Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-21М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Підлубний О.В.
«____» _____ 2022 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Доренський О.П.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Підлубному Олексію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж*

2. Керівник роботи *Доренський Олександр Павлович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Доренський О.П.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Підлубний О.В.
(прізвище та ініціали)

АНОТАЦІЯ

Підлубний О.В. Дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віддаленого доступу з використанням WAN-мереж.

Метою розробки є дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

Об'єктом дослідження є процес віддаленого доступу з використанням WAN-мереж.

Предметом дослідження є методи віддаленого доступу з використанням WAN-мереж.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi.

Ключові слова: комп'ютерні науки, віддалений доступ, WAN-мережі

ABSTRACT

Pidlubnyi O.V. Research and software implementation of a remote access system using WAN networks. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for a remote access system using WAN networks.

The purpose of the development is research and software implementation of a remote access system using WAN networks.

The object of research is the process of remote access using WAN networks.

The subject of research is methods of remote access using WAN networks.

Research methods are based on computer network theory methods, mathematical statistics methods, and software development methods.

The result of the work is a software implementation of a remote access system using WAN networks.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi environment.

Keywords: computer science, remote access, WAN networks

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АВН	–	автоматичний визначник номера
АТМ	–	автоматична телефона мережа
ЛОМ	–	локальна обчислювальна мережа
ОП	–	оперативна пам'ять
ОС	–	операційна система
ПК	–	персональний комп’ютер
ЦП	–	центральний процесор
AES	–	алгоритм шифрування
DNS	–	сервер домених імен
IP	–	Internet Protocol
LAN	–	локальна мережа
MAC	–	Media Access Control – управління доступом до носія
NAT	–	Network Address Translation
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol – протокол користувальницьких дейтаграмм
UPS	–	блок безперебійного живлення

ВСТУП

Актуальність теми. Сучасний світ тяжко представити без наявності мереж. У тому або іншому вигляді кожна людина на даний момент стикається з мережами: починаючи від роботи та спілкування в Інтернет й закінчуючи роботою з локальною мережею дома, або на роботі. Ще більше проникнення мереж у життя відбулося з введенням електронного документообігу й електронних платежів (у тому числі й отримання грошей через мережу банкоматів).

Таке глибоке проникнення мережевих технологій приводить до того, що доволі часто виникає потреба у віддаленому управлінні ЕОМ через локальну мережу або Інтернет та контролі того, що відбувається на віддаленій ЕОМ.

До кругу таких задач можна віднести наступні:

– необхідно контролювати, чим займаються працівники за комп'ютерами на робочому місті;

– при зверненні до мережевої служби підтримки, доволі часто більш зручно не вислуховувати діагноз поломки, або суть проблеми у виконанні клієнта, а зайти на його ПК й самому розібратися, яка у нього проблема й як її можна вирішити;

– у системі банкоматів, доволі часто виникає потреба у необхідності зайти в програмне забезпечення, яке керує банкоматом, або відслідити дії налагоджувача банкомату;

– перебуваючи в гостях або на роботі віддалено зайти на свій домашній комп'ютер й передивитись якусь інформацію, або віддалено запустити якусь програму.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем віддаленого доступу з використанням WAN-мереж.

– Дослідження системи віддаленого доступу з використанням WAN-мереж.

– Програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

Об'єктом дослідження є процес віддаленого доступу з використанням WAN-мереж.

Предметом дослідження є методи віддаленого доступу з використанням WAN-мереж.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод віддаленого доступу з використанням WAN-мереж.

– Розроблено вітчизняний продукт віддаленого доступу з використанням WAN-мереж, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віддаленого доступу з використанням WAN-мереж.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Якщо магістральні зв'язки між локальними мережами завжди будуються шляхом з'єднання локальних мереж з територіальним транспортом через маршрутизатори, то для організації віддаленого доступу можуть використовуватися різні схеми й продукти. Продукти віддаленого доступу можуть істотно відрізнятися реалізованими в них функціями, а виходить, і можливостями при рішенні конкретної практичної задачі.

Віддалений доступ – дуже широке поняття, що містить у собі різні типи й варіанти взаємодії комп'ютерів, мереж і додатків. Якщо розглядати всі численні схеми взаємодії, які звичайно відносять до віддаленого доступу, то всім їм властиве використання глобальних каналів або глобальних мереж при взаємодії. Крім того, для віддаленого доступу, як правило, характерна несиметричність взаємодії, коли, з одного боку, є центральна велика мережа або центральний комп'ютер, а з іншого боку – окремий віддалений термінал, комп'ютер або невелика мережа, які хочуть одержати доступ до інформаційних ресурсів центральної мережі. Кількість віддалених від центральної мережі вузлів і мереж, що вимагають цей доступ, постійно ростуть, тому сучасні засоби віддаленого доступу розраховані на підтримку великої кількості віддалених клієнтів.

Типи взаємодіючих систем

Основні схеми віддаленого доступу, відрізняються типом взаємодіючих систем:

- термінал-комп'ютер;
- комп'ютер-комп'ютер;
- комп'ютер-мережа;
- мережа-мережа.

Перші три види віддаленого доступу часто поєднують поняттям індивідуального доступу, а схеми доступу мережа – мережу іноді ділять на два класи – ROBO і SOHO. Клас ROBO (Regional Office/Branch Office) відповідає випадку підключення до центральної мережі мереж середніх розмірів – мереж регіональних підрозділів підприємства, а класу SOHO (Small Office/Home Office) відповідає випадок віддаленого доступу мереж невеликих офісів і домашніх мереж.

Типи підтримуваних служб

Схеми віддаленого доступу можуть відрізнятися також і типом служб, які підтримуються для віддаленого клієнта. Найбільше часто використовується віддалений доступ до файлів, базам даних, принтерам у том же стилі, до якого користувач звик при роботі в локальній мережі. Такий режим називається режимом віддаленого вузла (remote node). Іноді при віддаленому доступі реалізується обмін із центральною мережею повідомленнями електронної пошти, за допомогою якого можна в автоматичному режимі одержати запитувані корпоративні дані, наприклад з бази даних.

Особливе місце серед всіх видів віддаленого доступу до комп'ютера займає спосіб, при якому користувач одержує можливість віддалено працювати з комп'ютером таким же способом, як якби він управляв їм за допомогою локально підключеного термінала. У цьому режимі він може запускати на виконання програми на віддаленому комп'ютері й бачити результати з виконання. При цьому прийнято підрозділяти такий спосіб доступу на термінальний доступ і віддалене керування. Якщо у віддаленого користувача в розпорядженні є тільки неінтелектуальний алфавітно-цифровий термінал або ж він запускає на своєму персональному комп'ютері програму емуляції такого термінала, то такий режим роботи називають термінальним доступом. Для власника алфавітно-цифрового термінала, наприклад VT-100, цей вид віддаленого доступу є єдино можливим. Доступ до мейнфрейму IBM, що працює під керуванням операційної системи MVS, за допомогою доступу через віддалений або убудований PAD, що потім працює з мейнфреймом через мережу X.25, також являє приклад термінального доступу. Відмінною рисою термінального доступу є те, що операційні системи на

комп'ютері, до якого одержують доступ користувачі, розраховані на багатотермінальний режим роботи, тому головне тут – відмінна від стандартного варіанта схема підключення термінала, орієнтована на глобальні мережі.

При **віддаленому керуванні** користувач запускає на своєму комп'ютері програму, що емулює йому на екрані сеанс роботи з операційною системою - Windows, Linux, Unix, FreeBSD OS/2, – яка не підтримує багатотермінальний режим роботи. Програма емуляції екрана через глобальні канали взаємодіє з додатковим програмним забезпеченням, що працює під керуванням відповідної операційної системи на віддаленому комп'ютері. Користувач, як і при термінальному доступі, також одержує повне керування віддаленим комп'ютером, при цьому він бачить на екрані графічний інтерфейс звичної йому операційної системи, у якості якої найчастіше виступає Windows. Результат виходить практично той же, але за рахунок нестандартного додаткового програмного забезпечення на віддаленому комп'ютері.

Типи використовуваних глобальних служб

Схема організації віддаленого доступу багато в чому визначається тими глобальними транспортними службами, які доступні в точках знаходження численних клієнтів віддаленого доступу. Крім ступеня поширеності необхідно враховувати й вартість глобальної служби. З обліком цих двох обставин найбільше часто для організації віддаленого доступу використовується служба телефонних мереж – аналогових (Plain Old Telephone Service – POTS) і, якщо це можливо, ISDN.

Служба виділених каналів економічно виправдана тільки при підключенні невеликого числа великих підрозділів підприємства, а для окремих користувачів її використання – занадто більша розкіш.

Служба мереж з комутацією пакетів, таких як X.25 або frame relay, через свою вартість також малопригодна для індивідуальних користувачів. Крім того, точки доступу до цих мереж далеко не так поширені, як точки доступу до телефонної мережі, наявні майже в кожній квартирі, не говорячи вже про невеликі офіси. Прямі підключення до мереж X.25 або frame relay доцільні для організації рівноправних зв'язків мереж або ж для підключення мереж класу

ROBO, тому що такого роду мереж у підприємства звичайно небагато, а зв'язок із центральною мережею їм потрібна постійно. Для індивідуальних користувачів проблема підключення до мережі X.25 вирішується доступом по телефонній мережі через пристрій PAD, оснащене модемним пулом, якщо такий доступ виправданий економічно.

1.2 Область застосування

У загальному область застосування наведена у вступі. У цьому пункті наведемо деякі практичні приклади. З ростом популярності високошвидкісних каналів в Інтернет стає актуальним питання про віддалене керування домашнім комп'ютером, перебуваючи на роботі, у гостях або з будь-якого іншого місця, де є вихід у глобальну мережу. Наприклад, можливо, подзвонивши на телефон і включивши комп'ютер, зайти віддалено на нього за допомогою програми, яка буде розроблена у результаті виконання магістерського проекту, (для цього серверна частина програми повинна бути прописана в автозавантаженні Windows), далі, запустивши FTP-сервер або убудовані засоби перекачування файлів, скачати необхідну нам інформацію з комп'ютера.

Також можливо продіагностувати його стан. Скажемо, є блок безперебійного живлення (UPS), можна контролювати його стан віддалено. Для цього, зрозуміло, він повинен бути з'єднаний з комп'ютером спеціальним кабелем через COM-порт або через шину USB. Програму моніторингу й налаштування для UPS можно знайти в Інтернеті на сайті виробника, або скачати утиліту сторонніх розроблювачів. Точно так само можна моніторить температури, швидкість обертання кулерів, напруги, вільне місце на диску й т.д.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Розглянемо існуючі програмні продукти які дозволяють здійснювати віддалене керування комп'ютером через мережу, та проаналізуємі їхні характеристики.

Radmin

Ви бачите екран віддаленого комп'ютера на своєму моніторі в окремому вікні або в повноекраному режимі. Ваші маніпуляції мишею або клавіатурою передаються на віддалений комп'ютер. Таким чином, Ви можете працювати за ним так, начебто він перебуває прямо перед Вами. Ви можете віддалено управляти одним комп'ютером з декількох точок доступу й використовувати обмін файлами, текстовий і голосовий чати, віддалене вимикання, режим Telnet і інші корисні функції.

Можливості Radmin

– Висока швидкість роботи. Radmin працює швидше інших програм віддаленого керування. Новітня технологія DirectScreenTransfer™ використовує драйвер відео-перехоплення, щоб прискорити частоту передачі зображення екрана до декількох сотень відновлень у секунду. Спеціальна оптимізація для каналів з низькою пропускною здатністю дозволяє Вам з комфортом працювати навіть по dial-up модему й через GPRS з'єднання.

– Високий рівень безпеки. Radmin працює в режимі захисту даних, при якому всі передані дані, зображення екрана, переміщення курсору й сигнали клавіатури надійно захищені за стандартом AES. Секретний ключ генерується випадковим образом для кожного підключення. Для автентифікації користувачів

в Radmin може бути використана або система безпеки Windows з підтримкою активних директорій (Active Directory) і протоколу Kerberos, або власна система безпеки Radmin з індивідуальними правами доступу для кожного користувача й захищеною автентифікацією по логину й паролю. Додатково таблиці IP-фільтрації дозволяють дозволити доступ тільки для певних хостів і підмереж.

– Віддалене керування на апаратному рівні з підтримкою технології Intel® AMT NEW. AMT (Active Management Technology) є розробкою корпорації Intel і являє собою технологію віддаленого керування, інтегрувальну в апаратну частину ПК на основі платформи vPro™. Завдяки підтримці даної технології нова версія Radmin дозволяє включати, виключати й перезавантажувати віддалений комп'ютер. За допомогою програми користувачі можуть одержати доступ до текстового режиму й BIOS віддаленого ПК. Крім того, у новій версії Radmin реалізована можливість завантаження віддаленого комп'ютера з локального CD/Floppy або файлу образа диска.

– Повна сумісність із Windows 10/11. Radmin повністю підтримує Windows 10/11 32-bit і 64-bit, включаючи керування користувальницькими аккаунтами й швидке перемикання користувачів. Radmin Server підтримує операційні системи Windows 10/11. Radmin Viewer підтримує операційні системи Windows 10/11.

– Текстовий і голосовий чати. Текстовий чат, Голосовий чат і функція Відправлення одиночного повідомлення – це можливості, розроблені для версії Radmin, які дозволять Вам спілкуватися з людиною, що перебуває за віддаленим комп'ютером.

– Простота у використанні. Добре продуманий інтерфейс є одним з головних переваг програми. Відсутність функцій, які в 99,9% випадків ніколи не знадобляться, робить Radmin легким в навчанні навіть для користувачів, які абсолютно не знайомі з мережевими технологіями.

– Безпечний "Drag and Drop" обмін файлами з функцією "докачки". Radmin дозволяє копіювати файли з одного комп'ютера на інший. Інтерфейс

режиму обміну файлами аналогічний інтерфейсу Провідника Windows, знайомому всім користувачам, що істотно спрощує освоєння програми. Реалізована "докачка" файлів: у випадку збоїти мережі можна продовжити передачу файлу з моменту збоїти, а не із самого початку.

– Підтримка декількох одночасних з'єднань. Radmin підтримує кілька одночасних підключень до екрана віддаленого ПК. Таким чином, Ви можете дозволити друзям або колегам спостерігати Ваш екран віддалено (дуже зручно для проведення конференцій) або Ви можете переглядати кілька віддалених екранів або управляти декількома комп'ютерами зі свого ПК (дуже зручно для віддаленої технічної підтримки або навчання).

– Підтримка перемикання сесій користувачів в Windows 10/11.

– Вибір режиму передачі екрана: 2, 4, 16, 256, 65 тисяч або 16 мільйонів квітів.

– Повна підтримка відображення курсору віддаленого ПК: його форми, анімації й прозорості.

– Підтримка прокручування за допомогою колеса миші.

– Підтримка спеціальних клавіш і сполучень клавіш.

– Підтримка високих дозволів (обмеження на максимальний дозвіл дисплея відсутній).

– Можливість відображення екрана віддаленого ПК в окремому вікні або в повноекраному режимі із плавною зміною масштабу й збереженням пропорцій.

– Підтримка декількох моніторів.

– Radmin Viewer тепер сполучимо з Wine (віддалений доступ з машин, де встановлена ОС Linux).

– Двостороння робота з буфером обміну з підтримкою Unicode.

– Адресна книга без обмежень на кількість записів, деревоподібною структурою, drag-and-drop для записів і папок.

– Підключення до віддаленого ПК із адресної книги в один клич.

– Вбудований сканер серверів Radmin.

– Вбудована довідкова система.

– Режим Telnet.

– Віддалене вимикання комп'ютера.

– Запуск Radmin Server винятково як системної служби.

– Сумісність із попередніми версіями Radmin Server 2.x.

– Захист від угадування пароля із затримкою після п'яти послідовних невдалих спроб.

– Запис у лог файл ім'я користувача й DNS розшифровки його адреси.

Radmin складається із двох модулів:

– Серверний модуль (Radmin Server).

– Клієнтський модуль (Radmin Viewer).

Розглянемо їхню роботу детальніше.

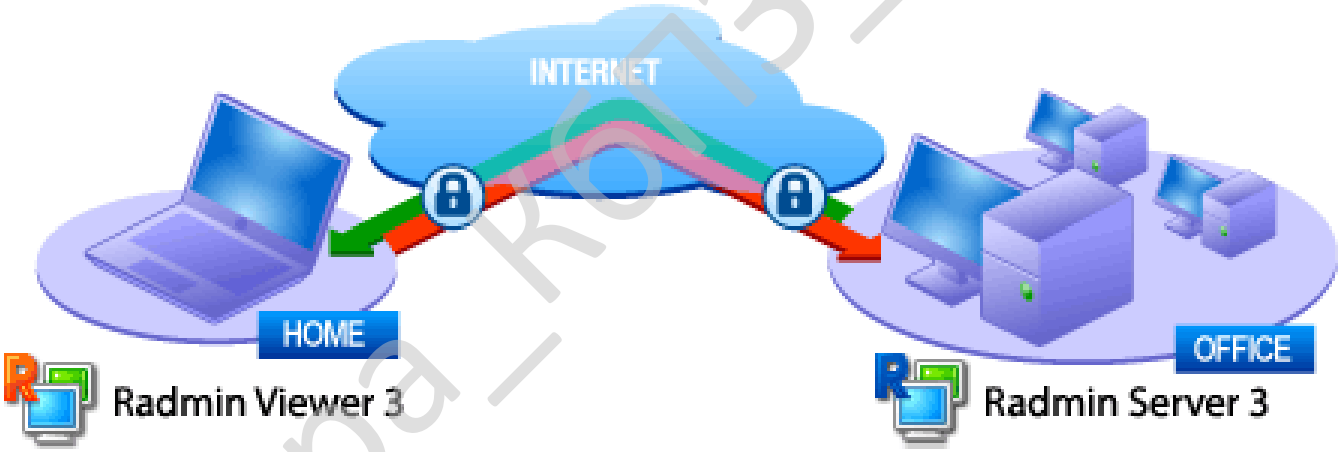

Рисунок 2.1 – Структурна схема роботи Radmin

Як використовувати Radmin

Насамперед необхідно запустити Radmin Server на віддаленому комп'ютері. Потім запустити Radmin Viewer на локальному комп'ютері. Обидва комп'ютери повинні мати вихід в Інтернет або бути приєднаними до загальної локальної мережі (LAN).

Треба установити Radmin Server на віддаленому ПК. Необхідно запустити файл rserv33ru.exe і додержуватися інструкцій на екрані.

Треба установити Radmin Viewer на локальному ПК. Необхідно запустити файл rview33ru.exe і додержуватися інструкцій на екрані.

Настройте Radmin Server на віддаленому ПК. Клацніть правою кнопкою миші на іконці Radmin Server у треї й виберіть "Налаштування Radmin Server". Натисніть кнопку "Права доступу...", щоб установити пароль для доступу до Radmin Server. Вам необхідно знати IP-адресу віддаленого ПК. Щоб відобразити його, наведіть курсор миші на іконку Radmin Server у треї.

Запустите Radmin Viewer на локальному ПК. У меню Пуск клацніть "Radmin Viewer", створіть нове з'єднання й уведіть IP-адресу віддаленого ПК. Потім виберіть тип з'єднання й натисніть "З'єднатися". Уведіть пароль, що Ви встановили на віддаленому ПК і почніть віддалену роботу!

Radmin Server

Серверний модуль Radmin повинен бути встановлений на тих комп'ютерах, до яких Ви збираєтеся підключитися. Він завжди запускається як сервіс і автоматично завантажується при запуску Windows.

Radmin відомий своїми стандартами безпеки. При спробі підключення до віддаленого комп'ютера, Radmin Server робить безпечну автентифікацію користувача за допомогою пароля. Програма перевіряє наявність у користувачів прав доступу й налаштування IP-фільтрації. Всі дані при передачі по мережі надійно захищені по стандарту AES.

Серверний модуль Radmin забезпечує повний ефект присутності за екраном віддаленого комп'ютера. Radmin містить у собі нову технологію DirectScreenTransfer™ на базі Radmin Mirror driver. Дана розробка дозволяє Radmin Server зчитувати дані з екрана віддаленого комп'ютера в обхід відеоконтролерів, знижуючи тим самим навантаження на процесор. Radmin використовує інтелектуальний алгоритм, що дозволяє зчитувати відновлення даних. Дана технологія не тільки забезпечує можливість максимального

збільшення швидкості відновлення екрана, але й знижує витрата мережевого трафіку. Це особливо важливо при роботі з повільним Інтернет-з'єднанням або з'єднанням через модем.

Дана частина Radmin працює також як сервер для підтримки можливостей текстового й голосового чату. У системі безпеки використовуються ті ж алгоритми безпеки й автентифікації, що й у режимі Full Control.

Radmin Server підтримує багато можливостей, серед яких точна передача форми й анімації курсору, коректна передача зображення з декількох моніторів, і багато чого іншого.

Можливості Radmin Server 3:

– Повна підтримка Windows 10/11.

– Нова технологія DirectScreenTransfer™ з використанням нового драйвера відео-перехоплення.

– Неперевершена швидкість роботи й низьке завантаження процесора.

– Підтримка перемикання сесій користувачів в Windows 10/11.

– Повна підтримка відображення курсору віддаленого ПК: його форми, анімації й прозорості.

– Підтримка декількох моніторів.

– Вбудований багатокористувальницький текстовий і голосовий чат.

– Підтримка миттєвого одержання коротких повідомлень від Radmin Viewer.

– Надійний захист всіх переданих і одержуваних даних.

– Система безпеки Windows з підтримкою активних директорій (Active Directory) і протоколу Kerberos.

– Захист від угадування пароля із введенням тимчасових затримок при підозрі на перебір пароля.

– Запис у лог файл ім'я користувача й DNS розшифровки його адреси.

– Новий оптимізований мережевий протокол.

– Кількість підключень до однієї копії програми Radmin Server 3.x у будь-якому режимі (Керування, Перегляд, Чат, Telnet і 'З'єднання через') обмежено 5 одночасними підключеннями. Це число можна збільшити придбанням ліцензії на додаткові підключення

Radmin Viewer

На додаток до своєї основної функції передачі зображення на інший комп'ютер, Radmin надає адміністраторам і постійним користувачам можливість виконувати й інші задачі.

Radmin істотно спрощує процес копіювання файлів з одного комп'ютера на інший. Використовуючи режим передачі файлів Radmin, Ви можете копіювати файли з/на віддалений комп'ютер, що не ставиться до Вашої мережі. Radmin дозволяє Вам копіювати файли прямо.

Іншою корисною особливістю Radmin є можливість підключення до віддаленого комп'ютера в режимі Telnet. Це дозволить вам здійснювати перенос текстових команд на віддалений комп'ютер за допомогою командного рядка у вигляді вхідного й вихідного потоку. Дана можливість дозволить вам працювати на віддаленому комп'ютері, не заважаючи працюючому за ним користувачеві – це практично термінальний доступ, тільки обмежений режимом командного рядка. Позитивною стороною цього методу є економія й зменшення витрати трафіку в тисячі разів у порівнянні із графічним режимом.

Radmin Viewer пропонує оновлений інтерфейс у стилі 10/11 і множину корисних можливостей:

– Підтримка функцій Intel® AMT (Active Management Technology).
– Повна сумісність із Windows 10/11.
– Зручний інтерфейс і вбудована довідкова система.
– Вбудований сканер серверів Radmin.
– Збереження адресної книги у файлі.
– Підтримка папок в адресній книзі.
– Підтримка експорту й імпорту адресної книги.

– Підтримка деревоподібної організації папок.

– Drag-and-drop для записів і папок.

– Можливість створювати на робочому столі ярлики елементів адресної книги.

– Можливість задавати значення за замовчуванням для налаштування нових з'єднань.

– Загальні опції для типів з'єднань «віддалений екран», «передача файлів», «текстовий чат», «голосовий чат» у головному вікні.

– Мінімізація головного вікна «у трей».

Нові можливості, доступні в Radmin при роботі з екраном віддаленого комп'ютера (режим керування):

– Технологія DirectScreenTransfer™, що забезпечує максимальну швидкість.

– Вибір режиму передачі екрана: 2, 4, 16, 256, 65 тисяч або 16 мільйонів кольорів.

– Обмеження на максимальний дозвіл дисплея відсутній.

– Повна підтримка декількох моніторів як на віддаленому, так і на локальному ПК.

– Повноекранний, масштабований і нормальний перегляд екрана віддаленого комп'ютера із плавною зміною масштабу й збереженням пропорцій.

– Повна підтримка відображення курсору віддаленого ПК: його форми, анімації й прозорості.

– Підтримка прокручування за допомогою колеса миші.

– Двустороня робота з буфером обміну з підтримкою Unicode.

– Сполучення клавіш, що набудовуються, під команди Radmin and optional Full keyboard control.

– Підтримка спеціальних клавіш і сполучень клавіш (Alt-Tab, клавіша Windows, і т.д.).

– Швидкий запуск інших типів з'єднань (передача файлів, текстовий чат, голосовий чат, telnet, і т.д.) з вікна віддаленого комп'ютера без повторного уведення пароля.

Hidden Administrator

Hidden Administrator, призначений для керування (адміністрування) і спостереження за віддаленими комп'ютерами під керуванням Windows 98/ME/NT/2000/XP/2003/10/11.

Програма проста як в освоєнні, так і у використанні, при цьому програма має зручний і інтуїтивно зрозумілий інтерфейс, так що розібратися з її роботою буде під силу навіть непідготовленому користувачеві.

Можливості програми Hidden Administrator

– Одержання повного доступу до ресурсів віддаленого комп'ютера.

– Сховане спостереження за віддаленими комп'ютерами.

– Керування віддаленими комп'ютерами.

– Одночасне спостереження за декількома комп'ютерами (до 256 комп'ютерів одночасно).

– Запис зображення з віддаленого екрана у відеофайл формату AVI.

– Моніторинг запущених програм і відвіданих сайтів.

– Одержання інформації з повноекраного DOS вікна.

– Обмін файлами з віддаленим комп'ютером.

– Передача звуку по мережі в режимі реального часу.

– Віддалена установка серверної частини програми.

– Посилка повідомлень на віддалений комп'ютер.

– Обмін повідомленнями з віддаленим комп'ютером (чат).

– Вимикання й перезавантаження віддаленого комп'ютера.

– Одержання інформації про систему.

– Робота з реєстром віддаленого комп'ютера.

– Одержання й передача буфера обміну.

– Завершення будь-якого додатка на віддаленому комп'ютері.

– Автоматичне завершення додатків і процесів.

– Запуск програм на віддаленому комп'ютері.

– Приховання/відображення Панелі задач, іконок робочого стола, курсору миші, мінімізація/приховання/відновлення вікон і т.п.

– Автоматичний пошук запущених серверів (пошук комп'ютерів для підключення).

– Захист паролем з'єднання з віддаленим комп'ютером.

– Обмеження прав доступу до віддаленого комп'ютера.

– Надання загального доступу до віддалених папок.

– Віддалене включення комп'ютерів (Wake on LAN).

– Установка пароля на зміну налаштувань сервера.

– Збереження скриншотів віддаленого екрана.

– Віддалена печатка документів.

– Фільтрація IP адрес.

– Шифрування даних.

Anyplace Control

Anyplace Control – програма, що дозволяє в цілковитій безпеці управляти віддаленим комп'ютером через Інтернет або ЛОМ.

Anyplace Control відображає робочий стіл віддаленого комп'ютера, і дозволяє управляти їм на вашім локальному ПК за допомогою миші й клавіатури. Інакше кажучи, незалежно від того, де Ви перебуваєте, Ви можете управляти віддаленим комп'ютером також, як якби Ви сиділи прямо перед ним. Вбудована функція Передача файлів дозволяє Вам здійснювати обмін файлами між комп'ютерами. За допомогою програми Anyplace Control Ви можете віддалено виключати комп'ютер, перезавантажувати, блокувати мишу й клавіатуру на віддаленому комп'ютері, а також здійснювати багато інших функцій. Крім того, програма оснащена унікальною можливістю Віддаленої установки, що дозволяє Вам установлювати й конфігурувати нашу програму віддалено. При цьому не

потрібно Ваша безпосередня присутність перед комп'ютерами, на які Ви проводите установку віддалено.

Програма дозволяє:

– Бачити робочий стіл віддаленого комп'ютера на вашім екрані в реальному часі.

– Використовувати мишу й клавіатуру для керування віддаленим ПК.

– Передавати файли з або на віддалений комп'ютер.

– Копіювати текст, графіку або інші дані з буфера обміну одного комп'ютера й вставляти їх на іншій.

– Включати, виключати, перезавантажувати віддалений ПК, а також блокувати його мишку й клавіатуру.

– Одержувати доступ до ПК, що перебуває за маршрутизатором або брендмауером без яких-небудь налаштувань.

– Системні вимоги: Windows/95/98/NT/2000/XP/2003/10/11

Як працює програма:

Вам знадобитися як мінімум 2 комп'ютери, підключених друг до друга у ЛОМ або через Інтернет, ну й звичайно програма Anyplace Control, установлена на обох ПК. З'єднання між цими комп'ютерами може бути встановлено двома способами:

Пряме підключення: Доступ до віддаленого ПК через IP адресу або DNS ім'я.

Account підключення: Доступ до віддаленого комп'ютера через Інтернет, використовуючи обліковий запис або ім'я комп'ютера. (IP адреса не використовується). Цей режим дозволяє обходити брендмауери, маршрутизатори й інші перешкоди.

Після того, як з'єднання встановлене, Ви побачите робочий стіл віддаленого комп'ютера, зможете управляти їм за допомогою своєї миші й клавіатури, а також використовувати функцію Обміну файлами між вашим ПК і віддаленим.

Програма складається із двох модулів:

– Host Module – установлюється на комп'ютер, що Ви хочете контролювати.

– Admin Module – установлюється на комп'ютер, з якого Ви будете контролювати віддалений ПК.

Інсталяційний пакет містить обидва модулі. Ви можете вибрати необхідний модуль під час установки. При бажанні, Ви також можете встановити обидва модулі на один комп'ютер. Щоб установити програму, необхідно притримуватись наступних інструкцій:

– Присядьте за комп'ютер, що буде контролюватися віддалено (Host PC). Скачайте інсталяційний пакет з нашого сайту на цей комп'ютер і встановіть на нього Host Module.

– Поверніться до свого комп'ютера (Admin PC). Скачайте інсталяційний пакет і встановите на свій комп'ютер Admin Module.

– Якщо Host комп'ютер перебуває в тій же локальній мережі, що й Admin комп'ютер, тоді Вам не потрібно підходити до одному з них безпосередньо щоб установити Host Module, тому що Ви можете встановити його віддалено, використовуючи функцію "Віддалена інсталяція ". Ця функція вбудована в Admin Module, і Ви можете запустити її в закладці "Інструменти".

TeamViewer

Схоже, але трохи інше рішення пропонує www.teamviewer.com. Якщо при використанні VNC для доступу до віддаленого комп'ютера на ньому повинна бути встановлена серверна частина VNC, то у випадку з TeamViewer – всі навпаки. На вашім комп'ютері ви ставите TeamViewer.

На клієнтському комп'ютері скачується невелика утиліта , що повинна бути запущена. Після цього вам повідомляється (краще по телефоні або e-mail) ідентифікаційний номер клієнта й пароль для доступу, що ви й вводите. З'єднання встановлюється наскрізне – міжмережеві екрани, різні підмережі, як правило, не перешкода.

Усе, що вам треба знати для керування віддаленим комп'ютером – це ідентифікаційний номер (ID) і пароль (Kennwort) в TeamViewer-Віддаленого комп'ютера.

У програмі досить багато корисних опцій – різні режими відображення й керування, перезапуск комп'ютера, імітація сполучення клавіш Ctr+Alt+Del, можливість перекачування файлів з одного комп'ютера на інший, начебто ви працюєте під Norton Comander, VPN, чат, можливість презентації – тобто показ вашого робочого стола й вироблених дій на екрані клієнтського комп'ютера.

Порівняння різних програм віддаленого керування

Які вимоги ви пред'являєте програмі для віддаленого керування? Головне, мабуть, ця зручність роботи! Для комфортної віддаленої роботи має значення якість переданого зображення, відсутність перекручувань. Також важлива швидкість роботи й синхронізація, що дозволяє працювати в реальному часі.

При виборі програми для віддалених керувань варто звернути увагу на такі параметри як швидкість роботи і якість зображення (по десятибальній шкалі, максимальний бал – 10).

Таблиця 2.1 – Результати порівняння програм віддаленого керування

Назва програми	Швидкість роботи	Якість зображення	Загальний бал
Radmin	10	10	10
Remote Assistance	8	8	8
Symantec pcAnywhere	6	4	5
DameWare	5	5	5
RealVNC	2	6	4
TeamViewer	1	3	2

Методика проведення тестування

Компанія Фаматек провела ряд тестів у порівнянні Radmin з іншими програмами для віддаленого керування.

Серверний модуль був установлений на один комп'ютер (комп'ютер А), а клієнтський модуль – на другий комп'ютер (комп'ютер В). Комп'ютери були з'єднані в ізольовану локальну мережу щоб уникнути можливого впливу зовнішніх факторів. Програми були настроєні таким чином, що на дисплеї комп'ютера В відображався екран віддаленого комп'ютера А. На комп'ютері В була запущена програма BB FlashBack для запису зображення в реальному часі. На віддаленому комп'ютері А виконувалися дії, задані командами в скрипті.

Для тестування були обрані наступні дії: відкриття вікна Display Properties, передпрогляд заставки, перегляд вмісту папки Wallpapers за допомогою Windows Picture Viewer.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

– У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion,

навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи віддаленого доступу з використанням WAN-мереж.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які

забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Опис загальної технології віддаленого керування комп'ютером через Інтернет

Апаратно-програмні вимоги

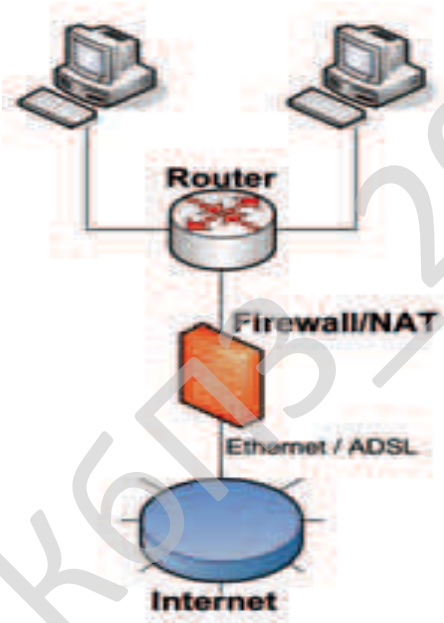

Рисунок 3.1 – Загальна схема віддаленого керування ПК

Отже, що нам буде потрібно, щоб одержати можливість віддаленого керування нашим домашнім комп'ютером:

– По-перше, це, звичайно ж, доступ домашнього комп'ютера в Інтернет, і необхідно, щоб інтернет-провайдер виділив нам пряму (зовнішню) IP-адресу.

– По-друге, необхідно встановити на комп'ютер спеціальне програмне забезпечення для віддаленого адміністрування.

– По-третє, комп'ютер повинен бути включений, із завантаженою операційною системою й всім комплексом програмного забезпечення, необхідного нам для віддаленого керування. Тримати ПК постійно включеним

незручно, але проблема розв'язувана. Якщо залишився старий аналоговий dial-up модем, і BIOS материнської плати підтримує технологію Wake-on-Ring, то комп'ютер може залишатися виключеним. Включеним залишиться тільки модем, що при першому ж вхідному дзвінку «розбудить» комп'ютер, і до нього стане можливим звернутися через Інтернет. Головним мінусом даної технології є саме те, що модем спрацює на будь-який вхідний дзвінок, і, відповідно, це може привести до помилкового включення комп'ютера, але адже його потім можна знову відключити.

Детальніше варто зупинитися на підключенні до Інтернету. Найважливішу роль отут грає IP-адреса. Так, у випадку якщо домашньому комп'ютеру привласнюється пряма IP-адреса, ніяких труднощів виникнути не повинно, а от у випадку підключення через шлюз, і, відповідно, з «сірим» IP усе буде трохи складніше. Якщо підключатися через районну будинкову мережу, то прийдеться домовлятися із провайдером про надання прямої IP-адреси, а от якщо шлюз перебуває в будинку (наприклад, при використанні ADSL-модему в режимі роутера), треба просто забезпечити наскрізне проходження пакетів, адресованих нашому комп'ютеру. Як це зробити? Для цього існує технологія NAT (Network Address Translation), що транслює запити із внутрішніх IP-адрес у зовнішню мережу, і навпаки. За замовчуванням NAT транслює тільки запити із внутрішньої мережі в зовнішню, а запити, що приходять із зовнішнього миру скидаються, просто тому що подальший маршрут проходження NAT'у невідомий. Для того щоб NAT перенаправляв запити на нашу машину, потрібно жорстко закріпити зовнішні TCP/UDP-порти шлюзу за певним комп'ютером. Для цього NAT'у варто вказати, запити для яких портів необхідно відправляти на адресу нашого ПК.

Підготовка комп'ютера

Отже, необхідно почати підготовку комп'ютера до віддаленого керування. Розглянемо випадок, коли вихід в Інтернет здійснюється через роутер (наприклад, нині популярні роутери CISCO 2801). Тоді схема підключення виходить приблизно така:

Wake-on-Ring

Для початку настроїмо опцію Wake-on-Ring. Якщо в тебе внутрішній модем, то необхідно з'єднати його зі спеціальним рознімання на материнській платі. Із зовнішнім модемом нічого додаткового не потрібно. Заходимо в BIOS материнської плати, у розділ з налаштуваннями живлення, знаходимо щось подібне Resume on Ring або Wake on Ring, і активуємо цю опцію. Тепер після вимикання комп'ютера (у випадку із зовнішнім модемом не забудь залишити його включеним) модем включить комп'ютер при першому ж дзвінку.

Невелике зауваження для тих, у кого будинку АВН. Принцип його функціонування такий, що для визначення номера він знімає трубку (ініціалізує з'єднання з АТМ) і далі дзвінки на телефони/модеми/факси, підключені паралельно АВНу, уже не проходять. Таким чином, у ланцюзі підключення АВН повинен бути першим пристроєм, а всі інші – включатися за ним.

Якщо ти хочеш, щоб комп'ютер автоматично відключався після помилкового дзвінка, необхідно встановити одну із програм, спеціально призначених для цих цілей.

У налаштуваннях програми обов'язково ставимо галочку «автоматично завантажуватися після запуску Windows», створюємо задачу «вимикання живлення», за умови відсутності активності курсору (рухів мишки) протягом десяти мінут (думаю, цього часу сповна вистачить, щоб вийти в Інтернет, зайти RAdmin'ом на комп'ютер і деактивувати цю утиліту) і підтверджуємо додавання задачі. Тепер комп'ютер при надходженні дзвінка ввімкнеться, завантажить Windows, після чого десять мінут буде покірно чекати твоїх вказівок, а після закінчення цього строку знову відключиться до наступного дзвінка.

Wake-on-LAN

Хтось може запитати, а чому б не використовувати для цих цілей технологію включення комп'ютера за допомогою локальної мережі Wake-on-LAN? Вся справа в принципі роботи цієї технології. Wake-on-LAN (а вірніше, її найпоширеніший різновид, Magic Packet) у чомусь схожа на Wake-on-Ring. Тут

ключову роль виконує мережева карта, що при включенні функції Wake-on-LAN, продовжує працювати навіть після вимикання комп'ютера, і очікує спеціальний кадр, що будить. Інформація, що перебуває в цьому кадрі, являє собою шість байт синхронізації й шістнадцять разів повторену MAC-адресу мережевої карти-приймача. Послідовність упаковується в UDP, потім у пакет IP із широкомовною адресою, у кадр Ethernet, і адресується приймачу. Як адреса призначення використовується MAC-адреса Ethernet-адаптера, тобто адресація відбувається тільки на канальному рівні моделі OSI. Тому застосовувати технологію можна тільки в локальних мережах, не розділених на сегменти, або усередині одного сегмента. По цій же причині з'являються складності при посилці пакета з послідовністю, що будить, через мережу Internet.

Все це обмежує область застосування даної технології вузьким колом задач. Наприклад, при наявності в будинку декількох комп'ютерів, можно із одного з них включати інші. Для того щоб можна було скористатися цією функцією, її повинні підтримувати й мережева карта, і BIOS материнської плати. Налаштування так само просте, як і налаштування Wake-on-Ring. Заходимо в BIOS материнської плати в той же самий розділ з налаштуваннями живлення й знаходимо щось подібне Resume on LAN або Wake on LAN. Активуємо цю опцію. Якщо материнська плата має шину PCI специфікації до 2.2 (довідатися, яку специфікацію шини PCI підтримує твоя материнська плати, ти можеш із інструкції до неї), то на ній повинен бути трьохштирьковий роз’єм «Wake On Lan». Аналогічний роз’єм повинен бути на мережевому адаптері. Їх потрібно з'єднати спеціальним кабелем, що входить у комплект поставки мережевого адаптера. Для випадку із шиною PCI 2.2 таке з'єднання вже виконане прямо. Тепер залишається тільки виключити комп'ютер.

Тепер, щоб віддалено включити цей комп'ютер, нам потрібно по мережі послати кадр із послідовністю, що будить. Для цього існує кілька програм, таких як wol.exe або broadc.exe. Всі що нам потрібно знати для запуску програми – це

MAC-адреса мережевого адаптера віддаленого комп'ютера. Наприклад, для broadc.exe, що запускається з консолі, вхідний рядок буде така:

broadc.exe (MAC-адреса мережевої карти) 255.255.255.255 67

Допустимо, що MAC-адреса мережевого адаптера – 00:02:B3:D8:B4:E6, тоді рядок прийме вид:

broadc.exe 0002b3d8b4e6 255.255.255.255 67

Інші вхідні параметри змінювати не потрібно. 255.255.255.255 – це широкомовний IP-адреса, завдяки якому сформований кадр пройде через всю мережу, а 67 – номер порту протоколу UDP, у дейтаграмі якого й буде перебувати послідовність, що будить. Використання wol.exe і інших подібних програм повністю аналогічно.

Вибір програмного забезпечення

Наступним кроком є визначення необхідного нам програмного забезпечення, для здійснення функцій віддаленого керування й контролю за системою. Існує множина програм для віддаленого керування комп'ютером. Принципово можна розділити все це різноманіття на дві групи, що розрізняються по способу керування. Управляти можна через командний рядок/консоль (Telnet, SSH) або за допомогою графічного подання робочого стола віддаленої операційної системи (Remote Desktop, Remote Administrator). Найбільш зручним і зрозумілим для кінцевого користувача, звичайно, представляється другий спосіб, а найпоширенішою й відомою програмою для такого доступу є Remote Administrator. У якості ftp-сервера можу порекомендувати Bullet Proof або Serv-U – вони досить прості й гнучкі в налаштуванні.

Налаштування NAT

Тепер, перейдемо до підготовки нашого роутера. Отут прийде затурбуватися налаштуванням двох речей: це NAT і вбудований пакетний фільтр (або, як його частіше називають, брандмауер або firewall). І те, і інше зручніше набудовувати за допомогою web-інтерфейсу.

Для початку створимо запис в таблицю статичної NAT-адресації. Залежно від виробника конкретної моделі роутера, ця опція може позначатися по-різному. У роутерах D-link це називається «Virtual Server». У кожному разі настроюється сам NAT скрізь однаково. Задається зовнішній порт роутера, на який приходить запит, IP-адреса й порт, на який роутер повинен перенаправляти цей запит. Звичайно номер порту задається в обох випадках однаковий. Усе, що нам треба знати, це те, які порти використовує необхідне нам програмне забезпечення. Так, стандартний порт для Remote Administrator – 4899. З метою безпеки й зменшення ймовірності несанкціонованого доступу, можна замінити в налаштуваннях серверної частини RAdmin'a стандартний порт на будь-який інший. Для FTP-сервера стандартний порт – 21. Рекомендую також застосовувати нестандартний порт, наприклад 2121. Обумовлено, це тим, що деякі провайдери фільтрують запити, які поступають ззовні, адресовані на стандартний для FTP порт.

Окремо варто поговорити про правильне налаштування FTP. Існують два режими роботи FTP-сервера: пасивний і активний. Для коректної роботи за NAT'ом, потрібно настроїти сервер на пасивний режим роботи. У такому режимі клієнт, з'єднуючись із сервером, одержує від нього список портів, по яких надалі він повинен ініціювати з'єднання для передачі файлів. У налаштуваннях самої програми FTP-сервера необхідно вказати зовнішній IP-адресу, на який будуть надсилати запити клієнти, і діапазон портів, по яких їм варто встановлювати з'єднання для передачі даних (наприклад, 47990-48000). Їх же варто вказати в таблиці статичної NAT-адресації.

Налаштування файрвола

Але настроїти на роутері NAT – недостатньо для повноцінного функціонування цих сервісів. Необхідно ще зконфігурувати пакетний фільтр, реалізований в ADSL-роутері. Основними правилами завжди повинні бути: пропускати всі запити із внутрішнього інтерфейсу на зовнішній, і скидати всі запити із зовнішнього на внутрішній. Таким чином, ми позбуваємося від зайвого трафіку ззовні, також охороняючи слабкі місця операційної системи від

«промацування». Залишається лише додати правила для пропущення запитів, адресованих FTP-серверу й Remote Administrator'у. Знову таки, налаштування пакетного фільтра схожі в різних виробників роутеров, і завжди містить у собі:

– завдання протоколу передачі інформації (звичайно вибір лежить між TCP, UDP і ICMP; необхідно вказати той або інший набір протоколів, які використовує додаток для передачі свого трафіку);

– завдання маршруту проходження (з якого інтерфейсу/порту на який іде трафік);

– завдання IP-адреси або діапазону IP-адрес відправника (потрібно, тільки якщо ми дозволяємо одержувати ці запити тільки з яких те конкретних IP-адрес);

– завдання номера порту або діапазону портів відправника (для вхідних запитів практичного застосування фактично ні, і в деяких роутерах цей пункт цілком логічно відсутній);

– завдання IP-адреси або діапазону IP-адрес одержувача (потрібно, тільки у випадку наявності в нас декількох зовнішніх IP-адрес, для розмежування функціонального навантаження між ними);

– завдання номера порту або діапазону портів одержувача (цим правилом, ми дозволяємо певним сервісам приймати ззовні запити й обробляти їх);

– завдання розкладу дії правила (другорядний параметр, що рекомендується залишати в значенні за замовчуванням, звичайно – always);

– вибір дії із запитом: пропустити або скинути (вибираємо дія, що буде робити роутер, одержавши пакет, що підходить під ці параметри).

Так, для Remote Administrator'a необхідно дозволити вхідні запити по протоколі TCP, з будь-яких IP-адрес, на будь-які IP-адреси, на порт 4899. Для FTP-сервера варто дозволити вхідні запити по протоколі TCP, з будь-яких IP-адрес, на будь-які IP-адреси, на порт 21 (2121) і на діапазон портів 47990-48000.

Отже, що маємо в підсумку. Ми настроїли комп'ютер, що включається по нашому телефонному дзвінку й відключається через десять мінут, у випадку якщо це був помилковий дзвінок. Ми одержали можливість віддалено включати

цей комп'ютер з локальної або домашньої мережі, настроїти на роутері NAT і пакетний фільтр так, що до комп'ютера стало можливим звертатися через Інтернет. І тепер у нас з'явилася повноцінна можливість діагностувати й віддалено управляти повною мірою домашнім комп'ютером.

Опис архітектури клієнт-сервер та функцій API для цієї архітектури

Архітектура клієнт – сервер

Архітектура мережі визначає основні елементи мережі, характеризує її загальну логічну організацію, технічне забезпечення, програмне забезпечення, описує методи кодування. Архітектура також визначає принципи функціонування й інтерфейс користувача.

Архітектура клієнт – сервер (client-server architecture) – це концепція інформаційної мережі, у якій основна частина її ресурсів зосереджена в серверах, що обслуговують своїх клієнтів (рисунку 3.2). Розглянута архітектура визначає два типи компонентів: *сервери й клієнти*.

Сервер – це об'єкт, що надає *сервіс* іншим об'єктам мережі по їхніх запитах. *Сервіс* – це процес обслуговування клієнтів.

Сервер працює по завданнях клієнтів і управляє виконанням їхніх завдань. Після виконання кожного завдання сервер посилає отримані результати клієнтові, що послав це завдання.

Сервісна функція в архітектурі клієнт – сервер описується комплексом прикладних програм, відповідно до якого виконуються різноманітні прикладні процеси.

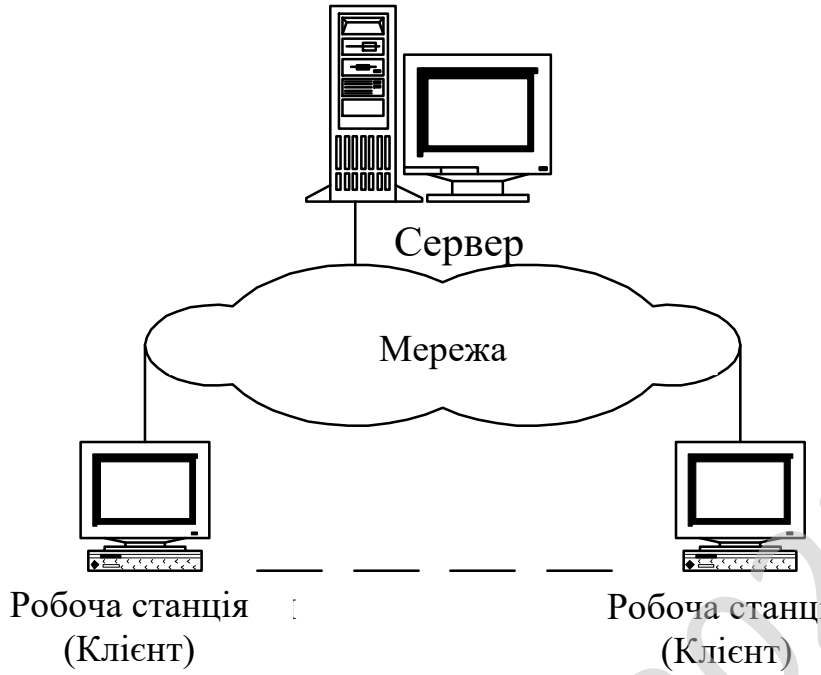

Рисунок 3.2 – Архітектура клієнт – сервер

Процес, що викликає сервісну функцію за допомогою певних операцій, називається *клієнтом*. Єю може бути програма або користувач. На рисунку наведений перелік сервісів в архітектурі клієнт – сервер.

Клієнти – це робочі станції, які використовують ресурси сервера й надають зручні *інтерфейси користувача*. *Інтерфейси користувача* це процедури взаємодії користувача із системою або мережею.

Клієнт є ініціатором і використовує електронну пошту або інші сервіси сервера. У цьому процесі клієнт запитує вид обслуговування, установлює сеанс, одержує потрібні йому результати й повідомляє про закінчення роботи.

У *мережах з виділеним файловим сервером* на виділеному автономному ПК установлюється серверна мережева операційна система. Цей ПК стає сервером. Програмне забезпечення (ПЗ), установлене на робочій станції, дозволяє їй обмінюватися даними із сервером.

Крім мережевої операційної системи необхідні мережеві прикладні програми, що реалізують переваги, надавані мережею.

Мережі на базі серверів мають кращі характеристики й підвищену надійність. Сервер володіє головними ресурсами мережі, до яких звертаються інші робочі станції.

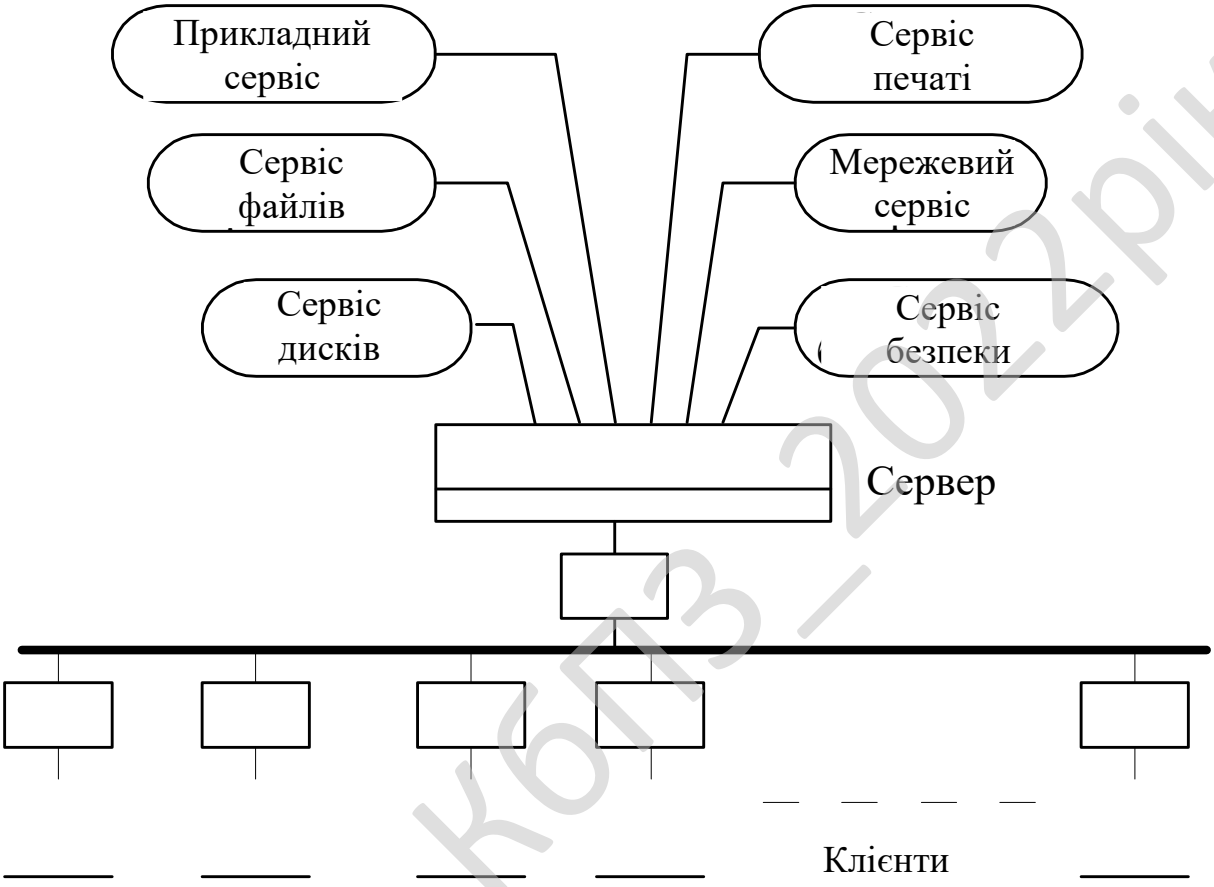

Рисунок 3.3 – Модель клієнт-сервер

У сучасної клієнт – серверній архітектурі виділяється чотири групи об'єктів: клієнти, сервери, дані й мережеві служби. Клієнти розташовуються в системах на робочих місцях користувачів. Дані в основному зберігаються в серверах. Мережеві служби є спільно використовуваними серверами й даними. Крім того служби управляють процедурами обробки даних.

Мережі клієнт – серверної архітектури мають наступні переваги:

– дозволяють організовувати мережі з великою кількістю робочих станцій;

– забезпечують централізоване керування обліковими записами користувачів, безпекою й доступом, що спрощує мережеве адміністрування;

– ефективний доступ до мережевих ресурсів;

– користувачеві потрібний один пароль для входу в мережу й для одержання доступу до всіх ресурсів, на які поширюються права користувача.

Поряд з перевагами мережі клієнт – серверної архітектури мають і ряд недоліків:

– несправність сервера може зробити мережа непрацездатною, як мінімум втрату мережевих ресурсів;

– вимагають кваліфікованого персоналу для адміністрування;

– мають більше високу вартість мереж і мережевого встаткування.

Application Programming Interface (API) – інтерфейс прикладного програмування. Набір стандартних програмних переривань, викликів процедур (функцій, методів) і форматів даних, які повинні використовувати прикладні програми для запиту й одержання від операційної системи, телекомунікаційного протоколу або програмного інтерфейсу (механізму) і т.п. пов'язаного з ними обслуговування. API визначає на рівні вихідного тексту деякий рівень абстракції, що дозволяє переносити вихідні тексти програм на комп'ютери із процесорами, що відрізняються, де вони після перекомпіляції зможуть відразу виконуватися.

Internet Server API фірми Microsoft

Коли застосовується інтерфейс Internet Server API (ISAPI) фірми Microsoft, то взаємодія між сервером і прикладною програмою організується через спеціальну структуру даних, іменовану ECB (Extension Control Block). У ній утримується інформація про те, як прикладній програмі варто обробляти поточний запит клієнта. Спочатку сервер завантажує прикладний модуль і передає йому ECB-блок. За допомогою функцій GetServerVariable і ReadClient модуль зчитує передані клієнтом вхідні дані. Обробивши їх, модуль звертається до функції WriteClient і відправляє дані обернено клієнтові. Після цього він викликає функцію ServerSupportFunction, щоб повідомити сервер про закінчення

обробки запиту. Інсталяція прикладних ISAPI-модулів виробляється за допомогою Internet Service Manager.

Існують два типи ISAPI-програм для нарощування можливостей сервера: фільтри й прикладні модулі. Фільтри виконують, задають або змінюють запити клієнта до початку їхньої обробки сервером. Звертання до фільтрів відбувається при надходженні будь-якого запиту від клієнта. Відомості про наявні DLL-фільтри втримуються в системному Реєстрі Windows, і сервер звертається до них при своїй ініціалізації. На відміну від фільтрів прикладні модулі повинні запитуватися самим клієнтом.

Крім ISAPI існує ще інтерфейс OLEISAPI, що дозволяє вашої допоміжної API-програмі взаємодіяти із серверами OLE Automation (автоматизація OLE). У результаті можна буде створювати програми в середовищі Visual Basic, що працюють із ISAPI. (Для розробки ISAPI-модулів необхідно використовувати Visual C++ або Delphi.) Специфікація OLEISAPI містить тільки частина ISAPI і офіційно ще не прийнята фірмою Microsoft.

Функції API для роботи з мережею в Delphi

Delphi існують вбудовані класи для роботи з мережею – це компонента Delphi 6 на закладці Internet (`TServerSocket` та `TclientCocket`) і компоненти `FastNet`, або компоненти `Indy` в Delphi 7. Але в цій статті розглядається програмування мережевих додатків на низькому рівні – з використанням WinSock API.

Навіщо це потрібно, якщо існує великий набір вбудованих компонентів і велика кількість безкоштовно розповсюджуваних класів?

1. У випадку, якщо для додатка критичний розмір (це найбільше ставиться до серверних додатків, які найчастіше не мають ніякої візуальної частини), те використання будь-яких VCL-компонентів украй небажано.

2. Програмування на більш низькому рівні традиційно вимагає більших витрат, але надає більш повний контроль за роботою додатка.

3. І, нарешті, знання основ роботи Windows Socket необхідно всім бажаючим розробляти мережеві додатки й буде гарною практикою для більше повного розуміння всіх процесів, що відбуваються.

Для підтримки мережевих додатків існує технологія, названа "сокети". Сокет – це модель одного кінця з'єднання, з усіма властивими йому властивостями й методами. По суті, це прикладний програмний інтерфейс, що входить до складу багатьох операційних систем (ОС) і покликаний для підтримки мережевих можливостей ОС. У стандарті структури протоколів семирівневої моделі OSI – сокети лежать на так званому транспортному рівні, нижче перебуває мережевий протокол IP, а вище – протоколи сеансового рівня, такі як FTP, POP3, SMTP і т.д.

В Windows підтримка сокетів включена починаючи з версії 3.11 і названа WinSock. Для написання додатків з мережевою підтримкою існує спеціальний WinSock API.

Сокети можуть базуватися на TCP/IP, IPX/SPX але надалі ми будемо говорити тільки про з'єднання TCP/IP.

Всі мережеві додатки побудовані на технології клієнт-сервер; це значить, що в мережі існує принаймні один додаток, що виявляє сервером, типова задача якого – це очікування запиту на підключення від додатків-клієнтів, яких може бути теоретично скільки завгодно, і виконання всіляких процедур у відповідь на запити клієнтів. Для клієнт-серверної технології абсолютно неважливо, де розташовані клієнт і сервер – на одній машині або на різні. Звичайно, для успішного з'єднання клієнта із сервером клієнтові необхідно мати мінімальний набір даних про розташування сервера – для мереж TCP/IP це IP-адреса комп'ютера, де розташований сервер, і адреса порту, на якому сервер очікує запити від клієнтів.

Кожний з комп'ютерів у мережі TCP/IP має свою унікальну IP-адресу, що використовується для обміну даними з іншими комп'ютерами. Кожний пакет, що посилається, від одного комп'ютера іншому має адреса відправника й

одержувача, що дозволяє його однозначно ідентифікувати. Однак у випадку, якщо на комп'ютері працює багато додатків, що одночасно використовують мережу, то такого набору атрибутів явно недостатньо.

Для дозволу неоднозначності крім адреси кожне з'єднання на кожному кінці має ідентифікатор за назвою "порт", цей ідентифікатор представляє число від 0 до 65535. Таким чином, пари адреса+порт являє собою сокет-канал, по якому два комп'ютери обмінюються даними один з одним. Тільки один додаток на одному комп'ютері в те саме час може використовувати конкретний порт, однак для серверних частин можливе створення декількох сокетів на одному порту для роботи з декількома клієнтами.

Значення порту не обов'язково повинне збігатися на сервері й клієнті – клієнтові для з'єднання важливо тільки знати порт сервера, порт клієнта може вибиратися клієнтом довільно й стає відомий серверу в момент запиту клієнта на з'єднання. Коли з'єднання буде встановлено, ОС створить для серверного додатка відповідний сокет, з яким і буде працювати додаток, так що порт клієнта для сервера зовсім не важливий.

Механізм роботи сокетів такий: на серверній стороні запускається серверний сокет, що після запуску відразу переходить у режим прослуховування (тобто очікування з'єднання клієнтів). На стороні клієнта створюється сокет, для якого вказується IP-адреса й порт сервера й дається команда на з'єднання. Коли сервер одержує запит на з'єднання, ОС створює новий екземпляр сокету, за допомогою якого сервер може обмінюватися даними із клієнтом. При цьому сокет, що створений для прослуховування, продовжує перебувати в режимі прийому з'єднань, у такий спосіб програмісти може створити сервер, що працює з декількома підключеннями від клієнтів.

Робота із сокетами, по суті, це операції вводу-виводу, які бувають синхронні й асинхронні. У термінології сокетів робота в асинхронному режимі називається що блокують сокетами, а в синхронному – що не блокують сокети. Спроба з'єднання або прийому даних у режимі, що блокує (відправлення завжди

синхронне, тому що фактично є постановкою в чергу) означає, що поки програма не з'єднається або не прикмет дані, передачі керування на наступний оператор не відбудеться.

Давайте тепер розглянемо мінімальний набір функцій з WinSock API, необхідних для написання елементарного клієнта й сервера. Самі функції перебувають у файлі winsock.dll. Файл winsock.pas містить необхідні оголошення імпортованих функцій WinSock API і базові структури даних. На жаль, цей файл імпортує не всі необхідні нам функції, і пізніше ми напишемо свій файл імпорту.

```
function WSAStartup(wVersionRequired: word; var WSData: TWSAData): Integer; stdcall;
```

Функція повідомляє ОС, що в будь-якому процесі додатка можуть бути використані функції WinSock. Функція повинна бути викликана один раз при запуску додатка перед використанням будь-якої функції WinSock.

```
function WSACleanup: Integer; stdcall;
```

Функція повідомляє ОС, що додаток більше не використовує WinSock. Повинна бути викликана перед завершенням додатка.

```
function socket(af, Struct, protocol: Integer): TSocket; stdcall;
```

Функція створює сокет. Порт і адреса задається у функції bind (сервер) або connect (клієнт). Вхідний параметр af – специфікація сімейства сокетів (AF_INET, AF_IPX і ін.), Struct – специфікація типу нового сокету (приймає значення SOCK_STREAM або SOCK_DGRAM), protocol – специфічний протокол, що буде використовуватися сокетом. Якщо функція виконана без помилок, вона повертає дескриптор на новий сокет, якщо помилки є, вертається INVALID_SOCKET.

```
function connect(s: TSocket; var name: TSockAddr; namelen: Integer): Integer; stdcall;
```

Функція з'єднання для клієнта. Структура адреси містить порт (необхідно привести функцією htons) і адресу (для клієнта необхідно привести з ім'я або специфікації ip4 – xxx.xxx.xxx.xxx).

```
function bind(s: TSocket; var addr: TSockAddr; namelen: Integer): Integer; stdcall;
```

Функція асоціює адресу із сокетом. Структура адреси містить порт (необхідно привести функцією htons) і адресу (для сервера звичайно вказується INADDR_ANY – кожної).

```
function send(s: TSocket; var Buf; len, flags: Integer): Integer; stdcall;
```

Функція відправлення даних. Поміщає в чергу сокету s шматок даних з buf, довжиною len. Останній параметр відповідає за вид передачі повідомлення. Може бути зігнорований (0).

```
function recv(s: TSocket; var Buf; len, flags: Integer): Integer; stdcall;
```

Функція одержання даних.

Отже, розглянемо приклади елементарного сервера й клієнта. Домовимося, що сервер буде працювати в асинхронному (блокируючому режимі). Єдина функціональність сервера – це одержання рядків даних від клієнта й висновок їх на екран. Зв'язок клієнта із сервером розривається після одержання рядка, що складається з єдиного символу 'q'. Для забезпечення можливості підключення до сервера множини клієнтів сервер на кожне з'єднання запускає окремий потік.

Вихідний код клієнта не представляє із себе нічого цікавого й не вимагає ніяких коментарів.

Отже, ми розглянули, як працюють сокети в асинхронному режимі, давайте подивимося тепер, які можливості WinSock нам надає для роботи із що не блокують сокетами.

Для того щоб перевести сокет у режим, що не блокує, використовується `функція ioctlsocket(...)`, що дозволяє контролювати режими робота сокету.

Т.я. тепер функція `recv` сервера буде повертати керування відразу, незалежно від наявності даних у буфері сокету, те тепер нам потрібний механізм, що дозволяє визначати які-небудь події, що відбуваються із сокетом. Для цього існує кілька механізмів. Перший, котрий ми розглянемо, це використання функції `select(...)`.

```
function select(nfds: Integer; readfds, writefds, exceptfds: PFDSet;
timeout: PTimeVal): Longint; stdcall;
```

Ця функція дозволяє контролювати стан набору сокетів.

Аргумент nfds ігнорується й залишений тільки для сумісності. Повинен бути дорівнює 0. readfs,writefds,exceptfds – покажчики на набори сокетів, для яких потрібно контролювати стан читання, відправлення даних і помилок відповідно. Набори зберігаються в структурі PFDSet, керування якої здійснюється спеціальними макросами, описаними в winsock.pas:

`procedure FS_ZERO(var FDSet: TFDSet)` – обнуляє структуру, установлює кількість контрольованих сокетів в 0;

`procedure FD_SET(Socket: TSocket; var FDSet: TFDSet)` – додає зазначений сокет у структуру;

`procedure FD_CLR(Socket: TSocket; var FDSet: TFDSet)` – видаляє зазначений сокет зі структури;

`function FD_ISSET(Socket: TSocket; var FDSet: TFDSet): Boolean` – повертає true, якщо зазначений сокет є членом зазначеної структури.

Аргумент timeout є посиланням на структуру типу PTimeVal, у якій можна вказати час очікування спрацьовування функції select. У випадку вказівки як значення часу затримки 0 або nil як аргумент timeout функція select буде чекати нескінченно, як при виконанні операції в режимі, що блокує.

Як видно з опису, функція може стежити відразу за декількома сокетами, таким чином, тепер ми можемо або також запускати окремий процес на кожний відкритий сокет, що буде стежити за конкретним сокетом, або створити один процес, що буде стежити за всіма відкритими сокетами. У першому випадку нам будуть потрібні лише незначні зміни процедури SocketThread, у другому випадку будуть потрібні досить значні зміни.

Давайте розглянемо логіку роботи в другому випадку, тому що перший випадок банальний.

Для початку в основній частині програми необхідно перевести знову створений сокет у режим, що не блокує:

```
arg := 1;
ioctlsocket(Socket,FIONBIO,arg);
```

Увага, перед закриттям сокету його необхідно буде повернути в режим, що блокує:

```
arg := 0;
ioctlsocket(Socket,FIONBIO,arg);
```

Потім необхідно реалізувати можливість збереження кожного сокету в деякий масив `SockArray`. Далі, потрібно забезпечити, щоб потік, що буде займатися обробкою клієнтів, запускався тільки один раз. Це зробити нескладно, знаючи, що процедура `CreateThread` повертає в змінне посилання на знову створений потік, що і потрібно використовувати для перевірки, чи був створений потік. У потоці обробки потрібно внести наступні зміни для використання функції `select`:

Нові використовувані змінні `wfds :TFDSet ; i : integer;`

```
tv : Ttimeval;
```

Потім визначимо основний цикл, у якому будемо обробляти дані:

```
Repeat
...
until connum>0;
```

У цьому циклі першою справою необхідно сформувати структуру wfds, що містить набір контрольованих сокетів. Для цього ми переносимо туди сокети з масиву `SockArray`:

```
FD_ZERO(wfds);
for i:=1 to connum do
begin
FD_SET(sock[i],wfds);
end;
```

Далі, указуємо в структурі `tv` час затримки для функції `select`:

```
tv.tv_sec := 5;
tv.tv_usec := 0;
```

Тепер можна викликати функцію `select` (тому що ми стежимо тільки за прийомом даних, то в якості writefds, exceptfds ми вказуємо nil):

```
select(0,@wfds,nil,nil,@tv);
```

Тепер, коли функція select поверне керування в змінної `wfds`, ми будемо мати набір сокетів, для яких необхідно зробити читання, і можемо обробити дані, що надійшли,:

```
if wfds.fd_count=0 then continue;
for i:=0 to wfds.fd_ count-1 do
begin
vSocket := wfds.fd_array[i];
//Обробка даних, що надійшли, із сокету vSocket.
end;
```

Умова виходу з основного циклу – відсутність відкритих сокетів у масиві `SockArray`. Умови закриття сокету я залишаю на совісті читача, додам тільки лише, що сокет потрапить в обробку `select` і при настанні події розриву зв'язку (для обробки можна використовувати те, що кількість прийнятих байт функцією recv буде дорівнює нулю, а також функцію `WSAGetLastError`).

Давайте тепер розберемося, для чого ми вказали час очікування функції send, а не зробили його нескінченним? Справа в тому, що в поточній реалізації, при створення нового сокету, він не потрапить в обробку select, поки не буде встановлений функцією `FS_SET`, що, природно, не пройде в режимі, що блокує, поки select не поверне керування по події з одним із клієнтів, що відслідковуються. Установка значення timeout гарантує, що сокет потрапить в обробку незалежно від стану інших сокетів.

Який варіант використовувати – потік на кожний сокет, або один потік обробляє всі з'єднання – питання, до якого потрібно підходити по-різному в кожному конкретному випадку. Якщо в процесі спілкування із клієнтом серверу потрібне значний час на підготовку й обробку даних, те, звичайно, варто створити різні потоки, щоб іншим клієнтам не довелося очікувати завершення обробки, а працювати паралелльно. У противному випадку, якщо обробка мінімальна, то створення великої кількості процесів вплине в гіршу сторону на загальну продуктивність системи.

Крім функції `select` існує ще два методи роботи з асинхронними сокетами:

```
function WSAAsyncSelect(s: TSocket; HWindow: HWND; wMsg: u_int; lEvent: Longint): Integer; stdcall;
```

Ця функція зв'язує сокет з одержанням повідомлень вікна. При виклику цієї функції, повідомлення про з'єднання, читання/запису даних у сокет і закритті сокету можна обробляти у функції обробки повідомлень від вікна.

```
const
WM_MYSOCKET = WM_USER + 1;
...
type
TForm1 = class(TForm)
...
private
procedure Socket_Proc(var Msg:TMessage);message WM_MYSOCKET;
...
WSAAsyncSelect(vSocket,Form1.Handle,WM_
MYSOCKET,FD_ACCEPT+FD_READ);
....
procedure TForm1.Socket_Proc(var Msg: TMessage);
begin
if ((Msg.Msg = WM_MYSOCKET)
and (Msg.lParam = FD_ACCEPT))
then ShowMessage('Connected');
end;
```

Не будемо затримуватися на обробці повідомлень вікна, а розглянемо ще один спосіб, що придасться, якщо у вас немає вікна додатка; цей спосіб заснований на системних подіях (Events). На жаль, файл winsock.pas не імпортує відповідні функції, у результаті чого багато програмістів зневажають можливостями подій. Але для нас це не лихо – напишемо власний імпорт необхідних процедур:

```
function WSAEventSelect(s: TSocket; Event: THandle; lEvent:
Longint):integer;stdcall;
external 'ws2_32.dll' name 'WSAEventSelect';
function WSAWaitForMultipleEvents(nCount: DWORD; lpHandles: PWOHandleArray;
bWaitAll: BOOL; dwMilliseconds: DWORD; fAlertable:BOOL):integer;stdcall;
external 'ws2_32.dll' name 'WSAWaitForMultipleEvents';
function WSACreateEvent:THandle;stdcall;
external 'ws2_32.dll' name 'WSACreateEvent';
function WSAResetEvent(Event : THandle):BOOL;stdcall;
external 'ws2_32.dll' name 'WSAResetEvent';
function WSAEnumNetworkEvents(const s : TSocket;
```

```
const Event : THandle; lpNetworkEvents : LPWSANetworkEvents): longint ;
stdcall;far;
external 'ws2_32.dll' name 'WSAEnumNetworkEvents';
function WSACloseEvent(Event : THandle):integer;
stdcall; external 'ws2_32.dll' name 'WSACloseEvent';
Також нам буде потрібно опис структури WSANetworkEvents
const
fd_max_event = 10;
type
TWSANetworkEvents = record
lNetworkEvents: LongInt;
iErrorCode: Array[0..fd_max_ event-1] of Integer;
end;
PWSANetworkEvents = ^TWSANetworkEvents;
LPWSANetworkEvents = PWSANetworkEvents;
```

Принцип роботи із цим набором функцій складається в створенні спеціального об'єкта типу Event, потім зв'язування цієї події із сокетом за допомогою функції `WSAEventSelect`, у якій також вказується набір станів, що відслідковуються, сокету. Один сокет може бути зв'язаний тільки з одним об'єктом типу `Event`.

Потім, у циклі обробки ми організуємо очікування надходження події від сокету; це реалізується за допомогою API функцій `WaitForSingleObject` – для очікування однієї події, або `WaitForMultipleObjects` – для очікування набору подій. При настанні події функція повертає керування. Для однозначної ідентифікації, від якого сокету прийшло повідомлення, у зв'язку із чим використовується функція `WSAEnumNetworkEvents`, що повертає структуру типу `TWSANetworkEvents`.

```
var
FEvent : THandle;
//Створюємо серверний сокет
...
FEventClose := WSACreateEvent;
WSAEventSelect(Socket,FEvent, FD_CLOSE + FD_READ );
repeat
WaitForSingleObject(FEvent,INFINITE);
WSAEnumNetworkEvents(FSocket,FEvent,@NI);
case NI.lNetworkEvents of
FD_Close:break;
```

```
FD_Read: begin
ReceiveData;
end;
end;
WSAResetEvent(FEventClose);
Until false;
WSACloseEvent(FEventClose);
```

Поза залежністю від того, чи використовуємо ми синхронні або асинхронні сокети і які методи обрані для обробки подій, що приходять від сокетів, при відправленні й прийомі даних є один підводний камінь, на який попадають всі починаючі програмісти мережевих додатків.

Розглянемо відправлення даних – справа в тому, що, як ми вже говорили, відправлення даних є, по суті, постановка порції даних у чергу. Ми не може управляти окремими пакетами, більше того, дані, що потрапили в буфер, відправляються не відразу, а можуть накопичуватися для відправлення надалі одним пакетом. Таким чином, послідовний виклик

```
send(vSocket,@buf1,Length(buf1),0);
send(vSocket,@buf2,Length(buf2),0);
```

фактично буде ідентичний одному виклику `send` з об'єднаним буфером `buf1+buf2`.

Таким чином, при прийманні даних, хоча ми послали дві порції даних, ми одержимо одну. Зовсім інший випадок, коли ми посилаємо порцію даних, більшу, ніж буфер сокету, тоді функція send відправить тільки частина даних із зазначеного буфера, рівно стільки, скільки влізло в буфер сокету. Для того щоб відстежити таку ситуацію й відправити неопрацьовану частину буфера, потрібно скористатися тим, що функція send повертає кількість фактично відісланих даних. Функцію send потрібно запускати в циклі, умовою завершення якого буде повне відправлення всього буфера.

Таким чином, при читанні із сокету даних, ми можемо спостерігати як би "склейку" порцій даних, або, навпаки, фрагментацію (не плутати із фрагментацією пакетів на рівні TCP/IP). Такі ситуації повинна обробляти наша програма. Вирішити проблему можна додаванням сигнатури ознаки кінця блоку

даних. Це має сенс, якщо додатки часто обмінюються невеликими блоками даних, де найчастіше виникає ефект "склеювання", але неефективно при більших об'ємах, тому що сканування великого буфера на предмет сигнатури забирає багато часу. Звичайно все-таки це вирішується таким способом – на початку кожного пакета додається 32-бітне число, що визначає довжину порції даних у байтах. Таким чином, що приймає частина, знаючи розмір кожного блоку, може розпізнати "склейку" і фрагментацію.

Викладеної інформації досить для розуміння ідеології мережевих додатків.

Опис TCP/IP

Стек протоколів TCP/IP (Transmission Control Protocol/Internet Protocol) – збірна назва для мережевих протоколів різних рівнів, використовуваних у мережах. Слово «стек» має на увазі, що протокол TCP працює поверх IP.

У моделі OSI даний стек займає (реалізує) всі рівні й ділиться сам на 4 рівні: прикладний, транспортний, міжмережевий, рівень доступу до мережі (в OSI це рівні – фізичний, канальний і частково мережевий). На стеці протоколів TCP/IP побудована вся взаємодія користувачів у мережі, від програмної оболонки до канального рівня моделі OSI. По суті це база, на якій зав'язане вся взаємодія. При цьому стек є незалежним від фізичного середовища передачі даних.

Рівні стека TCP/IP

Існують розбіжності в тім, як вписати модель TCP/IP у модель OSI, оскільки рівні в цих моделях не збігаються.

До того ж, модель OSI не використовує додатковий рівень – «Internetworking» – між транспортним і мережевим рівнями. Прикладом спірного протоколу може бути ARP або STP.

От як традиційно протоколи TCP/IP уписуються в модель OSI.

Таблиця 3.1 – Відповідність протоколів TCP/IP рівням у моделі OSI

	Рівень OSI	Протоколи TCP/IP
7	Прикладний	HTTP, SMTP, SNMP, FTP, Telnet, scp, NFS, RTSP, BGP
6	Представницький	XML, XDR, ASN.1, SMB, AFP
5	Сеансовий	TLS, SSL, ISO 8327/CCITT X.225, RPC, NetBIOS, ASP
4	Транспортний	TCP, UDP, RTP, SCTP, SPX, ATP, DCCP, GRE
3	Мережевий	IP, ICMP, IGMP, CLNP, ARP, RARP, OSPF, RIP, IPX, DDP
2	Канальний	Ethernet, Token ring, PPP, HDLC, X.25, Frame relay, ISDN, ATM, MPLS, Wi-Fi
1	Фізичний	електрика, радіо, лазер

Звичайно в стеці TCP/IP верхні рівні (прикладний, представницький і сеансовий) моделі OSI поєднують в один – прикладний. Оскільки в такому стеці не передбачається уніфікований протокол передачі даних, функції по визначенню типу даних передаються додатку. Спрощено інтерпретацію стека TCP/IP можна представити наступним чином (таблиця 3.2).

Фізичний рівень

Фізичний рівень описує середовище передачі даних (будь то кабель, оптоволокно або радіоканал), фізичні характеристики такого середовища й принцип передачі даних (поділ каналів, модуляцію, амплітуду сигналів, частоту сигналів, спосіб синхронізації передачі, час очікування відповіді й максимальна відстань).

Таблиця 3.2 – Відповідність протоколів рівням TCP/IP

	Рівень TCP/IP	Протоколи
7	Прикладний	HTTP, FTP, DNS (RIP, що працює поверх UDP, і BGP, що працює поверх TCP, є частиною мережевого рівня)
4	Транспортний	TCP, UDP, RTP, SCTP, DCCP (протоколи маршрутизації, подібні OSPF, що працюють поверх IP, є частиною мережевого рівня)
3	Міжмережевий	Для TCP/IP це IP (IP) (допоміжні протоколи, начебто ICMP і IGMP працюють поверх IP, але є частиною мережевого рівня; ARP не працює поверх IP)
2	Канальний	Ethernet, Token ring, і подібні.
1	Фізичний	фізичне середовище й принципи кодування інформації, T1, E1

Канальний рівень

Канальний рівень описує, яким образом передаються пакети даних через фізичний рівень, включаючи кодування (тобто спеціальні послідовності біт, що визначають початок і кінець пакета даних). Ethernet, наприклад, у полях заголовка пакета містить вказівку того, якій машині або машинам у мережі призначений цей пакет.

Приклади протоколів канального рівня – Ethernet, IEEE 802.11 Wireless Ethernet, SLIP, Token Ring, ATM і MPLS.

PPP не зовсім уписується в таке визначення, тому звичайно описується у вигляді пари протоколів HDLC/SDLC.

MPLS займає проміжне положення між канальним і мережевим рівнем і, строго говорячи, його не можна віднести до жодного з них.

Канальний рівень іноді розділяють на 2 підрівня – LLC і MAC.

Мережевий рівень

Розроблений для передачі даних з однієї (під)мережі в іншу. Прикладами такого протоколу є X.25 і IPC у мережі ARPANET.

З розвитком концепції глобальної мережі урівень були внесені додаткові можливості по передачі з будь-якої мережі в будь-яку мережу, незалежно від протоколів нижнього рівня, а також можливість запитувати дані від віддаленої сторони, наприклад у протоколі ICMP (використовується для передачі діагностичної інформації IP-З'єднання) і IGMP (використовується для керування multicast-потоками).

ICMP і IGMP розташовані над IP і повинні потрапити на наступний – транспортний – рівень, але функціонально є протоколами мережевого рівня, а тому їх неможливо вписати в модель OSI.

Пакети мережевого протоколу IP можуть містити код, що вказує, який саме протокол наступного рівня потрібно використовувати, щоб витягти дані з пакета. Це число – унікальний IP-номер протоколу. ICMP і IGMP мають номера, відповідно, 1 і 2.

Транспортний рівень

Протоколи транспортного рівня можуть вирішувати проблему негарантованої доставки повідомлень (« чи дійшло повідомлення до адресата?»), а також гарантувати правильну послідовність приходу даних. У стеці TCP/IP транспортні протоколи визначають для якого саме додатка призначені ці дані.

Протоколи автоматичної маршрутизації, логічно представлені на цьому рівні (оскільки працюють поверх IP), насправді є частиною протоколів мережевого рівня; наприклад OSPF (IP ідентифікатор 89).

TCP (IP ідентифікатор 6) – «гарантований» транспортний механізм із попереднім установленням з'єднання, що надає додатку надійний потік даних, що дає впевненість у безпомилковості одержуваних даних, що перезапитує дані у випадку втрати й усуючий дублювання даних. TCP дозволяє регулювати навантаження на мережу, а також зменшувати час очікування даних при передачі

на більші відстані. Більше того, TCP гарантує, що отримані дані були відправлені точно в такій же послідовності. У цьому його головну відмінність від UDP.

UDP (IP ідентифікатор 17) протокол передачі датаграмм без установлення з'єднання. Також його називають протоколом «ненадійної» передачі, у змісті неможливості впевнитися в доставці повідомлення адресатові, а також можливого перемішування пакетів. У додатках, що вимагають гарантованої передачі даних, використовується протокол TCP.

UDP звичайно використовується в таких додатках, як потокове відео й комп'ютерні ігри, де допускається втрата пакетів, а повторний запит утруднений або не виправданий, або в додатках виду запит-відповідь (наприклад, запити до DNS), де створення з'єднання займає більше ресурсів, чим повторне відправлення.

Й TCP, і UDP використовують для визначення протоколу верхнього рівня число, називане портом.

Прикладний рівень

На прикладному рівні працює більшість мережевих додатків.

Ці програми мають свої власні протоколи обміну інформацією, наприклад, HTTP для WWW, FTP (передача файлів), SMTP (електронна пошта), SSH (безпечне з'єднання з віддаленою машиною), DNS (перетворення символьних імен в IP-адреси) і багато які інші.

У своєй масі ці протоколи працюють поверх TCP або UDP і прив'язані до певного порту, наприклад:

- HTTP на TCP-порт 80 або 8080;
- FTP на TCP-порт 20 (для передачі даних) і 21 (для керуючих команд);
- SSH на TCP-порт 22;
- запити DNS на порт UDP (рідше TCP) 53;
- відновлення маршрутів по протоколі RIP на UDP-порт 520.

Ці порти визначені Агентством по виділенню імен і унікальних параметрів протоколів (IANA).

Безперечно, до цього рівня відносяться: DHCP, Echo, Finger, Gopher, HTTP, HTTPS, IMAP, IMAPS, IRC, NNTP, NTP, POP3, POPS, QOTD, RTSP, SNMP, SSH, Telnet, XDMCP.

Опис протоколу TELNET і SSH

TELNET (**TEL**ecommunication **NET**work) – мережевий протокол для реалізації текстового інтерфейсу по мережі (у сучасній формі – за допомогою транспорту TCP). Назву «telnet» мають також деякі утиліти, що реалізують клієнтську частину протоколу.

Хоча в сесії Telnet виділяють клієнтську й серверну сторону, протокол насправді повністю симетричний. Після встановлення транспортного з'єднання (як правило, TCP) обоє його кінця відіграють роль «мережевих віртуальних терміналів» (Network Virtual Terminal, NVT), що обмінюються двома типами даних:

– Прикладними даними (тобто даними, які йдуть від користувача до текстового додатка на стороні сервера й обернено);

– Опціями протоколу Telnet, які служать для з'ясування можливостей і переваг сторін.

Прикладні дані проходять через протокол без змін, тобто на виході другого віртуального термінала ми бачимо саме те, що було введено на вхід першого. З погляду протоколу дані представляють просто послідовність байтів (октетів), за замовчуванням приналежному набору ASCII, але при включеній опції Binary – будь-яких. Хоча були запропоновані розширення для ідентифікації набору символів, але на практиці ними не користуються.

Всі значення октетів прикладних даних крім \377 (десяткове 255) передаються по транспорті як є. Октет \377 передається послідовністю \377\377 із двох октетів. Це пов'язане з тим, що октет \377 використовується на транспортному рівні для кодування опцій.

Історично Telnet служив для віддаленого доступу до інтерфейсу командного рядка операційних систем. Згодом його стали використовувати для

інших текстових інтерфейсів, аж до ігор MUD і анімованого ASCII-art. Теоретично, навіть обидві сторони протоколу можуть бути програмами, а не людиною.

Іноді клієнти telnet використовуються для доступу до інших протоколів на основі транспорту TCP.

Протокол telnet використовується в керуючому з'єднанні FTP, тобто заходити на сервер командою telnet ftp.example.net ftp для виконання налагодження й експериментів не тільки можливо, але й правильно (на відміну від застосування клієнтів telnet для доступу до HTTP, IRC і більшості інших протоколів).

У протоколі не передбачене використання ні шифрування, ні перевірки дійсності даних. Тому він уразливий для будь-якого виду атак, до яких уразливий його транспорт, тобто протокол TCP. Для функціональності віддаленого доступу до системи в цей час застосовується мережевий протокол SSH (особливо його версія 2), при створенні якого упор робився саме на питання безпеки. Так що варто мати на увазі, що сесія Telnet досить беззахисна, якщо тільки не здійснюється в повністю контрольованій мережі або із застосуванням захисту на мережевому рівні (різні реалізації віртуальних приватних мереж). Через ненадійність від Telnet як засобу керування операційними системами давно відмовилися.

SSH (**S**ecure **Sh**ell – «безпечна оболонка») – мережевий протокол прикладного рівня, що дозволяє робити віддалене керування операційною системою й туннельованне TCP-з'єднаня (наприклад, для передачі файлів). Подібний по функціональності із протоколами Telnet і rlogin, але, на відміну від них, шифрує весь трафік, включаючи й передані паролі. SSH допускає вибір різних алгоритмів шифрування. SSH-клієнти й SSH-сервери є для більшості мережевих операційних систем.

SSH дозволяє безпечно передавати в незахищеному середовищі практично будь-який інший мережевий протокол, таким чином, можна не тільки віддалено

працювати на комп'ютері через командну оболонку, але й передавати по шифрованому каналі звуковий потік або відео (наприклад, з веб-камери). Також SSH може використовувати стиск переданих даних для наступного їхнього шифрування, що зручно, наприклад, для віддаленого запуску клієнтів X Window System.

Більшість хостинг-провайдерів за певну плату надають клієнтам доступ до їхнього домашнього каталогу по SSH. Це може бути зручно як для роботи в командному рядку, так і для віддаленого запуску програм (у тому числі графічних додатків).

Telnet і інші протоколи

У середовищі фахівців з технологій internet поширене думка, що клієнт Telnet придатний для здійснення ручного доступу (наприклад, з метою налагодження) до таких протоколів прикладного рівня як HTTP, IRC, SMTP, POP3 і прочими текст-орієнтованим протоколам на основі транспорту TCP. Однак, використання клієнта telnet як клієнт TCP викликає наступні небажані ефекти:

– Клієнт може передати дані, які Ви не вводили (опції Telnet);

– Клієнт не буде приймати октет \377;

– Клієнт буде спотворювати октет \377 при передачі;

– Клієнт взагалі може відмовитися передавати октети зі старшим бітом, який дорівнює 1.

Такі програми як netcat дійсно забезпечують чистий доступ до TCP, однак використання чистого доступу до TCP викликає іншу проблему. Переведення рядка, що вводиться з Unix-системи, буде переданий одним символом LF, у той час як названі протоколи вимагають CR LF як роздільник рядка – втім, більшість серверів задовольняються й одним LF, хоча по стандарту робити це не зобов'язані. Звичайний клієнт Telnet за замовчуванням передає будь-який переклад рядка саме як CR LF. Найбільш коректним рішенням для відладочного доступу до прикладних протоколів (крім FTP і, властиво, Telnet) є використання

клієнта PuTTY у режимі «Raw» (чистий доступ до TCP) – PuTTY перетворить переклади рядка окремо від підтримки протоколу Telnet.

3.2 Розробка структурної схеми

Проаналізувавши, всі досліджені технології віддаленого керування комп'ютером, движок програмного забезпечення розроблений у даному магістерському проекті виконує такі наступні основні дії:

Спершу відбувається з'єднання через Інтернет або локальну мережу із клієнтською частиною, що встановлена на комп'ютері який буде віддаленно управлятися.

Після цього клієнтом, за допомогою функцій API, грабую весь екран і передаю його серверу, той, у свою чергу, відтворює його у своїй робочій області й там можна рухати мишею й робити довільні дії; все це програма-сервер відслідковує, перехоплює й посилає клієнтові, а той їх відтворює.

Перераховані вище дії утворюють основне ядро програми.

На рисунку 3.4 зображена структурна схема віддаленого управління ЕОМ у загальному випадку.

З цієї схеми ми бачимо, що усі віддалені користувачі зв’язуються один з одним за допомогою RAS (серверів віддаленого управління доступом), модемів та маршрутизаторів.

При цьому для доступу у мережу підприємства, сервер віддаленого управління доступом використовує файрвол, для надання взаємного захисту між внутрішньою мережею підприємства, яку можно моніторити за допомогою, розробленого, у результаті виконання магістерського проектування, програмного забезпечення, з однієї сторони та сервером віддаленого управління доступом з іншої сторони.

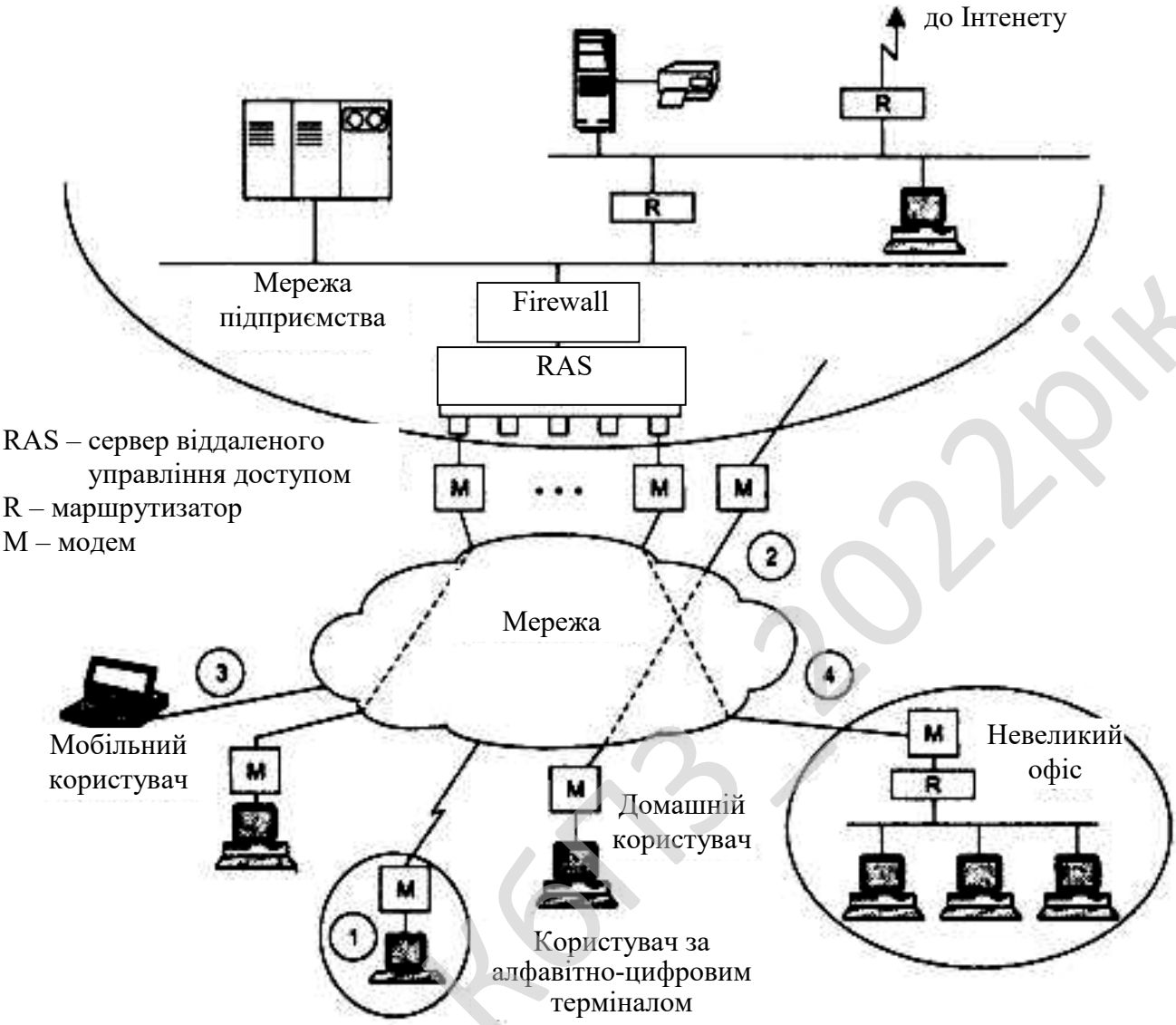

Рисунок 3.4 – Структуран схема системи

Схеми віддаленого керування наведені на рисунку 3.4, відрізняються типом взаємодіючих систем:

(1) – термінал-комп'ютер;

(2) – комп'ютер-комп'ютер;

(3) – комп'ютер-мережа;

(4) – мережа-мережа.

Відповідно при реалізації різних типів взаємодіючих систем використовується різне апаратне обладнання.

3.3 Розробка функціональної схеми

На рисунку 3.5 зображена функціональна схема розробленої системи дистанційного управління ЕОМ через мережу.

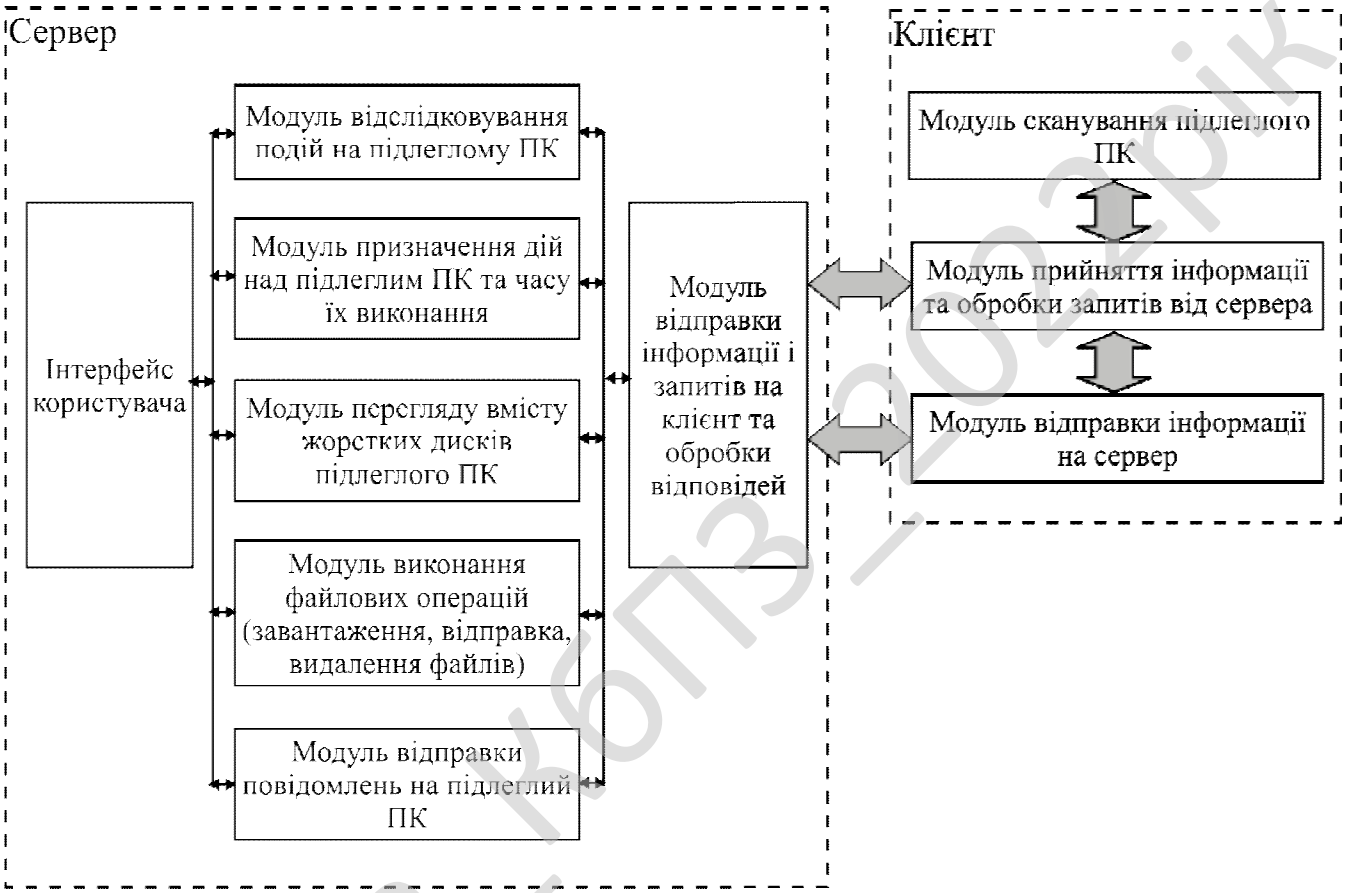

Рисунок 3.5 – Функціональна схема системи

З рисунка ми бачимо, що розроблена система функціонально складається з двох основних блоків:

– серверної частини;

– клієнтської частини.

Розглянемо ці функціональні частини більш докладно.

Серверна частина включає в себе наступні модулі.

Інтерфейс користувача, який дозволяє користувачу наочно працювати з програмою використовуючи, ті або інші функціональні модулі. До цих модулів

відносяться наступні:

– Модуль відслідковування подій на підлеглому ПК.

– Модуль призначення дій над підлеглим ПК та часу їх виконання.

– Модуль перегляду вмісту жорстких дисків підлеглого ПК.

– Модуль виконаня файлових операцій (завантаження, відправка, видалення файлів).

– Модуль відправки повідомлень на підлеглиий ПК.

Усі вищеперераховані модулі, взаємодіють з однієї сторони з інтерфейсом користувача, а з іншої сторони з модулем відправки інформаціїі запитів на клієнт та обробки відповідей. Цей модуль взаємодіє вже безпеосередньо з клієнтською частиноою розробленої системи.

Відповідно перейдемо до розгляду клієнтської частини розробленої програми віддаленого управління комп'ютером через мережу.

Клієнтська частина складається з наступних функціональних блоків:

– Модуль прийняття інформації та обробки запитів від серверу.

– Модуль відправки інформації на сервер.

– Модуль сканування підлеглого ПК.

Перший та другий модуль безпосередньо взаємодіють з серверною частиною, а точніше з таким її функціональним блоком, як модуль відправки інформації і запитів на клієнт та обробки відповідей.

3.4 Розробка діаграми процесів

На рисунку 3.6 зображена діаграма процесів системи. З неї ми бачимо що у системі відбуваються наступні процеси. Робота починається з процесу початку/кінця роботи системи, який взаємодіє з одного боку з зміною параметрів програми, а з іншого з з'єднанням з мережою. Після того як відбулося з'єднання з мережою запускається процес виведення списку підлеглих комп'ютерів. Після цього й починається основна частина роботи програми.

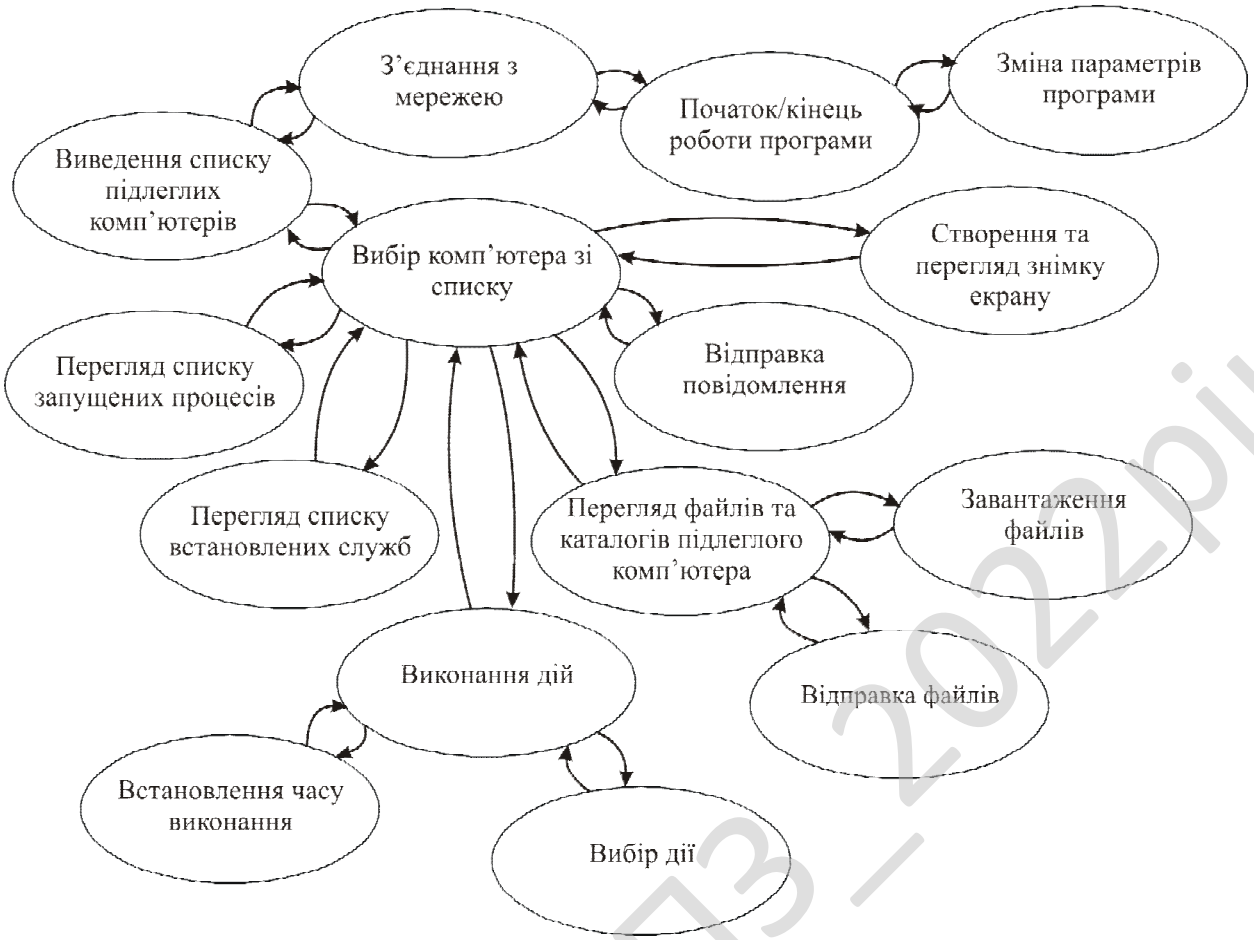

Рисунок 3.6 – Діаграма процесів системи

У основній частині головну роль грає процес вибора комп'ютера зі спику, після чого з ним можно проводити наступні дії:

– Перегляд списку запущених процесів.

– Перегляд списку встановлених служб.

– Створення та перегляд знімку екрану.

– Відправка повідомлення на вибраний комп'ютер.

– Перегляд файлів та каталогів підлеглого комп'ютера.

– Виконання дій: встановлення дії та часу її виконання.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Блок-схема основної програми наведена на рисунку 4.1. З неї ми бачимо що програма виконується за наступним алгоритмом.

Спершу відбувається ініціалізація зміних.

Після цього блокуються кнопки, що стають доступними тільки після вибору комп’ютера зі списку.

За цією дією виводиться головне вікно програми й відбувається з’єднання з мережею.

Після з’єднання з мережею виводиться список підлеглих комп’ютерів, якими адміністратор може керувати віддалено.

Якщо комп’ютер вибрано зі списку то розблокується кнопка та пункти меню, що відносяться до роботи з цим підлеглим комп’ютером.

Після цього відбуваються наступні дії.

Запропоновується показати файли та каталоги.

Якщо «так» – то відображається список логічних дисків та розміщених на них файлів й каталогів.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується завантажити файл.

Якщо «так» – то вибраний файл завантажується.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується відправити файл.

Якщо «так» – то вибраний файл відправляється на вказаний комп’ютер.

Якщо «ні» – то переходимо до наступної дії.

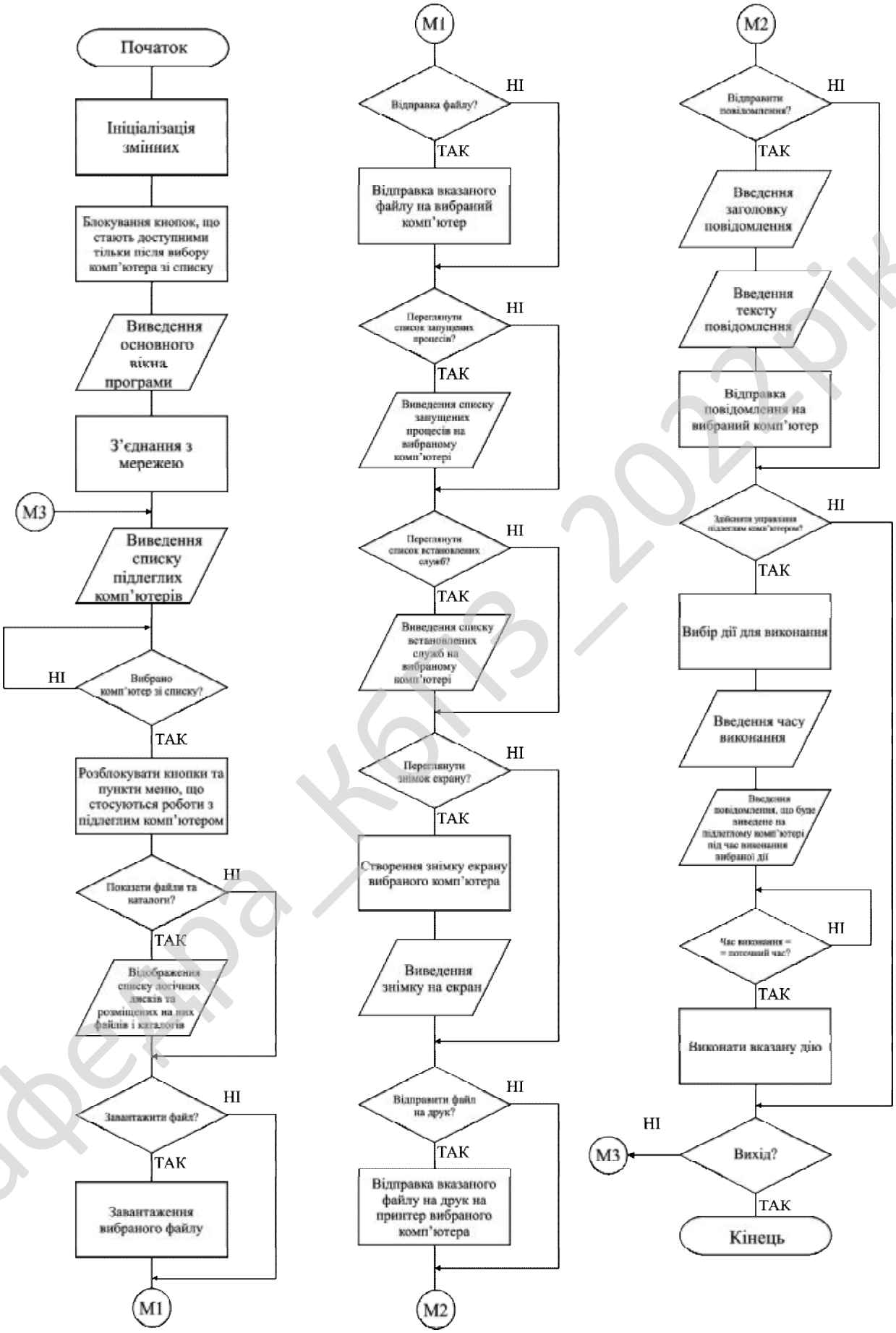

Рисунок 4.1 – Блок-схема роботи основної програми

Запропоновується переглянути список запущених процесів.

Якщо «так» – то виводиться список запущених процесів на вибраному комп'ютері.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується переглянути список встановлених служб.

Якщо «так» – то виводиться список встановлених служб на вибраному комп'ютері.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується створення знімку екрану вибраного комп'ютера.

Якщо «так» – то створюється знімок екрану вибраного комп'ютера та вивидитья на екран комп'ютера адміністратора.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується відправити файл на друк.

Якщо «так» – то відправляється вказаний файл на друк вибраного комп'ютера.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується відправити повідомлення.

Якщо «так» – то вводиться заголовок повідомлення. Після чого вводиться текст повідомлення й воно відправляється на вибраний комп'ютер.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується здійснити управління підлеглим комп'ютером.

Якщо «так» – то спершу вибирається дія викоання. Потім вводиться час виконання. Після чого вводиться повідомленя, яке буде виведене на підлеглому комп'ютері під час виконання заданої дії. Якщо часом виконання буде заданий поточний час, то вказана дія виконується негайно.

Якщо «ні» – то переходимо до наступної дії.

Запропоновується закінчити роботу програми

Якщо «так» – то програма закінчується

Якщо «ні» – то переходимо до початку програми.

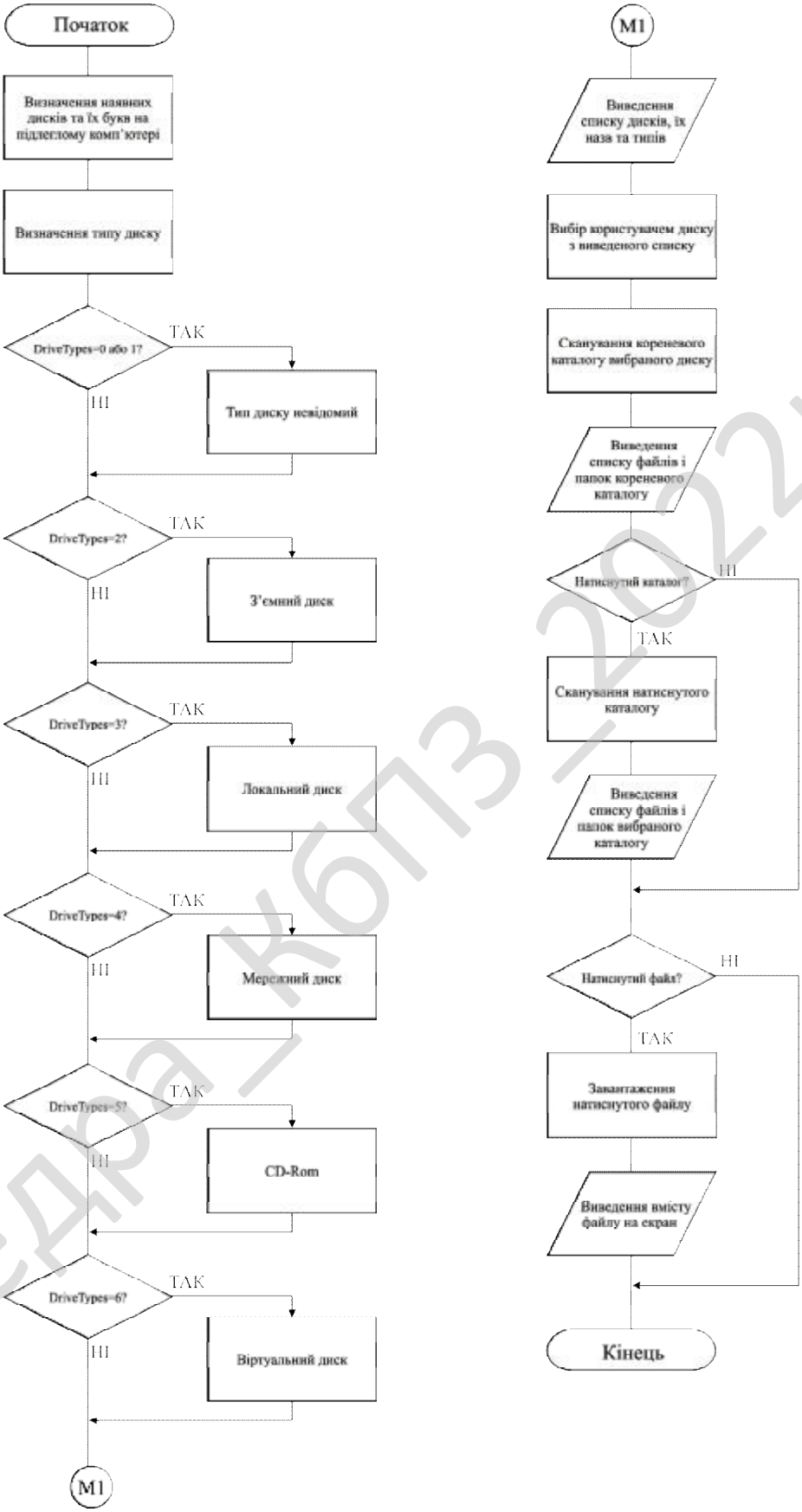

Рисунок 4.2 – Блок-схема роботи підпрограми перегляду файлів та папок підлеглого комп'ютера

На рисунку 4.2 зображено блок-схему роботи підпрограми перегляду файлів та папок підлеглого комп'ютера

Ця підпрограма починається з визначення наявних дисків, та їх букв на підлеглому комп’ютері.

Після цього відбувається визначення типу дисків наступним чином.

Якщо DriveTypes = 0 або 1 то тип диску невідомий.

Якщо DriveTypes = 2 то з’ємний диск.

Якщо DriveTypes = то локальний диск.

Якщо DriveTypes = 4 то мережевий диск.

Якщо DriveTypes = 5 то CD-Rom.

Якщо DriveTypes = 6 то віртуальний диск.

Після цього відбувається виведення списку дисків, їхї назв та типів.

Потім користувачем вибирається диск з наведеного списку.

З цим диском відбуваються наступні дії. Спершу сканується кореневий каталого вибраного диску.

За результатами сканування виводиться список файлів й папок кореневого каталогу.

Якщо на каталог наводиться мишка й натискається ліва кнопка, то відбувається сканування вибраного каталогу й виведення списку файлів та папок вибраного каталогу.

Після чого є можливість вибрати натиснути на файл або ні. Якщо на файл натискаємо, то завантажується натиснутий файл, та виводиться вміст файлу на екран.

Після цього підпрограма закінчує свою роботу.

Нижче наведемо деякі з функцій, які дозволяють реалізувати передачу даних з клієнта на сервер.

Функція одержання даних на стороні сервера.

```
// Додаток-Сервер.
//Сокети працюють у режимі, що блокує.
//На кожне з'єднання створюється окремий потік.
{$APPTYPE CONSOLE}
```

```
uses
SysUtils,
Winsock,
Windows;
var
vWSAData : TWSAData;
vListenSocket,vSocket : TSocket;
vSockAddr : TSockAddr;
trId : THandle;
const
cPort = word(33);
cSigExit = 'q';
//Процедура окремого потоку для кожного клієнта.
procedure SocketThread;
var SockName : TSockAddr;
aBuf : array of char;
vBuf : string;
vSize : integer;
s :TSocket;
BufSize : integer;
begin
s := vSocket;
if s = INVALID_SOCKET then exit;
vSize := SizeOf(TSockAddr);
getpeername(s, SockName, vSize);
Writeln(format('Клієнт допустимий, віддалена адреса [%s].',[inet_ntoa
(SockName.sin_addr)]));
//Визначаємо розмір буфера читання для сокету
vSize := sizeOf(BufSize);
getsockopt(s,SOL_SOCKET,SO_RCVBUF,PChar(@
BufSize),vSize);
writeln(format('Розмір буферу приймача [%d]',[BufSize]));
SetLength(aBuf,BufSize);
repeat
//Одержуємо дані. Процедура працює в режимі, що блокує,
//у такий спосіб наступний рядок коду не одержить керування,
//поки не надійдуть дані від клієнта.
vSize := recv(s,aBuf[0],BufSize,0);
if vSize<=0 then Break;
SetLength(vBuf,vSize);
lstrcpyn(@vBuf[1],@aBuf[0],vSize);
writeln(format('Прийнято від клієнта: %s',[vBuf]));
```

```
        until vBuf = 'q';
        Writeln(format('      Клієнт      роз'єднаний,      віддалена      адреса
[%s].',[inet_ntoa(SockName.sin_addr)]));
        SetLength(aBuf,0);
        closesocket(s);
        end;
        begin
        Writeln(' Стартує додаток...');
        //Повідомляємо, що програма буде використовувати Windows Sockets.
        if WSAStartup($101,vWSAData)<>0 then Halt(1);
        Writeln(' Використовуємо сокети Windows.');
        //Створюємо сокет що прослуховує.
        vListenSocket := socket(AF_INET,SOCK_STREAM,IPPROTO_IP);
        Writeln(format(' Створюємо сокет на порт [%d].',[cPort]));
        if vListenSocket = INVALID_SOCKET then Halt(1);
        FillChar(vSockAddr,SizeOf(TSockAddr),0);
        vSockAddr.sin_family := AF_INET;
        vSockAddr.sin_port := htons(cPort);
        vSockAddr.sin_addr.S_addr := INADDR_ANY;
        Writeln(' Зв'язок з сокетом...');
        //Прив'язуємо адреса й порт до сокету.
        if bind(vListenSocket,vSockAddr,SizeOf(TSockAddr)) <> 0
        then Halt(1);
        //Починаємо прослуховувати.
        if listen(vListenSocket,SOMAXCONN) <> 0
        then Halt(1);
        Writeln(' Статус сокету: слухає.');
        repeat
        //Очікуємо підключення.
        vSocket := accept(vListenSocket,nil,nil);
        //Клієнт підключився, запускаємо новий процес на з'єднання.
        CreateThread(nil,0,@SocketThread,0,0,trId);
        until false;
        closesocket(vListenSocket);
        WSACleanup;
        end.
```

Отут використовувалася не описана раніше функція `getpeername()`, що повертає інформацію про канал, асоційованому із сокетом. Я використовував її для одержання інформації про IP-адресу клієнта, що підключився. Докладно про цю функцію, як і про всі інші функції WinSock, ви можете прочитати в Windows

Sockets 2 Application Program Interface, що входить до складу Win32 Programmer's Reference.

Вихідний код клієнта не представляє із себе нічого цікавого й не вимагає ніяких коментарів.

```
{$APPTYPE CONSOLE}
uses
SysUtils,
winsock;
const
cPort = 33;
cSigExit = 'q';
var
vWSAData : TWSAData;
vSocket : TSocket;
vSockAddr : TSockAddr;
buf : string;
begin
if WSAStartup($101,vWSAData)<>0 then Halt(1);
vSocket := socket(AF_INET,SOCK_STREAM,IPPROTO_IP);
if vSocket = INVALID_SOCKET then Halt(1);
FillChar(vSockAddr,SizeOf(TSockAddr),0);
vSockAddr.sin_family := AF_INET;
vSockAddr.sin_port := htons(cPort);
vSockAddr.sin_addr.S_addr := inet_addr('127.0.0.1');
if   connect(vSocket,vSockAddr,SizeOf(TSockAddr))   =   SOCKET_ERROR   then
Halt(1);
repeat
Readln(buf);
if send(vSocket,buf[1],Length(buf),0) = SOCKET_ERROR then Break;
until buf = cSigExit;
closesocket(vSocket);
WSACleanup;
end.
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір

блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і теорії комп'ютерних мереж. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних [2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Перед запуском програм необхідно встановити наступні бібліотеки:

- ZLibEx (перебуває в папці zlib).
- CoolTrayIcon (перебуває в папці CoolTrayIcon).
- Drag&Drop (перебуває в папці Components).
- WSockets (перебуває в папці Components).
- GHIPEdit (перебуває в папці Components).

Після цього потрібно виконати наступні кроки:

0. В Tools>EnvironmentOptions на закладці Library в LibraryPath додати шлях $(DELPHI)\Lib\Delphi2.

1. Установити ZLibEx.

2. Установити CoolTrayIcon.

4. Установити DropSource і DropTarget.

5. Установити WSockets.

6. Установити uServiceController.

7. Установити uTCPParser.

8. Установити uBaseLanClient і uBaseLanServer.

9. Установити uFileInfo – якщо він запросить модуль OLE2, значить ви не виконали пункт 0 не плутати з повідомленням Ole2 implicity imported into ...).

10. Установити uProcessInfo.

11. Установити GHIPEdit.

Використання:

Передбачається, що всі зазначені вище компоненти встановлені й прописані шляхи до них в опціях проектів.

1. Скомпілювати клієнтську частину (svcclean.dpr) і установник (Setup.dpr). Отримані exe файли покласти в окрему папку (ніяких сторонніх dll.

не використовується).

2. Скомпілювати серверну частину (ProjectServer.dpr), отриманий exe файл покласти в папку окремо від svcclean.exe і setup.exe.

3. Скопіювати файли svcclean.exe і setup.exe на підлеглі комп'ютери, які потрібно контролювати, у будь-який каталог.

4. Запустити файл setup.exe на клієнтських машинах, програма встановиться в каталог Windows\System32.

5. Перевірити підключення комп'ютерів до локальної мережі.

6. Запустити на головному комп'ютері файл ProjectServer.exe.

Можливості програми:

1. Перегляд файлів та каталогів підлеглих комп'ютерів.
2. Завантаження файлів з підлеглих комп'ютерів.
3. Копіювання файлів з головного копм'ютера на підлеглі.
4. Видалення файлів з клієнтських комп'ютерів.
5. Перегляд списоку служб підлеглого комп'ютера.
6. Перегляд списоку запущених процесів на підлеглому комп'ютері.
7. Відправлення повідомлення на клієнтські комп'ютери.
8. Дозвіл роботи клієнту.
9. Створення та перегляд знімку екрана підлеглого комп'ютера.
10. Відсилання документу на друк на принтер підлеглого комп'ютера.
11. Перегляд властивостей підлеглих комп'ютерів.
12. Оновлення програми клієнта на підлеглих комп'ютерах.

Інтерфейс розробленої програми зображений на рисунках 5.1–5.7.

Головне вікно прогами зображене на рисунку 5.1. Для здійснення підключення до мережі та виведення списку підлеглих комп'ютерів, треба натиснути кнопку  , або пункт меню **З'єднання->Підключити**.

Для перегляду файлів та каталогів підлеглого комп'ютер, треба натиснути на потрібному комп'ютері у списку та вибрати пункт меню користувача **Дія->Показати->Файли та каталоги**, а потім вибрати логічний диск зі списку,

що з'явиться, після чого його вміст буде виведено на екран (рисунок 5.1).

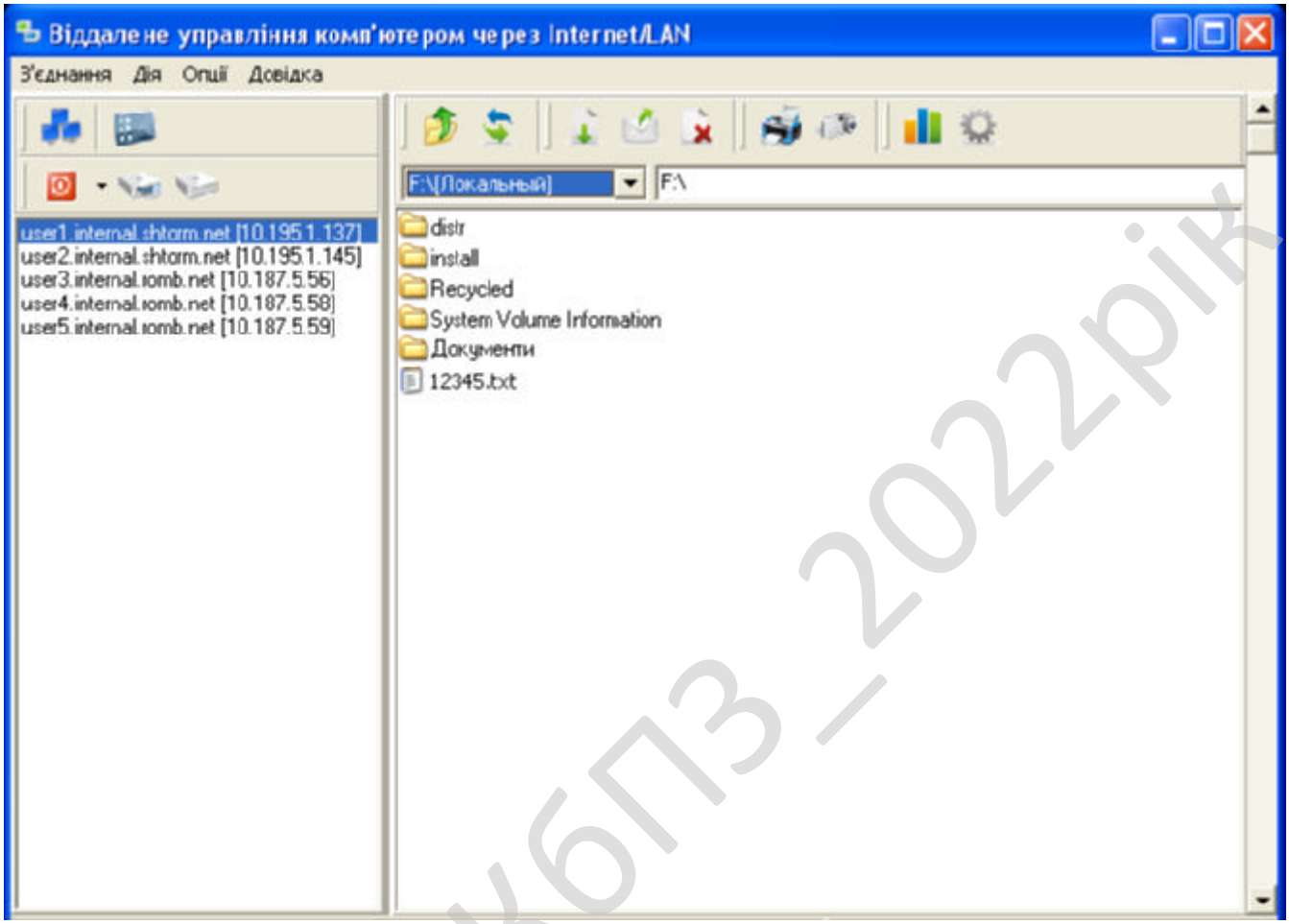

Рисунок 5.1 – Головне вікно програми (відображення файлів та каталогів підлеглого комп'ютера)

Для перегляду списоку служб підлеглого комп'ютера, треба вибрати пункт меню користувача **Дія->Показати->Список служб**, або натиснути кнопку  (рисунок 5.2). На екрані з'явиться список служб, що містить:

- ім'я сервіса;
- назву;
- опис;
- стан;
- тип запуску;

– вхід від імені.

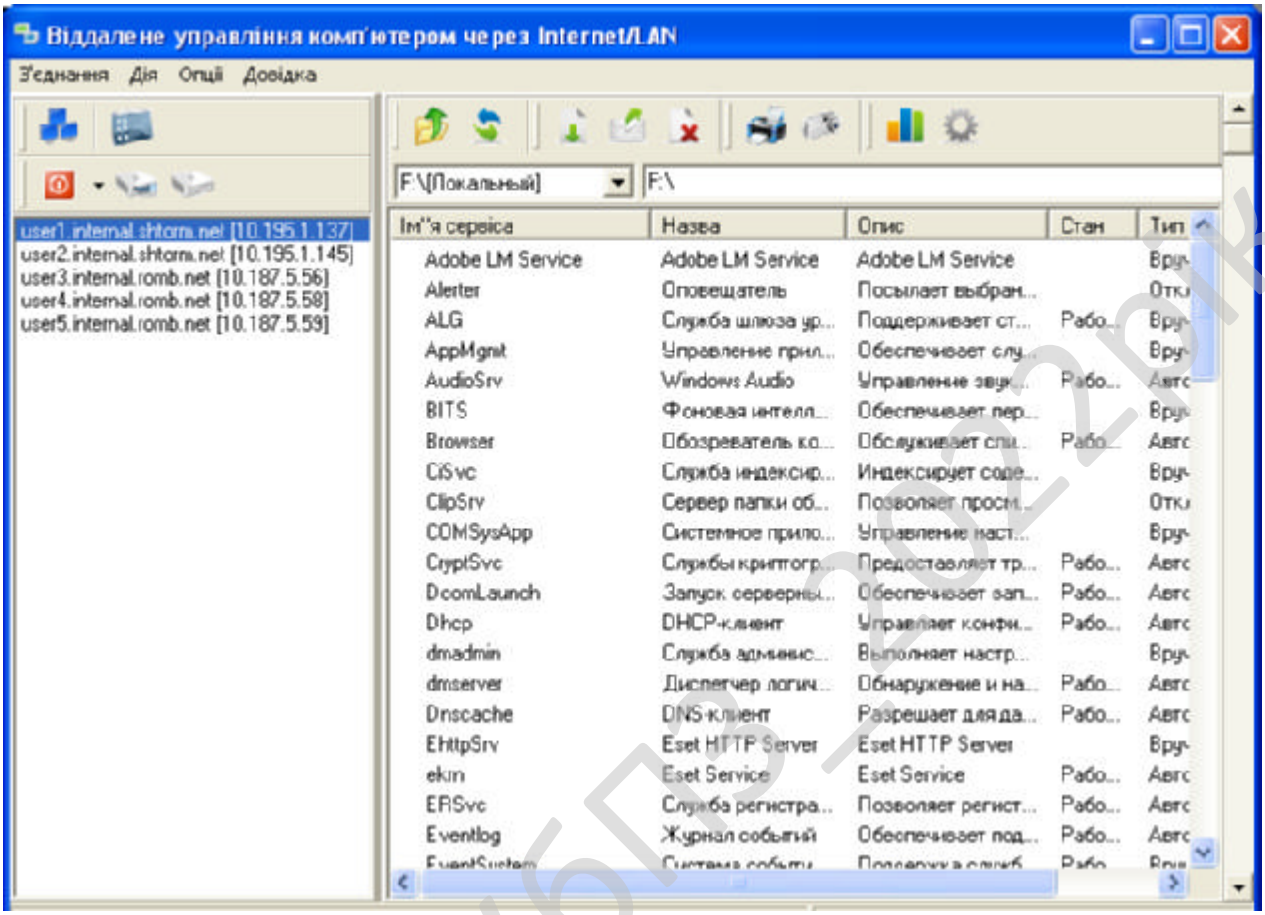

Рисунок 5.2 – Головне вікно програми (список служб підлеглого комп'ютера)

Для перегляду списку запущених процесів на підлеглому комп'ютері, треба вибрати пункт меню користувача **Дія->Показати->Список процесів**, або натиснути кнопку  (рисунок 5.3). У списку, що з'явиться на екрані будуть вказані:

– назва файлу процесу;

– ім'я користувача;

– займана пам'ять.

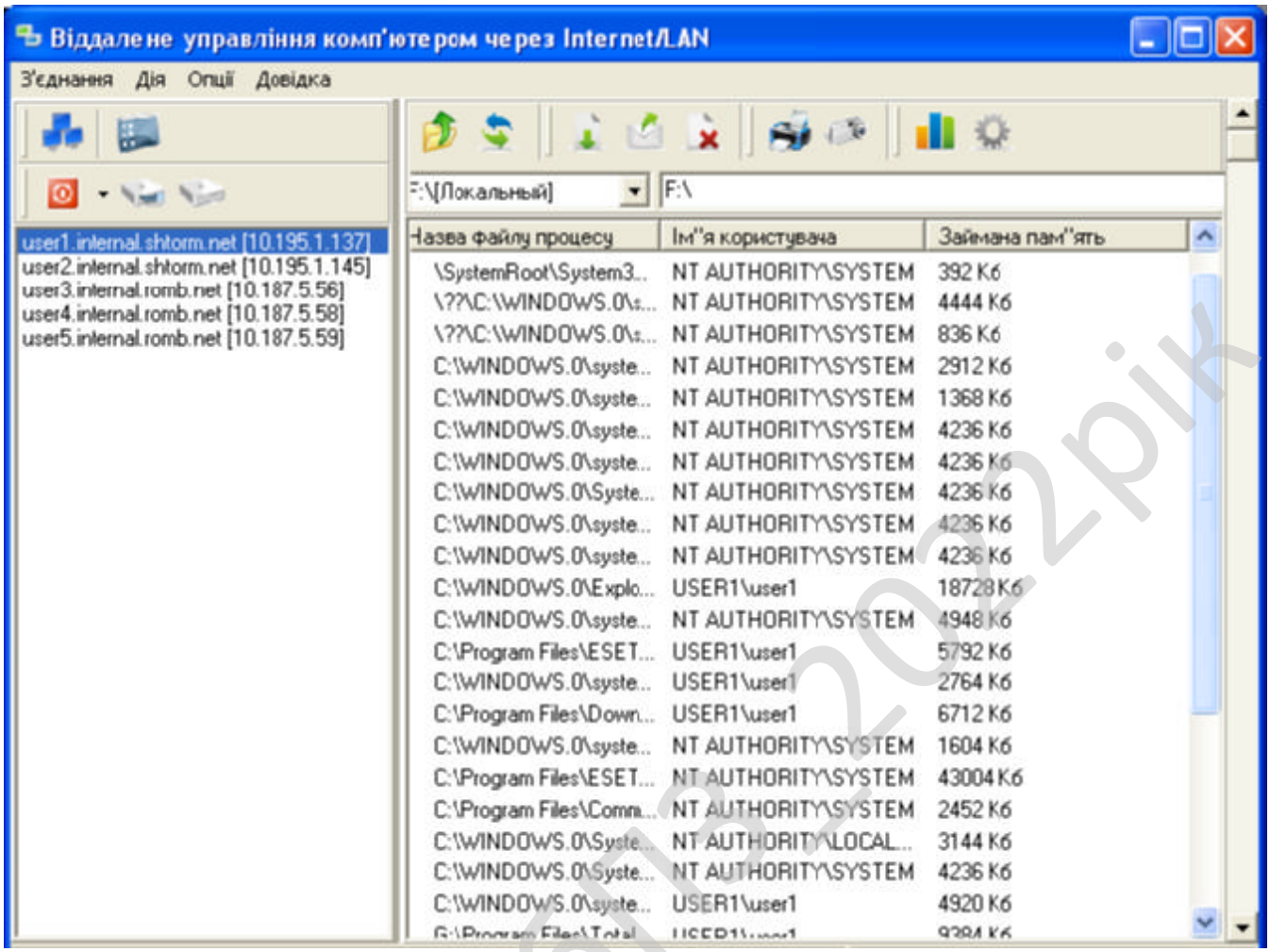

Рисунок 5.3 – Головне вікно програми (список запущених процесів на підлеглому комп'ютері)

Розроблена програма дозволяє обмежувати роботу на підлеглих комп'ютерах, вказувати час їх вимкнення, перезавантаження, завершення сеансу роботи користувача та завершення роботи запущених програм. Для здійснення даних дій слід вибрати в меню користувача пункт **Дія->Вимикання->Дозволити роботу на…** Вікно, що з'явиться після цього на екрані зображене на рисунку 5.4. В ньому слід вказати час виконання, вибрати дію та ввести повідоблення, що виведеться на екран підлеглої машини під час виконання вказаної дії.

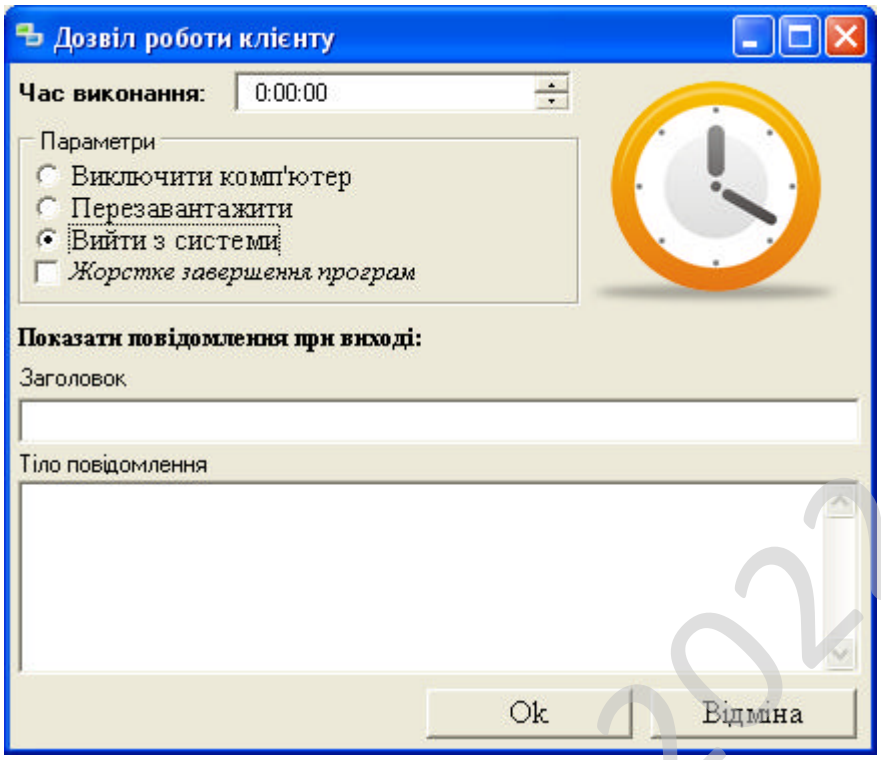

Рисунок 5.4 – Дозвіл роботи клієнту

Для відправлення повідомлення на підлеглий копм'ютер, слід вибрати його зі списку комп'ютерів та натиснути пунк меню користувача **Дія->Відправити повідомлення**. У діалоговому вікні, що з'явиться на екрані треба набрати заголовок та текст повідомлення і натиснути кнопку «ОК» (рисунок 5.5).

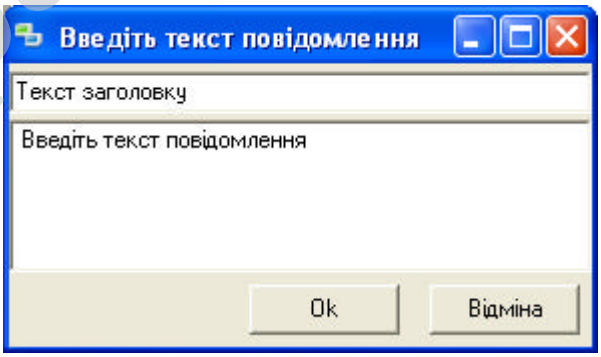

Рисунок 5.5 – Відправка повідомлення

Для зміни параметрів програми слід вибрати пункт меню користувача

Опції->Змінити параметри програми… Після чого з'явиться вікно зображене на рисунку 5.6.

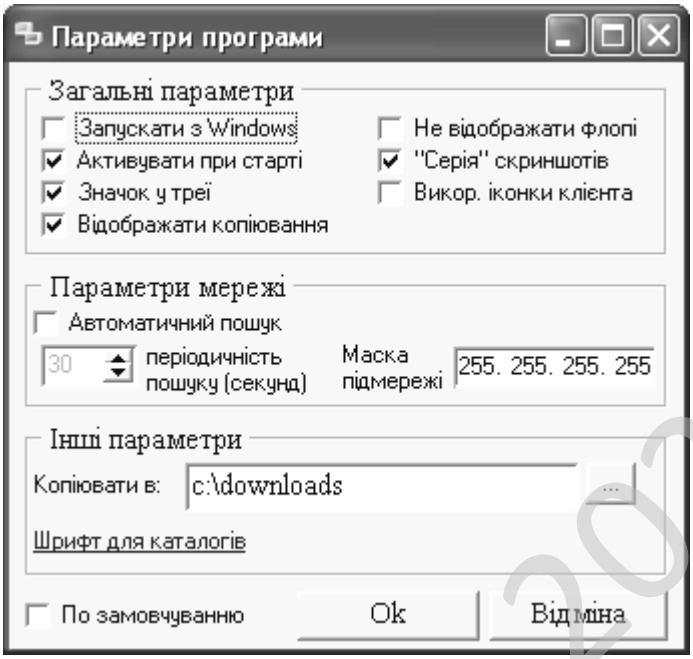

Рисунок 5.6 – Параметри програми

Для перегляду довідки про програму та її автора, слід вибрати пункт меню **Довідка->Про програму…** Після чого з'явиться вікно зображене на рисунку 5.7.

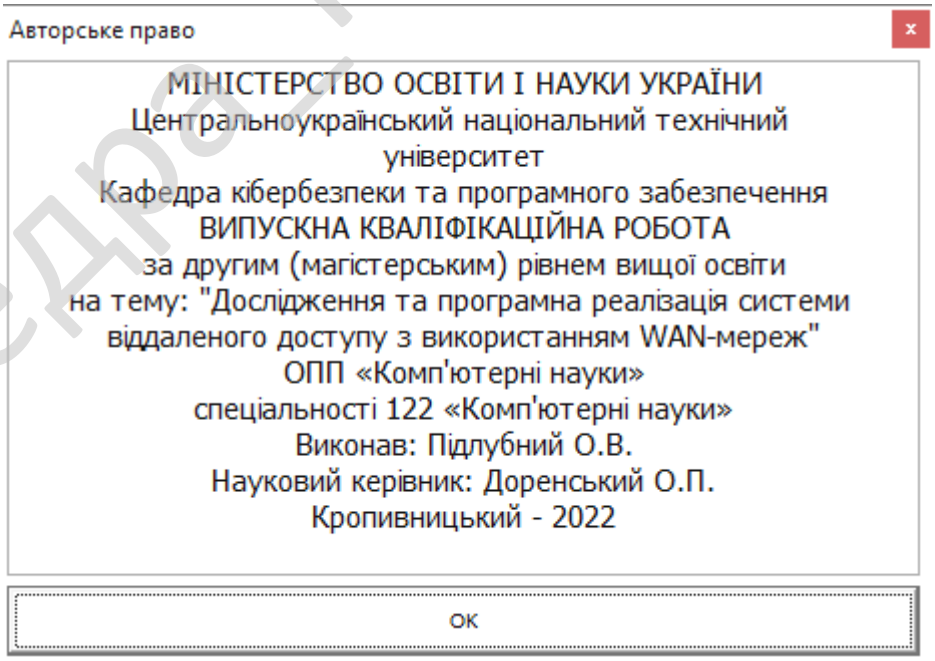

Рисунок 5.7 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віддаленого доступу з використанням WAN-мереж.

Метою розробки є дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

Об'єктом дослідження є процес віддаленого доступу з використанням WAN-мереж.

Предметом дослідження є методи віддаленого доступу з використанням WAN-мереж.

Методи дослідження базуються на методах теорії комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод віддаленого доступу з використанням WAN-мереж.

– Розроблено вітчизняний продукт віддаленого доступу з використанням WAN-мереж, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 днів (один місяць).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	30
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Г
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	8
8. Кількість форм вихідної інформації.	–	6
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	1
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	3
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	4
17. Складність кінцевого програмного продукту (1-6)	–	5
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	3
20. Вимоги до швидкодії ПП (1-6)	–	3
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	4
23. Професійний рівень аналітиків (1-6)	–	3
24. Професійний рівень програмістів (1-6)	–	4
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	1
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	30000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	40
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \quad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+4{,}22+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}2^{1{,}027} = 5{,}5 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \quad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}5 \cdot (1 \cdot 1{,}09 \cdot 1{,}30 \cdot 0{,}91 \cdot 1 \cdot 1 \cdot 1 \cdot 1{,}15 \cdot 1 \cdot 0{,}87 \cdot 1{,}10 \cdot$$

$$\cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}10 \cdot 1 \cdot 1 \cdot 1{,}10) = 12{,}9 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \quad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 12{,}9^{0{,}33+0{,}2(1{,}027-1{,}01)} \cdot 60 = 112 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	112	Ф 7.1-7.4
Впровадження	15	Д13
Всього	161	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{161 \cdot 1}{24\text{-}3} = 7,7 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	10	900	15
Монітор	60	10	600	10
Клавіатура	30	10	300	5
Маніпулятор «мишка»	30	10	300	5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	2	60	1
Кабельні господарства ЛОМ на 1 м.п.	2,5	150	375	6,25
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	47,91

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{48 \cdot 1}{1,2} = 40 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 40/(24 \cdot 8) = 0{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	1	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	1	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	1	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	0,25	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,75	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,5
	Створення графічних і стилістичних елементів сайту	1	
	Оформлення банерів і промо-сторінок	1	
	Розміщення графіки і контенту на Інтернет сторінках	1	
Всього		4	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,5
	Верстка друкованих видань	1	
	Додрукова підготовка макетів	1	
	Розміщення графіки і контенту на Інтернет сторінках	1	
Всього		4	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	8385	8385
Продакт-менеджер	0,25	8500	2125
Інженер-програміст	7,7	8780	67606
Інженер-електронщик	0,2	7000	1400
Інженер-системотехнік	0,5	7000	3500
Адміністратор мережі	0,5	8000	4000
Дизайнер WEB	0,5	10000	5000
Всього за період розробки	$R_{сп}$ = 10,65	-	$Ф_{роб}$ = 92016

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{92016}{10,65 \cdot 24} = 360 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 13 \cdot 8 \cdot 20000 = 2080000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 208000 грн. Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 13 \cdot 3500 = 45500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет магазину Компбест за 14.11.22 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11457
Системний блок		7509
Процесор	Intel Core i7-4790 (4(8) ядра по 3.6 - 4. GHz); Cache Memory 8 MB	4200

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системна плата	1st Player ATX NEW	1525
Відеокарта	PCIeX: ATI HD5670 SAPPHIR 1024MB/128bit/DDR3/TV/DualDVI	430
Жорсткий диск	HDD: 500 Gb 7200 Serial ATA WD 16MB	490
Оперативна пам'ять	Kingston DDR3 2GB (KVR1333D3N9/2G Intel/AMD – 2 шт	333
DVD-привод	-	-
Корпус	ATX Middle Tower GIGABYTE GZ-X Silver 500W (GZ-X4 Silver)	411
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	120
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1348

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	13	11457	14894,1	163835,1
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	297	3267
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	185653,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	2080000	-	-
2. Передавальні пристрої	208000	-	-
Всього по групі	2288000	5	114400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	185654	-	-
Всього по групі	185654	50	92827
Група 5, 6			
4. Вимірювальні пристрої	3999	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	45500	25	-
Всього по групі	49499	-	12374,75
7. Нематеріальні активи	30000	10	3000
Разом	$К_р = 2553153$		$А_р =$ 222602

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_e}, \quad (7.11)$$

де: N_e – кількість екземплярів програм, шт.

$$З_о = 360 \cdot 161/30 = 1932 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 1932 \cdot 10 \cdot 0{,}01 = 193 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $H_c = 22\%$ від суми основної та додаткової зарплати:

$$C_{оц} = 0{,}01 \cdot H_c (З_о + З_д), \quad (7.13)$$

де: H_c – відрахування на соціальні потреби, %.

$$C_{оц} = 0{,}01 \cdot 22(1932+193) = 468 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $H_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot H_г \cdot 0{,}01, \quad (7.14)$$

де: $H_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 1932 \cdot 15 \cdot 0{,}01 = 290 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам’ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картріджей, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,2 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м \cdot n. \quad (7.16)$$

$$З_{М1} = 200 \cdot 1 \cdot 0{,}2 = 40 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам’ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 3):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 35 \cdot 2 + 23 = 93 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{МЗ} = \sum Ц_{з}., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570 = 1702 \text{ грн.}$$

$$З_{М} = (40 + 93 + 1702)/30 = 61 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 1932 \cdot 15 \cdot 0{,}01 = 290 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 30 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 222602 \cdot 1/(30 \cdot 12) = 618 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 1932+193+468+290+61+290+618 = 3852 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	1932
2. Додаткова зарплата виконавців	$З_д$	193
3. Відрахування на соціальні потреби	$С_{оц}$	468
4. Загальногосподарські витрати	$Г_{осп}$	290
5. Витрати на матеріали	$З_М$	61
6. Освоєння нових операційних систем, мов програмування	$О_п$	290
7. Амортизація основних фондів	$А_м$	618
8. Повна собівартість програмного забезпечення	$С_п$	3852
9. Плановий прибуток	$П_р$	1541
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	5393
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	1077
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	6470

Визначимо плановий прибуток за рівнем рентабельності ($Р_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 40%.

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \qquad (7.22)$$

де: $Р_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 40 \cdot 3852 = 1541 \text{ грн.}$$

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	6470
Всього капітальних витрат	–	6470

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Витрати на профілактичні роботи:

$$З_р = T_р \cdot З_г \cdot (1 + 0{,}01 \cdot H_q) \cdot (1 + 0{,}01 \cdot H_c), \qquad (7.23)$$

де: $T_р$ – кількість годин обслуговування кожного комп’ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	40260	21472
2. Витрати на електроенергію	$З_{ел}$	22680	22050
3. Витрати на амортизацію	$З_{ам}$	0	3235
Всього витрат за рік	$І$	62940	46757

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 300 годин на рік до 160 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 300 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 40260\ грн,$$

до:

$$З_{р\ нов} = 160 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 21472\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 10 \cdot 0{,}15 \cdot 7200 \cdot 2{,}1 = 22680 \text{ грн.}$$

$$З_{ел\ нов} = 10 \cdot 0{,}15 \cdot 7000 \cdot 2{,}1 = 22050 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	6470	–	3235
Всього відрахувань	-	–	6470	–	3235

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (5393 - 3852) \cdot 30 - (0,05 \cdot 2288000 + 0,5 \cdot 185654 + \\ +0,25 \cdot 49499 + 0,1 \cdot 30000) \cdot 1/12 = 27680 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{265153}{(5393 - 3852) \cdot 30 \cdot 12 / 1} = 0,5 \text{ року}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	30
2. Повна собівартість розробленої програми	Грн.	3852
3. Ціна розробленої програми	Грн.	5393
4. Плановий прибуток від реалізації розробленої програми	Грн.	1541
5. Рентабельність програмної продукції	%	40
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	2553153
7. Загальний прибуток від реалізації програмної продукції	Грн.	46230
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	27680
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	6470
11.Величина економічного ефекту у користувача програмної продукції	Грн.	12948
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,4

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (62940\text{-}46757)\text{-}0{,}5 \cdot 6470 = 12948 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{6470}{62940 - 46757} = 0,4 \ року .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні.

– Положення про охорону праці.

– Накази з охорони праці.

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи

з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території;

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м'язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5
Довжина	6
Висота	3,4

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6
Об'єм, V	$м^3$	не менше 20.0	20,4

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин)

У зазначеному приміщенні працюють двоє людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-

обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри). Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами. У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-60	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер *HP EU Laser 107a (4ZB77A*), електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень.

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Проводемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 5 м, довжина – 6 м, висота – 3,4 м.

У зазначеному приміщенні працює 5 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S=5 х 6 = 30 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, *S* = 30 м2;

h – розрахункова висота підвісу, *h* = 3 м (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

A – ширина приміщення, *A* = 5 м;

B – довжина приміщення, *B* = 6 м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=0,43.$$

Знаючи індекс приміщення, за знаходимо *n* = 0,23 (з табличних даних коефіцієнтів використання світлового потоку (*n*) світильників з відповідним типом лампам). Підставимо всі значення у формулу, визначемо світловий потік: *F=64565 Лм.*

Для розрахунку дудемо використовувати світлодіодні панелі *LED панель 42Вт 6000К SUNLED 000000127*, світловий потік яких $F_л$ *= 3990 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$N= 64565/ 3990=16,18 \text{ шт.}$$

Приймаємо необхідну кількість світлодіодних світильників 17 шт.

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи віддаленого доступу з використанням WAN-мереж.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів віддаленого доступу з використанням WAN-мереж.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем віддаленого доступу з використанням WAN-мереж.

– Досліджена система віддаленого доступу з використанням WAN-мереж.

– На основі отриманих результатів досліджень створена програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віддаленого доступу з використанням WAN-мереж.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 12948 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,4 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Підлубний О.В. Дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Терри Оглтри Модернизация и ремонт сетей = Upgrading and Repairing Networks. – 4-е изд. – М.: «Вильямс», 2005. – С. 1328. – ISBN 0-7897-2817-6

3. Дуглас Камер Сети TCP/IP, том 1. Принципы, протоколы и структура = Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture. – М.: «Вильямс», 2003. – С. 880. – ISBN 0-13-018380-6

4. . Дуглас Камер, Дэвид Л. Стивенс, Сети TCP/IP, том 3. Разработка приложений типа клиент/сервер, издательство «Вильямс», 2002 г., 592 стр.

5. Гэри Неббет. Справочник по базовым функциям API Windows NT/2000. 528 стр., с ил.; ISBN 5-8459-0238-X, 1-57870-199-6; формат 70x100/16; серия Circle; 2002, 1 кв.; Вильямс.

6. Microsoft Windows API. Справочник системного программиста. ДиаСофт. – 2004 – 1216 с.

7. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

8. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

9. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis,*

G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

10. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

11. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

12. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

13. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

14. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

15. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. **(Scopus)**.

16. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings.* Volume 2740, 2020, Pages 102-114. **(Scopus)**.

17. Smirnov O.A., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346. **(Scopus)**.

18. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

19. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

20. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

21. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T.**, «**Representation of Cascade Codes in the Frequency Domain**».** In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. **Springer**, Cham. 2021. pp 557-587. **(Scopus)**.

22. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136. **(Scopus)**.

23. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379. **(Scopus)**.

24. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No., 2020. PP.33-43. **(Scopus)**.

25. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

27. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407. **(Scopus)**.

28. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

29. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019. **(Scopus)**.

30. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629. **(Scopus)**.

31. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

32. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures*. ***Collective monograph***. Edited by Ivan D. Gorbenko, Victor A.

Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

33. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures**. Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

34. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. **Collective monograph**. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

35. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

36. Смірнов О.А., Дрєєва Г.М., «Метод генерування фрактального трафіку за допомогою моделі генератора на графі» у Інформаційна безпека та інформаційні технології: монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139.

37. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

38. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

39. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої

функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98. 2022. (**Фахове видання. Категорія «Б»**)

40. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

41. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

42. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

43. Смирнов А., Кузнецов А., Кузнецова Т. «Шумоподобные дискретные сигналы для асинхронных систем кодового разделения радиоканалов». *Радиотехника*, № 2(205), 175–183. 2021.

44. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». *CEUR Workshop Proceedings* Volume 2732, 2020, Pages 214-227.

45. Смірнов, О.А., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю.Усік П.С., «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв’язку». *Проблеми телекомунікацій*. № 1(26). С. 83-96. 2020.

46. Смирнов А.А, Кузнецов А.А., Киян А.С., Кузнецова Е.А. «Сокрытие данных на основе адресации шумоподобных сигналов». *Всеукраїнський міжвідомчий науково-технічний збірник "Радіотехніка"* – Харьків: ХНУРЕ. – 2020. – Вип. 203. – С. 38-49.

47. Смирнов А.А., Дудан А.В., Смирнова Т.В. «Формализация структуры технологического процесса электродугового напыления». *Сборник научных трудов «Актуальные вопросы машиноведения».* Объединенный институт машиностроения Национальной Академии Наук Беларуси. №9. С. 308-312, 2020.

48. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

49. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

50. А.А. Смирнов, Т.В. Смирнова, А.Н. Дреев, А.В. Дудан. «Оптимизация технологического процесса восстановления и упрочнения поверхностей с заданными характеристиками в виде облачного сервиса». Вестник Полоцкого государственного университета. Серия В, Промышленность. Прикладные науки. Республика Беларусь – 2020. – № 3. – С. 50-61.

51. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

52. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», *Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура* № 1 (11). с. 48-57, 2019.

53. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

54. Смірнов О.А., Смірнова Т.В., Солових Є.К., Дрєєв О.М., «Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей». Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

55. Смірнов О.А., Смірнова Т.В.,. Дрєєв О.М, «Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням», Системи управління, навігації та зв'язку, № 2 (54). с. 149-154, 2019.

56. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

57. Смирнов А.А., Лысенко И.А., Информационная технология проектирования тестовых наборов на основе требований к программному обеспечению, Системи управління, навігації та зв'язку. – Випуск 4 (44). – Полтава: ПолтНТУ. – 2017. – С. 112-115.

58. Смірнов О.А., Мелешко Є.В., Хох В.Д., Дослідження методів аудиту систем управління інформаційною безпекою, Системи управління, навігації та зв'язку. – Випуск 1 (41). – Полтава: ПолтНТУ. – 2017. – С. 38-42.

59. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

60. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

61. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

62. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

63. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

64. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

65. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : "Колос", 1973. – 238 с.

66. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

67. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-122.22.0026.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Підлубний О.В.			Дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж	Літ. М	Аркуш 1 / Аркушів 6
Перевірив		Доренський О.П.					
Н. Контр.		Гермак В.С.				ЦНТУ КН-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи віддаленого доступу з використанням WAN-мереж.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи віддаленого доступу з використанням WAN-мереж.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи віддаленого доступу з використанням WAN-мереж;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 126 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Доренський О.П.

Дослідження та програмна реалізація

системи віддаленого доступу з використанням WAN-мереж

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 81

Літера: РП

Кропивницький – 2022 року

Серверна частина

Файл ProjectServer.dpr - файл проекту

```
program ProjectServer;

uses
  Forms,
  UnitServer in 'UnitServer.pas' {fmMain},
  uServerTCPParser in 'uServerTCPParser.pas',
  uOptions in 'uOptions.pas' {fmOptions},
  uAbout in 'uAbout.pas' {fmAbout},
  uInfo in 'uInfo.pas' {fmInfo},
  uAllowWork in 'uAllowWork.pas' {fmAllowWork},
  uRemoteConnect in 'uRemoteConnect.pas' {fmRemoteConnect};

{$R *.RES}

begin
  Application.Initialize;
  Application.HintHidePause:=65500;
  Application.CreateForm(TfmMain, fmMain);
  Application.CreateForm(TfmAbout, fmAbout);
  Application.CreateForm(TfmInfo, fmInfo);
  Application.CreateForm(TfmAllowWork, fmAllowWork);
  Application.CreateForm(TfmRemoteConnect, fmRemoteConnect);
  Application.CreateForm(TfmOptions, fmOptions);
  Application.Run;
end.
```

Файл ProjectServer.dpr - програма-сервер

```
unit UnitServer;

interface

uses
  Windows, ShellAPI, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Menus, ComCtrls, CommCtrl, ExtCtrls, Buttons,
uTCPParser, uBaseLanServer, uServerTCPParser, uServiceController, scktcomp,
  DropTarget, DropSource, ImgList, CoolTrayIcon, ToolWin;

type
  TDisplayStatus=(dsUnknown,dsFolders,dsProcesses,dsServices,dsSearchResults);

  TfmMain = class(TForm)
    BaseLanServer: TBaseLanServer;
    TCPParserCollector: TTCPParserCollector;
    MainMenu: TMainMenu;
    mmAbout: TMenuItem;
    mmOptions: TMenuItem;
    mmConnect: TMenuItem;
    mmDisconnect: TMenuItem;
    mmExit: TMenuItem;
    mmAction: TMenuItem;
    mmLogOut: TMenuItem;
    mmShutDown: TMenuItem;
    mmReboot: TMenuItem;
    mmForceShutDown: TMenuItem;
    mmSendMessage: TMenuItem;
    mmFindInFiles: TMenuItem;
    mmAllowWork: TMenuItem;
    mmCloseCD: TMenuItem;
    mmOpenCD: TMenuItem;
    mmBroadCastClients: TMenuItem;
    mmRemoteSearch: TMenuItem;
    mmShutDownClient: TMenuItem;

    pmTray: TPopupMenu;
    pmConnect: TMenuItem;
    pmDisconnect: TMenuItem;
    pmRestore: TMenuItem;
    pmCollapse: TMenuItem;
    pmOptions: TMenuItem;
    pmExit: TMenuItem;

    pmlvDirs: TPopupMenu;
    pmCopy: TMenuItem;
    pmPrint: TMenuItem;
    pmDelete: TMenuItem;
    pmRefreshDir: TMenuItem;
    pmProperty: TMenuItem;
    pmOpen: TMenuItem;
    pmSendFiles: TMenuItem;
    pmCreateShortCut: TMenuItem;
    pmRename: TMenuItem;

    pmClients: TPopupMenu;
    pmPCInfo: TMenuItem;
    pmSendMessage: TMenuItem;
    pmOpenAsSystem: TMenuItem;
    pmClientVersion: TMenuItem;
    pmUpdateClient: TMenuItem;
    mmConnection: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
```

```
N5: TMenuItem;
N6: TMenuItem;
N7: TMenuItem;
N8: TMenuItem;
N9: TMenuItem;
N10: TMenuItem;
N11: TMenuItem;
N12: TMenuItem;
N13: TMenuItem;
N14: TMenuItem;
N15: TMenuItem;
N16: TMenuItem;

Panel1: TPanel;
Panel3: TPanel;
Panel4: TPanel;

lbNetClients: TListBox;
cbDrive: TComboBox;
lvDirs: TListView;
edDir: TEdit;

Splitter1: TSplitter;
sbVolume: TScrollBar;
SystemLargeImageList: TImageList;
ButtonImageList: TImageList;

DropFileSource: TDropFileSource;
DropFileTarget: TDropFileTarget;

OpenDialog: TOpenDialog;

TrayIcon: TCoolTrayIcon;

CoolBar1: TCoolBar;
CoolBar2: TCoolBar;
XPToolBarClients: TToolBar;
XPToolBarFileOperations: TToolBar;
XPToolBarFolderOperations: TToolBar;
XPToolBarProcessOperations: TToolBar;
XPToolBarActions: TToolBar;
XPToolBarShut: TToolBar;
tbUp: TToolButton;
tbRefresh: TToolButton;
tbCopy: TToolButton;
tbSendFiles: TToolButton;
tbDeleteFiles: TToolButton;
tbPrintFile: TToolButton;
tbScreenShot: TToolButton;
tbProcesses: TToolButton;
tbServices: TToolButton;

tbBroadcastClients: TToolButton;
tbShutdownActions: TToolButton;
tbOpenCD: TToolButton;
tbCloseCD: TToolButton;
mmUpdateClient: TMenuItem;
ClientImageList: TImageList;
N1: TMenuItem;
pmScreenShot: TMenuItem;
pmShowProcess: TMenuItem;
pmServices: TMenuItem;
N17: TMenuItem;
mmShow: TMenuItem;
mmShowFolders: TMenuItem;
mmScreenShot: TMenuItem;
mmProcesses: TMenuItem;
mmServices: TMenuItem;
N18: TMenuItem;
```

```
    mmPCInfo: TMenuItem;
    mmClientVersion: TMenuItem;
    N19: TMenuItem;
    pmActivateClient: TMenuItem;
    tbViewStyle: TToolButton;
    SystemSmallImageList: TImageList;
    ToolButton1: TToolButton;
    N20: TMenuItem;
    StatusBar1: TStatusBar;

    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormResize(Sender: TObject);

    procedure BaseLanServerChangeClients(Sender: TObject);

    procedure TCPParserCollectorAdddingParser(Sender: TObject;var TCPParser:
TCustomTCPParser);
    procedure TCPParserCollectorAddParser(Sender: TObject;
      var TCPParser: TCustomTCPParser);
    procedure TCPParserCollectorStartDataReceive(
      const Socket: TCustomWinSocket; const DataHeader: TDataHeader;
      var DataStream: TStream);



    procedure mmDisconnectClick(Sender: TObject);
    procedure mmConnectClick(Sender: TObject);
    procedure mmBroadcastClientsClick(Sender: TObject);
    procedure mmExitClick(Sender: TObject);
    procedure mmOptionsClick(Sender: TObject);
    procedure mmAboutClick(Sender: TObject);
    procedure mmLogOutClick(Sender: TObject);
    procedure mmShutDownClick(Sender: TObject);
    procedure mmRebootClick(Sender: TObject);
    procedure mmForceShutDownClick(Sender: TObject);
    procedure mmAllowWorkClick(Sender: TObject);
    procedure mmSendMessageClick(Sender: TObject);
    procedure mmHelpClick(Sender: TObject);
    procedure mmRemoteSearchClick(Sender: TObject);
    procedure mmUpdateClientClick(Sender: TObject);

    procedure pmRestoreClick(Sender: TObject);
    procedure pmCollapseClick(Sender: TObject);
    procedure pmRenameClick(Sender: TObject);
    procedure pmPropertyClick(Sender: TObject);
    procedure pmlvDirsPopup(Sender: TObject);
    procedure pmOpenAsSystemClick(Sender: TObject);
    procedure mmPCInfoClick(Sender: TObject);
    procedure mmClientVersionClick(Sender: TObject);
    procedure mmRefreshDirClick(Sender: TObject);

    procedure pmActivateClientClick(Sender: TObject);

    procedure lvDirsMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure lvDirsMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure lvDirsMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure pmOpenClick(Sender: TObject);
    procedure lvDirsKeyDown(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure lvDirsEdited(Sender: TObject; Item: TListItem;
      var S: String);
    procedure lvDirsColumnClick(Sender: TObject; Column: TListColumn);
    procedure lvDirsCompare(Sender: TObject; Item1, Item2: TListItem;
      Data: Integer; var Compare: Integer);
```

```
    procedure tbUpClick(Sender: TObject);
    procedure tbRefreshClick(Sender: TObject);
    procedure pmCopyClick(Sender: TObject);
    procedure pmSendFilesClick(Sender: TObject);
    procedure pmPrintClick(Sender: TObject);
    procedure pmDeleteClick(Sender: TObject);
    procedure mmScreenShotClick(Sender: TObject);
    procedure sbOpenCDClick(Sender: TObject);
    procedure sbCloseCDClick(Sender: TObject);
    procedure mmProcessesClick(Sender: TObject);
    procedure mmServicesClick(Sender: TObject);
    procedure sbVolumeScroll(Sender: TObject; ScrollCode: TScrollCode;
      var ScrollPos: Integer);
    procedure tbShutdownActionsClick(Sender: TObject);

    procedure cbDriveChange(Sender: TObject);

    procedure Splitter1CanResize(Sender: TObject; var NewSize: Integer;
      var Accept: Boolean);
    procedure DropFileTargetDrop(Sender: TObject; ShiftState: TShiftState;
      Point: TPoint; var Effect: Integer);
    procedure DropFileSourceDrop(Sender: TObject; DragType: TDragType;
      var ContinueDrop: Boolean);
    procedure DropFileTargetDragOver(Sender: TObject;
      ShiftState: TShiftState; Point: TPoint; var Effect: Integer);
    procedure CoolBar2Resize(Sender: TObject);
    procedure tbViewStyleClick(Sender: TObject);
    procedure N20Click(Sender: TObject);
    procedure lvDirsChange(Sender: TObject; Item: TListItem;
      Change: TItemChange);
  private
    { Private declarations }
    DirectoryIndex:integer;
    FDontShowFloppy: boolean;
    FAllCopyDirectory:string;
    FDisplayStatus: TDisplayStatus;
    FSortColumn: integer;
    FSortDirection:integer;
    FAllowFilePropertys: Boolean;
    FAllowLargeFileIcons: Boolean;

    procedure SetDontShowFloppy(const Value: boolean);
    procedure SetFAllCopyDirectory(const Value: string);
    procedure SetDisplayStatus(const Value: TDisplayStatus);
    procedure SetSortColumn(const Value: integer);
    procedure SetAllowFilePropertys(const Value: Boolean);
    procedure SetAllowLargeFileIcons(const Value: Boolean);

  public
    { Public declarations }
    Header:TDataHeader;
    Stream:TStream;
    ActiveParser:TTCPParser;

    ShowScreenShot:Boolean;
    ShowCopyProgress:boolean; // установлювати ця властивість на самому
      // парсері безглуздо - однаково, відображатися повинні тільки
      // файлові копіювання, а Custom Parser'у однаково, що він приймає

    MultiScreenShot:Boolean;

    DragFlag:boolean;

    procedure ActivateButtons(const Active:Boolean);
    procedure SetOptions(TCPParser:TTCPParser);


    procedure ParserOnGetDrives(Sender:
TObject;ErrCode:integer;ErrString:string);
```

```
    procedure ParserOnGetDirs(Sender:TObject);
    procedure ParserOnProgress(Sender:TObject);
    procedure ParserOnGetProcesses(Sender:TObject);
    procedure ParserGetProcessInfo(Sender:TObject);
    procedure ParserOnGetFile(Sender: TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnExecute(Sender: TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnPrint(Sender: TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnSendFile(Sender:
TObject;ErrCode:integer;ErrString:string);
    procedure ParserOnScreenShot(Sender:
TObject;ErrCode:integer;ErrString:string);
    procedure ParserServicesUpdate(Sender:
TObject;ErrCode:integer;ErrString:string);

    property DontShowFloppy:boolean read FDontShowFloppy write
SetDontShowFloppy;
    property AllowFilePropertys:Boolean read FAllowFilePropertys write
SetAllowFilePropertys;
    property AllowLargeFileIcons:Boolean read FAllowLargeFileIcons write
SetAllowLargeFileIcons default True;
    property AllCopyDirectory:string read FAllCopyDirectory write
SetFAllCopyDirectory;
    property DisplayStatus:TDisplayStatus read FDisplayStatus write
SetDisplayStatus;
    property SortColumn:integer read FSortColumn write SetSortColumn;
  end;


var
  fmMain: TfmMain;
  AlreadyDragging: boolean; // чи перетаскуємо із чи програми ні
                            // щоб не DropTarget не прийняв випадково
                            // те, що намагається передати DropSource
  DragPoint:TPoint;{ початкове положення курсору при старті
                       Drag&Drop файлів}

implementation

uses uConsts,
     ZLibEx,
     uOptions,
     uAbout,
     uInfo,
     uAllowWork,
     uRemoteConnect,
     uFileInfo,
     uProcessInfo;

const
  Flag = SHGFI_ADDOVERLAYS or
         SHGFI_ICON or
         SHGFI_SYSICONINDEX or
         SHGFI_USEFILEATTRIBUTES;

{$R *.DFM}

//=====================================================================
//                    загальні процедури
//=====================================================================

function GetTempDir: string;
var
  Res: DWORD;
begin
  SetLength (Result, MAX_PATH);
  Res := GetTempPath(MAX_PATH, PChar(Result));
  SetLength (Result, Res);
end;
```

```
procedure TfmMain.ActivateButtons(const Active: Boolean);
begin
  tbBroadcastClients.Enabled:=not BaseLanServer.AutoSearchClients and
BaseLanServer.Active;
  mmBroadcastClients.Enabled:=tbBroadcastClients.Enabled;

  tbUp.Enabled:=Active;
  tbRefresh.Enabled:=Active;
  tbCopy.Enabled:=Active;
  tbSendFiles.Enabled:=Active;
  tbPrintFile.Enabled:=Active;
  tbDeleteFiles.Enabled:=Active;
  tbScreenShot.Enabled:=Active;
  tbProcesses.Enabled:=Active;
  tbServices.Enabled:=Active;

  tbShutDownActions.Enabled:=Active;
  tbOpenCD.Enabled:=Active;
  tbCloseCD.Enabled:=Active;

  cbDrive.Enabled:=Active;
  edDir.Enabled:=Active;

  lvDirs.Enabled:=Active;
  sbVolume.Enabled:=Active;
 mmAction.Enabled:=True; // mmAction.Enabled:=Active;
  if not Active then
    lvDirs.Items.Clear;
end;




//=============================================================================
//                        процедури форми
//=============================================================================

procedure TfmMain.FormCreate(Sender: TObject);
var
  fi: _SHFILEINFOA;
begin
  CoInitialize(nil);

  SystemLargeImageList.Handle:=SHGetFileInfo(
    PChar(''),
    0,
    fi,
    SizeOf(TSHFileInfo),
    Flag or SHGFI_LARGEICON);

  SystemSmallImageList.Handle:=SHGetFileInfo(
    PChar(''),
    0,
    fi,
    SizeOf(TSHFileInfo),
    Flag or SHGFI_SMALLICON);

  ActivateButtons(False);

  DropFileTarget.Register(lvDirs);
  FSortDirection:=1;

  tbShutdownActions.ImageIndex:=2;
  FAllowLargeFileIcons:=lvDirs.ViewStyle=vsIcon;
end;

procedure TfmMain.FormDestroy(Sender: TObject);
begin
```

```
  CoUninitialize;
end;

procedure TfmMain.FormResize(Sender: TObject);
begin
  if Panel3.Width<425 then
   Panel1.Width:=fmMain.ClientWidth-Splitter1.Width-(425+14);
  lvDirs.Arrange(arDefault);
end;

//=============================================================================
//              процедури BaseLanServer'a
//=============================================================================

procedure TfmMain.BaseLanServerChangeClients(Sender: TObject);
begin
  lbNetClients.Items.Assign(BaseLanServer.Clients);
  if (BaseLanServer.ClientCount=0)or(ActiveParser=nil) then
    begin
      ActivateButtons(False);

    end;
end;

//=============================================================================
//                      процедури TCPParserCollector
//=============================================================================

procedure TfmMain.SetOptions(TCPParser:TTCPParser);
begin
  TCPParser.OnGetDrives:=ParserOnGetDrives;
  TCPParser.OnGetDirs:=ParserOnGetDirs;
  TCPParser.OnProgress:=ParserOnProgress;
  TCPParser.OnGetProcesses:=ParserOnGetProcesses;
  TCPParser.OnGetFile:=ParserOnGetFile;
  TCPParser.OnExecute:=ParserOnExecute;
  TCPParser.OnPrint:=ParserOnPrint;
  TCPParser.OnSendFile:=ParserOnSendFile;
  TCPParser.OnScreenShot:=ParserOnScreenShot;
  TCPParser.OnGetProcessInfo:=ParserGetProcessInfo;
  TCPParser.OnServicesUpdate:=ParserServicesUpdate;

  TCPParser.CopyDirectory:=IncludeTrailingBackSlash(AllCopyDirectory)+
     TCPParser.Socket.RemoteHost+' ['+TCPParser.Socket.RemoteAddress+']';
  TCPParser.DontShowFloppy:=DontShowFloppy;
  TCPParser.ShowProgress:=ShowCopyProgress;
  TCPParser.ShowScreenShot:=ShowScreenShot;
  TCPParser.AllowFilePropertys:=AllowFilePropertys;
  TCPParser.AllowLargeFileIcons:=AllowLargeFileIcons;
end;

procedure TfmMain.TCPParserCollectorAdddingParser(Sender: TObject;
  var TCPParser: TCustomTCPParser);
begin
  TCPParser:=TTCPParser.Create(Self);
end;

procedure TfmMain.TCPParserCollectorAddParser(Sender: TObject;
  var TCPParser: TCustomTCPParser);
begin
  SetOptions(TCPParser as TTCPParser);
end;

procedure TfmMain.TCPParserCollectorStartDataReceive(
  const Socket: TCustomWinSocket; const DataHeader: TDataHeader;
  var DataStream: TStream);
begin
  // установлювати ця властивість на самому
  // парсері безглуздо - однаково, відображатися повинні тільки
```

```
  // файлові копіювання, а Parser'у однаково, що він приймає
    ActiveParser.ShowProgress:=ShowCopyProgress and DataHeader.isFileOperation;
end;

//=====================================================================
//                          події парсерів
//=====================================================================

procedure TfmMain.ParserOnExecute(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  ShowMessage(ErrString);
end;

procedure TfmMain.ParserOnGetDirs(Sender: TObject);
var
  l:TListItem;
  i:integer;
  fi:_SHFileInfo;
  FileInfo:TFileInfo;
  TempFlag:UINT;
begin
  if not Assigned(ActiveParser) then
    exit;// незважаючи на те, що подія викликається тільки активним парсером
         // і кнопки блокуються, якщо не обраний жоден клієнт
  edDir.Text:=ActiveParser.CurrentDir;
  lvDirs.Items.BeginUpdate;
  lvDirs.Items.Clear;

  if AllowFilePropertys then
    begin
      ClientImageList.Clear;
      if AllowLargeFileIcons then
        begin
          ClientImageList.Height:=32;
          ClientImageList.Width:=32;
        end
      else
        begin
          ClientImageList.Height:=16;
          ClientImageList.Width:=16;
        end;
      lvDirs.LargeImages:=nil;// без цього при відображенні vsSmallIcon після
vsIcon
      lvDirs.SmallImages:=nil;// у всіх TListItem залишається висота 32 пікселя.
      lvDirs.LargeImages:=ClientImageList;
      lvDirs.SmallImages:=ClientImageList;
    end
  else
    begin
      lvDirs.LargeImages:=SystemLargeImageList;
      lvDirs.SmallImages:=SystemSmallImageList;
    end;

  // розподіляємо папки
  for i:=0 to ActiveParser.Directorys.Count-1 do
    begin
      l:=lvDirs.Items.Add;
      l.Caption:=ActiveParser.Directorys[i];
      if ActiveParser.AllowFilePropertys then
        begin
          FileInfo:=TFileInfo(ActiveParser.Directorys.Objects[i]);
          ClientImageList.AddIcon(FileInfo.FileIcon);
          l.ImageIndex:=i;
          l.SubItems.Add('');
          l.SubItems.Add(FileInfo.FileType);
          l.SubItems.Add(FileTimeToStr(FileInfo.LocalLastWriteTime));
        end
      else
```

```
        begin
          case lvDirs.ViewStyle of
            vsIcon:
              TempFlag:=Flag or
                        SHGFI_LARGEICON or
                        SHGFI_TYPENAME;
          else
            TempFlag:=Flag or
                      SHGFI_SMALLICON or
                      SHGFI_TYPENAME;
          end;
          SHGetFileInfo(PChar('.folder'),
                FILE_ATTRIBUTE_NORMAL,
                fi,
                SizeOf(TSHFileInfo),
                TempFlag);
          l.imageIndex:=fi.iIcon;
          l.SubItems.Add('');
          l.SubItems.Add(fi.szTypeName);
          l.SubItems.Add('невідомо');
        end;
      l.Checked:=True;
    end;

  // розподіляємо файли
  for i:=0 to ActiveParser.Files.Count-1 do
    begin
      l:=lvDirs.Items.Add;
      l.Caption:=ActiveParser.Files[i];
      if ActiveParser.AllowFilePropertys then
        begin
          FileInfo:=TFileInfo(ActiveParser.Files.Objects[i]);
          ClientImageList.AddIcon(FileInfo.FileIcon);
          l.ImageIndex:=i+ActiveParser.Directorys.Count;
          l.SubItems.Add(FileInfo.StrFileSize);
          l.SubItems.Add(FileInfo.FileType);
          l.SubItems.Add(FileTimeToStr(FileInfo.LocalLastWriteTime));
        end
      else
        begin
          case lvDirs.ViewStyle of
            vsIcon:
              TempFlag:=Flag or
                        SHGFI_LARGEICON or
                        SHGFI_TYPENAME;
          else
            TempFlag:=Flag or
                      SHGFI_SMALLICON or
                      SHGFI_TYPENAME;
          end;
          SHGetFileInfo(PChar(ActiveParser.Files[i]),
                FILE_ATTRIBUTE_NORMAL,
                fi,
                SizeOf(TSHFileInfo),
                TempFlag);

          l.imageIndex:=fi.iIcon;
          l.SubItems.Add('');
          l.SubItems.Add(fi.szTypeName);
          l.SubItems.Add('невідомо');
        end;
      l.Checked:=False;
    end;
  DisplayStatus:=dsFolders;
  lvDirs.Items.EndUpdate;
end;

procedure TfmMain.ParserOnGetDrives(Sender: TObject; ErrCode: integer;
  ErrString: string);
```

```
begin
  cbDrive.Items.Assign(ActiveParser.Drives);
  cbDrive.ItemIndex:=0;
end;

procedure TfmMain.ParserOnGetFile(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin

end;

procedure TfmMain.ParserOnGetProcesses(Sender: TObject);
var
  i:integer;
  l:TListItem;
  pi:TProcessInfo;
begin
  lvDirs.LargeImages:=nil; // при відображенні процесів
  lvDirs.SmallImages:=nil; // іконки не використовуємо

  SortColumn:=0;
  FSortDirection:=1;

  lvDirs.Items.BeginUpdate;
  lvDirs.Items.Clear;

  for i:=0 to ActiveParser.Processes.ProcessCount-1 do
    begin
      l:=lvDirs.Items.Add;
      pi:=TProcessInfo(ActiveParser.Processes.Process[i]);
      l.Caption:=pi.ProcessName;
      l.SubItems.Add(pi.ProcessDomainName+'\'+pi.ProcessUserName);
      l.SubItems.Add(IntToStr(pi.ProcessMemoryInfo.WorkingSetSize div 1024)+'
Кб');
      l.ImageIndex:=-1;
    end;

  DisplayStatus:=dsProcesses;
  lvDirs.Items.EndUpdate;
end;

procedure TfmMain.ParserOnPrint(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  ShowMessage(ErrString)
end;

procedure TfmMain.ParserOnProgress(Sender: TObject);
begin

end;

procedure TfmMain.ParserOnScreenShot(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  if not ActiveParser.ShowScreenShot then
    begin
      ActiveParser.ShowProgress:=False;

ShellExecute(Handle,'open',@ActiveParser.DataHeader.CommStr[1],nil,nil,SW_Show);
    end;
end;

procedure TfmMain.ParserOnSendFile(Sender: TObject; ErrCode: integer;
  ErrString: string);
begin
  ActiveParser.DoGetDirs;
  if errCode<>0 then
    ShowMessage(ErrString);
```

```
end;

procedure TfmMain.ParserGetProcessInfo(Sender: TObject);
var
  pi:TProcessInfo;
begin
  if Assigned(ActiveParser) then
    with fmInfo do
      begin
        pi:=TProcessInfo.Create(nil);
        pi.ReadFromStream(ActiveParser.DataStream);
        lbName.Caption:=ExtractFileName(pi.ProcessName);
        lbFullName.Caption:=pi.ProcessName;
        lbPageFaults.Caption:=IntToStr(pi.ProcessMemoryInfo.PageFaultCount);

lbPeakWorkingSetSize.Caption:=IntToStr(pi.ProcessMemoryInfo.PeakWorkingSetSize);
        lbWorkingSetSize.Caption:=IntToStr(pi.ProcessMemoryInfo.WorkingSetSize);
        lbPageFileUsage.Caption:=IntToStr(pi.ProcessMemoryInfo.PagefileUsage);

lbPeakPageFileUsage.Caption:=IntToStr(pi.ProcessMemoryInfo.PeakPagefileUsage);
        lbUserName.Caption:=pi.ProcessDomainName+'\'+pi.ProcessUserName;
        pi.Free;
        ShowModal;
      end;
end;

procedure TfmMain.ParserServicesUpdate(Sender: TObject; ErrCode: integer;
  ErrString: string);
var
  l:TListItem;
  i:integer;
  SS:TServiceStatus;
begin
  SortColumn:=0;
  FSortDirection:=1;
  lvDirs.LargeImages:=nil;
  lvDirs.SmallImages:=nil;
  lvDirs.Items.BeginUpdate;

  lvDirs.Items.Clear;
  for i:=0 to ActiveParser.Services.ServiceStatusCount-1 do
    begin
      l:=lvDirs.Items.Add;
      SS:=ActiveParser.Services[i];
      l.Caption:=SS.ServiceName;
      l.SubItems.Add(SS.DisplayName);
      l.SubItems.add(SS.Description);
      l.SubItems.add(SrvStatus[SS.DWCurrentState]);
      l.SubItems.add(SrvStartType[SS.dwStartType]);
      l.SubItems.add(SS.ServiceStartName);
      l.ImageIndex:=-1;
    end;
  DisplayStatus:=dsServices;
  lvDirs.Items.EndUpdate;
end;


//=====================================================================
//                    Процедури головного меню
//=====================================================================

// -------меню "З'єднання" ----------

procedure TfmMain.mmBroadcastClientsClick(Sender: TObject);
begin
  BaseLanServer.DoSearchClients;
end;

procedure TfmMain.mmRemoteSearchClick(Sender: TObject);
```

```
begin
  fmRemoteConnect.ShowModal;
end;

procedure TfmMain.mmConnectClick(Sender: TObject);
begin
  BaseLanServer.Active:=True;
  mmConnect.Checked:=True;
  pmConnect.Checked:=True;

end;

procedure TfmMain.mmDisconnectClick(Sender: TObject);
begin
  BaseLanServer.Active:=False;
  mmDisconnect.Checked:=True;
  pmDisconnect.Checked:=True;
  ActivateButtons(False);

end;

procedure TfmMain.mmExitClick(Sender: TObject);
begin
  fmMain.Close;
end;

// ------ меню "Дія" -----------

procedure TfmMain.sbOpenCDClick(Sender: TObject);
begin
  ActiveParser.DoOpenCD;
end;

procedure TfmMain.sbCloseCDClick(Sender: TObject);
begin
  ActiveParser.DoCloseCD;
end;

procedure TfmMain.mmSendMessageClick(Sender: TObject);
var
  fmMes:TForm;
  edCaption:TEdit;
  mmText:TMemo;
  bbOk:TButton;
  bbCancel:TButton;
begin
  if not Assigned(ActiveParser) then
    exit;
  fmMes:=TForm.Create(Self);
  fmMes.Caption:=' Введіть текст повідомлення';
  fmMes.BorderStyle:=bsSingle;
  fmMes.ClientHeight:=141;
  fmMes.ClientWidth:=291;
  fmMes.FormStyle:=fsStayOnTop;
  fmMes.Position:=poScreenCenter;

  edCaption:=TEdit.Create(fmMes);
  edCaption.Parent:=fmMes;
  edCaption.Text:='Текст заголовку';
  edCaption.Left:=0;
  edCaption.Top:=4;
  edCaption.Width:=291;

  mmText:=TMemo.Create(fmMes);
  mmText.Parent:=fmMes;
  mmText.height:=76;
  mmText.Left:=0;
  mmText.Top:=28;
  mmText.Width:=291;
```

```
  mmText.Lines.Add(' Введіть текст повідомлення');

  bbOk:=TButton.Create(fmMes);
  bbOk.Parent:=fmMes;
  bbOk.Left:=120;
  bbOk.Top:=110;
  bbOk.Caption:='Ok';
  bbOk.ModalResult:=mrOk;

  bbCancel:=TButton.Create(fmMes);
  bbCancel.Parent:=fmMes;
  bbCancel.Left:=210;
  bbCancel.Top:=110;
  bbCancel.Caption:='Відміна';
  bbCancel.ModalResult:=mrCancel;

  if fmMes.ShowModal=mrOk then
    ActiveParser.doShowMessage(edCaption.Text,mmText.Lines.Text);
  fmMes.Free;
end;

procedure TfmMain.mmUpdateClientClick(Sender: TObject);
var
  s:string;
  Filter:string;
begin
  if not Assigned(ActiveParser) then
    exit;
  s:=OpenDialog.Title;
  Filter:=OpenDialog.Filter;
  OpenDialog.Title:='Виберіть нову версію клієнта';
  OpenDialog.Filter:='Програма клієнт|svcclean.exe';
  if OpenDialog.Execute then
    ActiveParser.DoUpdateClient(OpenDialog.FileName);

  OpenDialog.Title:=s;
  OpenDialog.Filter:=Filter;
end;

// ------- підміню "Вимикання" --- -------

procedure TfmMain.mmLogOutClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoLogout(mmForceShutDown.Checked);
end;

procedure TfmMain.mmShutDownClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoShutdownPC(mmForceShutDown.Checked);
end;

procedure TfmMain.mmRebootClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoRebootPC(mmForceShutDown.Checked);
end;

procedure TfmMain.mmForceShutDownClick(Sender: TObject);
begin
  mmForceShutdown.Checked:=not mmForceShutdown.Checked;
end;

procedure TfmMain.mmAllowWorkClick(Sender: TObject);
var
  AW:TAllowWorkTime;
begin
  { дозволити роботу на клієнтському комп’ютер'ютері на х мінут}
```

```
  if assigned(ActiveParser) then
    begin
      if (fmAllowWork.ShowModal<>mrOk) then
        exit;
    end
  else
    exit;
  AW.CanAllow:=False;
  ActiveParser.AllowWork:=AW;

  AW.TimeToWork:=fmAllowWork.dtpAllowTime.DateTime;
  AW.CanAllow:=True;
  if fmAllowWork.rbReboot.Checked then
    AW.PCCloseAction:=caReboot;
  if fmAllowWork.rbShutdown.Checked then
    AW.PCCloseAction:=caShutdown;
  if fmAllowWork.rbLogout.Checked then
    AW.PCCloseAction:=caLogout;
  AW.ForceCloseApp:=fmAllowWork.cbForce.Checked;
  AW.MessCaption:=fmAllowWork.edCaption.Text;
  AW.MessText:=fmAllowWork.mmText.Text;

  if (AW.MessCaption='')or(AW.MessText='') then
    AW.NeedShowMessage:=False
  else
    AW.NeedShowMessage:=True;

  ActiveParser.AllowWork:=aw;
end;

// ------ меню "Опції" --------------

procedure TfmMain.mmOptionsClick(Sender: TObject);
begin

end;

// -------меню "Допомога" ------------

procedure TfmMain.mmHelpClick(Sender: TObject);
var
  s:string;
  HlpFile:string;
begin
  s:=ExtractFilePath(Application.ExeName)+'help';
  HlpFile:=s+'\Help.hlp';
  ShellExecute(Handle,'open',@HlpFile[1],nil,@s[1],SW_SHOW);
end;

procedure TfmMain.mmAboutClick(Sender: TObject);
begin
  fmAbout.ShowModal;
end;

//=====================================================================
//                          меню іконки в треї
//=====================================================================

procedure TfmMain.pmRestoreClick(Sender: TObject);
begin
  TrayIcon.ShowMainForm;
end;

procedure TfmMain.pmCollapseClick(Sender: TObject);
begin
  TrayIcon.HideMainForm;
end;

//=====================================================================
```

```
//                          меню списку клієнтів
//=============================================================================

procedure TfmMain.pmActivateClientClick(Sender: TObject);
begin
  if lbNetClients.ItemIndex>-1 then
    begin
      ActivateButtons(True);
      if Assigned(ActiveParser) then
        ActiveParser.Active:=False;
      ActiveParser:=TCPParserCollector.FindTCPParser(lbNetClients.ItemIndex) as
TTCPParser;
      // гарантоване знаходження активного парсера, т.до ItemIndex>-1
      ActiveParser.Active:=True;
      if ActiveParser.Drives.Count<>0 then
        begin  // якщо цей парсер уже був задіяний
               // і в нього є поточна папка
          cbDrive.Items.Assign(ActiveParser.Drives);
          cbDrive.ItemIndex:=ActiveParser.CurrentDrive;
          ActiveParser.DoGetDirs;
        end
      else   // він тільки підключився
        ActiveParser.DoGetDrives;

     end
  else
    begin // не обраний жоден із клієнтів
      ActivateButtons(False);
      ActiveParser:=Nil;
      end;

end;

procedure TfmMain.mmScreenShotClick(Sender: TObject);
begin
  if MultiScreenShot then
    begin
      ActiveParser.ShowScreenShot:=True;
      ActiveParser.StopSendScreen:=False;
      ActiveParser.DoLoadScreenShot;
      ActiveParser.DoLoadPartScreenShot;
    end
  else
    begin
      ActiveParser.ShowScreenShot:=False;
      ActiveParser.StopSendScreen:=False;
      ActiveParser.DoLoadScreenShot;
    end;
end;

procedure TfmMain.mmProcessesClick(Sender: TObject);
begin
  ActiveParser.DoGetProcessList;
end;

procedure TfmMain.mmServicesClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoGetServiceList;
end;

procedure TfmMain.mmPCInfoClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoGetPCInfo;
end;

procedure TfmMain.mmClientVersionClick(Sender: TObject);
begin
```

```
  if Assigned(ActiveParser) then
    ActiveParser.DoGetClientVersion;
end;

//=============================================================================
//               меню списку файлів/папок/процесів/сервісів
//=============================================================================

procedure TfmMain.pmlvDirsPopup(Sender: TObject);
var
  i:integer;
begin
  // забираємо функції, доступні тільки для виділених елементів
  if lvDirs.SelCount=0 then
    begin
      for i:=0 to pmlvDirs.Items.Count-1 do
        if pmlvDirs.Items.Items[i].Tag=1 then
           pmlvDirs.Items.Items[i].Enabled:=False;
    end
  else
    begin
      for i:=0 to pmlvDirs.Items.Count-1 do
        if pmlvDirs.Items.Items[i].Tag=1 then
           pmlvDirs.Items.Items[i].Enabled:=True;
      pmOpenAsSystem.Enabled:=lvDirs.Selected.ImageIndex<>DirectoryIndex;
    end;

  Case DisplayStatus of
    dsFolders:// залишаємо як їсти
               ;
    dsProcesses:// забираємо недоступне для процесів
      begin
        pmOpen.Enabled:=False;
        pmOpenAsSystem.Enabled:=False;
        pmCopy.Enabled:=False;
        pmSendFiles.Enabled:=False;
        pmPrint.Enabled:=False;
        pmCreateShortCut.Enabled:=False;
        pmRename.Enabled:=False;
      end;
    dsServices:
      begin
        pmOpen.Enabled:=False;
        pmOpenAsSystem.Enabled:=False;
        pmCopy.Enabled:=False;
        pmSendFiles.Enabled:=False;
        pmPrint.Enabled:=False;
        pmCreateShortCut.Enabled:=False;
        pmRename.Enabled:=False;
      end;
  end;
end;

procedure TfmMain.pmOpenClick(Sender: TObject);
var
   l:TListItem;
   PathFile:string;
begin
  if (lvDirs.SelCount=0)or not  Assigned(ActiveParser) then
    exit;

  l:=lvDirs.Selected;
  case DisplayStatus of
    dsFolders:
      if l.Checked then
        {directory}
        begin

ActiveParser.CurrentDir:=IncludeTrailingBackSlash(edDir.Text+l.Caption);
```

```
        end
      else{file}
        begin
          PathFile:=ActiveParser.CurrentDir+l.Caption;
          ActiveParser.DoExecFile(PathFile);
        end;
    dsProcesses:
      ActiveParser.DoGetProcessInfo(l.Caption);
    dsServices:
      ActiveParser.Services.DisplayServiceStatusFromList(
        ActiveParser.Services.FindServiceStatusID(l.Caption));
  end;
end;

procedure TfmMain.pmOpenAsSystemClick(Sender: TObject);
var
  l:TListItem;
  PathFile:string;
begin
  if lvDirs.SelCount=0 then
    exit;
  l:=lvDirs.Selected;
  if l.Checked then
    begin {directory}
      edDir.Text:=IncludeTrailingBackSlash(edDir.Text+l.Caption);
      ActiveParser.CurrentDir:=edDir.Text;
    end
  else{file}
    begin
      PathFile:=edDir.Text+l.Caption;
      ActiveParser.DoExecFileAsSystem(PathFile);
    end;
end;

procedure TfmMain.mmRefreshDirClick(Sender: TObject);
begin
  case DisplayStatus of
    dsFolders:
      tbRefresh.Click;
    dsProcesses:
      ActiveParser.DoGetProcessList;
    dsServices:
      ActiveParser.DoGetServiceList;
  end;
end;

procedure TfmMain.pmCopyClick(Sender: TObject);
var
  s:TStringList;
  i:integer;
  l:TListItem;
begin
  ActiveParser.FileOperationComplete:=False;
  s:=TStringList.Create;
  if lvDirs.SelCount>=1 then
    begin
    { проходимо файли й папки в поточній директорії}
      s.Add(ActiveParser.CurrentDir);
      for i:=lvDirs.Selected.Index to lvDirs.Items.Count-1 do
        begin
          l:=lvDirs.Items.Item[i];
          if l.Selected then
            if l.Checked then
              S.add('?'+l.Caption)
            else
              s.Add(l.Caption);
        end;
      ActiveParser.ShowProgress:=ShowCopyProgress;
      ActiveParser.DoCopyFile(s.Text);
```

```
    end;
  s.Free;
end;

procedure TfmMain.pmSendFilesClick(Sender: TObject);
var
  i:integer;
begin
  if OpenDialog.Execute then
    begin
      for i:=0 to OpenDialog.Files.Count-1 do
        ActiveParser.DoSendFile(OpenDialog.Files.Strings[i]);
      ActiveParser.DoGetDirs;
    end;
end;

procedure TfmMain.pmPrintClick(Sender: TObject);
begin
  if Assigned(lvDirs.Selected) then
    if not lvDirs.Selected.Checked then //  виділений файл
      ActiveParser.DoPrintFile(edDir.Text+'\'+lvDirs.Selected.Caption);
end;

procedure TfmMain.pmDeleteClick(Sender: TObject);
var
  i:integer;
  l:TListItem;
  s:TStringList;
begin
  Case DisplayStatus of
    dsProcesses:
      if Assigned(ActiveParser) then
        if Assigned(lvDirs.Selected) then
          ActiveParser.DoTerminateProcess(lvDirs.Selected.Caption);
    dsServices:
      if Assigned(ActiveParser) then
        if Assigned(lvDirs.Selected) then
          begin
            ActiveParser.DoDeleteService(lvDirs.Selected.Caption);
          end;
    dsFolders:
      if Assigned(lvDirs.Selected) then
        begin
          s:=TStringList.Create;
          for i:=lvDirs.Selected.Index to lvDirs.Items.Count-1 do
            begin
              l:=lvDirs.Items.Item[i];
              if l.Selected then

s.Add(IncludeTrailingBackSlash(ActiveParser.CurrentDir)+l.Caption);
            end;
          ActiveParser.DoDeleteFile(s.Text,true);
          s.Free;
        end;
  end;
end;

procedure TfmMain.pmRenameClick(Sender: TObject);
begin
  if lvDirs.SelCount<>0 then
    lvDirs.Selected.EditCaption;
end;

procedure TfmMain.pmPropertyClick(Sender: TObject);
var
  SL:TStringList;
  i:integer;
begin
  Case DisplayStatus of
```

```
    dsProcesses :
      ActiveParser.DoGetProcessInfo(lvDirs.Selected.Caption);

    dsServices :

ActiveParser.Services.DisplayServiceStatusFromList(lvDirs.Selected.Caption);

    dsFolders :
      begin
        SL:=TStringList.Create;
        for i:=lvDirs.Selected.Index to lvDirs.Items.Count-1 do
          if lvDirs.Items.Item[i].Selected then
            SL.Add(IncludeTrailingBackSlash(ActiveParser.CurrentDir)+
                   lvDirs.Items.Item[i].Caption);
        ActiveParser.DoGetFileInfo(SL.Count,SL.Text);
        SL.Free;
      end;

  end;
end;

//=====================================================================
//                          процедури Drag & Drop
//=====================================================================

procedure TfmMain.lvDirsMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  DragPoint.x:=x;
  DragPoint.y:=y;
  inherited MouseDown(button,Shift,x,y);
  AlreadyDragging:=false;
end;

procedure TfmMain.lvDirsMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var
  TempPath:string;
  i:integer;
  tmpFile:string;
begin
  inherited MouseMove(Shift,x,y);
  if (AlreadyDragging) then
    exit;
  if  not (ssLeft in Shift) or ((abs(DragPoint.X - X) <10) and
   (abs(DragPoint.Y - Y) <10)) then
   exit;  //Якщо немє вибраних файлів то вихід...
  if lvDirs.SelCount = 0 then
    exit;
  //Відаліть що небуть з  dragdrop...
  DropFileSource.Files.clear;
  TempPath := GetTempDir;
  for i := lvDirs.Selected.index to LvDirs.items.Count-1 do
    if (LvDirs.items.item[i].Selected) then
      DropFileSource.Files.Add(TempPath+LvDirs.items.item[i].caption);
  AlreadyDragging := true;

  //Файли не існують, але запускаються dragdrop...
  DropFileSource.execute;
    AlreadyDragging := false;
  //Очищення у цьому випадку...
  //Це не є обов'язковим, якщо файли успішно переміщені.
  for i := 0 to LvDirs.items.Count-1 do
    if (LvDirs.items.item[i].Selected) then
      begin
        tmpFile:=TempPath+LvDirs.items.item[i].caption;
        if fileexists(tmpFile) then
          DeleteFile(tmpFile);
      end;
```

```
end;

procedure TfmMain.lvDirsMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  DragPoint.x:=-1;
  DragPoint.y:=-1;
  AlreadyDragging:=False;
end;

procedure TfmMain.DropFileSourceDrop(Sender: TObject; DragType: TDragType;
  var ContinueDrop: Boolean);
var
  PrevCopyDir:string;
begin
  PrevCopyDir:=ActiveParser.CopyDirectory;
  ActiveParser.CopyDirectory:=ExtractFileDir(GetTempDir);
  tbCopy.Click;
  while not ActiveParser.FileOperationComplete do
    Application.ProcessMessages;

  ActiveParser.CopyDirectory:=PrevCopyDir;
end;

procedure TfmMain.DropFileTargetDragOver(Sender: TObject;
  ShiftState: TShiftState; Point: TPoint; var Effect: Integer);
begin
  if AlreadyDragging then
    Effect:=0;
end;

procedure TfmMain.DropFileTargetDrop(Sender: TObject;
  ShiftState: TShiftState; Point: TPoint; var Effect: Integer);
var
  i:integer;
begin
  if not AlreadyDragging then
    begin
      for i:=0 to DropFileTarget.Files.Count-1 do
        if FileExists(DropFileTarget.Files.Strings[i]) then
          ActiveParser.DoSendFile(DropFileTarget.Files.Strings[i]);
      ActiveParser.DoGetDirs;
    end;
end;

//=====================================================================
//                    установка властивостей
//=====================================================================

procedure TfmMain.SetDontShowFloppy(const Value: boolean);
var
  i:integer;
begin
  FDontShowFloppy := Value;
  for i:=0 to TCPParserCollector.ParserCount-1 do
    TTCPParser(TCPParserCollector.Parser[i]).DontShowFloppy:=Value;
end;

procedure TfmMain.SetFAllCopyDirectory(const Value: string);
var
  i:integer;
  TCPParser:TTCPParser;
begin
  FAllCopyDirectory := Value;
  for i:=0 to TCPParserCollector.ParserCount-1 do
    begin
      TCPParser:=TCPParserCollector.Parser[i] as TTCPParser;
      TCPParser.CopyDirectory:=IncludeTrailingBackSlash(AllCopyDirectory)+
        TCPParser.Socket.RemoteHost+' ['+
```

```
        TCPParser.Socket.RemoteAddress+']';
    end;
end;

procedure TfmMain.SetAllowFilePropertys(const Value: Boolean);
var
  i:integer;
begin
  FAllowFilePropertys := Value;
  For i:=0 to TCPParserCollector.ParserCount-1 do
    TTCPParser(TCPParserCollector.Parser[i]).AllowFilePropertys:=Value;
end;

procedure TfmMain.SetDisplayStatus(const Value: TDisplayStatus);
var
  lc:TListColumn;
begin
  if FDisplayStatus<>Value then
    begin
      FDisplayStatus := Value;
      Case FDisplayStatus of
        dsFolders:
          begin
            case lvDirs.ViewStyle of
              vsIcon:AllowLargeFileIcons:=True;
              vsSmallIcon,vsList:AllowLargeFileIcons:=False;// нічого не робимо
              vsReport:
                begin
                  AllowLargeFileIcons:=False;
                  if lvDirs.Columns.Items[0].Caption<>'Ім"я' then
                    begin
                      lvDirs.Columns.BeginUpdate;
                      lvDirs.Columns.Clear;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Ім"я';
                          AutoSize:=True;
                        end;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Розмір';
                          AutoSize:=True;
                        end;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Тип';
                          AutoSize:=True;
                        end;
                      with lvDirs.Columns.Add do
                        begin
                          Caption:='Змінений';
                          AutoSize:=True;
                        end;
                      lvDirs.Columns.EndUpdate;
                    end;
                end;
            end;
          end;

        dsProcesses:
          begin
            AllowLargeFileIcons:=False;
            lvDirs.Columns.BeginUpdate;
            lvDirs.Columns.Clear;
            With lvDirs.Columns.add do
              begin
                Caption:='Назва файлу процесу';
                AutoSize:=True;
              end;
```

```
            With lvDirs.Columns.add do
              begin
                Caption:='Ім"я користувача';
                AutoSize:=True;
              end;
            With lvDirs.Columns.add do
              begin
                Caption:='Займана пам"ять';
                AutoSize:=True;
              end;
            lvDirs.ViewStyle:=vsReport;
            lvDirs.Columns.EndUpdate;
          end;
        dsServices:
          begin
            AllowLargeFileIcons:=False;
            lvDirs.Columns.BeginUpdate;
            lvDirs.Columns.Clear;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Ім"я сервіса';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Назва';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Опис';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Стан';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Тип запуску';
            lc.AutoSize:=True;

            lc:=lvDirs.Columns.Add;
            lc.Caption:='Вхід від імені';
            lc.AutoSize:=True;

            lvDirs.ViewStyle:=vsReport;
            lvDirs.Columns.EndUpdate;
          end;
      end;
      SortColumn:=0;
      FSortDirection:=1;
    end;
end;

procedure TfmMain.SetSortColumn(const Value: integer);
begin
  if Value=FSortColumn then
    FSortDirection:=-FSortDirection
  else
    FSortDirection:=1;
  FSortColumn := Value;
end;

procedure TfmMain.SetAllowLargeFileIcons(const Value: Boolean);
var
  i:integer;
begin
  FAllowLargeFileIcons := Value;
  for i:=0 to TCPParserCollector.ParserCount-1 do
    TTCPParser(TCPParserCollector.Parser[i]).AllowLargeFileIcons:=Value;
end;
```

```
//=====================================================================
//            Всяке разне (натискання кнопок, різні події)
//=====================================================================

procedure TfmMain.Splitter1CanResize(Sender: TObject; var NewSize: Integer;
  var Accept: Boolean);
begin
  Accept:=True;
  if NewSize<150 then
    Accept:=False;
  if (fmMain.ClientWidth-NewSize)<(425+18) then
    Accept:=False;
end;

procedure TfmMain.tbUpClick(Sender: TObject);
var
  s:String;
  i:integer;
  pos:integer;
begin
  if Assigned(ActiveParser) then
    begin
      s:=ActiveParser.CurrentDir;
      pos:=-1;
      for i:=length(s)-1 downto 1 do
        if s[i]='\' then
          begin
            pos:=i+1;
            break;
          end;
      if pos=-1 then
        exit;
      Delete(s,pos,length(s)-pos+1);
      ActiveParser.CurrentDir:=s;
    end;
end;

procedure TfmMain.tbRefreshClick(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.DoGetDirs;
end;

procedure TfmMain.cbDriveChange(Sender: TObject);
begin
  if Assigned(ActiveParser) then
    ActiveParser.CurrentDrive:=cbDrive.ItemIndex;;
end;

procedure TfmMain.lvDirsKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Case Key of
    VK_DELETE:
      tbDeleteFiles.Click;
    VK_RETURN:
      lvDirs.OnDblClick(Self);
    VK_F5:
      tbRefresh.Click;
  end;
end;

procedure TfmMain.lvDirsEdited(Sender: TObject; Item: TListItem;
  var S: String);
begin
  if (DisplayStatus=dsFolders) and (s<>'') then
```

```
ActiveParser.DoRenameFile(IncludeTrailingBackslash(ActiveParser.CurrentDir)+Item
.Caption,
      IncludeTrailingBackslash(ActiveParser.CurrentDir)+s)
  else
    S:=Item.Caption;// якщо відображаються не файли, те забороняємо
перейменування
      // проста установка ReadOnly=True при відображенні сервісів і процесів
      // в SetDisplayStatus приводить до якогось незвільнення ресурсів у
програмі.
end;

procedure TfmMain.lvDirsColumnClick(Sender: TObject; Column: TListColumn);
begin
  SortColumn:=Column.Index;
  lvDirs.AlphaSort;
end;

procedure TfmMain.lvDirsCompare(Sender: TObject; Item1, Item2: TListItem;
  Data: Integer; var Compare: Integer);
var
  s1,s2:string;
  b1,b2:boolean;
begin
  if SortColumn=0 then
    begin
      s1:=item1.Caption;
      s2:=Item2.Caption;
    end
  else
    begin
      s1:=Item1.SubItems[ SortColumn-1];
      s2:=Item2.SubItems[ SortColumn-1];
    end;
  b1:=Item1.Checked;
  b2:=Item2.Checked;
  s1:=AnsiLowerCase(s1);
  s2:=AnsiLowerCase(s2);
  if b1=b2 then // якщо обидва елементи - папки, або обоє файли
    begin
      if (s1>s2) then
        Compare:=FSortDirection
      else
        if s1<s2 then
          Compare:=-FSortDirection
        else
          Compare:=0;
    end
  else
    if b2 then   // якщо другий елемент - папка
     Compare:=FSortDirection
    else
     Compare:=-FSortDirection;
end;

procedure TfmMain.CoolBar2Resize(Sender: TObject);
begin
  Panel4.Height:=CoolBar2.Height+edDir.Height+11;
end;

procedure TfmMain.sbVolumeScroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
begin
  if ScrollCode=scEndScroll then
    ActiveParser.DoChangeVolume( MaxWord-ScrollPos);
end;

procedure TfmMain.tbShutdownActionsClick(Sender: TObject);
begin
```

```
  if Assigned (ActiveParser) then
    mmLogout.Click;
end;

procedure TfmMain.tbViewStyleClick(Sender: TObject);
var
  i:integer;
begin
  if DisplayStatus=dsFolders then
    begin
      i:=integer(lvDirs.ViewStyle);
      inc(i);
      if i>3 then
        i:=0;
      if i=1 then
        i:=2;// не виходить нормально відобразити стиль vsSmallIcon,
             // довгий текст ListItem'ов налазить один на одного :(
      lvDirs.ViewStyle:=TViewStyle(i);
      AllowLargeFileIcons:=lvDirs.ViewStyle=vsIcon;
      pmRefreshDir.Click;
    end;
end;

procedure TfmMain.N20Click(Sender: TObject);
begin
fmOptions.ShowModal;
end;

procedure TfmMain.lvDirsChange(Sender: TObject; Item: TListItem;
  Change: TItemChange);
begin
StatusBar1.Panels.Text:='Підключення до '+
lbNetClients.Items.Strings[lbNetClients.ItemIndex];

end;

end.
```

Файл uServerTCPParser - програма-сервер

```
unit uServerTCPParser;

interface

uses
  Windows,
  Forms,
  uTCPParser,
  uServiceController,
  uProcessInfo,
  classes,
  graphics,
  scktcomp,
  JPe,
  ZLibEx,
  Psapi,
  extctrls;

type
  TPCCloseAction=(caShutdown,caReboot,caLogout); // тип обмеження роботи
                                                 // підлеглого комп'ютер'ютера

  TTCPParserEvent = procedure (Sender: TObject;ErrCode:integer;ErrString:string)
of object;

  TAllowWorkTime=record
    CanAllow:boolean;         // дозвіл роботи
    TimeToWork:TDateTime;     // обмеження часу роботи
    NeedShowMessage:boolean; // показувати чи ні повідомлення по закінченні часу
    MessCaption:string;       // заголовок повідомлення
    MessText:string;          // текст повідомлення
    PCCloseAction:TPCCloseAction; // вид виходу
    ForceCloseApp:Boolean;    // закривати швидко додатка, чи ні
  end;

  TTCPParser=class(TCustomTCPParser)
  private
    FActive: Boolean;         // активний чи ні (тобто потрібно активізувати
події )
    FDontShowFloppy: Boolean;// показувати чи немає флоппі в списку дисків
    FcurrentDir: string;      // поточна папка
    FCurrentDrive: integer;   // поточний диск
    FShowScreenShot:Boolean; // відображати чи ні знімок екрана

    Header:TDataHeader;       // заголовок даних
    FAllowWork: TAllowWorkTime; // структура дозволу роботи
    FCurrTime:TDateTime;      { час прийому заявки AllowWork}

    fmScreenShot:TForm;       // форма відображення скріншота

    FOnGetProcesses: TNotifyEvent;
    FOnGetProcessInfo:TNotifyEvent;
    FOnGetDirs: TNotifyEvent;
    FOnProgress: TNotifyEvent;
    FOnGetFile: TTCPParserEvent;
    FOnPrint: TTCPParserEvent;
    FOnExecute: TTCPParserEvent;
    FOnSendFile: TTCPParserEvent;
    FOnGetDrives: TTCPParserEvent;
    FOnScreenShot: TTCPParserEvent;
    FOnServicesUpdate: TTCPParserEvent;
    FAllowFilePropertys: Boolean;
    FAllowLargeFileIcons: Boolean;
```

```
    procedure ClearFolderStrings;

    function  DefineDrives:boolean; // прийняли список дисків
    procedure DefineDir;            // прийняли список папок
    procedure DefineDirEx;          // прийняли розширений список папок
    procedure DefineGetFile;        // прийняли файл
    procedure DefineFileInfo;       // прийняли властивості файлу
    function  DefineExec:string;    {підтвердження виконання}
    function  DefinePrint:string;   // підтвердження печатки файлу
    procedure DefineFileOpComplete; // підтвердження виконання копіювання
    procedure DefineProcess;        // прийняли список процесів
    procedure DefineCloseProcess;   // підтвердження закриття процесу
    procedure DefineProcessInfo;    // прийняли властивості процесу
    procedure DefinePCInfo;         // прийняли інформацію про комп'ютер'ютер
    procedure DefineSendFile;       // визначення правильності передачі файлу
    procedure DefineFullScreen;     // прийняли повний екран (відсилається
першим
                                    // при відеоскріншоті або при одиночному
пересиланні
    procedure DefinePartScreen;     // прийняли частину, що змінилася, екрана
    procedure DefineServiceList;    // прийняли список сервісів
    procedure DefineServiceStatus;  // підтвердження статусу сервісу
    procedure DefineServiceStateStep;// крок зміни статусу сервісу
    procedure Desider;               // загальна процедура рішення того, що
прийнято

    procedure SetActive(const Value: Boolean); // установка активності
компонента
    procedure SetCurrentDir(const Value: string);// установка поточної папки
    procedure SetCurrentDrive(const Value: integer);// поточного диска
    procedure SetDontShowFloppy(const Value: Boolean);// відображення флоппі

    procedure CreatefmScreenShot;   // створення форми відображення скріншота
    procedure fmScreenShotClose(Sender: TObject; var Action: TCloseAction);
    procedure fmScreenShotCanResize(Sender: TObject; var NewWidth, NewHeight:
Integer; var Resize: Boolean);
    procedure SetAllowWork(const Value: TAllowWorkTime);

    procedure DefineClientVersion;

  protected

    procedure StartDataReceive;override;
    procedure CompleteDataReceive;override;
  public
    { Public declarations }
    CopyDirectory:string;

    MyJpeg:TJPEGImage;
    MyImage:TBitMap;
    img:TBitmap;

    StopSendScreen:boolean;

    Drives:TStringList;
    Directorys:TStringList;
    Files:TStringList;
    Processes:TProcessManager;
    Services:TServiceController;
    ProcessInfo:PROCESS_MEMORY_COUNTERS;

    FileOperationComplete:boolean;

    {посилаємо запити}
    procedure DoGetDrives;
    procedure DoGetDirs;
    procedure DoExecFile(f:string);
    procedure DoExecFileAsSystem(f:string);
```

```
    procedure DoCopyFile(f:string);
    procedure DoPrintFile(f:string);
    procedure DoRenameFile(OldFileName,NewFileName:string);
    procedure DoSendFile(sFileName:String);
    procedure DoDeleteFile(sFileName:String; RefreshDir:Boolean);

    procedure DoGetProcessList;
    procedure DoTerminateProcess(p:string);
    procedure DoGetProcessInfo(p:string);
    procedure DoGetFileInfo(const Count:integer;const FileNames: string);

    procedure DoGetServiceList;

    procedure DoChangeServiceStatus(Sender:TObject;
      OldStatus,NewStatus:TCustomServiceStatus; var AllowChange:Boolean);
    procedure DoChangeServiceState(Sender:TCustomServiceStatus;
      Command:TServiceStateCommand; var AllowChange:boolean);
    procedure DoDeleteService(ServiceName:string);


    procedure DoGetPCInfo;
    procedure DoShutdownPC(Force:Boolean);
    procedure DoRebootPC(Force:Boolean);
    procedure DoLogout(Force:Boolean);

    procedure DoOpenCD;
    procedure DoCloseCD;
    procedure DoChangeVolume(NewVolume:Integer);

    procedure DoLoadScreenShot;
    procedure DoLoadPartScreenShot;
    procedure DoStopSendScreen;

    procedure DoShowMessage(const MessCaption,MessText:string);

    procedure DoGetClientVersion;
    procedure DoUpdateClient(FileName:string);
    {Наступні 2 ф- не можна перенести в меншу видимість, тому що
     у будь-який момент може знадобитися пересилання яких-небудь
     даних (наприклад, однократних), для яких заклад
     спадкоємця недоцільно}
    function DoSendData(var Header:TDataHeader;var
Stream:TStream):boolean;override;
    function DoSendImmediateData(var Header:TDataHeader;var
Stream:TStream):boolean;override;

    constructor Create(const aSender:TObject);override;
    destructor destroy;override;

    function CheckIdle(AddTime: Integer):Boolean;override;// перевизначаємо, щоб
    // використовувати при установці часу клієнта


    property Active:Boolean read FActive write SetActive;
    property DontShowFloppy:Boolean read FDontShowFloppy write
SetDontShowFloppy;
    property CurrentDrive:integer read FCurrentDrive write SetCurrentDrive;
    property CurrentDir:string read FcurrentDir write SetCurrentDir;
    property AllowFilePropertys:Boolean read FAllowFilePropertys write
FAllowFilePropertys;
            // дозволяє при запиті вмісту папки завантаження властивостей
файлів включаючи іконки.
    property AllowLargeFileIcons:Boolean read FAllowLargeFileIcons write
FAllowLargeFileIcons;
    property ShowScreenShot:boolean read FShowScreenShot write FShowScreenShot;

    property AllowWork:TAllowWorkTime read FAllowWork write SetAllowWork;

    { події }
```

```
    property OnGetDrives:TTCPParserEvent read FOnGetDrives write FOnGetDrives;
    property OnGetDirs:TNotifyEvent read FOnGetDirs write FOnGetDirs;
    property OnProgress:TNotifyEvent read FOnProgress write FOnProgress;
    property OnGetProcesses:TNotifyEvent read FOnGetProcesses write
FOnGetProcesses;
    property OnGetProcessInfo:TNotifyEvent read FOnGetProcessInfo write
FOnGetProcessInfo;
    property OnGetFile:TTCPParserEvent read FOnGetFile write FOnGetFile;
    property OnExecute:TTCPParserEvent read FOnExecute write FOnExecute;
    property OnPrint:TTCPParserEvent read FOnPrint write FOnPrint;
    property OnSendFile:TTCPParserEvent read FOnSendFile write FOnSendFile;
    property OnScreenShot:TTCPParserEvent read FOnScreenShot write
FOnScreenShot;


    property OnServicesUpdate:TTCPParserEvent read FOnServicesUpdate write
FOnServicesUpdate;
  end;

implementation

uses uConsts,
     Controls,
     ShellAPI,
     SysUtils,
     FileCtrl,
     uFileInfo;

{ TTCPParser }

//=============================================================
//                 створення й знищення
//=============================================================

constructor TTCPParser.Create(const aSender: TObject);
begin
  inherited;
  FOnGetDrives:=nil;
  FOnGetDirs:=nil;
  FOnProgress:=nil;
  FOnGetProcesses:=nil;
  FOnGetFile:=nil;
  FOnExecute:=nil;
  FOnPrint:=nil;
  FOnSendFile:=nil;
  FOnScreenShot:=nil;

  Drives:=TStringList.Create;
  Directorys:=TStringList.Create;
  Files:=TStringList.Create;
  Processes:=TProcessManager.Create(nil);
  Drives.Sorted:=True;
  Directorys.Sorted:=True;
  Files.Sorted:=True;

  Services:=TServiceController.Create(nil);
  Services.OnChangeServiceStatus:=DoChangeServiceStatus;
  Services.OnChangeServiceState:=DoChangeServiceState;



  MyJPEG:=TJPEGImage.Create;
  MyImage:=TBitMap.Create;
  MyImage.PixelFormat:=pf32bit;
  img:=TBitMap.Create;

  FActive:=False;

  CopyDirectory:='c:\Downloads';
```

```
end;

destructor TTCPParser.destroy;
begin
  ClearFolderStrings;
  Drives.Free;
  Directorys.Free;
  Files.Free;
  Processes.Free;
  Services.Free;
  MyJPEG.Free;
  MyImage.Free;
  img.Free;
  if Assigned(fmScreenShot) then
    if fmScreenShot.Visible then
      DoStopSendScreen;
  fmScreenShot.Free;
  inherited;
end;

//============================================================
//       визначення додаткових властивостей, створення форм
//============================================================

procedure TTCPParser.CreatefmScreenShot;
begin
  fmScreenShot:=TForm.Create(nil);
  fmScreenShot.ClientWidth:=640;
  fmScreenShot.ClientHeight:=480;
  fmScreenShot.DoubleBuffered:=True;
  fmScreenShot.BorderIcons:=[biSystemMenu];
  fmScreenShot.Caption:='Скріншот
'+Socket.RemoteHost+'['+Socket.RemoteAddress+']';
  fmScreenShot.FormStyle:=fsNormal;
  fmScreenShot.OnClose:=fmScreenShotClose;
  fmScreenShot.OnCanResize:=fmScreenShotCanResize;
  fmScreenShot.Show;
end;

procedure TTCPParser.fmScreenShotClose(Sender: TObject;
  var Action: TCloseAction);
begin
  DoStopSendScreen;
  Action:=caHide;
end;

procedure TTCPParser.fmScreenShotCanResize(Sender: TObject; var NewWidth,
  NewHeight: Integer; var Resize: Boolean);
var
  AspectRatio:Extended;
begin
  Resize:=False;
  if Assigned(MyImage) then
    begin
      AspectRatio:=MyImage.Height/MyImage.Width;
      NewHeight:=Round(NewWidth*AspectRatio);
      Resize:=True;
    end;
end;

procedure TTCPParser.SetActive(const Value: Boolean);
begin
  FActive := Value;
end;

procedure TTCPParser.SetCurrentDir(const Value: string);
begin
  FcurrentDir := Value;
  DoGetDirs;
```

```
end;

procedure TTCPParser.SetCurrentDrive(const Value: integer);
begin
  FCurrentDrive := Value;
  CurrentDir:=copy(Drives.Strings[Value],1,3);
end;

procedure TTCPParser.SetDontShowFloppy(const Value: Boolean);
begin
  FDontShowFloppy := Value;
end;

procedure TTCPParser.SetAllowWork(const Value: TAllowWorkTime);
begin
  FAllowWork := Value;
  FCurrTime:=Now();
end;

//============================================================
//    доповнення операцій по прийому/передачі, перекриття
//                          методів
//============================================================

function TTCPParser.CheckIdle(AddTime: Integer): Boolean;
var
  DateTime:TDateTime;
begin
  if FAllowWork.CanAllow then
    begin
      DateTime:=Now;
      if ( DateTime-FCurrTime)>=FAllowWork.TimeToWork then
        begin
          if FAllowWork.NeedShowMessage then ;
            DoShowMessage(FAllowWork.MessCaption,FAllowWork.MessText);
          FAllowWork.CanAllow:=False;
          case FAllowWork.PCCloseAction of
            caShutDown:DoShutdownPC(FAllowWork.ForceCloseApp);
            caReboot:DoRebootPC(FAllowWork.ForceCloseApp);
            caLogout:DoLogout(FAllowWork.ForceCloseApp);
          end;
        end;
    end;
  Result:=inherited CheckIdle(AddTime);
end;

procedure TTCPParser.StartDataReceive;
begin
    if DataHeader.isFileOperation then
      begin
        ForceDirectories(ExtractFileDir(DataHeader.CommStr));
        DataStream:=TFileStream.Create(DataHeader.CommStr,fmCreate);
        ShowProgress:=True;
      end
    else
      begin
        DataStream:=TMemoryStream.Create;
        ShowProgress:=False;
      end;
end;

procedure TTCPParser.CompleteDataReceive;
var
  DecompressStream:TMemoryStream;
begin
  if DataHeader.isArchive and (DataStream.Size<>0)then
    begin
      DataStream.Position:=0;
      DecompressStream:=TMemoryStream.Create;
```

```
      ZDecompressStream(DataStream,DecompressStream);
      DecompressStream.Position:=0;
      DataStream.Size:=0;
      DataStream.CopyFrom(DecompressStream,0);
      DataStream.Position:=0;
      DecompressStream.Free;
    end;
  Desider;
end;

function TTCPParser.DoSendData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
    end;
  Result:=inherited DoSendData(Header,Stream);
end;

function TTCPParser.DoSendImmediateData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
    end;
  Result:=inherited DoSendImmediateData(Header,Stream);
end;

//=========================================================
//      прийом інформації, розшифровка даних
//=========================================================

procedure TTCPParser.Desider;
begin
  case DataHeader.Command of
    GetDrives:DefineDrives;
    GetDirs:DefineDir;
    GetFiles:DefineGetFile;
    GetFileInfo:DefineFileInfo;
    SendFiles:DefineSendFile;
    GetDirsEx: DefineDirEx;

    {операції над файлами (виконання, печать...}
    ExecuteFile:DefineExec;
    PrintFile:DefinePrint;
    //SendToClipBoard=220;
    //MakeShortCut=230;
    //FindInFiles=240;
    FileOpComplete:DefineFileOpComplete;
```

```
    { операції над системними функціями}
    ViewProcess:DefineProcess;
    CloseProcess:DefineCloseProcess;
    GetProcessInfo:DefineProcessInfo;

    GetPCInfo:DefinePCInfo;

    { пересилання екрана}
    SendScreen:DefineFullScreen;
    SendPartScreen:if not StopSendScreen then
                     DefinePartScreen;

    { версія програми}
    ClientVersionInfo:DefineClientVersion;
    //GetUpdateVersion=2010;

    GetServiceList:DefineServiceList;

    SetServiceStatus,SetServiceState:DefineServiceStatus;
    SetServiceStateStep: DefineServiceStateStep;

  end;
end;

{========== інформація про диски, папки,файли==========}

function TTCPParser.DefineDrives: boolean;
var
   i:integer;
   Part:String;
  procedure DelFloppy;
  begin
    if Drives.Strings[0][1]='A' then
      Drives.Delete(0);
    if Drives.Strings[0][1]='B' then
      Drives.Delete(0);
  end;
begin
  { повернення імен дисків в DataStream,
  збережених як TStringList.SaveToStream}
  Drives.Clear;
  Drives.LoadFromStream(DataStream);
  {назви всіх дисків через "}
  if Drives.Count=0 then
    Result:=False
  else
    begin
      if FDontShowFloppy then
        DelFloppy;
      FCurrentDrive:=0;
      FCurrentDir:=copy(Drives.Strings[0],1,3);//FCurrentDrive;
      DoGetDirs;
      Result:=True;
    end;
  if Result then
    begin
      i:=0;
      Part:=Error_None;
    end
  else
    begin
      i:=1;
      Part:=Error_Get_Drives;
    end;
  if FActive and assigned(FOnGetDrives) then
    FOnGetDrives(Self,i,Part);
end;

procedure TTCPParser.ClearFolderStrings;
```

```
var
  i:integer;
begin
  Directorys.BeginUpdate;
  Files.BeginUpdate;
  for i:=Directorys.Count-1 downto 0 do
    begin
      Directorys.Objects[i].Free;
      Directorys.Objects[i]:=nil;
    end;

  for i:=Files.Count-1 downto 0 do
    begin
      Files.Objects[i].Free;
      Files.Objects[i]:=nil;
    end;
  Directorys.Clear;
  Files.Clear;

  Directorys.EndUpdate;
  Files.EndUpdate;
end;

procedure TTCPParser.DefineDir;
var
  i:integer;
  FilesFolders:TStringList;
begin
  ClearFolderStrings; // очистили старі списки файлів і папок
  Directorys.BeginUpdate;
  Files.BeginUpdate;

  FilesFolders:=TStringList.Create;

  FilesFolders.LoadFromStream(DataStream);

  for i:=0 to FilesFolders.Count-1 do
    if Length(FilesFolders[i])>0 then
      if FilesFolders[i][1]='?' then
        Directorys.Add(Copy(FilesFolders[i],2,Length(FilesFolders[i])-1))
      else
        Files.Add(FilesFolders[i]);

  Directorys.EndUpdate;
  Files.EndUpdate;

  FilesFolders.Free;
  if FActive and assigned(FOnGetDirs) then
    FOnGetDirs(Self);
end;

procedure TTCPParser.DefineDirEx;
var
  FileInfo:TFileInfo;
begin
  ClearFolderStrings;

  Directorys.BeginUpdate;
  Files.BeginUpdate;

  While DataStream.Position<DataStream.Size do
    begin
      FileInfo:=TFileInfo.Create;
      FileInfo.LoadFromStream(DataStream);
      if (FileInfo.FileAttributes and FILE_ATTRIBUTE_DIRECTORY)<>0 then
        // папка
        Directorys.AddObject(
          ExtractFileName(FileInfo.FileName),
          FileInfo)
```

```
      else
        // файл
        Files.AddObject(
          ExtractFileName(FileInfo.FileName),
          FileInfo);
    end;

  Directorys.EndUpdate;
  Files.EndUpdate;
  if FActive and assigned(FOnGetDirs) then
    FOnGetDirs(Self);
end;

procedure TTCPParser.DefineGetFile;
begin

end;

procedure TTCPParser.DefineSendFile;
var
  f:TFileStream;
  i,size:integer;
  Err:string;
begin
  f:=TFileStream.Create(DataHeader.CommStr,fmOpenRead);
  size:=f.Size;
  f.Free;
  i:=DataHeader.Unknown;
  if i=size then
    begin
      Err:=Error_None;
      i:=0;
    end
  else
    begin
      Err:=Error_Send_File;
      i:=1;
    end;
  if FActive and assigned(FOnSendFile) then
    FOnSendFile(Self,i,Err);
end;

procedure TTCPParser.DefineFileInfo;
var
  fi:TFileInfo;
begin
  fi:=TFileInfo.Create;
  fi.LoadFromStream(DataStream);
  fi.ShowInfo;
  fi.Free;
end;

{========== файлові операції (копіювання, etc)==========}

procedure TTCPParser.DefineFileOpComplete;
begin
  FileOperationComplete:=True;
  if FActive and Assigned(FOnGetFile) then
    FOnGetFile(Self,0,Error_None);
end;

function TTCPParser.DefinePrint: string;
var
   Err:integer;
   s:string;
begin
     Err:=DataHeader.Unknown;
     s:='Файл успішно відправлений на печать !';
     case Err of
```

```
          0:s:='На віддаленому комп’ютер'ютері мало пам'яті';
          ERROR_FILE_NOT_FOUND:s:='Не вдалося знайти файл';
          ERROR_PATH_NOT_FOUND:s:='Не знайдено шлях до файлу';
          ERROR_BAD_FORMAT:s:='файл невідомого формату';
     end;
     Result:=s;
     if FActive and assigned(FOnPrint) then
        FOnPrint(Self,Err,Result);
end;

function TTCPParser.DefineExec: string;
var
  Err:integer;
  s:string;
begin
  Err:=DataHeader.Unknown;
  case Err of
    0,997:s:='Файл успішно запущений !';
    ERROR_FILE_NOT_FOUND:s:='Не вдалося знайти файл';
    ERROR_PATH_NOT_FOUND:s:='Не знайдений шлях до файлу';
    ERROR_BAD_FORMAT:s:='файл невідомого формату';
    SE_ERR_ACCESSDENIED:s:='доступ до файлу заборонений';
    SE_ERR_DLLNOTFOUND:s:='необхідна DLL не знайдена';
    SE_ERR_NOASSOC:s:='жодне додаток не зіставлений із цим файлом';
    ERROR_SECTOR_NOT_FOUND:s:='не знайдений шлях до команди для запуску
додатка';
    ERROR_BAD_COMMAND:s:='не знайдена команда запуску додатка';
    ERROR_INVALID_STARTING_CODESEG:s:='не вдалося запустити додаток';
  else
    s:='Відбулася невідома помилка на віддаленому комп’ютер'ютері
'+IntToStr(DataHeader.Unknown);
  end;
  Result:=s;
  if FActive and Assigned(FOnExecute) then
    FOnExecute(self,err,Result);
end;


{===============процеси====================}

procedure TTCPParser.DefineProcess;
begin
  Processes.LoadFromStream(Datastream);
  if FActive and Assigned(FOnGetProcesses) then
    FOnGetProcesses(Self);
end;

procedure TTCPParser.DefineCloseProcess;
var
   s:string;
begin
  s:=DataHeader.CommStr;
  if s='Process Closed' then
    begin
      Sleep(1000);
      DoGetProcessList;
    end;
end;

procedure TTCPParser.DefineProcessInfo;
begin
  if not Active then
    exit;
  if FActive and Assigned(FOnGetProcessInfo) then
    FOnGetProcessInfo(Self);
end;
```

```
{=========пересилання екрана============================}

procedure TTCPParser.DefineFullScreen;
begin
  if ShowScreenShot then
    begin
      DataStream.Seek(0,soFromBeginning);
      MyJPEG.LoadFromStream(DataStream);
      MyImage.Width:=MyJPEG.Width;
      MyImage.Height:=MyJPEG.Height;
      MyImage.Assign(MyJPEG);
      if not Assigned(fmScreenShot) then
        CreatefmScreenShot;

StretchBLT(fmScreenShot.Canvas.Handle,0,0,fmScreenShot.ClientWidth,fmScreenShot.
ClientHeight,MyImage.Canvas.Handle,
        0,0,MyImage.Width,MyImage.Height,SRCCopy);
      fmScreenShot.Visible:=True; // не забирати !!! форма створюється один
      // раз і потім ховається. Якщо її не показувати назад, то ми не
      // побачимо видеоскріншот ще раз.
    end;
  if FActive and assigned(FOnScreenShot) then
    FOnScreenShot(Self,0,Error_None);
end;

procedure TTCPParser.DefinePartScreen;
var
  s:string;
  mcomp,mdecomp:TMemoryStream;
  xpos,ypos,horLines,vertlines,partsize:integer;
begin
  if not StopSendScreen then
    DoLoadPartScreenShot;
  if (DataStream=nil)or(DataStream.Size=0) then
    exit;
  DataStream.Position:=0;
  SetLength(s,79);
  mcomp:=TMemoryStream.Create;
  mdecomp:=TMemoryStream.Create;
  while DataStream.Position<DataStream.Size do
    begin
      mcomp.Clear;
      mdecomp.Clear;
      DataStream.Read(s[1],79);
      //s:=Format('%15d %15d %15d %15d
%15d',[xpos*PartImg.Width,yPos*PartImg.Height,PartImg.Width,PartImg.Height,Compr
essStream.Size]);
      xpos:=StrToInt(Copy(s,1,15));
      ypos:=StrToInt(copy(s,17,15));
      HorLines:=StrToInt(Copy(s,33,15));
      VertLines:=StrToInt(Copy(s,49,15));
      PartSize:=StrToInt(Copy(s,65,15));

      img.Width:=HorLines;
      img.Height:=VertLines;
      mcomp.CopyFrom(DataStream,PartSize);
      mcomp.Position:=0;
      ZDecompressStream(mcomp,mdecomp);
      mdecomp.Position:=0;
      img.LoadFromStream(mdecomp);

bitblt(myImage.Canvas.Handle,xpos,ypos,horLines,VertLines,img.Canvas.Handle,0,0,
SRCCopy);
    end;
  mcomp.Free;
  mdecomp.Free;
  { а отут уже в MyImage - повністю оновлене зображення}
  if FShowScreenShot then
        begin
```

```
          if not Assigned(fmScreenShot) then
             CreatefmScreenShot;

StretchBLT(fmScreenShot.Canvas.Handle,0,0,fmScreenShot.ClientWidth,fmScreenShot.
ClientHeight,MyImage.Canvas.Handle,
            0,0,MyImage.Width,MyImage.Height,SRCCopy);
          //fmScreenShot.Visible:=True;// не забирати !!! форма створюється один
            // раз і потім ховається. Якщо її не показувати назад, то ми не
            // побачимо видеоскріншот ще раз.
        end;
  if FActive and assigned(FOnScreenShot) then
    FOnScreenShot(Self,0,Error_None);
end;

{=======================система=======================}

procedure TTCPParser.DefinePCInfo;
begin

end;

procedure TTCPParser.DefineClientVersion;
begin
  MessageBox(0,@DataHeader.CommStr[1],'Версія клієнта',MB_OK);
end;

procedure TTCPParser.DefineServiceList;
begin
  Services.LoadServicesStatusFromStream(DataStream);
  if FActive and Assigned(FOnServicesUpdate) then
    FOnServicesUpdate(Self,0,'');
end;

procedure TTCPParser.DefineServiceStatus;
var
  ServiceStatus:TServiceStatus;
begin
  ServiceStatus:=TServiceStatus.Create(Services);
  ServiceStatus.LoadFromStream(DataStream);
  Services.SetServiceStatusFromList(ServiceStatus.ServiceName,ServiceStatus);
  ServiceStatus.Free;
  ServiceStatus:=Services.FindServiceStatusFromList(DataHeader.CommStr);
  if assigned(ServiceStatus.fmServiceStatus) then
    begin
      ServiceStatus.fmServiceStatus.pbStateSetting.hide;
      ServiceStatus.fmServiceStatus.Enabled:=True;
    end;
  if FActive and Assigned(FOnServicesUpdate) then
    FOnServicesUpdate(Self,0,'');
end;

procedure TTCPParser.DefineServiceStateStep;
var
  ServiceStatus:TServiceStatus;
begin
  ServiceStatus:=Services.FindServiceStatusFromList(DataHeader.CommStr);
  ServiceStatus.fmServiceStatus.pbStateSetting.Visible:=True;
  ServiceStatus.fmServiceStatus.pbStateSetting.Position:=DataHeader.Unknown;
end;


//============================================================
//              відправлення запитів
//============================================================

procedure TTCPParser.DoGetDrives;
var
  Stream:TStream;
begin
```

```
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetDrives;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendImmediateData(Header,Stream);
end;

procedure TTCPParser.DoGetDirs;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;

  if FAllowFilePropertys then
    Header.Command:=GetDirsEx
  else
    Header.Command:=GetDirs;

  Header.Unknown:=0;
  Header.Unknown2:=FAllowLargeFileIcons;
  Header.CommStr:=FCurrentDir;
  Header.ReturnStr:=' ';
  Stream:=Nil;
  DoSendImmediateData(Header,Stream);
end;

procedure TTCPParser.DoCopyFile(f: string);
var
  Stream:TStream;
begin
  { sFileName - TStringList.Text де [0] - поточна папка, кожний
  наступний елемент повний шлях до файлу або '?'папці,
  підлягаючій копіюванню}
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetFiles;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=IncludeTrailingBackslash(CopyDirectory);

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendFile(sFileName: String);
var
  Stream:TStream;
begin

  Header.isFileOperation:=True;
  Header.isArchive:=False;
  Header.Command:=SendFiles;

Header.CommStr:=IncludeTrailingBackSlash(CurrentDir)+ExtractFileName(sFileName);
  Header.ReturnStr:=' ';

  try
    Stream:=TFileStream.Create(sFileName,fmOpenRead);
    DoSendData(Header,Stream);
  except
    Stream:=nil;
  end;
end;

procedure TTCPParser.DoDeleteFile(sFileName: String; RefreshDir: Boolean);
```

```
var
  Stream:TStream;
begin
  { sFileName - TStringList.Text, де елементи - повні імена
  файлів}
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=DeleteFiles;
  Header.Unknown:=0;
  Header.CommStr:=sFileName;
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
  if RefreshDir then
    DoGetDirs;
end;

procedure TTCPParser.DoRenameFile(OldFileName, NewFileName: string);
var
  stream:TStream;
begin
  if (OldFileName<>'')and(NewFileName<>'') then
    begin
      Stream:=nil;
      Header.Command:=ChangeFileName;
      Header.isFileOperation:=False;
      Header.isArchive:=False;
      Header.CommStr:=OldFileName;
      Header.ReturnStr:=NewFileName;
      DoSendData(Header,Stream);
    end;
end;

procedure TTCPParser.DoExecFile(f: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ExecuteFile;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=' ';

  Stream:=Nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoExecFileAsSystem(f: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ExecuteFileAsSystem;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=' ';

  Stream:=Nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoPrintFile(f: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
```

```
  Header.isArchive:=False;
  Header.Command:=PrintFile;
  Header.Unknown:=0;
  Header.CommStr:=f;
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoChangeVolume(NewVolume: Integer);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SetVolume;
  Header.Unknown:=NewVolume;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,stream);
end;

procedure TTCPParser.DoOpenCD;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=OpenCD;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoCloseCD;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=CloseCD;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetProcessList;
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ViewProcess;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoLoadPartScreenShot;
```

```
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SendPartScreen;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoLoadScreenShot;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SendScreen;
  Header.Unknown:=0;
  Header.Unknown2:=not ShowScreenshot;
  Header.CommStr:=' ';

Header.ReturnStr:=IncludeTrailingBackSlash(CopyDirectory)+'ClientScreenShot.jpg'
;
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoStopSendScreen;
var
  Stream:TStream;
begin
  StopSendScreen:=True;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=StopSendPartScreen;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoRebootPC(Force: Boolean);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=RebootPC;
  Header.Unknown2:=Force;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoShutdownPC(Force: Boolean);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ShutdownPC;
  Header.Unknown2:=Force;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
```

```
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoTerminateProcess(p: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=CloseProcess;
  Header.Unknown:=0;
  Header.CommStr:=p;
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetProcessInfo(p: string);
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetProcessInfo;
  Header.CommStr:=p;
  Header.ReturnStr:=' ';
  Stream:=nil;
  DoSendData(header,Stream);
end;

procedure TTCPParser.DoLogout(Force:Boolean);
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=LogOut;
  Header.Unknown2:=Force;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoShowMessage(const MessCaption, MessText: string);
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ShowUserMessage;
  Header.CommStr:=MessCaption;
  Header.ReturnStr:=MessText;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetPCInfo;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetPCInfo;
  Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  Stream:=Nil;
```

```
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetFileInfo(const Count:integer;const FileNames: string);
var
  S:TStream;
begin
  {FileNames - імена файлів і папок, збережені як TStringList.Text}
  s:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=GetFileInfo;
  Header.Unknown:=Count;
  Header.CommStr:=FileNames;
  Header.ReturnStr:=' ';
  DoSendData(Header,S);
end;

procedure TTCPParser.DoGetClientVersion;
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ClientVersionInfo;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoUpdateClient(FileName: string);
var
  Stream:TStream;
begin
  try
    Stream:=TFilestream.Create(FileName,fmOpenRead);
  except
    MessageBox(0,' Не вдалося відкрити файл',' Помилка відновлення !',MB_OK);
    exit;
  end;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=GetUpdateVersion;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoChangeServiceState(Sender: TCustomServiceStatus;
  Command: TServiceStateCommand; var AllowChange: boolean);
var
  Stream:TStream;
  ServiceStatus:TServiceStatus;
begin
  AllowChange:=False;
  Stream:=nil;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=SetServiceState;
  Header.Unknown:=Integer(Command);
  Header.CommStr:=Sender.ServiceName;
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
  ServiceStatus:=Services.FindServiceStatusFromList(Header.CommStr);
  if assigned(ServiceStatus.fmServiceStatus) then
    ServiceStatus.fmServiceStatus.Enabled:=False;
end;
```

```
procedure TTCPParser.DoChangeServiceStatus(Sender:TObject;
  OldStatus,NewStatus:TCustomServiceStatus;var AllowChange:Boolean);
var
  Stream:TStream;
begin
  AllowChange:=False;
  Stream:=TMemoryStream.Create;
  NewStatus.SaveToStream(Stream);
  Stream.Position:=0;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=SetServiceStatus;
  Header.CommStr:=OldStatus.ServiceName;
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoGetServiceList;
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=false;
  Header.isArchive:=False;
  Header.Command:=GetServiceList;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoDeleteService(ServiceName: string);
var
  Stream:TStream;
begin
  Stream:=nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=SetDeleteService;
  Header.CommStr:=ServiceName;
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;
end.
```

Файл uOptions.pas - вікно зміни параметрів програми

```
unit uOptions;

interface

uses
  Windows,
  Classes,
  Controls,
  Forms,
  Dialogs,
  Spin,
  Graphics,
  StdCtrls,
  IniFiles,
  Registry,
  GHIPEdit;

const
  General='General';
  Clients='Clients';
  Fonts='Fonts';
  DownloadDir='DownloadDir';

type
  TfmOptions = class(TForm)
    Label1: TLabel;
    Label4: TLabel;

    gbGeneral:     TGroupBox;
    gbLanSettings: TGroupBox;

    cbAutoSearchClients: TCheckBox;
    cbWinStart:          TCheckBox;
    cbActivate:          TCheckBox;
    cbTrayIcon:          TCheckBox;
    cbFloppyShow:        TCheckBox;
    cbShowScreenShot:    TCheckBox;
    cbShowProgress:      TCheckBox;
    cbAllowClientIcons:  TCheckBox;
    cbDefaultSettings:   TCheckBox;

    SESpeed:     TSpinEdit;
    meIPMask:    TGHIPEdit;
    bbOk:     TButton;
    bbCancel: TButton;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    edDirectory: TEdit;
    bbDir: TButton;
    FontDialog: TFontDialog;

    procedure cbTrayIconClick(Sender: TObject);
    procedure cbFloppyShowClick(Sender: TObject);
    procedure cbShowScreenShotClick(Sender: TObject);
    procedure cbShowProgressClick(Sender: TObject);
    procedure cbAutoSearchClientsClick(Sender: TObject);
    procedure SESpeedChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure bbDirClick(Sender: TObject);
    procedure bbFontClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure bbOkClick(Sender: TObject);
    procedure cbAllowClientIconsClick(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    procedure ApplySettings;
    procedure LoadSettings;
    procedure SaveSettings;
  public
    { Public declarations }
    Options:TIniFile;
  end;

var
  fmOptions: TfmOptions;

implementation

uses FileCtrl,
     SysUtils,
     UnitServer,
     uAllowWork,
     uInfo,
     uRemoteConnect;

{$R *.DFM}

procedure TfmOptions.cbTrayIconClick(Sender: TObject);
begin
     fmMain.TrayIcon.MinimizeToTray:=cbTrayIcon.Checked;
     fmMain.TrayIcon.IconVisible:=cbTrayIcon.Checked;
end;

procedure TfmOptions.cbFloppyShowClick(Sender: TObject);
begin
  fmMain.DontShowFloppy:=cbFloppyShow.Checked;
end;

procedure TfmOptions.cbShowScreenShotClick(Sender: TObject);
begin
  fmMain.MultiScreenShot:=cbShowScreenShot.Checked;
end;

procedure TfmOptions.cbShowProgressClick(Sender: TObject);
begin
  fmMain.ShowCopyProgress:=cbShowProgress.Checked;
end;

procedure TfmOptions.cbAutoSearchClientsClick(Sender: TObject);
begin
  seSpeed.Enabled:=cbAutoSearchClients.Checked;
  fmMain.BaseLanServer.AutoSearchClients:=cbAutoSearchClients.Checked;
  fmMain.tbBroadcastClients.Enabled:=not cbAutoSearchClients.Checked and
fmMain.BaseLanServer.Active;
end;

procedure TfmOptions.SESpeedChange(Sender: TObject);
begin
  fmMain.BaseLanServer.RefreshRate:=SESpeed.Value*1000;
end;

procedure TfmOptions.FormCreate(Sender: TObject);
begin
  Options:=TIniFile.Create(ExtractFilePath(Application.ExeName)+'Options.ini');
  LoadSettings;
  ApplySettings;
end;

procedure TfmOptions.bbDirClick(Sender: TObject);
var
  s:WideString;
```

```
  Dir:string;
begin
  s:=   '';
  Dir:= '';
  if SelectDirectory('Укажіть папку для збереження',s,Dir) then
    edDirectory.Text:=Dir;
end;

procedure TfmOptions.bbFontClick(Sender: TObject);
begin
  FontDialog.Execute;
end;

procedure TfmOptions.FormDestroy(Sender: TObject);
begin
  Options.Free;
end;

procedure TfmOptions.ApplySettings;
var
  r:TRegistry;
begin
  r:=TRegistry.Create;
  r.RootKey:=HKEY_CURRENT_USER;
  r.OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',True);

  if cbWinStart.Checked then
    r.WriteString('',Application.ExeName)
  else
    r.DeleteValue('');

  r.CloseKey;
  r.Free;

  if cbActivate.Checked then
    fmMain.mmconnect.Click;

  fmMain.TrayIcon.MinimizeToTray:=cbTrayIcon.Checked;
  fmMain.TrayIcon.IconVisible:=cbTrayIcon.Checked;

  fmMain.DontShowFloppy:=cbFloppyShow.Checked;

  fmMain.ShowScreenShot:=cbShowScreenShot.Checked;

  fmMain.ShowCopyProgress:=cbShowProgress.Checked;

  fmMain.AllowFilePropertys:=cbAllowClientIcons.Checked;

  fmMain.BaseLanServer.RefreshRate:=SESpeed.Value*1000;
  fmMain.BaseLanServer.AutoSearchClients:=cbAutoSearchClients.Checked;
  fmMain.BaseLanServer.IPMask:=meIPMask.Text;

  if Trim(edDirectory.Text)='' then
    edDirectory.Text:='C:\Downloads';
  fmMain.AllCopyDirectory:=edDirectory.Text;

  if not DirectoryExists(fmMain.AllCopyDirectory) then
    ForceDirectories(fmMain.AllCopyDirectory);

  fmMain.Font.Assign(FontDialog.Font);
  fmAllowWork.Font.Assign(FontDialog.Font);
  fmInfo.Font.Assign(FontDialog.Font);
  fmOptions.Font.Assign(FontDialog.Font);
  fmRemoteConnect.Font.Assign(FontDialog.Font);

  if cbDefaultSettings.Checked then
    SaveSettings;
end;
```

```
procedure TfmOptions.LoadSettings;
begin
  cbWinStart.Checked          := Options.ReadBool(General,'WinStart',False);
  cbActivate.Checked          := Options.ReadBool(General,'Activate',True);
  cbTrayIcon.Checked          := Options.ReadBool(General,'TrayIcon',True);
  cbFloppyShow.Checked        := Options.ReadBool(General,'FloppyShow',False);
  cbShowScreenShot.Checked    :=
Options.ReadBool(General,'VideoScreenShot',True);
  cbShowProgress.Checked      := Options.ReadBool(General,'ShowProgress',True);
  cbAllowClientIcons.Checked  :=
Options.ReadBool(General,'AllowClientIcons',False);
  cbAutoSearchClients.Checked :=
Options.ReadBool(Clients,'AutoSearchClients',False);

  seSpeed.Value               := Options.ReadInteger(Clients,'RefreshSpeed',30);
  meIPMask.Text               :=
Options.ReadString(Clients,'IPMask','255.255.255.255');


  edDirectory.Text            :=
Options.ReadString(DownloadDir,'Directory','c:\downloads');

  FontDialog.Font.Charset     :=
Options.ReadInteger(Fonts,'Charset',DEFAULT_CHARSET);
  FontDialog.Font.Color       :=
Options.ReadInteger(Fonts,'Color',clWindowText);
  FontDialog.Font.Name        := Options.ReadString(Fonts,'FontName','MS Sans
Serif');
  FontDialog.Font.Size        := Options.ReadInteger(Fonts,'FontSize',8);
end;

procedure TfmOptions.SaveSettings;
begin
  Options.WriteBool (General,'WinStart',cbWinStart.Checked);
  Options.WriteBool (General,'Activate',cbActivate.Checked);
  Options.WriteBool (General,'TrayIcon',cbTrayIcon.Checked);
  Options.WriteBool (General,'FloppyShow',cbFloppyShow.Checked);
  Options.WriteBool (General,'VideoScreenShot',cbShowScreenShot.Checked);
  Options.WriteBool (General,'ShowProgress',cbShowProgress.Checked);
  Options.WriteBool (General,'AllowClientIcons',cbAllowClientIcons.Checked);

  Options.WriteBool   (Clients,'AutoSearchClients',cbAutoSearchClients.Checked);
  Options.WriteInteger(Clients,'RefreshSpeed',seSpeed.Value);
  Options.WriteString (Clients,'IPMask',meIPMask.Text);


  Options.WriteString(DownloadDir,'Directory',edDirectory.Text);

  Options.WriteInteger(Fonts,'Charset',FontDialog.Font.Charset);
  Options.WriteInteger(Fonts,'Color',FontDialog.Font.Color);
  Options.WriteString (Fonts,'FontName',FontDialog.Font.Name);
  Options.WriteInteger(Fonts,'FontSize',FontDialog.Font.Size);
end;

procedure TfmOptions.bbOkClick(Sender: TObject);
begin
  ApplySettings;
end;

procedure TfmOptions.cbAllowClientIconsClick(Sender: TObject);
begin
  fmMain.AllowFilePropertys:=cbAllowClientIcons.Checked;
end;

procedure TfmOptions.FormShow(Sender: TObject);
begin
  cbDefaultSettings.Checked:=False;
end;
end.
```

Клієнтська частина

Файл svcclean.dpr - файл проекту

```
program svcclean;

uses
  uSVC in 'uSVC.pas' {AllSCManager: TService},
  uClientTCPParser in 'uClientTCPParser.pas';

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TAllSCManager, AllSCManager);
  Application.Run;
end.
```

Файл uClientTCPParser.pas - програма-клієнт

```
unit uClientTCPParser;

interface
uses
  Windows,
  Forms,
  Graphics,
  Dialogs,
  uTCPParser,
  classes,
  JPe,
  ZLibEx,
  uServiceController,
  uProcessInfo;

const
     PI_NOUI=1;
     AppPath='SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\';

type
  // початкові оголошення структур, які я не знайшов
  //у стандартних модулях
  WTS_INFO_CLASS=(WTSInitialProgram,
                  WTSApplicationName,
                  WTSWorkingDirectory,
                  WTSOEMId,
                  WTSSessionId,
                  WTSUserName,
                  WTSWinStationName,
                  WTSDomainName,
                  WTSConnectState,
                  WTSClientBuildNumber,
                  WTSClientName,
                  WTSClientDirectory,
                  WTSClientProduct,
                  WTSClientHardwareId,
                  WTSClientAddress,
                  WTSClientDisplay,
                  WTSClientProtocolType);

  _WTS_CONNECTSTATE_CLASS=(WTSActive,
                  WTSConnected,
                  WTSConnectQuery,
                  WTSShadow,
                  WTSDisconnected,
                  WTSIdle,
                  WTSListen,
                  WTSReset,
                  WTSDown,
                  WTSInit);

  SessionInfo=^_WTS_SESSION_INFO;
  _WTS_SESSION_INFO=record
    SessionId:DWord;
    pWinStationName:PChar;
    State:_WTS_CONNECTSTATE_CLASS;
  end;

  MySessionInfo=array of SessionInfo;
  ppSessionInfo=^MySessionInfo;

  PProfileInfo = ^TProfileInfo;
  TProfileInfo = packed record
    dwSize:        DWORD ;
```

```
    dwFlags:       DWORD;
    lpUserName:    PAnsiChar;
    lpProfilePath: PAnsiChar;
    lpDefaultPath: PAnsiChar;
    lpServerName:  PAnsiChar;
    lpPolicyPath:  PAnsiChar;
    hProfile:      THandle;
  end;

  TTCPParserEvent = procedure (Sender: TObject;ErrCode:integer;ErrString:string)
of object;

  TTCPParser=class(TCustomTCPParser)
  private
    CanSendScreen:Boolean;
    {порівняння частин екрана (старої й нової) для виключення з передачі
однакових}
    function BitmapsEqual(var Bmp1, Bmp2: TBitmap): Boolean;
    {знаходження файлу, що виконується, відповідального за запуск файлу в
UserFileName}
    function FindExec(const h:HKEY;const UserFileName: string; var command:
string): boolean;
    { запуск файлу із сервісу під поточним користувачем}
    function RunFile(var h:THandle;AppName, FileName: string):Boolean;
    { одержання прав для перезавантаження/вимикання}
    function GrantShutdownAccess: Boolean;
    { копіювання зміненої частини екрана в старе зображення для відновлення }
    function CopyImgToPart(yPos: word; const Source: tbitmap;
      var PartImg: TBitMap): Boolean;
    { вирішує, що потрібно зробити із прийнятим DataStream (у предку) на основі
DataHeader}
    procedure Desider;
    { додає в потік для передачі змінену частину екрана й інфо про неї}
    function DoSend(yPos: integer; var PartImg: TBitMap): Boolean;

    procedure ChangeServiceStatus(Sender:TObject;
      OldStatus,NewStatus:TCustomServiceStatus;
      var AllowChange:Boolean);
    procedure
ChangeServiceState(Sender:TCustomServiceStatus;Command:TServiceStateCommand; var
AllowChange:boolean);
    procedure ChangeServiceStateStep( Sender: TCustomServiceStatus; Step:
integer);
  protected
    procedure StartDataReceive;override;
    procedure CompleteDataReceive;override;
  public
    Header:TDataHeader; // заголовок даних на передачу

    ProcessList:TProcessManager;// зберігає список процесів комп’ютер'ютера
    divider:integer;        // визначає, на скільки частин ділимо екран
                            // для порівняння й передачі вроздріб
    img1:TBitMap;           // "старе" зображення
    img2:TBitMap;           // "нове" зображення екрана
    img:TBitMap;
    PartImg1:TBitMap;       // частина "нового" зображення
    PartImg2:TBitMap;       // частина "старого" зображення
    Jp:TJPEGImage;          // повний Screenshot, що не відноситься до
попередньої,
                            // використовується тільки для пересилання повного
екрана
    Rect:TRect;
    DC:HDC;
    {transport streams}
    CompressStream, DStream:TStream;// потоки для пересилання частин зображення
    PartStream:TMemoryStream;

    PIDArray: array [0..1023] of DWORD;// відносяться до списку процесів
    PIDW:array[0..1023] of DWORD;
```

```
    Services:TServiceController;

    { перекриття методу для архівування даних при необхідності}
    function DoSendData(var Header:TDataHeader;var
Stream:TStream):boolean;override;
    function DoSendImmediateData(var Header:TDataHeader;var
Stream:TStream):boolean;override;

    constructor Create(const aSender:TObject);override;
    destructor destroy;override;

    function  DoSendText(s:string):boolean;
    procedure DoSendDir;
    procedure DoSendDirsEx;
    procedure DoSendDrives;
    procedure DoExecFile;
    procedure DoExecFileAsSystem;
    procedure DoSendFile;
    procedure DoPrintFile;
    procedure DoRenameFile;
    procedure DoSendToClipBoard;
    procedure sFileOpComplete;
    procedure DoShutdownPC;
    procedure DoLogOut;
    procedure DoSendProcess;
    procedure DoSendProcessInfo;
    procedure DoGetPCInfo;
    procedure DoCloseProcess;
    procedure DoReceiveFile;
    procedure DoDeleteFile;
    procedure DoSetVolume;

    procedure DoSendFileInfo;
    procedure DoSetFileInfo;

    procedure DoSendScreen;
    procedure DoSendPartScreen;

    procedure DoSendClientVersionInfo;
    procedure DoUpdate;

    procedure DoSendServiceList;
    procedure DoSetServiceStatus;
    procedure DoSetServiceState;
    procedure DoDeleteService;
  end;

    function  LoadUserProfile(Token: THandle; var ProfileInfo: TProfileInfo):
bool; stdcall; external 'Userenv.dll';
  function  UnloadUserProfile(Token :THandle; Profile: THandle): bool; stdcall;
external 'Userenv.dll';

  function
RegOpenUserClassesRoot(hToken:THANDLE;dwOptions:DWORD;samDesired:REGSAM;phkResul
t:PHKey):LongWord;stdcall;external 'advapi32.dll';

  function  WTSGetActiveConsoleSessionId:DWord;stdcall;external 'kernel32.dll';
  function
WTSQuerySessionInformation(hServer:THandle;SessionId:DWord;WTSInfoClass:WTS_INFO
_CLASS;ppBuffer:PChar;pBytesReturned:PDword):Bool;stdcall;external
'wtsapi32.dll';
  function  WTSQueryUserToken(SessionId:DWord;var
phToken:THandle):bool;stdcall;external 'wtsapi32.dll';
  procedure WTSFreeMemory(p:pointer);stdcall;external 'wtsapi32.dll';
implementation

uses uConsts,
     ShellAPI,
```

```
    SysUtils,
    mmSystem,
    Registry,
    PSApi,
    FileCtrl,
    uFileInfo;
var

  constdivider:integer;

{ TTCPParser }

//=============================================================================
//                          створення й видалення
//=============================================================================

constructor TTCPParser.Create(const aSender: TObject);
begin
  inherited;
  CoInitialize(nil);
  Services:=TServiceController.Create(nil);
  Services.OnChangeServiceStatus:=ChangeServiceStatus;
  Services.OnChangeServiceState:=ChangeServiceState;
  Services.OnStateChangeStep:=ChangeServiceStateStep;

  ProcessList:=TProcessManager.Create(nil);
  dc:=GetDC(GetDesktopWindow);
  Jp:=TJpegImage.Create;
  jp.Performance:=jpBestSpeed;
  jp.PixelFormat:=jf24Bit;
  jp.CompressionQuality:=100;

  Img1:=TBitMap.Create;
  img1.PixelFormat:=pf4bit;

  img2:=TBitMap.Create;
  img2.PixelFormat:=pf4bit;

  PartImg2:=TBitMap.Create;
  PartImg2.PixelFormat:=pf4bit;

  PartImg1:=TBitMap.Create;
  PartImg1.PixelFormat:=pf4bit;

  img:=TBitMap.Create;
  img.PixelFormat:=pf32bit;
  {підготовка транспортних потоків для компресії й
  зберігання частин переданого скріншота}
  CompressStream:=TMemoryStream.Create;
  PartStream:=TMemoryStream.Create;
end;

destructor TTCPParser.destroy;
begin
  FreeAndNil(Services);
  FreeAndNil(ProcessList);

  FreeAndNil(Jp);
  FreeAndNil(img1);
  FreeAndNil(img2);
  FreeAndNil(PartImg1);
  FreeAndNil(PartImg2);
  FreeAndNil(img);

  ReleaseDC(0,DC);
  FreeAndNil(CompressStream);
  FreeAndNil(PartStream);
  CoUninitialize;
  inherited;
```

```
end;

//=============================================================================
//     перекриття методів предка, у тому числі абстрактних
//=============================================================================


procedure TTCPParser.StartDataReceive;
begin
  ShowProgress:=False;
  if DataHeader.isFileOperation then
    begin
      ForceDirectories(ExtractFilePath(DataHeader.CommStr));
      DataStream:=TFileStream.Create(DataHeader.CommStr,fmCreate);
    end
  else
    DataStream:=TMemoryStream.Create;
end;

procedure TTCPParser.CompleteDataReceive;
var
  DecompressStream:TStream;
begin
  if DataHeader.isArchive then
    begin
      DataStream.Position:=0;
      DecompressStream:=TMemoryStream.Create;
      ZDecompressStream(DataStream,DecompressStream);
      DataStream.Size:=0;
      DecompressStream.Position:=0;
      DataStream.CopyFrom(DecompressStream,DecompressStream.Size);
      DecompressStream.Free;
    end;
  Desider;
end;


function TTCPParser.DoSendData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Assigned(Stream) and Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
    end;
  Result:=inherited DoSendData(Header,Stream);
end;

function TTCPParser.DoSendImmediateData(var Header: TDataHeader;
  var Stream: TStream): boolean;
var
  CompressedStream:TMemoryStream;
begin
  if Assigned(Stream) and Header.isArchive then
    begin
      CompressedStream:=TMemoryStream.Create;
      ZCompressStream(Stream,CompressedStream,zcDefault);
      CompressedStream.Position:=0;
      Stream.Size:=0;
      Stream.CopyFrom(CompressedStream,0);
      CompressedStream.Free;
      Stream.Seek(0,soFromBeginning);
```

```
    end;
  Result:=inherited DoSendImmediateData(Header,Stream);
end;

//=============================================================================
//                              обробка даних
//=============================================================================

procedure TTCPParser.Desider;
begin
  case DataHeader.Command of
    { FileSystem operations}
    GetDrives:DoSendDrives;
    GetDirs:DoSendDir;
    GetFiles:DoSendFile; { клієнт передає, сервер приймає}
    GetFileInfo:DoSendFileInfo;
    SetFileInfo:DoSetFileInfo;
    GetDirsEx:DoSendDirsEx;

    SendFiles:DoReceiveFile; { сервер передає, клієнт приймає}
    DeleteFiles:DoDeleteFile;

    //GetDirToStringList=25;


    {операції над файлами (виконання, печать...}
    ExecuteFile:DoExecFile;
    ExecuteFileAsSystem:DoExecFileAsSystem;
    ChangeFileName:DoRenameFile;
    PrintFile:DoPrintFile;
    SendToClipBoard:DoSendToClipBoard;
    //MakeShortCut=230;
    //FindInFiles=240;
    //FileOpComplete=250;


ShowUserMessage:MessageBox(0,@DataHeader.ReturnStr[1],@DataHeader.CommStr[1],MB_
OK);


    { операції над системними функціями}
    ViewProcess:DoSendProcess;
    CloseProcess:DoCloseProcess;
    GetProcessInfo:DoSendProcessInfo;

    GetPCInfo:DoGetPCInfo;

    OpenCD:mciSendString('Set cdaudio Door Open Wait', nil, 0, 0);
    CloseCD:mciSendString('Set cdaudio Door Closed Wait', nil, 0, 0);

    ShutdownPC,RebootPC,ForceShutdownPC:DoShutdownPC;
    LogOut: DoLogOut;

    SetVolume:DoSetVolume;
    //GetVolume;


    { пересилання екрана}
    SendScreen:DoSendScreen;
    SendPartScreen:DoSendPartScreen;
    StopSendPartScreen: CanSendScreen:=False;

    { версія програми}
    ClientVersionInfo:DoSendClientVersionInfo;
    GetUpdateVersion:DoUpdate;

    GetServiceList:DoSendServiceList;
    SetServiceStatus:DoSetServiceStatus;
    SetServiceState:DoSetServiceState;
```

```
    SetDeleteService: DoDeleteService;
  end;
end;

function TTCPParser.DoSendText(s: string): boolean;
var
   Stream:TStream;
begin
     { викликається звичайно при пересиланні інформації про підлеглий
комп'ютер'ютер
      - як то: диски, папки, процеси etc...}
     Stream:=nil;
     if s<>'' then
       begin
         Stream:=TMemoryStream.Create;
         Stream.Write(s[1],Length(s));
         Stream.Seek(0,soFromBeginning);
       end;
     Result:=DoSendImmediateData(Header,Stream);
end;

//=============================================================================
//                           робота з файловою системою
//=============================================================================
procedure TTCPParser.sFileOpComplete;
var
  Stream:TStream;
begin
  Stream:=Nil;
  Header.isFileOperation:=False;
  Header.Command:=FileOpComplete;
  Header.isArchive:=False;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendDrives;
const
  DriveTypes:array[0..6] of string=('Невідомий','Не існує','Знімний диск',
     'Локальний','Мережний',' CD-Rom','Віртуальний');
var
   i,j,DriveType:integer;
   Sl:TStringList;
   Str:TStream;
   s:string;
   p:PChar;
begin
     { визначається кількість і букви дисків, що є присутні у системі}
     { і відправляється на головний комп'ютер }
     Sl:=TStringList.Create;
     getMem(p,500);
     i:=GetLogicalDriveStrings(500,p);
     s:='';
     for j:=0 to i-1 do
         begin
              if (p[j]<>#0) and (p[j]<>'\') then
                s:=s+p[j];
              if p[j]='\' then
                begin
                  s:=s+'\';
                  DriveType:=GetDriveType(@s[1]);
                  s:=s+'['+DriveTypes[DriveType]+']';
                  Sl.Add(s);
                  s:='';
                end;
         end;
     Str:=TMemoryStream.Create;
     Sl.SaveToStream(Str);
```

```
     FreeMem(p);
     Sl.Free;
     Str.Seek(0,soFromBeginning);
     Header.isFileOperation:=False;
     Header.isArchive:=True;
     Header.Command:=GetDrives;
     Header.CommStr:=' ';
     Header.ReturnStr:=' ';
     DoSendImmediateData(Header,Str);
end;

procedure TTCPParser.DoSendDir;
var
   attr:integer;
   SearchRec:TSearchRec;
   DosError:integer;
   Dir:string;
   FilesFolders:TStringList;
   Stream:TStream;
begin
     { сканування й передача підпапок і файлів зазначеної папки}
     attr:=faAnyFile;
     Dir:=DataHeader.CommStr;
     Dir:=IncludeTrailingBackSlash(dir)+'*.*';
     FilesFolders:=TStringList.Create;
     DosError:=FindFirst(dir,attr,SearchRec);
     while DosError=0 do{скануємо папки}
           begin
                if ((SearchRec.Attr and faDirectory)<>0)and (SearchRec.Name[1]
<> '.') then
                   //s:=s+'"?'+SearchRec.Name+'"'
                   FilesFolders.Add('?'+SearchRec.Name)
                else
                  if (SearchRec.Attr and faDirectory)=0 then
                    //s:=s+'"'+SearchRec.Name+'"';
                    FilesFolders.Add(SearchRec.Name);
                DosError:=FindNext(SearchRec);
           end;
     FindClose(SearchRec);
     Stream:=TMemoryStream.Create;

     FilesFolders.SaveToStream(Stream);
     Stream.Position:=0;

     Header.isFileOperation:=False;
     Header.isArchive:=True;
     Header.Command:=GetDirs;
     Header.CommStr:=DataHeader.CommStr;
     Header.ReturnStr:=' ';

     DoSendData(Header,Stream);

     FilesFolders.Free;

     //DoSendText(s);
end;

procedure TTCPParser.DoSendFile;
var
  Sl:TStringList;
  ReturnDir,RelativeDir:string;
  i:integer;
  function SendFile(f:string):boolean;
  {сюди передаємо повне ім'я файлу( як воно існує на цьому комп'ютері)
   обчислюємо відносне(яке повинне прийти на комп'ютер замовника}
  var
    TempFileName:string;
    TempFs:TStream;
  begin
```

```
TempFileName:=ExtractRelativePath(RelativeDir,ExtractFilePath(f))+ExtractFileNam
e(f);
      Header.Command:=GetFiles;
      Header.isFileOperation:=True;
      Header.isArchive:=False;
      Header.CommStr:=IncludeTrailingBackSlash(ReturnDir)+TempFileName;
      Header.ReturnStr:=' ';
      try
        TempFs:=TFileStream.Create(f,fmOpenRead);
      except
        Result:=False;
        exit;
      end;
      TempFs.Seek(0,soFromBeginning);
      Result:=DoSendData(Header,TempFs);
  end;

  function SendDirectory(DirName:string):boolean;
  {сюди передаємо повне ім'я папки}
  var
    iIndex:integer;
    SearchRec:TSearchRec;
    sxFileName:String;
  begin
    Result:=False;
    DirName:=DirName+'*.*';
    iIndex := FindFirst(DirName, faAnyFile, SearchRec);
    While iIndex=0 do
      begin
        sxFileName := ExtractFilePath(DirName)+SearchRec.Name;
        if (SearchRec.Attr and faDirectory) = faDirectory then
           begin
             if (SearchRec.Name <> '' ) and (SearchRec.Name <> '.')
               and (SearchRec.Name <> '..') then
                SendDirectory(IncludeTrailingBackSlash(sxFileName))
           end
        else
          SendFile(sxFileName);
        iIndex := FindNext(SearchRec);
      end;
    FindClose(SearchRec);
  end;
begin
  Sl:=TStringList.Create;
  Sl.Text:=DataHeader.CommStr;
  ReturnDir:=IncludeTrailingBackSlash(DataHeader.ReturnStr);
  RelativeDir:=IncludeTrailingBackSlash(Sl[0]);
    for i:=1 to Sl.Count-1 do
      if Sl.Strings[i][1]='?' then
      SendDirectory(RelativeDir+copy(Sl.Strings[i],2,Length(Sl.Strings[i])-
1)+'\')
      else
        SendFile(RelativeDir+Sl.Strings[i]);
    Sl.Free;
    sFileOpComplete;
end;

procedure TTCPParser.DoReceiveFile;
begin

end;

procedure TTCPParser.DoDeleteFile;
var
  Sl:TStringList;
  i:integer;
  lpFileOp:TSHFileOpStruct;
  procedure DelDir(DirectoryName:string);
```

```
   var
      iIndex : Integer;
      SearchRec : TSearchRec;
      sxFileName : String;
   begin
         DirectoryName:=IncludeTrailingBackslash(DirectoryName)+ '*.*';
         iIndex := FindFirst(DirectoryName, faAnyFile, SearchRec);
         while iIndex = 0 do
               begin
                    sxFileName := ExtractFilePath(DirectoryName)+SearchRec.Name;
                    if (SearchRec.Attr and faDirectory) = faDirectory then
                      begin
                        if (SearchRec.Name <> '' ) and (SearchRec.Name <> '.')
                         and (SearchRec.Name <> '..') then
                            DelDir(sxFileName);
                      end
                    else
                      begin
                        SetFileAttributes(PChar(SL[i]),FILE_ATTRIBUTE_NORMAL);
                        Windows.DeleteFile(PChar(SL[i]));
                      end;
                    iIndex := FindNext(SearchRec);
               end;
         FindClose(SearchRec);
         RemoveDir(ExtractFileDir(DirectoryName));
   end;
begin
  Sl:=TStringList.Create;
  Sl.Text:=DataHeader.CommStr;

  {for i:=0 to SL.Count-1 do
    if SL[i][1]='?' then
      DelDir(Copy(SL[i],2,Length(SL[i])-1))
    else
      begin
        SetFileAttributes(PChar(SL[i]),FILE_ATTRIBUTE_NORMAL);
        Windows.DeleteFile(PChar(SL[i]));
      end;   }

  for i:=0 to SL.Count-1 do
    begin
      lpFileOp.Wnd:=0;
      lpFileOp.wFunc:=FO_DELETE;
      GetMem(lpFileOp.pFrom, MAX_PATH+2);
      FillChar(lpFileOp.pFrom[0],MAX_PATH+1,0);
      StrPCopy(lpFileOp.pFrom,SL[i]);
      lpFileOp.pTo:=nil;
      lpFileOp.fFlags:=FOF_NOCONFIRMATION or FOF_NOERRORUI or FOF_SILENT;
      lpFileOp.fAnyOperationsAborted:=False;
      lpFileOp.hNameMappings:=nil;
      lpFileOp.lpszProgressTitle:=nil;
      SHFileOperation(lpFileOp);
      FreeMem(lpFileOp.pFrom);
    end;;
  Sl.Free;
end;

procedure TTCPParser.DoRenameFile;
begin
  RenameFile(DataHeader.CommStr,DataHeader.ReturnStr);
  DataHeader.CommStr:=ExtractFilePath(DataHeader.CommStr);
  DoSendDir;
end;
```

```
//=============================================================================
//                    робота із процесами, запуск файлів
//=============================================================================

function TTCPParser.FindExec(const h: HKEY; const UserFileName: string;
  var command: string): boolean;
var
   r:TRegistry;
   UserFileDir,FileExt,AppDefault:string;
   Comm:PChar;
begin
     { для не-exe-файлів шукаємо файл, що за замовчуванням працює з ним}
     Result:=False;

     UserFileDir:=ExtractFileDir(UserFileName);
     GetMem(comm,Max_Path);
     SetLastError(0);
     if FindExecutable(@UserFileName[1],@UserFileDir[1],Comm)>32 then
        begin
             Command:=comm;
             Result:=True;
             FreeMem(comm);
             exit;
        end;
     FreeMem(comm);
     {не знайшли розширення, шукаємо ручками
     тобто стандартна функція не змогла визначити, який файл
     повинен працювати із цим розширенням
     тому беремо отриманий хендл HKEY_CLASSES_ROOT і шукаємо в ньому}
     r:=TRegistry.Create(KEY_READ);
     r.RootKey:=h;

     FileExt:=ExtractFileExt(UserFileName);
     if r.KeyExists(FileExt) then
        begin
             r.OpenKey(FileExt,False);
             AppDefault:=r.ReadString('');
             r.CloseKey;
             if not r.KeyExists(AppDefault+'\shell') then
                begin
                     SetLastError(ERROR_SECTOR_NOT_FOUND);
                     r.Free;
                     exit;
                end;
             r.OpenKey(AppDefault+'\shell',false);
             command:='open';//r.ReadString('');
             if not r.KeyExists(command+'\command') then
                begin
                     SetLastError(ERROR_BAD_COMMAND);
                     r.Free;
                     exit;
                end;
             r.OpenKey(command+'\command',false);
             command:=r.ReadString('');
             if command[1]='"' then
                begin
                     delete(command,1,1);
                     command:=Copy(command,1,pos('"',command)-1);
                end;
        end
     else
         begin
              Result:=False;
              SetLastError(SE_ERR_NOASSOC);
         end;
     r.Free;
end;
```

```
function TTCPParser.RunFile(var h: THandle; AppName, FileName: string):Boolean;
var
   DefaultDir:PChar;
   s:TStartupInfo;
   p:TProcessInformation;
   ProfileInfo:TProfileInfo;
   UserName:Pchar;
   Pr:PDword;
   r:TRegistry;
   OldPath:PChar;
   Env:String;
begin
     {усе - довідалися, який додаток відповідально за запуск цього файлу
      відповідно - можна запускати}
     //GetMem(UserName,Max_Path);
     GetMem(pr,SizeOf(DWord));

WTSQuerySessionInformation(0,WTSGetActiveConsoleSessionId,WTSUserName,@UserName,
pr);
     ProfileInfo.dwSize:=SizeOf(ProfileInfo);
     ProfileInfo.dwFlags:=PI_NOUI;
     ProfileInfo.lpUserName:=UserName;
     ProfileInfo.lpProfilePath:=nil;
     ProfileInfo.lpDefaultPath:=nil;
     ProfileInfo.lpServerName:=nil;
     ProfileInfo.lpPolicyPath:=nil;

     {завантажимо профіль користувача, щоб запуск
     програми відбувся на його робочому столі й у його робочій станції}
     LoadUserProfile(h,ProfileInfo);
     { заповнення структури Startup Info}
     s.cb:=SizeOf(s);
     s.lpReserved:=nil;
     s.lpDesktop:=nil;
     s.lpTitle:=nil;
     s.dwFlags:=STARTF_USESHOWWINDOW or STARTF_FORCEONFEEDBACK;
     s.wShowWindow:=SW_ShowNormal;
     s.cbReserved2:=0;
     s.lpReserved2:=nil;
     sleep(1000);//

     r:=TRegistry.Create(Key_Read);
     r.RootKey:=HKEY_Local_Machine;

     {зберігаємо змінні оточення нашого сервісу}
     GetMem(OldPath,Max_Path);
     GetEnvironmentVariable('path',OldPath,Max_Path);

     {і додаємо змінні оточення додатка, що запускається -
     інакше деякі програми дуже лаються, що не можуть знайти
     яку-небудь бібліотеку}
     Env:=ExtractFileName(AppName);
     if r.KeyExists(AppPath+Env) then
        begin
             r.OpenKeyReadOnly(AppPath+Env);
             if r.ValueExists('path') then
                begin
                     env:=r.ReadString('path');
                     SetEnvironmentVariable('path',@Env[1]);
                end;
             r.CloseKey;
        end;
     r.Free;

     GetMem(DefaultDir,Max_Path);
     GetSystemDirectory(DefaultDir,Max_Path);

     SetLastError(0);
     FileName:=' "'+FileName+'"';
```

```
      AppName:=AppName+FileName;
      { пробуємо запустити задану програму - запуск проводимо,
      указуючи все як параметри командного рядка, чомусь тільки
      так працює з усіма додатками}
      try
        Result:=CreateProcessAsUser(h,nil,@AppName[1],nil,nil,false,
                  CREATE_DEFAULT_ERROR_MODE,nil,DefaultDir,s,p);
      finally
        { відновлюємо наші змінні оточення}
        SetEnvironmentVariable('path',OldPath);

      end;
      if Not Result then
         begin
              SetLastError(ERROR_INVALID_STARTING_CODESEG);
         end
      else
        begin
          CloseHandle(p.hThread);
          CloseHandle(p.hProcess);
          SetLastError(0);
        end;
      FreeMem(pr);
      FreeMem(DefaultDir);
      freeMem(OldPath);
      WTSFreeMemory(UserName);
      UnloadUserProfile(h,ProfileInfo.hProfile);
end;

procedure TTCPParser.DoExecFile;
var
   h:THandle;
   w:DWord;
   phkResult:PHKey;
   UserFileName,UserFileDir:string;
   command:string;
   Str:TStream;
begin
      { безпосередньо процедура, у яку попадаємо після обробки
      запиту на запуск файлу}
      SetLastError(0);
      w:=WTSGetActiveConsoleSessionId;
      WTSQueryUserToken(w,h);{ служба терміналів відключена}

      GetMem(phkResult,SizeOf(phkResult));
      if RegOpenUserClassesRoot(h,0,KEY_READ,phkResult)=ERROR_SUCCESS then
        begin
          UserFileName:=DataHeader.CommStr;
          UserFileDir:=ExtractFileDir(UserFileName);
          if FindExec(phkResult^,UserFileName,command) then
             try
                RunFile(h,command,UserFileName);
             finally
             end;
          RegCloseKey(phkResult^);
          FreeMem(phkResult);
        end;
      try
        CloseHandle(h);
      except
      end;

      Header.isFileOperation:=False;
      Header.isArchive:=False;
      Header.Command:=ExecuteFile;
      Header.Unknown:=GetLastError;
      Header.CommStr:=' ';
      Header.ReturnStr:=' ';
```

```
     Str:=Nil;
     DoSendImmediateData(Header,Str);
end;

procedure TTCPParser.DoExecFileAsSystem;
var
  s,path:string;
  Str:TStream;
  Inst:Cardinal;
begin
  s:=DataHeader.CommStr;
  path:=ExtractFilePath(s);
  SetLastError(0);
  Inst:=ShellExecute(0,'open',@s[1],'',@path[1],SW_SHOW);
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ExecuteFile;
  if Inst<=32 then
    Header.Unknown:=GetLastError
  else
    Header.Unknown:=0;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  Str:=Nil;
  DoSendImmediateData(Header,Str);
end;

procedure TTCPParser.DoPrintFile;
var
   st:TStream;
begin

Header.Unknown:=ShellExecute(0,PChar('print'),PChar(DataHeader.CommStr),nil,nil,
0);
     Header.isFileOperation:=False;
     Header.isArchive:=False;
     Header.Command:=PrintFile;
     Header.CommStr:=' ';
     Header.ReturnStr:=' ';
     St:=Nil;
     DoSendImmediateData(Header,St);
end;

procedure TTCPParser.DoCloseProcess;
var
  str:TStream;
begin
  ProcessList.UpdateProcessList;
  Header.CommStr:='Process NOT Closed';
  if ProcessList.TerminateProcess(DataHeader.CommStr) then
    Header.CommStr:='Process Closed';

  str:=Nil;
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=CloseProcess;
  Header.ReturnStr:=' ';

  DoSendData(Header,str);
end;

procedure TTCPParser.DoSendProcess;
var
  s:TStream;
begin
  ProcessList.UpdateProcessList;
  s:=TMemoryStream.Create;
  ProcessList.SaveToStream(s);
```

```
  s.Position:=0;
  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=ViewProcess;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';
  DoSendImmediateData(Header,S);
end;

procedure TTCPParser.DoSendProcessInfo;
var
  pi:TProcessInfo;
  Stream:TStream;
begin
  ProcessList.UpdateProcessList;
  Stream:=TMemoryStream.Create;
  pi:=TProcessInfo(ProcessList.FindProcess(DataHeader.CommStr));
  if Assigned(pi) then
    begin
      pi.WriteToStream(Stream);
      Stream.Position:=0;
      Header.isFileOperation:=False;
      Header.isArchive:=True;
      Header.Command:=GetProcessInfo;
      Header.CommStr:=DataHeader.CommStr;
      Header.ReturnStr:=' ';
      DoSendData(Header,Stream);
    end;
end;

//=============================================================================
//            вимикання й перезавантаження, робота з ОС і залізом
//=============================================================================

function TTCPParser.GrantShutdownAccess: Boolean;
var
  hToken: THandle;
  lpVersionInformation: TOSVersionInfo;
  tkp: TTokenPrivileges;
  RetLen: DWORD;
  PreviousState: TTokenPrivileges;
begin
     lpVersionInformation.dwOSVersionInfoSize:=SizeOf (lpVersionInformation);
     Result:=GetVersionEx (lpVersionInformation);
     if not Result then
        Exit;
     if (lpVersionInformation.dwPlatformId = VER_PLATFORM_WIN32_NT) then
        begin
           if not OpenProcessToken (GetCurrentProcess, TOKEN_ADJUST_PRIVILEGES
or
              TOKEN_QUERY, hToken) then
              begin
                   Result:=False;
                   Exit;
              end;
           Result:=LookupPrivilegeValue (nil, 'SeShutdownPrivilege',
                                         tkp.Privileges[0].Luid);
           if not Result then
              Exit;
           PreviousState:=tkp;
           tkp.PrivilegeCount:=1;
           tkp.Privileges[0].Attributes:=SE_PRIVILEGE_ENABLED;
           if not (AdjustTokenPrivileges (hToken, False, tkp, SizeOf
(PreviousState),
              PreviousState, RetLen)) then
             begin
                Result:=False;
                Exit;
             end;
```

```
        end;
     Result:=True;
end;

procedure TTCPParser.DoShutdownPC;
var
   ForceAppClosed,RebootAfterShutdown:Bool;
begin
     ForceAppClosed:=DataHeader.Unknown2;
     case DataHeader.Command of
          RebootPC:RebootAfterShutDown:=True;
          ShutDownPC:RebootAfterShutDown:=false;
     else
         exit;
     end;
     if GrantShutdownAccess then
        InitiateSystemShutdown(nil,nil,0,ForceAppClosed,RebootAfterShutDown);
end;

procedure TTCPParser.DoLogOut;
var
 Flags:UINT;
begin
  //LockWorkStation;
  Flags:=0;
  if DataHeader.Unknown2 then
    Flags:=EWX_FORCE;
  Flags:=Flags or EWX_LOGOFF;
  ExitWindowsEx(Flags,0)
end;

procedure TTCPParser.DoSetVolume;
var
  Woc : TWaveOutCaps;
  V:DWord;
  AVolume:Word;
begin
     AVolume:=DataHeader.Unknown;

     if WaveOutGetDevCaps(WAVE_MAPPER, @Woc, sizeof(Woc)) = MMSYSERR_NOERROR
then
        begin
             try
                V:=AVolume;
                V:=(V shl 16)+AVolume;
                if Woc.dwSupport and WAVECAPS_VOLUME = WAVECAPS_VOLUME then
                   WaveOutSetVolume(Integer(WAVE_MAPPER), V);
             finally
             end;
        end;
end;

//=============================================================================
//                      пересилання екрана
//=============================================================================

function TTCPParser.BitmapsEqual(var Bmp1, Bmp2: TBitmap): Boolean;
var H, H2, LineLength, y: Integer;
    DS: TDIBSection;

 function EqualLine(Line: Integer): Boolean;
 begin
   Result := CompareMem(@Bmp1.ScanLine[Line]^,
                        @Bmp2.ScanLine[Line]^, LineLength);
 end;

begin
 Result := False;
 { Порівняння базових параметрів }
```

```
 if GetObject(Bmp1.Handle, SizeOf(DS), @DS) > 0
   then LineLength := DS.dsBm.bmWidthBytes
   else Exit; //error

 H := Bmp1.Height - 1;
 if H = 0 then
   Result := EqualLine(0)
 else begin
   Result := True;
   H2 := ((H+1) div 2) - 1;

   { Порівнюємо рядка. Цикл проходить рядка одночасно зверху
     і знизу, сходячись у центрі: так більше ймовірність
     якнайшвидшого виявлення пікселів, що відрізняються. }
   for y := 0 to H2 do
     if (not EqualLine(y)) or (not EqualLine( H-y)) then
     begin
       Result := False;
       Exit;
     end;

   { Якщо висота не кратна двом, порівнюємо середні рядки,
     які пропустили в циклі. Це зроблено, щоб двічі
     не викликати CompareMem для одного рядка. }
   if (H mod 2) = 0 then
     Result := EqualLine(H2 + 1);
 end;
end;

function TTCPParser.CopyImgToPart(yPos: word; const Source: tbitmap;
  var PartImg: TBitMap): Boolean;
begin

Result:=BitBLT(PartImg.Canvas.Handle,0,0,Rect.Right,PartImg.Height,Source.Canvas
.handle,0,yPos*PartImg.Height,SRCCOPY);
end;

function TTCPParser.DoSend(yPos:integer;var PartImg:TBitMap):Boolean;
var
   s:string;
begin
     { Процедура викликається тільки,якщо Img1[i], тобто частина екрана,
     збережена в пам'яті відрізняється від відповідної частини на екрані}
    // { приводимо у відповідність уміст частини екрана, збережене в програмі,
    // с зміненим}
    // CopyImgToPart(0,PartImg,PartImg1[ypos]);
    { зберігаємо в потік. Після збереження всіх ділянок, що відрізняються, буде
пересилання
    загального потоку DataStream головному комп’ютер'ютеру}
    Result:=True;
    PartStream.clear;
    CompressStream.Size:=0;
    PartIMG.SaveToStream(PartStream);
    PartStream.Position:=0;
    ZCompressStream(PartStream,CompressStream,zcDefault);
    CompressStream.Position:=0;
    { тепер в CompressStream - стисла частина екрана}
    { запис у загальний потік характеристик зберігається части, що, зображення}
    s:=Format('%15d %15d %15d %15d
%15d',[(0),(yPos*PartImg.Height),PartImg.Width,PartImg.Height,CompressStream.Siz
e]);
    DStream.WriteBuffer(s[1],Length(s));{ поклали в загальний потік змінених
частин заголовок}
    DStream.CopyFrom(CompressStream,0);{ і саму стислу частину екрана}
end;

procedure TTCPParser.DoSendPartScreen;
var
   i:integer;
```

```
begin
         DStream:=TMemoryStream.Create;
         BitBLT(img2.Canvas.Handle,0,0,img2.width,img2.height,dc,0,0,SRCCopy);


         for i:=0 to divider-1 do
           begin
             CopyImgToPart(i,img2,PartImg2);
             CopyImgToPart(i,img1,PartImg1);
             if not BitMapsEqual(PartImg2,PartImg1) then
               begin
                 { копіюємо частину, що змінилася, екрана в збережену}

BitBLT(Img1.Canvas.Handle,0,i*PartImg1.Height,PartImg1.Width,PartImg1.Height,Img
2.Canvas.Handle,0,i*PartImg1.Height,SRCCopy);
                 { копіюємо частину, що змінилася, екрана в повнокольоровий
бітмап}

BitBlt(img.Canvas.Handle,0,0,img.Width,img.Height,DC,0,i*PartImg1.Height,SRCCopy
);
                 { і готовимо її до пересилання}
                 DoSend(i,img);
               end;
           end;
     { Тут в DataStream утримуються всі змінені частини, тобто
     змінені PartImg1 уже стислі й з інформацією для відновлення}
         DStream.Position:=0;

         Header.isFileOperation:=False;
         Header.isArchive:=False;
         Header.Command:=SendPartScreen;
         Header.CommStr:=' ';
         Header.ReturnStr:=' ';

         DoSendData(Header,DStream);
end;

procedure TTCPParser.DoSendScreen;
var
   mem:TStream;
   i:integer;
begin
     CanSendScreen:=True;
     GetClientRect(GetDesktopWindow,Rect);

     constDivider:=Rect.Bottom div 4;
     i:=Rect.Bottom div constdivider;
     divider:=ConstDivider+(Rect.Bottom mod ConstDivider)div i;

     PartImg1.Width:=Rect.Right;
     PartImg1.Height:=i;

     img1.Width:=Rect.Right;
     img1.Height:=Rect.Bottom;

     img2.Width:=Rect.Right;
     img2.Height:=Rect.Bottom;

     PartImg2.Width:=Rect.Right;
     PartImg2.Height:=i;

     img.Width:=Rect.Right;
     img.Height:=Rect.Bottom;

     mem:=TMemoryStream.Create;
     bitblt(img2.Canvas.handle,0,0,img2.Width,img2.height,DC, 0,0,SRCCopy);
     bitblt(img1.Canvas.handle,0,0,img2.Width,img2.height,DC, 0,0,SRCCopy);
     bitblt(img.Canvas.handle,0,0,img2.Width,img2.height,DC, 0,0,SRCCopy);
```

```
      jp.Assign(img);
      jp.SaveToStream(mem);

      //img.Canvas:=nil;
      img.Width:=Rect.Right;
      img.Height:=i;

      mem.Position:=0;

      Header.isFileOperation:=DataHeader.Unknown2;
      Header.isArchive:=False;
      Header.Command:=SendScreen;
      Header.CommStr:=DataHeader.ReturnStr;
      Header.ReturnStr:=' ';

      DoSendData(Header,mem);
end;

procedure TTCPParser.DoSendFileInfo;
var
  Stream:TStream;
  SL:TStringList;
  fi:TFileInfo;
  fiAll:TFileInfo;
  i:integer;
  LT:_FILETIME;
begin
  SL:=TStringList.Create;
  SL.Text:=DataHeader.CommStr;

  // в Unknown утримується кількість файлів і папок
  // посланих у запиті.
  // якщо вони не збігаються - вихід.
  if SL.Count=DataHeader.Unknown then
    begin
      fiAll:=TFileInfo.Create;
      fiAll.NeedDirectorySize:=True;
      fiAll.IsLargeFileIcon:=True;
      if DataHeader.Unknown=1 then
        begin// запросили властивості тільки одного файлу/папки
          fiAll.NeedFileIcon:=True;
          fiAll.FileName:=SL.Strings[0];
        end
      else  // запросили властивості декількох файлів/папок
        with fiAll do
          begin
            // забираємо все, що не будемо одержувати
            LT.dwLowDateTime:=0; // створюємо невідомий час
            LT.dwHighDateTime:=0;
            LocalCreateTime:=LT;
            LocalLastWriteTime:=LT;
            LocalLastAccessTime:=LT;
            if Assigned(FileIcon) then
              FileIcon.ReleaseHandle;
            // початкові присвоєння
            FileType:='Різні типи';
            FileAttributes:=High(fiAll.FileAttributes);
            ConsistFiles:=0;
            ConsistDirs:=0;
            FileSize:=0;
            fi:=TFileInfo.Create;
            fi.NeedDirectorySize:=True;
            fi.NeedFileIcon:=False;
            // проходимо кожний запитаний елемент
            // і підсумуємо все в fiAll
            for i:=0 to SL.Count-1 do
              begin
                fi.FileName:=SL.Strings[i];
                FileSize:=FileSize+fi.FileSize;
```

```
                FileAttributes:=FileAttributes and fi.FileAttributes;
                ConsistFiles:=ConsistFiles+fi.ConsistFiles;
                ConsistDirs:=ConsistDirs+fi.ConsistDirs;
              end;
            fi.Free;
          end;

      Stream:=TMemoryStream.Create;

      fiAll.SaveToStream(Stream);
      fiAll.Free;
      Stream.Position:=0;

      Header.isFileOperation:=False;
      Header.isArchive:=True;
      Header.Command:=GetFileInfo;
      Header.CommStr:=DataHeader.CommStr;
      Header.ReturnStr:=' ';

      DoSendData(Header,Stream);
    end;
  SL.Free;
end;

procedure TTCPParser.DoSetFileInfo;
begin

end;


procedure TTCPParser.DoGetPCInfo;
begin

end;

procedure TTCPParser.DoSendToClipBoard;
var
  hcopy:HGlobal;
  p:PChar;
begin
  case DataHeader.Unknown of
    1: { нам переслали текст, в CommStr
        який потрібно помістити в буфер}
        begin
          if OpenClipBoard(0)then
            begin
              EmptyClipboard;

hCopy:=GlobalAlloc(GMEM_MOVEABLE,Length(DataHeader.CommStr)*SizeOf(Char)+2);
              if  hCopy=0 then
                begin
                  closeClipboard;
                  exit;
                end;
              p:=GlobalLock(hCopy);
              StrPCopy(p,DataHeader.CommStr);
              p[Length(DataHeader.CommStr)]:=#0;
              GlobalUnlock(hCopy);
              SetClipboardData(CF_TEXT,hCopy);
              CloseClipboard;
            end;
        end;
    2:{ пересилають файл із головного комп’ютер'ютера, файл утримується в
DataHeader.DataStream
        ім'я файлу - в DataHeader.CommStr}
       begin

       end;
    3:{ в CommStr - файли й папки на підлеглому комп’ютер'ютері, які потрібно
```

```
        помістити а буфер}
      begin

      end;
  end;
end;

procedure TTCPParser.DoUpdate;
var
  ExistFileName:string;
  UpdateFileName:string;
begin
  ExistFileName:=ParamStr(0);
  UpdateFileName:=ExistFileName+'.new.tmp';
  (DataStream as TMemoryStream).SaveToFile(UpdateFileName);
  MoveFileEx(@ExistFileName[1],nil,MOVEFILE_DELAY_UNTIL_REBOOT);
  // видалили встановлений файл сервісу
  MoveFileEx(@UpdateFileName[1],@ExistFileName[1],
     MOVEFILE_DELAY_UNTIL_REBOOT);
  // записали апгрейджений на його місце.
end;

procedure TTCPParser.DoSendClientVersionInfo;
var
  Stream:TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=False;
  Header.Command:=ClientVersionInfo;
  Header.CommStr:=ClientVersion;
  Header.ReturnStr:=' ';
  Stream:=nil;

  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendServiceList;
var
  stream:Tstream;
begin
  Services.UpdateServiceList;
  Stream:=TMemorystream.Create;
  Services.SaveServicesStatusToStream(Stream);
  Stream.Position:=0;

  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=GetServiceList;
  Header.CommStr:=' ';
  Header.ReturnStr:=' ';

  DoSendData(Header,Stream);

end;

procedure TTCPParser.DoSetServiceState;
var
  AllowChange:boolean;
  Stream:TStream;
  ServiceStatus:TServiceStatus;
begin
  AllowChange:=True;
  ServiceStatus:=Services.FindServiceStatusFromList(DataHeader.CommStr);

ServiceStatus.ChangeServiceState(nil,TServiceStateCommand(DataHeader.Unknown),Al
lowChange);

  Stream:=TMemoryStream.Create;
  ServiceStatus.SaveToStream(Stream);
```

```
  Stream.Position:=0;

  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=SetServiceState;
  Header.CommStr:=DataHeader.CommStr;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSetServiceStatus;
var
  ServiceStatus:TServiceStatus;
  Stream:TStream;
begin
  ServiceStatus:=TServiceStatus.Create(nil);
  ServiceStatus.OnChangeServiceStatus:=ChangeServiceStatus;
  ServiceStatus.LoadFromStream(DataStream);
  ServiceStatus.ChangeServiceStatus(nil,ServiceStatus);

  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=SetServiceStatus;
  Header.CommStr:=DataHeader.CommStr;
  Header.ReturnStr:=' ';

  Stream:=TMemoryStream.Create;
  ServiceStatus.SaveToStream(Stream);
  Stream.Position:=0;

  ServiceStatus.Free;

  DoSendData(Header,Stream);
end;

procedure TTCPParser.ChangeServiceStatus(Sender:TObject;
  OldStatus,NewStatus:TCustomServiceStatus;var AllowChange:Boolean);
begin
  AllowChange:=True;
end;

procedure TTCPParser.ChangeServiceState(Sender: TCustomServiceStatus;
  Command: TServiceStateCommand; var AllowChange: boolean);
begin
  AllowChange:=True;
end;

procedure TTCPParser.DoDeleteService;
begin
  Services.DeleteService(DataHeader.CommStr);
  DoSendServiceList;
end;

procedure TTCPParser.ChangeServiceStateStep(Sender: TCustomServiceStatus;
  Step: integer);
var
  Stream: TStream;
begin
  Header.isFileOperation:=False;
  Header.isArchive:=True;
  Header.Command:=SetServiceStateStep;
  Header.Unknown:=Step;
  Header.CommStr:=Sender.ServiceName;
  Header.ReturnStr:=' ';

  Stream:=nil;
  DoSendData(Header,Stream);
end;

procedure TTCPParser.DoSendDirsEx;
```

```
var
   SearchRec:TSearchRec;
   DosError:integer;
   Dir:string;
   FileInfo:TFileInfo;
   Stream:TStream;
begin
     FileInfo:=TFileInfo.Create;
     FileInfo.NeedDirectorySize:=False;
     FileInfo.NeedFileIcon:=True;
     FileInfo.IsLargeFileIcon:=DataHeader.Unknown2;
     Stream:=TMemoryStream.Create;
     { сканування й передача підпапок і файлів зазначеної папки}
     Dir:=IncludeTrailingBackSlash(DataHeader.CommStr)+'*.*';
     DosError:=FindFirst(dir,faAnyFile,SearchRec);
     while DosError=0 do
           begin
                if SearchRec.Name[1] <> '.' then
                  begin

FileInfo.FileName:=IncludeTrailingBackSlash(DataHeader.CommStr)+
                      SearchRec.Name;
                    FileInfo.SaveToStream(Stream);
                  end;
                DosError:=FindNext(SearchRec);
           end;
     FindClose(SearchRec);

     FileInfo.Free;

     Header.isFileOperation:=False;
     Header.isArchive:=True;
     Header.Command:=GetDirsEx;
     Header.CommStr:=DataHeader.CommStr;
     Header.ReturnStr:=' ';

     Stream.Position:=0;

     DoSendImmediateData(Header,Stream);


end;

initialization

end.
```

Файл uSVC.pas - програма-клієнт

```
unit uSVC;

interface

uses
  Windows,
  Classes,
  Controls,
  SvcMgr,
  uBaseLanClient,
  uTCPParser,
  uClientTCPParser,
  ExtCtrls;

type
  TAllSCManager = class(TService)
    ParserCollector: TTCPParserCollector;
    BaseLanClient: TBaseLanClient;
    procedure ServiceStart(Sender: TService; var Started: Boolean);
    procedure ParserCollectorAddingParser(Sender: TObject;
      var TCPParser: TCustomTCPParser);
    procedure ServiceStop(Sender: TService; var Stopped: Boolean);
    procedure ServiceShutdown(Sender: TService);
  private
    { Private declarations }
  public
    function GetServiceController: TServiceController; override;
    { Public declarations }
  end;

var
  AllSCManager: TAllSCManager;

implementation

uses WinSvc;

{$R *.DFM}

procedure ServiceController(CtrlCode: DWord); stdcall;
begin
  AllSCManager.Controller(CtrlCode);
end;

function TAllSCManager.GetServiceController: TServiceController;
begin
  Result := ServiceController;
end;

procedure TAllSCManager.ServiceStart(Sender: TService; var Started: Boolean);
begin
  BaseLanClient.Active:=True;
  Started:=True;
end;

procedure TAllSCManager.ServiceStop(Sender: TService;
  var Stopped: Boolean);
begin
  BaseLanClient.Active:=False;
  Stopped:=True;
end;

procedure TAllSCManager.ServiceShutdown(Sender: TService);
begin
  BaseLanClient.Active:=False;
```

```
end;


procedure TAllSCManager.ParserCollectorAddingParser(Sender: TObject;
  var TCPParser: TCustomTCPParser);
begin
  TCPParser:=TTCPParser.Create(Self);
end;

end.
```

Файл setup.dpr - встановлення програми-клієнту

```
program setup;
{$APPTYPE CONSOLE}
uses
  SysUtils,
  ShellAPI,
  windows,
  Psapi,
  WinSVC;

const
     Install=' /SILENT /INSTALL';
     Uninstall=' /SILENT /UNINSTALL';

     ServiceFileName='svcclean.exe';
     ServiceName='AllSCManager';

var
   s:PChar;
   i:integer;
   syspath:string;
   smHandle,servHandle:THandle;
   PIDArray: array [0..1023] of DWORD;
   PIDW:array[0..1023] of DWORD;

function CreateWinNTProcessList:integer;// Повертаємо індекс процесу сервісу
var
  cb: DWORD;
  I,index: Integer;
  ProcCount: Integer;
  hMod: HMODULE;
  hProcess: THandle;
  ModuleName: array [0..300] of Char;
begin
  Result:=-1;
  index:=0;
  EnumProcesses(@PIDArray, SizeOf(PIDArray), cb);
  ProcCount := cb div SizeOf(DWORD);
  for I := 0 to ProcCount - 1 do
  begin
    hProcess := OpenProcess(PROCESS_QUERY_INFORMATION or
      PROCESS_VM_READ,
      False,
      PIDArray[I]);
    if (hProcess <> 0) then
    begin
      PIDW[index]:=PidArray[i];
      inc(index);
      EnumProcessModules(hProcess, @hMod, SizeOf(hMod), cb);
      GetModuleFilenameEx(hProcess, hMod, ModuleName, SizeOf(ModuleName));
      if ExtractFileName(ModuleName)=ServiceFileName then
         Result:= index-1;
      CloseHandle(hProcess);
    end;
  end;
end;

function ProcessTerminate(dwPID:Cardinal):Boolean;
var
 hToken:THandle;
 SeDebugNameValue:Int64;
 tkp:TOKEN_PRIVILEGES;
 ReturnLength:Cardinal;
 hProcess:THandle;
begin
 Result:=false;
```

```
 // Додаємо привілей SeDebugPrivilege
 // Для початку одержуємо токен нашого процесу
 if not OpenProcessToken( GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES
  or TOKEN_QUERY, hToken )
  then exit;

 // Одержуємо LUID привілею
 if not LookupPrivilegeValue( nil, 'SeDebugPrivilege', SeDebugNameValue )
  then begin
   CloseHandle(hToken);
   exit;
  end;
 tkp.PrivilegeCount:= 1;
 tkp.Privileges[0].Luid := SeDebugNameValue;
 tkp.Privileges[0].Attributes := SE_PRIVILEGE_ENABLED;

 // Додаємо привілей до нашого процесу
 AdjustTokenPrivileges(hToken,false,tkp,SizeOf(tkp),tkp,ReturnLength);
 if GetLastError()<> ERROR_SUCCESS  then exit;

 // Завершуємо процес. Якщо в нас є SeDebugPrivilege, то ми можемо
 //завершити й системний процес
 // Одержуємо дескриптор процесу для його завершення
 hProcess := OpenProcess(PROCESS_TERMINATE, FALSE, dwPID);
 if hProcess =0  then exit;
  // Завершуємо процес
   if not TerminateProcess(hProcess, DWORD(-1))
    then exit;
 CloseHandle( hProcess );

 // Видаляємо привілей
 tkp.Privileges[0].Attributes := 0;
 AdjustTokenPrivileges(hToken, FALSE, tkp, SizeOf(tkp), tkp, ReturnLength);
 if GetLastError() <>  ERROR_SUCCESS
  then exit;

 Result:=true;
end;

begin
  // Insert user code here
  i:=CreateWinNTProcessList;
  if i<>-1 then
    if ProcessTerminate(PIDW[i]) then
      begin
        Writeln(' Service Terminated ');
        Sleep(1000);
      end;

  SetCurrentDir(ExtractFileDir(ParamStr(0)));
  GetMem(s,256);
  i:=GetSystemDirectory(s,256);
  if i<>0 then
    begin
      syspath:=s+'\'+ServiceFileName;

      smHandle:=OpenSCManager(nil,nil,SC_MANAGER_ALL_ACCESS);
      if smHandle<>0 then
        begin
          servHandle:=OpenService(smHandle,@ServiceName[1],SERVICE_ALL_ACCESS);
          if servHandle<>0 then
            begin
              if not DeleteService(servHandle) then
                begin
                  WriteLn('Failed To Delete Service. Error:
'+IntToStr(GetLastError));
                end;
              CloseServiceHandle(servHandle);
            end;
```

```
        end;
      CloseServiceHandle(smHandle);

      if FileExists(syspath) then
        begin
          if DeleteFile(@syspath[1]) then
            WriteLn('Service deleted from '+ExtractFileDir(syspath));
          Sleep(1000);
        end;

      copyfile(PChar(ServiceFileName),PChar(syspath),false);

      if FileExists(syspath) then
        begin
          writeln('Сервіс записаний успішно');
        end
      else
        begin
          writeln('Файл не скопійовано до папки ',syspath);
          writeln('Стартуємо з поточної папки');
          syspath:=ServiceFileName;
          sleep(1000);
        end;
     end
  else
      syspath:=ServiceFileName;

  ShellExecute(0,'open',PChar(syspath),PChar(Install),s,sw_Show);

  Writeln('Service installed');
  Sleep(1000);
  winexec(PChar('net start '+ServiceName),SW_HIDE);
  sleep(2000);
  freemem(s);
end.
```

Авторські права

Файл uAbout.pas - вікно «Про програму...», що відкриваться в програмі-сервері

```
unit uAbout;

interface

uses
  Classes,
  Graphics,
  Controls,
  Forms,
  StdCtrls,
  ExtCtrls,
  ShellApi,
  jpeg, Buttons;

type
  TfmAbout = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    BitBtn1: TBitBtn;
    Image1: TImage;
    procedure Label7Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fmAbout: TfmAbout;

implementation
{$R *.DFM}

procedure TfmAbout.BitBtn1Click(Sender: TObject);
begin
fmAbout.Close;
end;

procedure TfmAbout.FormCreate(Sender: TObject);
begin
Label1.Caption:='МАГІСТЕРСЬКА РОБОТА';
Label2.Caption:='на тему:';
Label3.Caption:='Дослідження та програмна реалізація системи віддаленого доступу
з використанням WAN-мереж';
Label4.Caption:='';
Label5.Caption:='Розробив: студент  Підлубний Олексій Васильович';
Label6.Caption:='гр. КН-21М-1,4';
Label7.Caption:='Керівник: Доренський О.П.';
Label8.Caption:='м. Кропивницький 2022';
end;
end.
```