GetFailureData	Заповнити буфер параметрами відновлення сервісу
65	GetDescriptionData	Заповнити буфер описом сервісу
Файл: servicedlg.asm		
66	CheckData	Перевірити дані на правильність
67	GetDwordFromString	Перевести рядок у ціле
68	SetTokenPrivilege	Установити привілею для поточного маркера

Продовження таблиці 4.1

1	2	3
Файл: services.asm		
69	FillServicesList	Заповнити список сервісів
70	FillServicesCombo	Заповнити combobox списку сервісів
71	FillOrderGroupCombo	Заповнити combobox груп списку сервісів
72	UpdateServiceRecord	Обновити рядок таблиці
73	FillBufferFromFlag	Заповнити буфер строковими значеннями прапорів
74	FillBufferFromCode	Заповнити буфер строковими значенням коду
75	WaitForServiceState	Очікувати поки стану сервісу не зміниться до бажаного або не вийде тайм-аут
76	GetServiceStatus	Одержати статусну інформацію сервісу
77	ServiceStartOperation	Запустити сервіс
78	ServiceStopOperation	Зупинити сервіс
79	ServicePauseOperation	Призупинити сервіс
80	ServiceResumeOperation	Відновити роботу сервісу
81	ServiceRestartOperation	Запустити знову сервіс
82	ServiceControlOperation	Послати сервісу керуючий код
83	ServiceDeleteOperation	Видалити сервіс
Файл: subserventdlg.asm		
84	SCMAccessDialogFunc	Процедура діалогового вікна Права доступу до SCM
85	ServAccessDialogFunc	Процедура діалогового вікна Права доступу до сервісів
86	ConnectToDialogFunc	Процедура діалогового вікна Підключення до вилученого комп'ютера

4.3 Вхідні дані, робота програми і результати її виконання

Вхідними даними "Монітор драйверів і сервісів" є:

- необхідність завдання прав доступу до диспетчеру управління сервісами;
- вибір самого диспетчера управління сервісами (SCM).

Права доступу до диспетчера управління сервісами позначаються такими ж стандартними прапорами, які використовуються диспетчером безпеки та управління доступом (таблиця 4.2).

Таблиця 4.2 – Права доступу до диспетчера управління сервісами

Права доступу до SCM	Призначення
1	2
SC_MANAGER_ALL_ACCESS	Включає всі перераховані типи прав доступу
STANDARD_RIGHTS_READ	Комбінація прапорців: SC_MANAGER_ENUMERATE_SERVICE SC_MANAGER_QUERY_LOCK_STATUS
STANDARD_RIGHTS_WRITE	Комбінація прапорців: SC_MANAGER_CREATE_SERVICE SC_MANAGER_MODIFY_BOOT_CONFIG
SC_MANAGER_CONNECT	Дозволяє підключення до SCM. Цей вид доступу враховується завжди, навіть якщо не зазначений явно
SC_MANAGER_CREATE_-SERVICE	Дозволяє виклик функції CreateService, що додає служби в базу даних SCM.
SC_MANAGER_-ENUMERATE_SERVICE	Дозволяє виклик функції EnumServicesStatusEx, що дозволяє одержати список служб, які находяться в базі даних SCM і визначити їх стан.

Продовження таблиці 4.2

1	2
SC_MANAGER_BOOT_-CONFIG	Дозволяє виклик функції NotifyBootConfig-Status
SC_MANAGER_LOCK	Дозволяє виклик функції LockService-Database, що не дає можливості SCM запускати служби.
SC_MANAGER_QUERY-_LOCK_STATUS	Дозволяє виклик функції QueryService-LockStatus, яка знаходить користувача (якщо такий існує), що заблокував базу даних SCM.

Права доступу до сервісів позначаються такими ж стандартними прапорами, які використовуються диспетчером безпеки та управління доступом (табл.4.3).

Таблиця 4.3 – Права доступу до сервісів

Права доступу	Призначення
1	2
SERVICE_ALL_ACCESS	Включає всі перераховані типи прав доступу
DELETE	Дозволяє видалення служби за допомогою функції DeleteService
READ_CONTROL	Дозволяє виклик функції QueryService-ObjectSecurity на запит диспетчера безпеки
WRITE_DAC	Дозволяє виклик функції SetServiceObjectSecurity, щоб модифікувати елемент DACL об'єкту дескриптора безпеки
WRITE_OWNER	Дозволяє виклик функції SetServiceObjectSecurity, щоб модифікувати елементи Owner або Group об'єкту дескриптора безпеки

Продовження таблиці 4.3

1	2
SERVICE_START	Дозволяє запускати службу за допомогою функції StartService
SERVICE_STOP	Дозволяє завершувати роботу служби за допомогою функції ControlService
SERVICE_QUERY_CONFIG	Дозволяє запитати конфігурацію служби за допомогою функції QueryServiceConfig або QueryServiceConfig2
SERVICE_CHANGE_CONFIG	Дозволяє зміну конфігурації служби за допомогою функції ChangeServiceConfig або ChangeServiceConfig2
SERVICE_QUERY_STATUS	Дозволяє запитувати у SCM дані про стан служби за допомогою функцій Query-ServiceStatus
SERVICE_ENUMERATE_-DEPENDENTS	Дозволяє перебирати всі служби, залежні від даної служби, за допомогою функції EnumDependentServices
SERVICE_PAUSE_CONTINUE	Дозволяє тимчасово переривати роботу служби й продовжувати її після паузи за допомогою функції ControlService. Цей вид доступу також дозволяє зміну параметрів служби
SERVICE_INTERROGATE	Дозволяє запитувати службу про її стан за допомогою функції ControlService
SERVICE_USER_-DEFINED_CONTROL	Дозволяє вказувати користувачеві коди управління за допомогою функції Control-Service

Запуск програми здійснюється запуском файлу MonDrvSrv.exe без

параметрів. Результат запуску відображений у вигляді графічного інтерфейсу (рисунок 4.1).

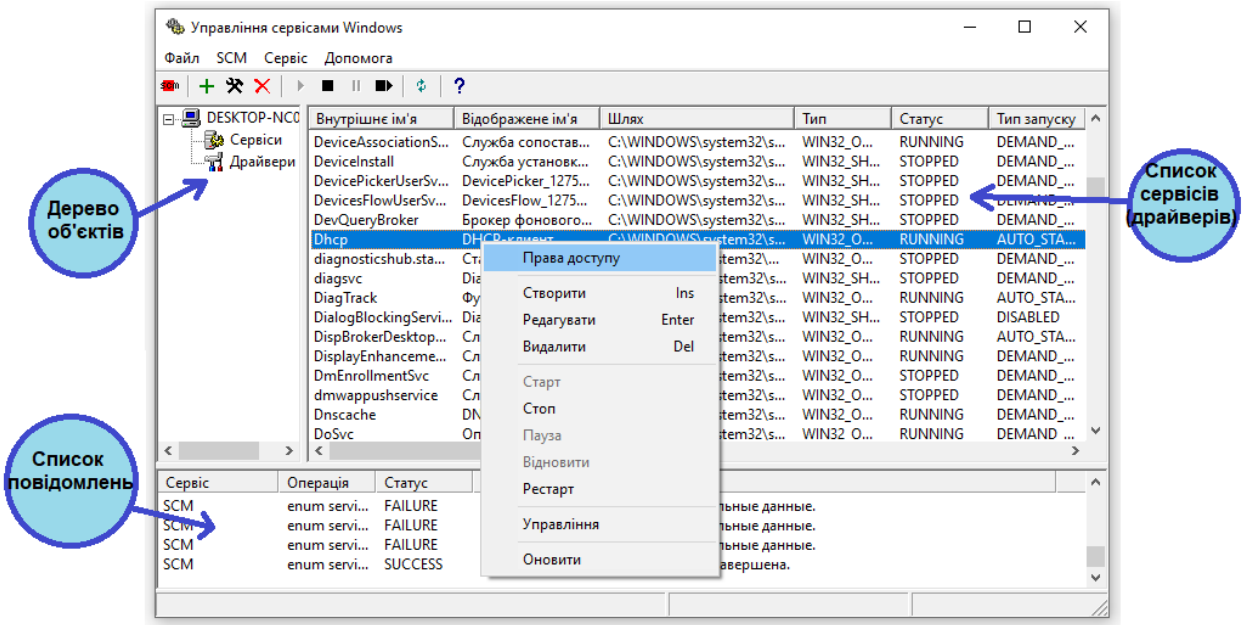


Рисунок 4.1 – Головне вікно програми після запуску

Головне меню програми містить чотири пункти. Вони мають наступне призначення:

Перший пункт головного меню "Файл" – використовується для доступу к діалоговому вікну встановлення з'єднання з віддаленим комп’ютером.

Другий пункт головного меню "SCM" – Service Control Manager – призначений для виконання операцій з диспетчером управління сервісами (драйверами).

Третій пункт головного меню – "Сервіс" – безпосередньо призначений для виконання операцій над сервісами та драйверами.

Останній четвертий пункт головного меню – "Помощь" – призначений для коротких пояснень щодо роботи та захисту прав розробника програми.

В якості приклада наведено управління властивостями сервісу AudioSrv (рисунок 4.2).

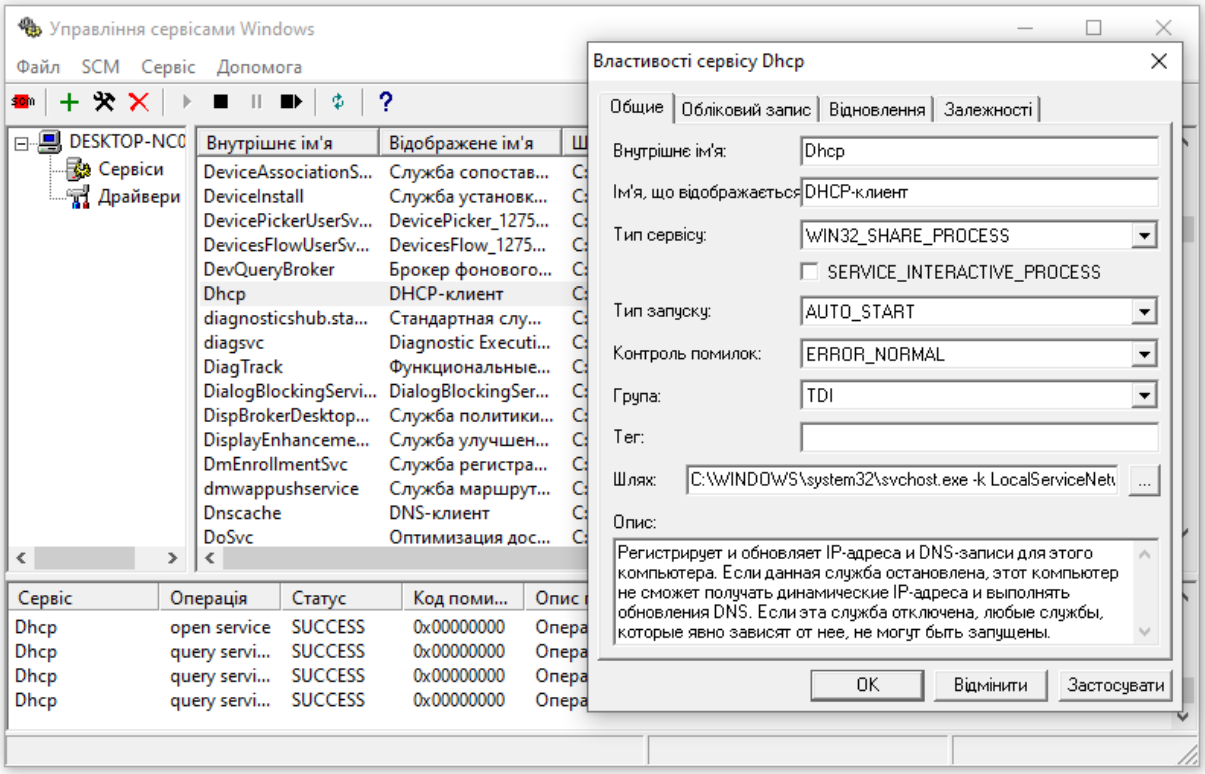


Рисунок 4.2 – Результат управління службою

4.4 Рекомендації користувачеві

Управління сервісами Windows – це утиліта призначена для роботи із сервісами й драйверами, які встановлені в системі. Програма дозволяє переглядати список установлених сервісів і драйверів як на локальному, так і на віддаленому комп'ютері. А також управляти службою/драйвером, реєструвати, змінювати параметри й видаляти сервіс або драйвер.

Нормальне функціонування розробленої програми можливо у випадку, якщо робота з нею відбувається на персональному комп'ютері типу IBM PC (або сумісному з ним) і що має у якості робочої платформи ОС Windows.

Для функціонування програмного виробу необхідні такі мінімальні набори периферійних пристроїв:

- кольоровий монітор;
- відеокартка;

- вінчестер;
- маніпулятор миша;
- клавіатура;
- мережна карта або модем для доступу до Internet.

Головне вікно програми розбите на три частини:

- дерево об'єктів;
- список сервісів (драйверів);
- список повідомлень.

Дерево об'єктів містить дві групи об’єктів: сервіси й драйвери. В залежності від побажань адміністратора проводиться вибір групи для відображення потрібної інформації.

Список сервісів містить інформацію про сервіси або драйвери, які встановлені у системі.

Вікно повідомлень містить всі проведені операції з описом статусу їх завершення.

Головне вікно програми має чотири пункти меню.

Перший пункт головного меню "Файл" (рисунок 4.3) – використовується для доступу к діалоговому вікну встановлення з'єднання з віддаленим комп’ютером. Для цього служить пункт меню "З'єднання з…", через функцію якого SCMConnectTo proc hWnd:HWND визивається функція ConnectToDialogFunc, яка і виконує обробку повідомлень.

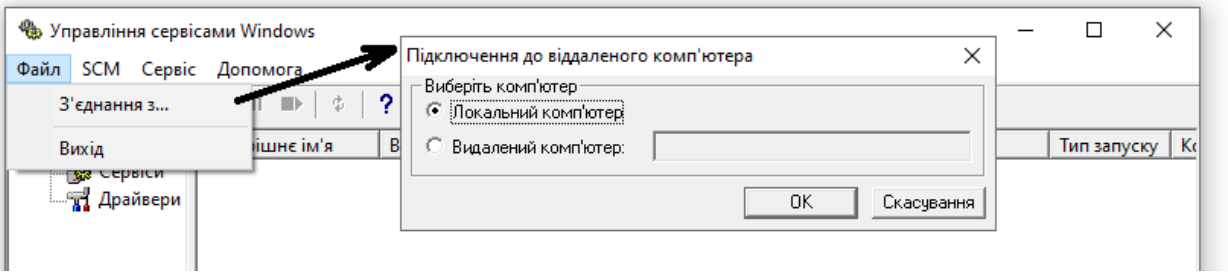


Рисунок 4.3 – Пункт меню "Файл" с діалоговім вікном встановлення з'єднання з віддаленим комп’ютером

Інший пункт меню "Файл" – Вихід – використовується для завершення роботи з програмою та закриття головного вікна.

Другий пункт головного меню "SCM" – Service Control Manager – призначений для виконання операцій з диспетчером управління сервісами (драйверами). Він містить наступні підпункти меню (рисунок 4.4).

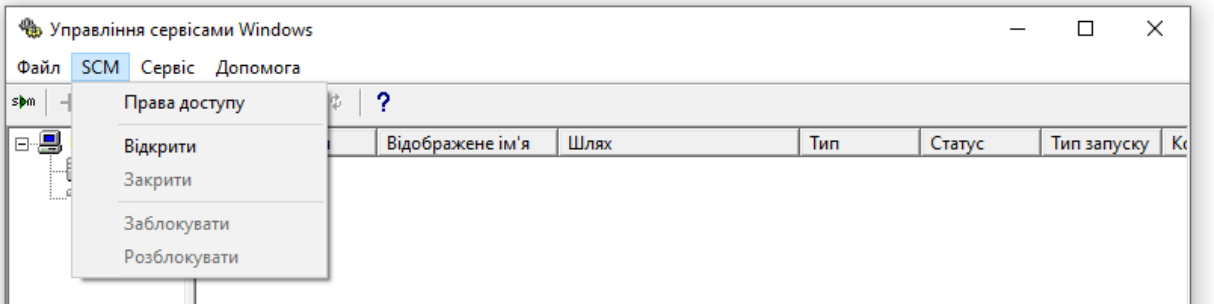


Рисунок 4.4 – Пункт меню " SCM "

Призначення підпунктів меню "SCM" визначається таким чином:

- "Права доступу" – права доступу до диспетчера управління сервісами (за замовчуванням задані мінімальні права). При виборі цього підпункту функцією OpenSCMAccess викликається діалогове вікно настройки прав доступу до SCM (рисунок 4.5) функцією DialogBoxParam (hInstance, IDD_SCM_ACCESS, hWnd, SCMAccessDialogFunc, addr dwSCMAccess);

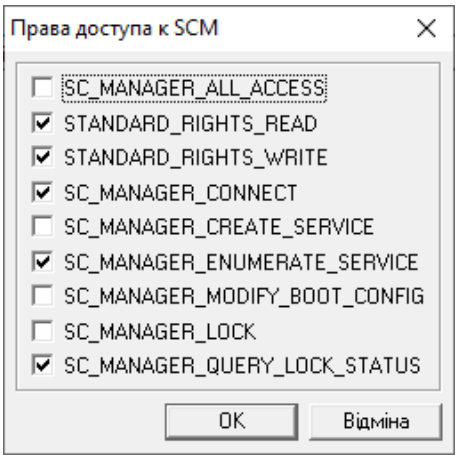


Рисунок 4.5 – Діалогове вікно настройки прав доступу до SCM

- "Відкрити" - відкрити з'єднання з SCM. При виборі цього підпункту викликається відповідна функція – OpenSCMConnection;

- "Закрити" - закрити з'єднання з SCM. При виборі цього підпункту викликається відповідна функція – CloseSCMConnection;

- "Заблокувати" - заблокувати базу даних SCM (для монопольного доступу до бази). При виборі цього підпункту викликається відповідна функція – LockSCM;

- "Розблокувати" - розблокувати базу даних SCM. При виборі цього підпункту викликається відповідна функція – UnLockSCM;

- третій пункт головного меню – "Сервіс" – безпосередньо призначений для виконання операцій над сервісами та драйверами. Він містить наступні підпункти меню (рисунок 4.6).

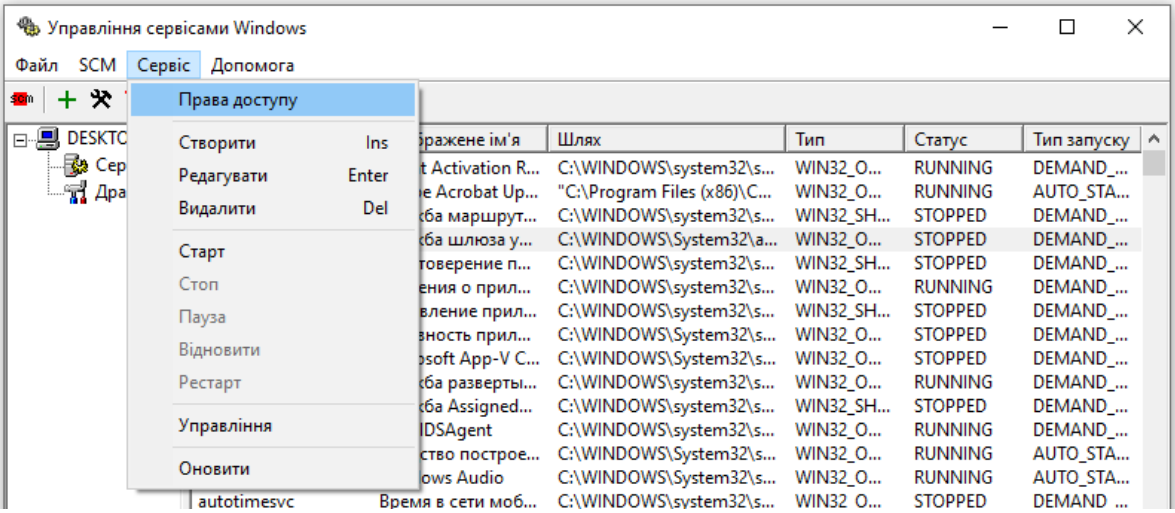


Рисунок 4.6 – Пункт меню "Сервіс"

Призначення підпунктів меню "SCM" визначається таким чином:

- "Права доступу" – права доступу до об'єкту сервісу (використовується у функціях відкриття, створення, видалення, керування сервісом тощо) При виборі цього підпункту функцією OpenServAccess викликається діалогове вікно настройки прав доступу до сервісу (рисунок 4.7) функцією DialogBoxParam (hInstance, IDD_SERV_ACCESS,

hWnd, ServAccessDialogFunc, addr dwServAccess. За замовчуванням встановлені мінімальні права, котрі показані на рисунок 4.7;

- "Створити" - зареєструвати сервіс у системі. При виборі цього підпункту викликається відповідна функція – AddServRecord;

- "Редагувати" – редагувати параметри сервісу. При виборі цього підпункту викликається відповідна функція – EditServRecord;

- "Видалити" - видалити інформацію про сервіс із бази даних SCM. При виборі цього підпункту викликається відповідна функція – DeleteServRecord.

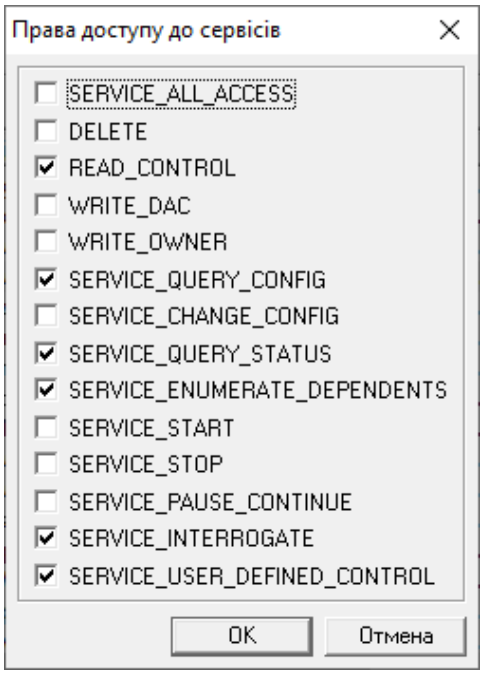


Рисунок 4.7 – Діалогове вікно настройки прав доступу до сервісу

Надалі йде група підпунктів, вибір одного з яких призведе до виклику функції ServControl, в якості другого параметру передається відповідне до нього значення:

- "Старт" - запуск сервісу. Викликається функція – ServControl з параметром ServiceStartOperation;

- "Стоп" - останов сервісу. Викликається функція – ServControl з

параметром ServiceStopOperation;

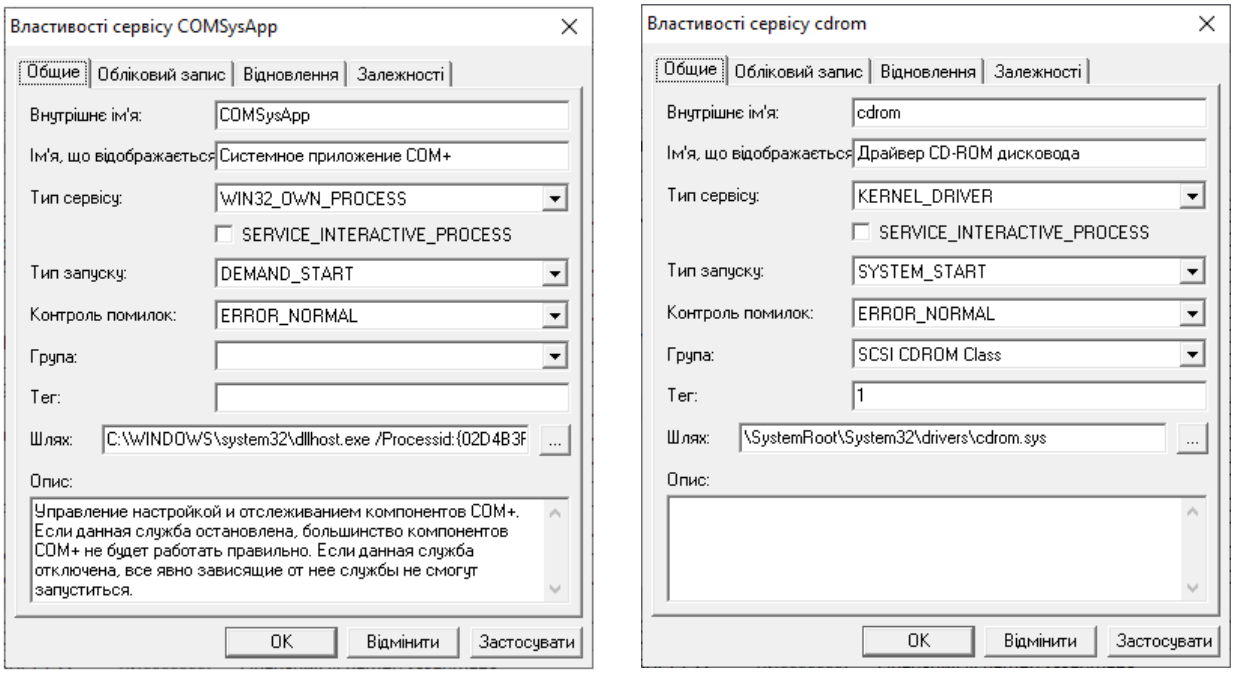
- "Пауза" - приостанов сервісу. Викликається функція – ServControl з параметром ServicePauseOperation;

- "Відновити" - поновлення роботи сервісу. Викликається функція – ServControl з параметром ServiceResumeOperation;

- "Рестарт" - перезапуск сервісу. Викликається функція – ServControl з параметром ServiceRestartOperation;

- "Управління" - відкриття діалогу управління сервісом (рис.5.6). При виборі цього підпункту викликається відповідна функція – ServControlDlg;

- "Обновити" - відновлення списку сервісів. При виборі цього підпункту викликається відповідна функція – RefreshData.



а) б)

Рисунок 4.8 – Діалогове вікно властивостей: а) сервісу; б) драйверу

Останній четвертий пункт головного меню – "Допомога" – призначений для коротких пояснень щодо роботи та захисту прав розробника програми. Він містить підпункт:

- "Про програму" - виклик діалогового вікна про розробника програми (рисунок 4.9). При виборі цього підпункту викликається відповідна функція – DialogBoxParam (hInstance, IDD_ABOUT, hWnd, AboutDialogFunc, NULL), яка й займається обробкою повідомлень для цього діалогового вікна.

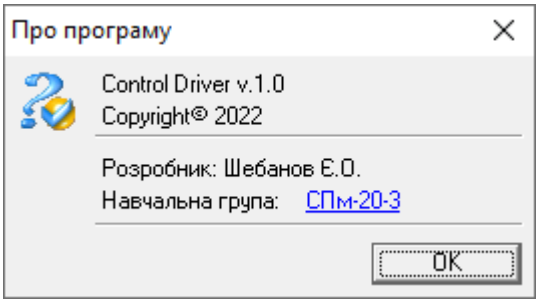


Рисунок 4.9 – Діалогове вікно про програму

Функції більшості підпунктів головного меню підтримуються в графічному вигляді панеллю інструментів, яка містить одинадцять кнопок (рисунок 4.10).

Рисунок 4.10 – Панель інструментів та призначення кнопок

Для початку роботи із програмою після завантаження необхідно задати права доступу до SCM і вибрати пункт меню або кнопку панелі інструментів "Відкрити з'єднання з SCM". Після запуску програма резидентна в оперативній пам'яті та займає всього приблизно 2 Мб.

Перевагою розробленого програмного забезпечення є можливість уніфікованим способом здійснювати адміністрування як сервісів, так й драйверів на локальному хості, а також підключатися до довільного вузла по локальній мережі.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи вивчені принципи побудови підсистеми вводу/виводу операційної системи Windows, інструментальні засоби управління сервісами. Функціональність підсистеми вводу/виводу полягає в:

- стандартних засобах безпеки й іменування пристроїв для захисту поділюваних пристроїв;
- високопродуктивному асинхронному пакетному вводі/виводі для підтримки масштабування;
- сервісах для написання драйверів пристроїв;
- підтримці багаторівневої моделі й розширюваності для додавання драйверів, що модифікують поводження інших драйверів або пристроїв без внесення змін у них;
- динамічному завантаженні та вивантаженні драйверів пристроїв;
- підтримці PnP, завдяки якої система знаходить і встановлює драйвери для нового обладнання, а потім виділяє їм потрібні апаратні ресурси;
- керуванні електроживленням;
- підтримці множини файлових систем;
- підтримці Windows Management Instrumentation і засобів діагностики.

Важливими моментами забезпечення функціональності є можливості динамічного завантаження і вивантаження драйверів, узагальнений вид формованих структур запитів на ввід/вивід і диспетчеризація.

З огляду на те, що ОС Windows у будь-яких версіях є однією із провідних операційних систем на ринку автоматизації, є сенс глибше вивчити питання будови цієї ОС і оволодіти технікою написання драйверів та сервісів для неї.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Пирогов В. Ю. Асемблер для Windows [Текст] / В.Ю. Пирогов. – Київ: Пітер в Україні, 2017. – 656 с.
2. Руссинович М. Внутреннее устройство Microsoft Windows. Ч.1 [Текст] / М. Руссинович, Д. Соломон. – СПб.: Питер, 2013. – 800 с.
3. Руссинович М. Внутреннее устройство Microsoft Windows. Ч.2. Основные подсистемы ОС [Текст] / М. Руссинович, Д. Соломон, А. Ионеску. – СПб.: Питер, 2014. – 672 с.
4. Рихтер Дж. Программирование серверных приложений для Microsoft Windows. Мастер-класс [Текст] / Дж. Рихтер, Д.Дж.Кларк. – СПб. : Питер, 2001. – 592 с.
5. Юров В.И. Ассемблер практикум [Текст] / В.И. Юров. – СПб.: Питер, 2016. – 399 с.
6. Юров В.И. Assembler. Учебник для вузов [Текст] / В.И. Юров. – СПб.: Питер, 2018. – 637 с.
7. Рихтер Дж. Windows для профессионалов: создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows. [Текст] / Дж. Рихтер. – СПб.: Питер, 2003. – 752 с.
8. Саймон Р. Windows API. Енциклопедія програміста. [Текст] / Р.Саймон. – Київ: ПАТ "ДіасофтЮП", 2004. – 1088 с.
9. Сорокина С.И. Программирование драйверов и систем безопасности: Учебное пособие [Текст] / С. И. Сорокина, А. Ю. Тихонов, А. Ю. Щербаков – СПб.: БХВ-Петербург, 2003. – 256 с.
10. Microsoft System Center Operations Manager Enterprise [Електронний ресурс] . – 2022. – Режим доступу до ресурсу: https://www.microsoft.com.
11. Диспетчер управління службами [Електронний ресурс] . – 2022. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/.