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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи адміністрування віддаленим комп’ютером за протоколом TCP/IP”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Гнида І.О.
«____» _____ 2023 р.

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"*Інформаційні технології*"*_____
Спеціальність *123 "*Комп'ютерна інженерія*"*
Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Гниді Івану Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи адміністрування віддаленим комп'ютером за протоколом TCP/IP*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 8-02 від 5.01.2023 року

3. Строк подання студентом роботи до захисту *23.05.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи адміністрування віддаленим комп'ютером за протоколом TCP/IP*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2023 р.	
3.	Розробка моделі компонента	20.03.2023 р.	
4.	Розробка структур даних	25.03.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2023 р.	
6.	Програмування алгоритмів	10.04.2023 р.	
7.	Оформлення ПЗ	17.04.2023 р.	
8.	Попередній захист роботи	23.05.2023 р.	

Дата видачі завдання
« 17 » січня 2023 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2023 р.

Підпис здобувача
_____ Гнида І.О.
(прізвище та ініціали)

АНОТАЦІЯ

Гнида І.О. Програмне забезпечення системи адміністрування віддаленим комп'ютером за протоколом TCP/IP. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адміністрування віддаленим комп'ютером за протоколом TCP/IP.

Метою розробки є програмне забезпечення системи адміністрування віддаленим комп'ютером за протоколом TCP/IP.

Результат роботи – програмна реалізація системи адміністрування віддаленим комп'ютером за протоколом TCP/IP.

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 10.4.

Ключові слова: комп'ютерна інженерія, адміністрування віддаленим комп'ютером

ABSTRACT

Hnyda I.O. Remote computer administration software using the TCP/IP protocol. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for a remote computer administration system using the TCP/IP protocol.

The purpose of the development is software for remote computer administration using the TCP/IP protocol.

The result of the work is a software implementation of the remote computer administration system using the TCP/IP protocol.

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 10.4 environment.

Keywords: computer engineering, remote computer administration

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЛОМ	–	локальна обчислювальна мережа
ПК	–	персональний комп’ютер
ПЗ	–	програмне забезпечення
DNS	–	Domain Name System
http	–	HyperText Transfer Protocol
IP	–	Internet Protocol

ВСТУП

Актуальність теми. Системи адміністрування віддаленим комп'ютером повинні забезпечувати необхідну гнучкість, щоб адміністратори мережі могли швидко відреагувати, наприклад, на відмову корпоративного сервера, при цьому зовсім не важливо, де вони перебувають – у серверній, удома або відрядженні.

Від ефективного рішення насамперед очікують забезпечення контролю за серверами різних платформ і послідовною периферією. Хоча й програмні, і апаратні рішення в області віддаленого керування пропонують порівнянні можливості, а саме – віддалене адміністрування сервера, роблять вони це різними способами.

Перші складаються, як правило, із двох різних додатків, які адміністратор повинен установити на керованому сервері й віддаленому ПК, щоб зображення й сигнали керування можна було передавати між двома комп'ютерами. Такі програмні рішення, як pcAnywhere від Symantec, Netop Remote Control від Danware або Remotelyanywhere від 03 AM Laboratories, використовуються найчастіше для керування окремими настільними комп'ютерами в корпоративній мережі або для контролю за серверами Windows. Користувач за віддаленим ПК може працювати за допомогою клавіатури й миші з іншим комп'ютером, що більшість виробників у своїй документації називають хостом. Взаємодія між ПК і хостом здійснюється по локальній мережі, що комутирується з'єднанню або Internet. Як протокол для доступу через Internet програми підтримують, як правило, TCP/IP, IPX, NetBIOS або HTTP. Крім цих основних функцій нерідко пропонуються й додаткові, завдяки яким системні адміністратори можуть передавати дані з комп'ютера на комп'ютер, установлювати нові програми, змінювати призначення завдань на печать або вести переговори з користувачем хоста.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи адміністрування віддаленим комп’ютером за протоколом TCP/IP.

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

– Огляд існуючих систем адміністрування віддаленим комп’ютером за протоколом TCP/IP.

– Дослідження системи адміністрування віддаленим комп’ютером за протоколом TCP/IP.

– Програмна реалізація системи адміністрування віддаленим комп’ютером за протоколом TCP/IP.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі адміністрування віддаленим комп’ютером за протоколом TCP/IP.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи адміністрування віддаленим комп’ютером за протоколом TCP/IP, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Всі програми віддаленого керування можна умовно розділити на дві категорії: програми, що надають доступ до комп'ютера через Інтернет, і програми, орієнтовані на роботу в локальній мережі.

Якщо мова йде про програми першої категорії, то потрібно мати на увазі, що їхня робота дуже сильно обмежена. І навіть у тому випадку, коли в рекламних проспектах вказується, що програма працює у всіх ситуаціях і переборює всі NAT-пристрої й брандмауери, на практиці виявляється, що це далеко не так. І навіть якщо ви користуєтеся такою потужною й популярною програмою, як Hamachi, для того щоб одержати доступ до свого домашнього ПК будучи на роботі, то аж ніяк не завжди це приведе до позитивного результату. Тут багато чого залежить від вашого провайдеру, якого, звичайно ж, ви контролювати не можете.

А тому надалі ми зосередимося на розгляді тих програм, які орієнтовані саме на роботу в локальній мережі. При цьому відзначимо, що більшість програм, призначених для одержання віддаленого доступу через Інтернет, можна застосовувати також для роботи в локальній мережі. Питання тільки в тому, наскільки це доцільно.

Крім того, по своєму призначенню програми віддаленого керування теж можна умовно розділити на дві категорії: програми, що надають доступ до робочого стола віддаленого ПК, і програми, що забезпечують доступ до командного рядка. Перші надають користувачам можливість роботи з віддаленим ПК точно так само, як з локальним, а другі дозволяють автоматизувати роботу мережі, запускаючи на декількох обраних комп'ютерах мережі те саме або різні додатки, а також, приміром, створювати розклад запуску програм на віддалених

ПК. Зовсім безглуздо намагатися порівнювати ці два типи утиліт, оскільки вони використовуються в різних ситуаціях і для різних цілей.

Більшість програм віддаленого керування функціонують за принципом «клієнт-сервер», тобто мають на увазі наявність серверної й клієнтської частин програми, які відповідно встановлюються на керованому комп'ютері й на ПК, з якого виробляється керування. Для одержання контролю над віддаленим ПК необхідно, щоб на ньому був запущений відповідний модуль програми (серверна частина). Деякі утиліти віддаленого керування забезпечують дистанційну установку серверної частини (при наявності відповідних адміністративних прав), причому іноді ця процедура абсолютно прозора для локального користувача ПК (як правило, у цьому випадку серверна частина встановлюється у вигляді служби на віддаленому ПК). Проте деякі утиліти вимагають установки вручну серверної частини програми.

У випадку застосування утиліт віддаленого керування в домашніх умовах, коли в мережі всього два-три комп'ютери й немає виділеного сервера, що виконує функцію контролера домену, тобто коли є вироджена мережа без домену (робоча група), у якій всі користувачі мають рівні права й немає мережного адміністратора, для використання утиліт віддаленого керування необхідно спочатку виконати ряд заходів щодо налаштування ПК.

По-перше, у переважній більшості випадків потрібно, щоб на всіх комп'ютерах робочої групи застосовувалася автентифікація користувачів, тобто вхід користувачів по паролі.

По-друге, для одержання можливості віддалено управляти комп'ютером у складі робочої групи необхідно, щоб на цьому комп'ютері був створений обліковий запис користувача (бажано із правами адміністратора). І якщо у випадку локальної мережі з контролером домену, маючи права адміністратора, можна завести нового користувача на ПК віддалено, то у випадку робочої групи віддалено створити користувача на комп'ютері не можна.

1.2 Область застосування

Колись програми для віддаленого адміністрування минулого потрібні тільки на підприємствах, де одній людині доводиться обслуговувати десятки, а то й сотні комп'ютерів, які розташовані в різних кабінетах і на різних поверхах. Сьогодні сфера їхнього застосування набагато ширше.

По-перше, у багатьох квартирах більше одного комп'ютера, і між ними для більш швидкої передачі інформації протягнена мережа. Якщо комп'ютери знаходяться у різних кімнатах, то програма для віддаленого адміністрування дає можливість працювати на двох ПК одночасно, не встаючи зі стільця.

По-друге, як правило, люди постійно працюють із двома комп'ютерами – домашнім і робочим. Програми для віддаленого адміністрування дозволяють через Інтернет стежити за тим, що відбувається на іншому комп'ютері, тому, сидячи дома, можна контролювати, як іде накачування через eMule на роботі або, навпаки, перебуваючи на робочому місці, спостерігати за тим, з ким дружина дома спілкується месенджером.

Одним словом, програма для віддаленого адміністрування просто необхідна кожному, у чийому розпорядженні перебуває більше одного комп'ютера.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи адміністрування віддаленим комп’ютером за протоколом TCP/IP, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Проведемо порівняльний аналіз існуючих програм віддаленого керування комп'ютером.

Remote Administrator (Radmin)

Radmin – це одна із самих популярних програм для спостереження за віддаленим комп'ютером серед російськомовних користувачів. Причин тому декілька: по-перше, програма розроблена російською компанією й тому має повноцінний російський інтерфейс. По-друге, Radmin має тільки самі необхідні засоби для керування віддаленим ПК і дуже проста в налаштуванні. Завдяки цьому її нескладно освоїти. Нарешті, у третіх, програма має невеликий розмір дистрибутива, який можна легко переслати поштою.

Radmin працює в декількох режимах: передача файлів, повне керування, тільки перегляд, телнет і вимикання. Є вбудований файловий менеджер, за допомогою якого файли передаються з одного ПК на інший. Програма веде статистику використовуваного трафіка й може шифрувати дані.

За допомогою Radmin можна управляти як одним віддаленим комп'ютером, так і відразу декількома. Причому, для кожного можна встановити свій пароль. Якщо потрібно забезпечити підвищений рівень безпеки, можна вжити додаткових заходів обережності: включити захист від перебору пароля й скласти список заборонених IP-адрес. Якщо ви працюєте з мережею, швидкість передачі даних у якій невисока, можна зменшити кількість квітів, які відображаються на екрані програми-клієнта. Це прискорить роботу з віддаленим ПК.

Рисунок 2.1 – Інтерфейс користувача Radmin

Безпека

1. Інтеграція із системою безпеки Windows. Radmin повністю підтримує NTLMv2, Active Directory, Kerberos.

2. Безпечна передача трафіку. Всі передані дані шифруються по стандарту AES.

3. Власна система безпеки. Radmin дозволяє створювати власні облікові записи, наділяючи їхніми певними правами.

4. Фільтрація IP-адрес клієнтів. Для підвищення безпеки можна давати дозвіл для підключення до сервера клієнтів тільки з певних хостов.

5. Ведення логів. У лог попадає ім'я користувача, що підключився, і його IP-адреса.

6. Захист від угадування пароля. Radmin дозволяє припинити підбор пароля, установлюючи блокування після декількох спроб неправильного введення.

Ключові можливості

1. Передача файлів на віддалений ПК.

2. Одержання виду віддаленого робочого стола.

3. Підтримка декількох одночасних з'єднань.

4. Telnet-сервер.

Symantec pcAnywhere

Symantec pcAnywhere, поза всяким сумнівом, можна назвати мрією системного адміністратора. Набір можливостей, які пропонує ця програма, просто величезний. Це – не просто засіб для "підглядання" за діями користувачів на віддалених ПК, а потужний інструмент для керування всіма функціями комп'ютера. Всі дії, які адміністратори, як правило, змушені робити, переміщаючись по будинку від одного комп'ютера до іншого, за допомогою pcAnywhere можна виконати віддалено.

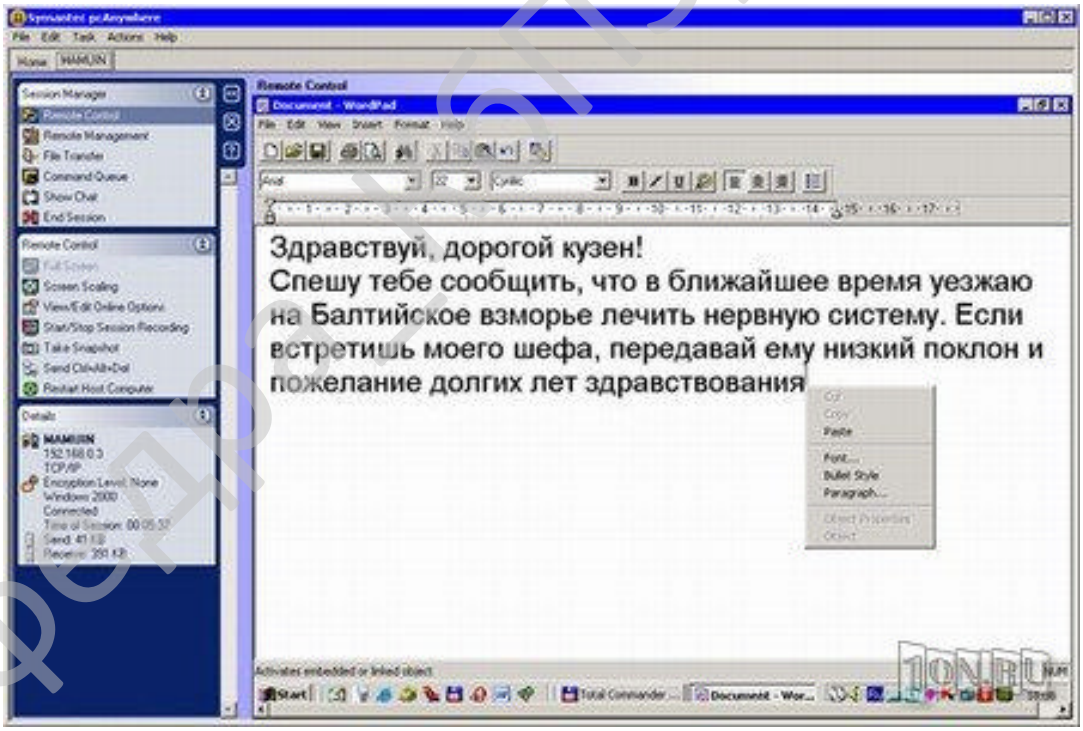

Рисунок 2.2 – Інтерфейс користувача Symantec pcAnywhere

Наприклад, у програмі є інструмент для керування службами операційної системи, редактор для роботи з реєстром, засіб для роботи з командним рядком, можливість видалення або припинення виконання додатків і навіть чат з користувачем. Ще одна корисна можливість – доступ до журналу подій. Він особливо корисний, якщо користувач скаржиться на проблему й затверджує, що він "нічого ТАКОГО не робив". Заглянувши в журнал, можна відразу ж побачити всі дії користувача й розібравшись із причиною проблеми, вирішити її. Окремо варто сказати про менеджера файлів. Працюючи з pcAnywhere, можна не тільки копіювати файли з одного ПК на інший і видаляти їх, але й виконувати безліч інших операцій. Наприклад, визначати черговість копіювання, переривати його й пізніше відновляти з того ж місця, порівнювати папки, робити синхронізацію й т.д.

UltraVNC

Основна перевага цієї програми перед іншими додатками, представленими в огляді, – безкоштовний статус. Вона нічим не уступає багатьом комерційним продуктам, а в деяких випадках навіть їх перевершує. Одна із цікавих можливостей програми – тонке налаштування серверної частини. Якщо ви з якихось причин хочете сховати наявність на віддаленому комп'ютері програми-сервера, ви можете викликати вікно налаштувань програми, клацнувши по значку в області повідомлень, і обмежити права користувача. Наприклад, можна заборонити закриття програми й зміна її налаштувань із сервера, а також сховати значок у системному треї, щоб у цікавих виникало менше питань.

Варто помітити, що засоби віддаленого адміністрування, які реалізовані в UltraVNC, наштовхують на думку, що програма створювалася саме з думкою про непокірливих користувачів. Крім описаних вище параметрів, є також корисна можливість блокування миші й клавіатури на віддаленому ПК під час сеансу підключення до нього. Якщо її включити, користувач не буде смикати мишку, заважаючи вам працювати. Є, до речі, і зворотна по призначенню можливість – режим View Only. Якщо його активувати, то можна буде тільки спостерігати за

віддаленим комп'ютером, не роблячи на ньому ніяких дій. Це зручно, якщо ви не хочете виявити свою присутність і випадково видати себе, зачепивши мишу. Всі основні інструменти для керування віддаленим комп'ютером зібрані у верхній частині вікна вьювера. Використовуючи ці кнопки, можна, наприклад, виконувати на ПК команду CTRL+ALT+DEL, відкривати меню "Пуск", перемикатися з віконного режиму перегляду в повноекраний, обновляти екран, якщо картинка "підвисла", викликати вікно чата або файлового менеджера. У цілому, інструменти керування дуже зручні.

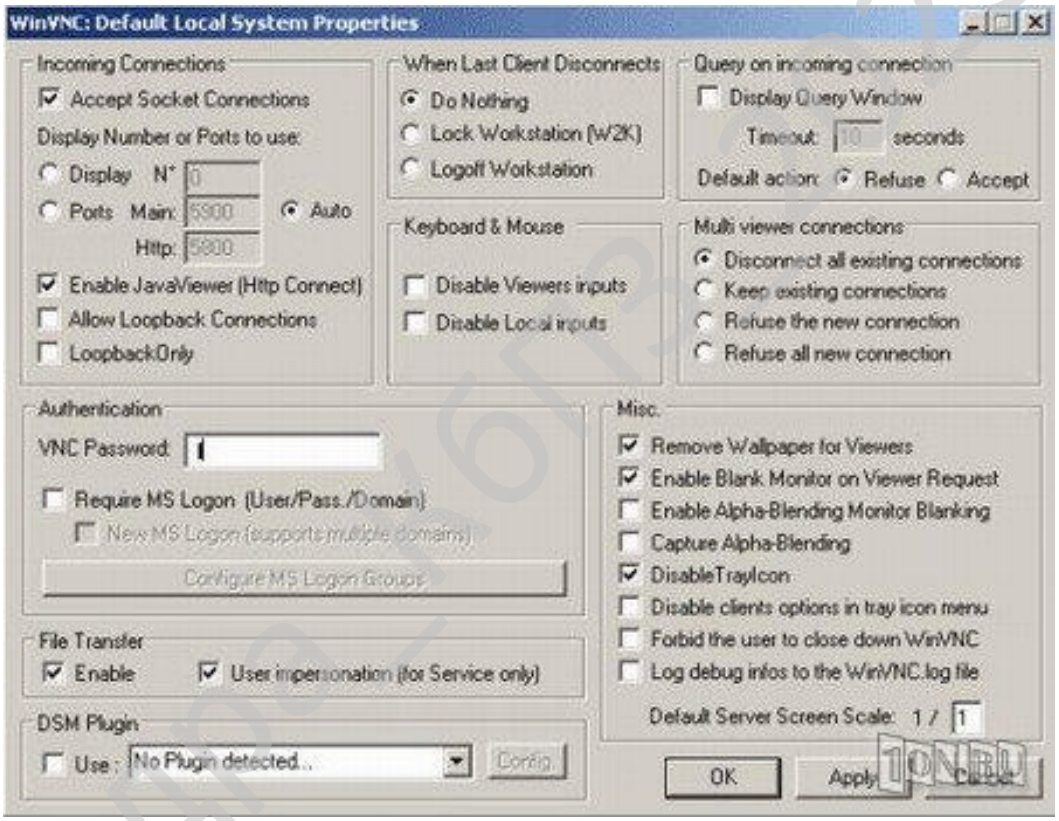

Рисунок 2.3 – Інтерфейс користувача UltraVNC

RemotelyAnywhere

RemotelyAnywhere – це не зовсім звичайна програма для адміністрування. На відміну від інших подібних утиліт, вона складається тільки з однієї частини – сервера, а роль клієнта виконує браузер. Іншими словами, установлювати програму потрібно тільки на тому, комп'ютері, до якого потрібно підключитися.

Це дуже зручно, якщо ви хочете проводити віддалене адміністрування з комп'ютера, на якому не можете встановлювати ніяких програм, наприклад, якщо цей комп'ютер знаходиться в навчальному закладі, в Інтернет-кафе або на роботі. Для роботи програма задіє спеціальний ActiveX-компонент браузера й використовує власний протокол обміну даними. Коли сервер установлений і запущений, до комп'ютера можна одержати доступ, увівши в браузері адреса https://ваш_IP:2000 (якщо ви збираєтеся працювати із програмою через Інтернет) або https://ім'я_комп'ютера:2000 (якщо передбачається підключатися по локальній мережі). Після цього потрібно вказати ім'я користувача й пароль, які використовуються для входу в Windows на віддаленому комп'ютері. Безпека підключення забезпечується NTLM-автентифікацією, захищеною 1024-бітовим ключем. Після підключення у вікні браузера будуть виведені докладні відомості про віддалений ПК, у тому числі, його конфігурація, ступінь заповнювання жорсткого диска, завантаженість процесора, мережна активність і т.д.

Рисунок 2.4 – Інтерфейс користувача RemotelyAnywhere

Всі команди, доступні для виконання, винесені в ліву частину вікна й представлені у вигляді дерева посилань. Їхній набір досить великий: чат, менеджер файлів, керування службами Windows, перегляд журналу подій і списку користувачів, які підключалися до комп'ютера. В RemotelyAnywhere також є планувальник завдань, що дає можливість виконувати різні завдання на віддаленому ПК за розкладом. Список запланованих завдань для зручності виводиться в головному вікні, а ще можна настроїти RemotelyAnywhere на відсилання звітів про їхнє виконання по електронній пошті.

DameWare NT Utilities

DameWare NT Utilitiest – знахідка для будь-якого системного адміністратора. Використовуючи цю «валізку» інструментів, можна забути про більшість рішень, які надає стандартна консоль MMC.

Безпека

1. Шифрування трафіка на основі Crypto API.

2. Можливість автентифікації по смарт-карті.

3. Підтримка системи безпеки серверних версій Windows, інтеграція з ActiveDirectory.

4. Власна система безпеки.

5. Обмеження підключення по IP-адресах.

Ключові можливості

1. Потужний браузер об'єктів ActiveDirectory. За допомогою даної функції можна легко й швидко управляти такими об'єктами AD як Organizational Units, Контейнери, Користувачі, Групи, Контакти, Комп'ютери, Загальні ресурси й т.д. Причому управляти об'єктами AD за допомогою DNTU набагато зручніше, ніж за допомогою стандартних засобів.

2. Велика кількість вбудованих утиліт для керування різними ресурсами віддаленого ПК: Служби, Диски, Процеси, Реєстр, Принтери й т.д.

3. Telnet-сервер.

4. Вбудовані TCP-утиліти: ping, traceroute, mx test і інші.

5. Віддалений вид робочого стола.

DameWare NT Utilities можливостей для адміністрування надає набагато більше, ніж Radmin, але щодо безпеки йому програє. DNTU краще використовувати усередині локальної мережі, а для адміністрування віддалених офісів варто використовувати VPN.

Tight VNC

Tight VNC – одна з програм для віддаленого адміністрування, що поширюється безкоштовно й має версії як під Unix-like системи, так і Windows. Хоч її функціонал не так багатий, це гарна знахідка для адміністраторів, яким доводиться одночасно адмініструвати Windows/Unix сервери.

Безпека

Власна система безпеки.

Можливості

1. Віддалений вид робочого стола.
2. Пересилання файлів.

У випадках, коли доводиться адмініструвати сервери під керуванням Windows і Linux одночасно, зручніше за все використовувати TightVNC**.**

Існує ще ряд програм, які в даному розділі ми не будемо розглядати більш докладно.

1. GoToMyPC – Оригінальне програмне рішення, що надає можливість управляти комп'ютером через звичайний web-браузер. Таким чином, стає можливим керування ПК із будь-якої точки планети, аби тільки там був інтернет і браузер.

2. Tele Desktop – Дозволяє управляти віддаленим робочим столом на зразок Radmin’a .

3. PC-IN-IE – Керування робочим столом прямо з вікна браузера – аналогія GoToMyPC.

4. Remote Anywhere – Ще один варіант програми для віддаленого адміністрування ПК прямо із браузера.

5. NetOP – Як відмітна риса можна вказати реалізацію маси надлишкових можливостей. Але встановити й подивитися радимо.

6. Ideal Administrator – Відмінна програма для віддаленого адміністрування NT-подібних систем. Безліч функцій, простота використання.

Висновок

Незважаючи на те, що всі програми, представлені в огляді, виконують ту саму функцію, навряд чи можна сказати однозначно, яка краще, а яка – гірше. Багаті можливості Symantec pcAnywhere оцінять адміністратори, що обслуговують великі корпоративні мережі, для домашнього використання на декількох комп'ютерах прекрасно підійдуть Remote Administrator і UltraVNC, а RemotelyAnywhere буде незамінна для тих, хто веде мобільний спосіб життя й не знає, з якого комп'ютера він буде виходити в Інтернет наступного разу.

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 першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи адміністрування віддаленим комп'ютером за протоколом TCP/IP.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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 програми адміністрування віддаленим комп'ютером

Програма адміністрування віддаленим комп'ютером – програма, що дозволяє одержати віддалений доступ до комп'ютера через Інтернет або ЛОМ і в цілковитій безпеці управляти віддаленим комп'ютером у режимі реального часу.

Програма адміністрування віддаленим комп'ютером відображає робочий стіл віддаленого комп'ютера, і дозволяє управляти їм з вашого локального ПК за допомогою миші й клавіатури. Інакше кажучи, незалежно від того, де перебуває, можна одержати віддалений доступ до комп'ютера й управляти їм так, ніби перебували безпосередньо перед ним.

Вбудована функція Обмін файлами дозволяє здійснювати обмін файлами між комп'ютерами. За допомогою програми Програма адміністрування віддаленим комп'ютером може віддалено виключати комп'ютер, перезавантажувати, блокувати мишу й клавіатуру на віддаленому ПК, а також здійснювати багато інших вивертів.

Можливості програми для віддаленого доступу

Програма адміністрування віддаленим комп'ютером дозволяє:

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

– Використовувати мишу й клавіатуру для керування віддаленим комп'ютером.

– Здійснювати обмін файлами між комп'ютерами.

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

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

– Одержати віддалений доступ через інтернет до ПК за маршрутизатором або брандмауером без яких-небудь налаштувань.

Системні вимоги: Windows 2000 / XP / 2003 / Vista / 2008 / 7.

Знадобитися як мінімум 2 комп'ютери, підключених друг до друга у ЛОМ або через Інтернет, та програма для віддаленого доступу, установлена на обох ПК. З'єднання між цими комп'ютерами може бути встановлено двома способами:

– Пряме підключення: віддалений доступ до комп'ютера, використовуючи IP адреск або DNS ім'я.

– Online Account підключення: віддалений доступ до комп'ютера через інтернет, використовуючи інтернет-аккаунт і псевдонім комп'ютера. (IP адреса не використовується). Цей режим дозволяє обходити брандмауери, маршрутизатори й інші перешкоди.

Після того, як з'єднання встановлене, 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 Модуль – установлюється на комп'ютер, що хочете контролювати.

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

Інсталяційний пакет містить обидва модулі. Можна брати необхідний модуль під час установки. При бажанні, також можете встановити два модулі на один комп'ютер. Щоб установити програму, будь ласка додержуйтесь наступних інструкцій:

1. На комп'ютер, що буде контролюватися віддалено (Host PC), скачайте інсталяційний пакет з нашого сайту на цей комп'ютер і встановите на нього Host Module.

2. 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 не потрібно підходити до одному з них безпосередньо щоб установити Host Module, тому що можна встановити його віддалено, використовуючи функцію "Віддалена інсталяція". Ця функція вбудована в Admin Module, і можна запустити її в закладці "Інструменти".

Програма адміністрування віддаленим комп'ютером відображає екрани віддалених ПК в окремих RScreen вікнах. RScreen – основний інтерфейс для роботи з віддаленим ПК. У цьому вікні можна управляти віддаленим ПК за допомогою миші й клавіатури. Декілька RScreen-вікон можуть бути відкриті одночасно; таким чином, можна спостерігати відразу за декількома моніторами. Можна розмістити ці вікна на вашім робочому столі як завгодно, а можете згорнути їх.

Режими роботи з віддаленим робочим столом

Програма пропонує два режими роботи з віддаленим робочим столом (можна перемикатися між двома способами):

1. Режим спостереження.
2. Режим керування.

Режим спостереження дозволяє переглядати віддалений робочий стіл (без можливості контролю над ним). Можна бачити віддалений робочий стіл і курсор віддаленої миші в реальному часі. Цей режим дозволяє бачити, що відбувається на віддаленому ПК, і чим власно займається віддалений користувач.

Використовуючи Режим керування, можна додатково контролювати віддалений робочий стіл за допомогою своєї миші й клавіатури.

Режими відображення віддаленого робочого стола

Віддалений робочий стіл може бути відображений на вашім екрані у двох режимах:

1. Повноекраний.
2. Віконний.

Повноекраний режим відображає віддалений робочий стіл на весь екран вашого монітора, у той час як Віконний режим відображає віддалений робочий стіл у вікні.

Функція "Зміни масштабу" і "Автопрокручування"

Якщо робочий стіл віддаленого ПК більше, ніж вікно, у якому воно відображається, тоді програма підганяє розмір вікна так, щоб віддалений робочий стіл помістився в ньому повністю. Таким чином, будете бачити віддалений робочий стіл у злегка зменшеному масштабі, але зате повністю.

Проте, якщо не використовуєте функцію масштабування, тоді можете переміщатися по віддаленому робочому столі за допомогою скролінга. За допомогою функції автоскролінгу не знадобитися прокручувати його вручну. Зображення великого віддаленого робочого стола автоматично прокрутиться туди, куди перемістите свою мишку.

Підтримка різних розрішень і кольорових схем

Програма адміністрування віддаленим комп'ютером підтримує всі екранні розрішення, і якість передачі кольору від 4 до 32 біт. За замовчуванням, програма встановлює передачу кольору для максимальної частоти відновлення екрана в мережах з високою пропускною здатністю (частота залежить від відеокарти). Природно можна встановити якість переданого зображення вручну. Якщо пропускна здатність мережі низька, тоді можна понизити якість передачі кольору, що у свою чергу підвищить продуктивність програми (обсяг переданих даних через мережу буде значно менше).

Панель для вікна віддаленого екрана

Вікно віддаленого екрана містить зручну панель, що дозволяє здійснювати швидкий доступ до часто використовуваних дій, необхідних для віддаленого керування. Наприклад, міняти якість передачі кольору, посилати натискання клавіш, відправляти вміст буфера обміну з локального ПК на віддалений і т.д.

Програма адміністрування віддаленим комп'ютером дозволяє передавати файли з або на віддалений комп'ютер. Вікно "Передачі файлів" у нашій програмі

схоже на вікно Windows Провідника, і складається із двох панелей: локальна (показує файли на локальному ПК) і віддалена (показує файли на віддаленому ПК). Можна просто перетаскувати файли й папки між двома панелями з host комп'ютера або на нього.

Всі файли й допоміжні команди надійно шифруються перед передачею.

Підтримувані операції з файлами

Функція "Передача файлів" дозволяє проводити в панелях локального й віддаленого ПК наступні операції з файлами й папками:

1. Копіювати й переміщати файли й папки з локального ПК на віддалений (і навпаки), або на тому самому ПК із однієї папки в іншу.

2. Видаляти файли й папки.

3. Перейменувати файли й папки.

4. Створювати нові папки.

5. Переглядати властивості файлів і папок.

6. Змінювати властивості файлів і папок.

Відкриття документів і запуск exe-файлів

Використовуючи функцію "Передача файлів", можна відкривати документи й запускати exe-файли на локальному або віддаленому ПК. Для цього знадобитися всього лише два рази нажати на кнопку мишки. Запускаючи інсталяційний файл, також можете задавати опції командного рядка (якщо це необхідно).

Поновлення процедури копіювання у випадку обриву мережного зв'язку

Функція "Передача файлів" відновляє копіювання файлів у випадку розриву зв'язку саме з того місця, де копіювання було перервано. Отже, не потрібно починати весь процес спочатку. Якщо хочете відновити копіювання файлу після відновлення зв'язку, просто перетягніть файл у ту ж папку. Програма проінформує про те, що частина скопійованого файлу вже існує, і запропонує або продовжити копіювання далі, або почати його із самого початку. Ця

можливість особливо корисна при копіюванні великих файлів. У випадку якщо потрібно йти з офісу, – просто поставте копіювання файлу на паузу, а в наступний день після повернення просто відновіть копіювання із припиненого місця.

Стиск файлів для підвищення швидкості передачі

Програма стискає файли перед тим, як передавати їх між комп'ютерами. Це здійснюється за допомогою швидкого алгоритму стиску, що зовсім не знижує швидкодію програми, і в теж час надає гарний коефіцієнт стиску. Більше того, це істотно знижує завантаження мережі, обсяг трафіка й значно підвищує швидкість передачі в мережах з низькою пропускною здатністю (наприклад, при dial-up з'єднанні). Дані передаються між комп'ютерами в стислому виді й у такий спосіб мають менший обсяг.

Функція нашої програми "Обмін фалами" інтелектуально обладнана: вона "розуміє" більшість розповсюджених розширень (наприклад, zip, jpg, gif, avi, etc.) і не стискає їх знову, що у свою чергу допомагає уникнути втрати часу для марного повторного стиску.

Функція стиску нашої програми вигідно відрізняється від рішень наших конкурентів. Звичайно, алгоритми стиску файлів конкурентних продуктів не стискають файли перед відправленням їх через мережу. Тому передача файлів може представляти тривалий процес для мереж з низькою пропускною здатністю.

Динамічне налаштування розміру буфера даних для збільшення швидкості передачі

Функція нашої програми "Обмін файлами" постійно оцінює швидкість з'єднання. Відповідно до отриманих результатів, вона динамічно змінює розмір свого буфера даних, у результаті чого досягається максимальна швидкість передачі даних.

Відповідно до наших тестів, алгоритм передачі файлів нашої програми значно випереджає по своїй швидкодії більшість інших програм (як то FTP, Remote Administrator, VNC, і т.д.)

Звичайно, ці програми використовують фіксований розмір буфера даних. Це неефективно, тому що якщо швидкість вашого з'єднання висока, а обсяг буфера даних малий, то швидкість передачі файлів не буде досягати свого максимально можливого значення. Однак, програма динамічно визначає обсяг для свого буфера обміну, залежно від швидкості з'єднання й інших параметрів. Таким чином, програма віддаленого керування комп'ютером надає максимально можливу швидкість для передачі файлів.

Функція передачі файлів інших програм передає кожний файл як окреме мережне повідомлення й чекає звіту про його доставку. І якщо потрібно відправити велику кількість файлів, розмір яких невеликий, – це значно понизить швидкодію, тому що програма, як уже було зазначено вище, запитує звіт після доставка кожного файлу на віддалений комп'ютер. Наша ж програма працює по іншому принципу. Вона може посилати кілька файлів одночасно в одному мережному повідомленні й чекає звіт уже від всієї групи файлів. Програма може розмістити стільки маленьких файлів в одному мережному повідомленні, скільки це дозволяє зробити обсяг буфера даних. Це значно підвищує швидкодію програми завдяки зменшенню кількості мережних повідомлень (і звітів про доставку) через мережу.

Альтернатива "Мережному оточенню" Microsoft

Якщо використовуєте "Мережне оточення" Microsoft для доступу до файлів через мережу, – потрібно відкривати для папок загальний доступ, щоб вони були видимими для всіх користувачів мережі. Однак робити папки загальнодоступними аж ніяк небезпечно.

Використовуючи програму адміністрування віддаленим комп'ютером зовсім необов'язково робити свої папки загальнодоступними. Є повний доступ до файлів на віддаленому ПК, але в "Мережному оточенні" ці папки залишаться недоступними для загального користування. На жаль, у більшості мереж "Мережне оточення" працює дуже повільно. Іноді доводиться швидко знайти який-небудь віддалений файл, але "Мережне оточення" просто не дозволяє цього

зробити, змушуючи чекати тривалий час. Програма адміністрування віддаленим комп'ютером пропонує миттєвий доступ до віддалених файлів і високошвидкісну передачу файлів, що однозначно є перевагою.

Сервіс "Пряме підключення"

Віддалене підключення через ЛОМ або Інтернет

"Пряме підключення" дозволяє підключатися до віддаленого ПК через IP адресу або DNS ім'я. Цей спосіб підключення доступний як через ЛОМ, так і через Інтернет.

Для успішного підключення, Host комп'ютер повинен мати зовнішній статичний IP адресу й правильно настроєні маршрутизатор і брендмауер.

При Прямому підключенні програма адміністрування віддаленим комп'ютером повинна бути встановлена принаймні на двох різних комп'ютерах (один ПК – Адмін, і другий – Host). В Admin модулі потрібно додати в адресну книгу IP адресу Host комп'ютера, або його DNS ім'я. Якщо обидва комп'ютери перебувають в одній локальній мережі, тоді IP адреса комп'ютера, яким хочете управляти, не знадобиться оскільки можна підключитися через ім'я комп'ютера в мережі.

Можливі проблеми і їхні рішення

Якщо обидва комп'ютери перебувають в одній ЛОМ, тоді проблем з підключенням не буде (хіба що із брендмауером, якщо його використовуєте). У більшості випадків можна також підключиться також і через Інтернет. Однак, якщо не вдається встановити підключення через Інтернет, додержуйтеся нижчеподаних інструкцій.

Динамічні IP адреси

Залежно від вашого Інтернет провайдеру, комп'ютер може мати постійно мінливу IP адресу. Цей вид адреси називається "динамічний", і характерний для більшості провайдерів, що роблять послуги з доступу в Інтернет. Буде важко встановити підключення з віддаленим комп'ютером, що має IP адресу яку не

знаєте. Однак є метод, що допоможе вирішити проблему з динамічними IP адресами.

Внутрішні IP адреси

Повсякденна ситуація, коли комп'ютер, до якого хочете підключитися, перебувє в корпоративній або домашній локальній мережі (за маршрутизатором). Як правило, комп'ютери, що перебувають усередині однієї ЛОМ, мають IP адреси, подібні такому 192.168.x.x. Ця адреса дійсна тільки в межах індивідуальних ЛОМ. таким чином, не зможете підключитися до цього комп'ютера через Інтернет, використовуючи внутрішній IP адреса. Проблему вирішує налаштування маршрутизатора.

Проблеми із брендмауером

Брендмауер може блокувати як зовнішні, так і внутрішні підключення з порту 5279 і порту 4279. Тому потрібно відкрити ці порти.

З'єднання через Proxy сервер

Якщо підключені до Інтернету через Proxy сервер, можуть виникнути труднощі із програмою адміністрування віддаленим комп'ютером під час установки підключення.

Сервіс "Account-підключення"

Безперешкодне віддалене керування через Інтернет

Програма адміністрування віддаленим комп'ютером пропонує легкий шлях керування віддаленим ПК через Інтернет. Основне розходження між нашим продуктом і продуктами інших виробників – рівень практичного використання. Безліч продуктів, що пропонують віддалене керування через Інтернет, насправді марні, тому що на практиці їх часто неможливо використовувати через ряд технічних проблем. У програмі адміністрування віддаленим комп'ютером ці проблеми вже вирішені завдяки функції "Account-підключення":

– Доступ через Інтернет до комп'ютерів, що не мають зовнішньої статичної IP адреси. Можна легко підключатися до комп'ютерів із внутрішнім (192.168.x.x) або динамічним IP адресою, без яких-небудь мережних

налаштувань. Більше того, взагалі не потрібно знати IP адресу віддаленого ПК. Підключаєтеся через ім'я облікового запису й імена комп'ютера, замість підключення через IP адресу.

– Обхід маршрутизаторів і брандмауерів. З легкістю можна підключатися до комп'ютерів, що перебуває за брандмауером і маршрутизатором, без необхідності їхнього налаштування. Не потрібно буде й відкривати порти брандмауерів.

Як працює служба Account-підключення

Host і Admin комп'ютери підключаються до нашого Gateway-серверу по http протоколу й постійно підтримують зв'язок з ним. Брандмауери й маршрутизатори не будуть блокувати таке з'єднання, тому що ви заходите в Інтернет як звичайно, – через http протокол. Після того, як з'єднання із сервером успішно встановлено, можна підключати Admin комп'ютер до Host комп'ютера через наш Gateway сервер. Інакше кажучи, Admin комп'ютер і Host комп'ютер з'єднуються між собою не прямо, а через наш Gateway сервер, що виконує роль посередника під час сесії підключення. Ця технологія вирішує проблему з IP адресами, маршрутизаторами й брандмауерами, і має подібність із ICQ.

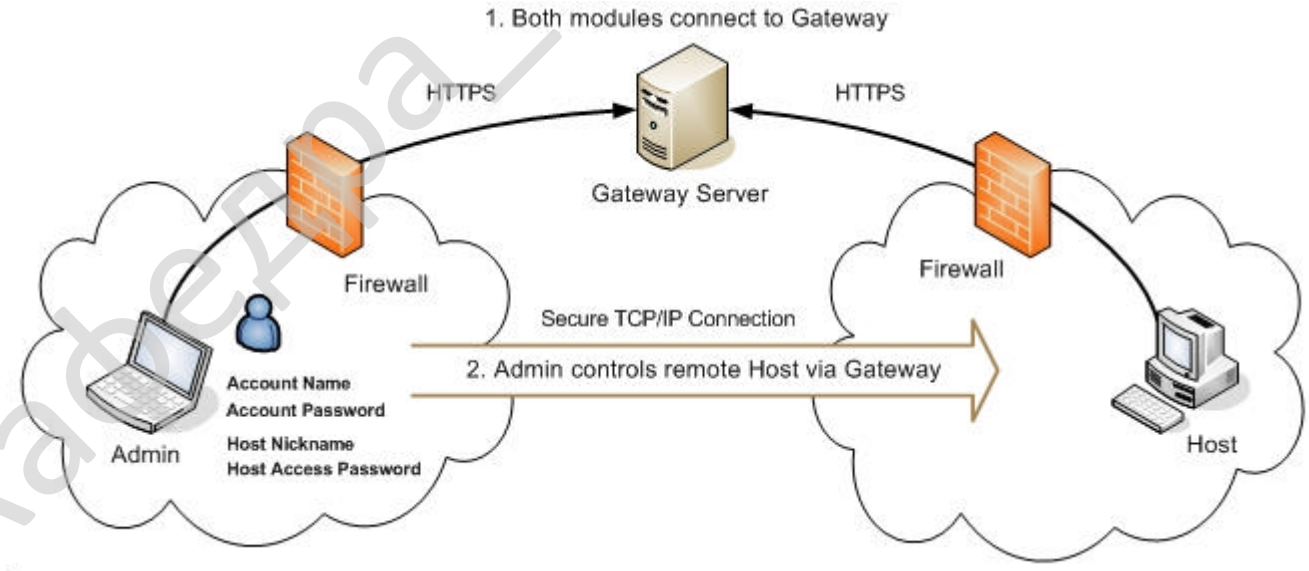

Рисунок 3.1 – Схема підключення через Gateway сервер

Реалізація служби Account-підключень у програмі адміністрування віддаленим комп'ютером

Щоб використовувати службу "Account-підключення" знадобитися як мінімум установити програму адміністрування віддаленим комп'ютером на 2 комп'ютери (Admin комп'ютер і Host комп'ютер).

Всі комп'ютери об'єднані в логічну групу – Account (обліковий запис). Кожний комп'ютер в обліковому записі має свої індивідуальні імена.

Під час установки програми адміністрування віддаленим комп'ютером можна створити новий обліковий запис або підключитися до уже існуючій. Щоб зайти в обліковий запис, потрібно знати її ім'я й пароль.

Після того, як додали свій Host комп'ютер в обліковий запис, можна легко підключитися до нього через Інтернет. Просто запустіть Admin-модуль і в закладці Account-підключення побачите список Host комп'ютерів облікового запису. Також можна бачити статус цих комп'ютерів (online/offline). Ваш обліковий запис майже ідентичний "Мережним оточенням" Windows. Це різновид віртуальної частної мережі. Беріть комп'ютер зі списку; натисніть кнопку Підключитися – і підключитеся.

Безпека

Незважаючи на те, що комп'ютери обмінюються даними через наш сервер, процедура ця абсолютно безпечна й конфіденційна.

Наша система використовує абсолютно безпечні алгоритми. Системи безпеки міжнародних банків при перекладі коштів через Інтернет використовують такі ж алгоритми, що й програма.

Безпека нашої системи здійснюється наступними технологіями:

– Безпечна ідентифікація й шифрування трафіку. Всі дані, які передаються через наш сервер, надійно шифруються. Ніхто, включаючи штат розроблювачів Anyplace-Control, не зможе розшифрувати ці дані у випадку їхнього перехоплення. Програма використовує CHAP алгоритм (тип

автентифікації з попереднім узгодженням виклику) для ідентифікації, а також RC4 алгоритм із 128-бітним випадковим ключем шифрування.

– Захист за допомогою подвійного пароля. Щоб підключитися до віддаленого ПК через Інтернет, потрібно знати як мінімум 2 паролі. Перший – це пароль облікового запису, що використовується для входу в облікові записи. Він дозволяє переглядати список доданих в обліковий запис комп'ютерів. Другий (пароль доступу) – більше важливий, тому що буде потрібно для віддаленого доступу на Host комп'ютер. Навіть якщо хто-небудь знає ваш пароль облікового запису, він не зможе підключитися до Host комп'ютера, не знаючи пароль доступу. Тому, зберігаєте, будь ласка, свій Пароль Доступу в безпеці, – і тоді не прийдеться хвилюватися за схоронність своїх даних.

– Засекреченість ваших даних. Ми не зберігаємо пароль доступу й особистих дані інших користувачів на своєму сервері.

– Ніяких відкритих портів у брандмауері. Щоб використовувати службу Інтернет з'єднань, не потрібно відкривати додаткові порти в брандмауері, отже, не створюєте "діри" у своїй системі безпеки.

Розповсюджені проблеми користувачів, розв'язувані програмою адміністрування віддаленим комп'ютером :

– Динамічні IP адреси. Залежно від Інтернет провайдеру, комп'ютер може мати постійно мінливу Інтернет адресу. Цей тип адрес називається "динамічний" і типовий для більшості провайдерів, що роблять послуги з доступу в Інтернет. Якщо хочете підключатися до віддаленого ПК, не знаючи його поточної IP адреси, тоді прийдеться зробити ряд складних налаштувань.

– Частні (внутрішні) IP адреси. Звичайна ситуація, коли комп'ютер, до якого хочете підключитися, перебуває в корпоративній або домашньій ЛОМ (за маршрутизатором або proxy сервером). Як правило, комп'ютери, що перебувають в одній ЛОМ, мають тільки внутрішні IP адреси, начебто цього 192.168.x.x. Ці адреси дійсні тільки у ЛОМ, але не дійсні в Internet. Таким чином, не зможете підключитися до цього комп'ютера через Інтернет, використовуючи внутрішній

IP адреса. У нашій програмі є сервіс "Account-підключення", що автоматично вирішує проблеми з IP адресами, тому що з'єднання між комп'ютерами реалізується не через IP адреси, а через ім'я облікового запису й ім'я комп'ютера.

– Дуже часто перед користувачами встає проблема налаштування маршрутизатора. Це стандартний шлях до рішення проблеми з підключенням через Інтернет до комп'ютера, що має тільки внутрішній IP адреса, не прибігаючи до використання нашої служби Account-підключень. Системний адміністратор швидше за все відмовиться налаштувати переадресацію портів, тому що це може привести до проблеми з безпекою. Наша служба Інтернет з'єднань дозволяє підключатися до віддаленого ПК, що перебуває у ЛОМ, без яких-небудь налаштувань маршрутизатора.

– Проблеми із брендмауером. Продукти інших виробників змушують брандмауери відкривати порти, що приводить до проблем з безпекою. Служба Account підключень не змушує відкривати порти брандмауера. Таким чином, ми вирішуємо питання з безпекою, тим самим не змушуючи прибігати до мудрованих налаштувань брандмауера

3.2 Розробка структурної схеми

На рисунку 3.2 зображена структурна схема розробленого програмного забезпечення віддаленого керування комп’ютером.

З неї ми бачимо, що розроблена система складається з двох структурних модулів:

– Admin Модуль – установлюється на комп'ютер, з якого буде відбуватися контролювання віддаленого ПК.

– Host Модуль – установлюється на комп'ютер, що буде контролюватися.

До одного Admin Модуля можливо приєднання декількох Host Модулів.

Інформація яка передається між модулями шифрується з використанням алгоритмів AES та SSL/TLS.

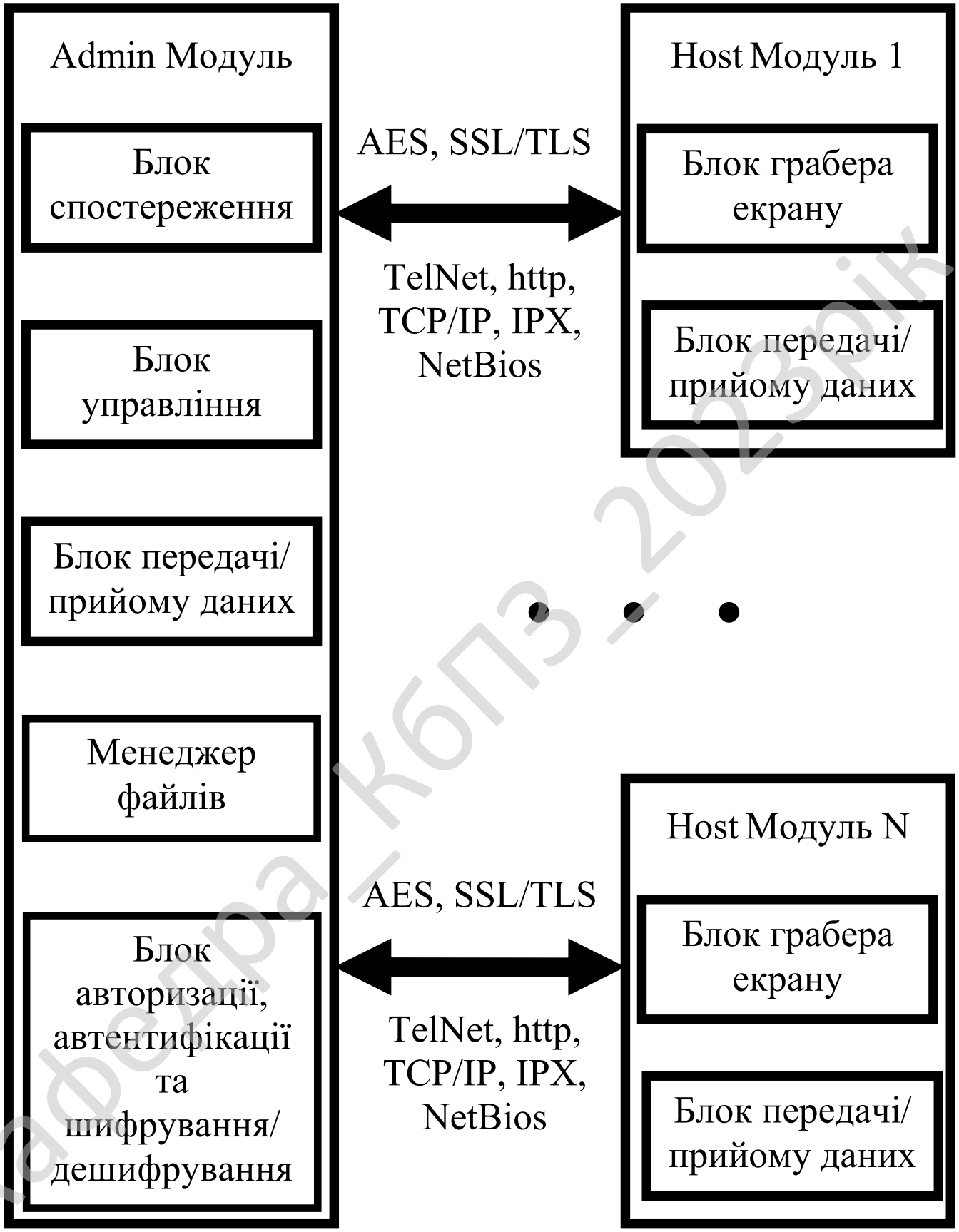

Рисунок 3.2 – Структурна схема розробленої системи

Для реалізації передачі даних використовуються наступні протоколи:

– TelNet.

– http.

– TCP/IP.

– IPX.

– NetBios.

На серверній частині (Admin Модуль) розташовані наступні структурні блоки:

– Блок спостереження.

– Блок управління.

– Блок передачі/прийому даних.

– Менеджер файлів.

– Блок авторизації, автентифікації та шифрування/дешифрування.

На клієнтській частині (Host Модуль) розташовані наступні структурні блоки:

– Блок грабера екрану.

– Блок передачі/прийому даних.

Розглянемо більш докладно функції серверної й клієнтської частини програми віддаленого адміністрування комп'ютера.

Програма віддаленого адміністрування комп'ютером – Серверний модуль

Серверний модуль програми віддаленого адміністрування комп'ютером повинен бути встановлений на віддалених комп'ютерах, до яких необхідно підключитися. Він завжди запускається як сервіс і автоматично завантажується при запуску Windows.

При спробі підключення до віддаленого комп'ютера, серверний модуль 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 стандарті AES.

При підключенні серверного модуля програми віддаленого адміністрування комп'ютером забезпечується повний ефект присутності за екраном віддаленого комп'ютера. Програма віддаленого адміністрування комп'ютером містить у собі технологію DirectScreenTransfer™. Дана розробка дозволяє серверному модулю зчитувати дані з екрана віддаленого комп'ютера в обхід відеоконтролерів, знижуючи тим самим навантаження на процесор. Програма віддаленого адміністрування комп'ютером використовує інтелектуальний алгоритм, що дозволяє зчитувати відновлення даних. Дана технологія не тільки забезпечує можливість максимального збільшення швидкості відновлення екрана, але й знижує витрата мережного трафіку. Це особливо важливо, коли підключення до віддаленого комп'ютера здійснюється при роботі з повільним Інтернет-З'єднанням або з'єднанням через модем.

Дана частина програми віддаленого адміністрування комп'ютером працює також як сервер для підтримки можливостей текстового й голосового чата. У системі безпеки використовуються ті ж алгоритми безпеки й автентифікації, що й у режимі Full Control.

Серверний модуль підтримує багато можливостей, серед яких точна передача форми й анімації курсору, коректна передача зображення з декількох моніторів, і багато чого іншого:

– 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 віддалених комп'ютерів.

– Вбудований багатокористувальницький текстовий і голосовий чат.

– Підтримка миттєвого одержання коротких повідомлень від клієнтського модуля.

– 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 мережний протокол.

Програма віддаленого адміністрування комп'ютером – Клієнтський модуль

На додаток до своєї основної функції, передачі зображення на інший комп'ютер, програма віддаленого адміністрування комп'ютером надає адміністраторам і постійним користувачам можливість виконувати й інші завдання.

Програма віддаленого адміністрування комп'ютером істотно спрощує процес копіювання файлів з одного комп'ютера на іншій. Використовуючи режим передачі файлів, програма віддаленого адміністрування комп'ютером, можете копіювати файли з/на віддалений комп'ютер, що не ставиться до Вашої мережі. Раніше використовували проміжний сервер або сервер LAN, експериментували з e-mail додатками або записували файли вручну на CD/DVD-RW, flash або флоппі-диски. Програма віддаленого адміністрування комп'ютером дозволяє копіювати файли прямо.

Іншою корисною особливістю програми віддаленого адміністрування комп'ютером є можливість підключення до віддаленого комп'ютера в режимі Telnet. Це дозволить 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 працюючий 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.

Клієнтський модуль пропонує оновлений інтерфейс у стилі Vista/XP і безліч корисних можливостей:

– Підтримка функцій Intel® AMT (Active Management Technology).

– Повна сумісність із Windows 7 і Windows Vista.

– Зручний інтерфейс і вбудована довідкова система.

– Вбудований сканер серверів програми віддаленого адміністрування комп'ютером.

– 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».

– Технологія DirectScreenTransfer™, що забезпечує максимальну швидкість.

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

– Обмеження на максимальний дозвіл дисплея відсутній.

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

– Повноекраний, 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.

– Двустороння робота з буфером обміну з підтримкою Unicode.

– Сполучення клавіш, що набудовуються, під команди Програма віддаленого адміністрування комп'ютером 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 введення пароля.

Виходячи з розгляду перерахованих вище можливостей, перейдемо до побудови функціональної схеми розробленого, у результаті виконання бакалаврського проекту, програмного забезпечення.

3.3 Розробка функціональної схеми

Функціонально розроблене програмне забезпечення складається з наступних блоків:

1. Блок спостереження.
2. Блок управління.
3. Блок обмежень прав користувача на Host.
4. Блокування клавіатури.
5. Блокування миші.
6. Керування службами.
7. Перегляд в повноекранному режимі.

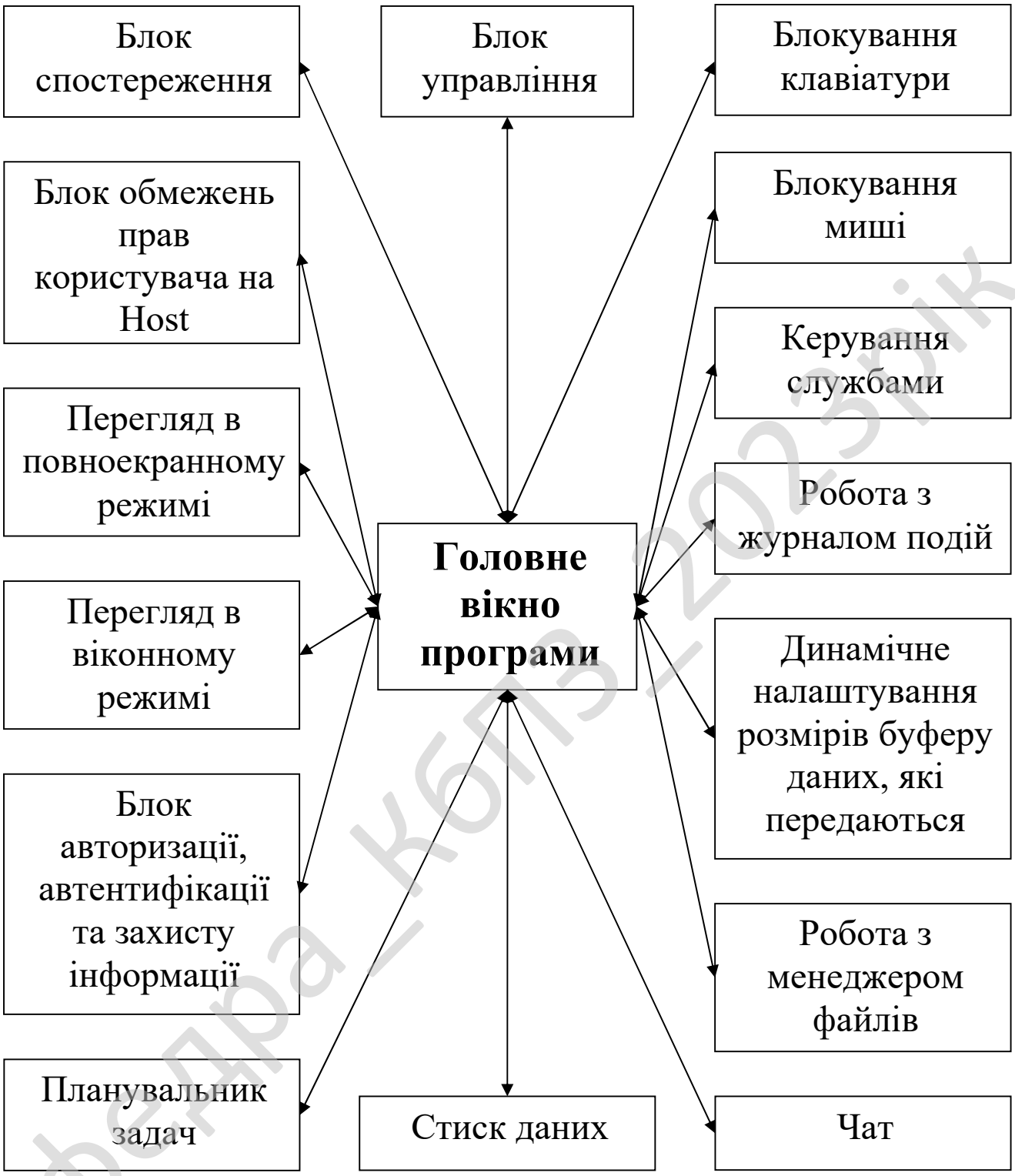

Рисунок 3.3 – Функціональна схема розробленої системи

8. Перегляд в віконному режимі.

9. Робота з менеджером файлів.

10. Робота з журналом подій.

11. Динамічне налаштування розмірів буферу даних, які передаються.

12. Блок авторизації, автентифікації та захисту інформації.

13. Стиск даних.

14. Планувальник задач.

15. Чат.

3.4 Розробка діаграми процесів

На рисунку 3.4 показана діаграма процесів розробленої системи.

З неї ми бачимо що у системі відбуваються наступні процеси.

Спершу запускається процес початку/кінця роботи програми, який взаємодіє з наступними процесами:

– З'єднання з мережею.

– Роз'єднання з мережею.

Процес з'єднання з мережею взаємодіє з процесом виведення списку комп’ютерів. Цей процес виводить усі комп’ютери які є у мережі.

Цей процес а також процес роз'єднання з мережею взаємодіють з процесом вибору комп’ютера зі списку.

Крім цих процесів, процес вибору комп’ютера зі списку взаємодіє з наступними процесами:

– Процес перегляду файлів та каталогів підлеглого комп’ютера.

– Процес виконання дії.

– Процес спостереження.

– Процес відправки повідомлень.

Процес перегляду файлів та каталогів підлеглого комп’ютера взаємодіє з наступними процесами:

– Процес завантаження файлів, призначений для завантаження файлів з підлеглого комп’ютера.

– Процес відправки файлів, призначений для відправки файлів на підлеглий комп’ютер.

Рисунок 3.4 – Діаграма процесів системи

Процес виконання дії взаємодіє з наступними процесами:

– Процес вибору дії, до яких відносяться наступні:

а) обмеження прав користувача на host.

б) блокування клавіатури.

в) блокування миші.

г) керування службами.

д) перегляд в повноекранному режимі.

е) перегляд в віконному режимі.

ж) робота з менеджером файлів.

з) робота з журналом подій.

и) динамічне налаштування розмірів буферу даних, які передаються.

к) авторизація, автентифікація та захист інформації.

л) стиск даних.

м) планувальник задач.

н) чат.

– Процес встановлення часу виконання дії, який призначений для контролю над часом виконання дії.

Процес спостереження взаємодіє з наступними процесами:

– Процес перегляду списку встановлених служб.

– Процес перегляду списку запущених процесів.

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.

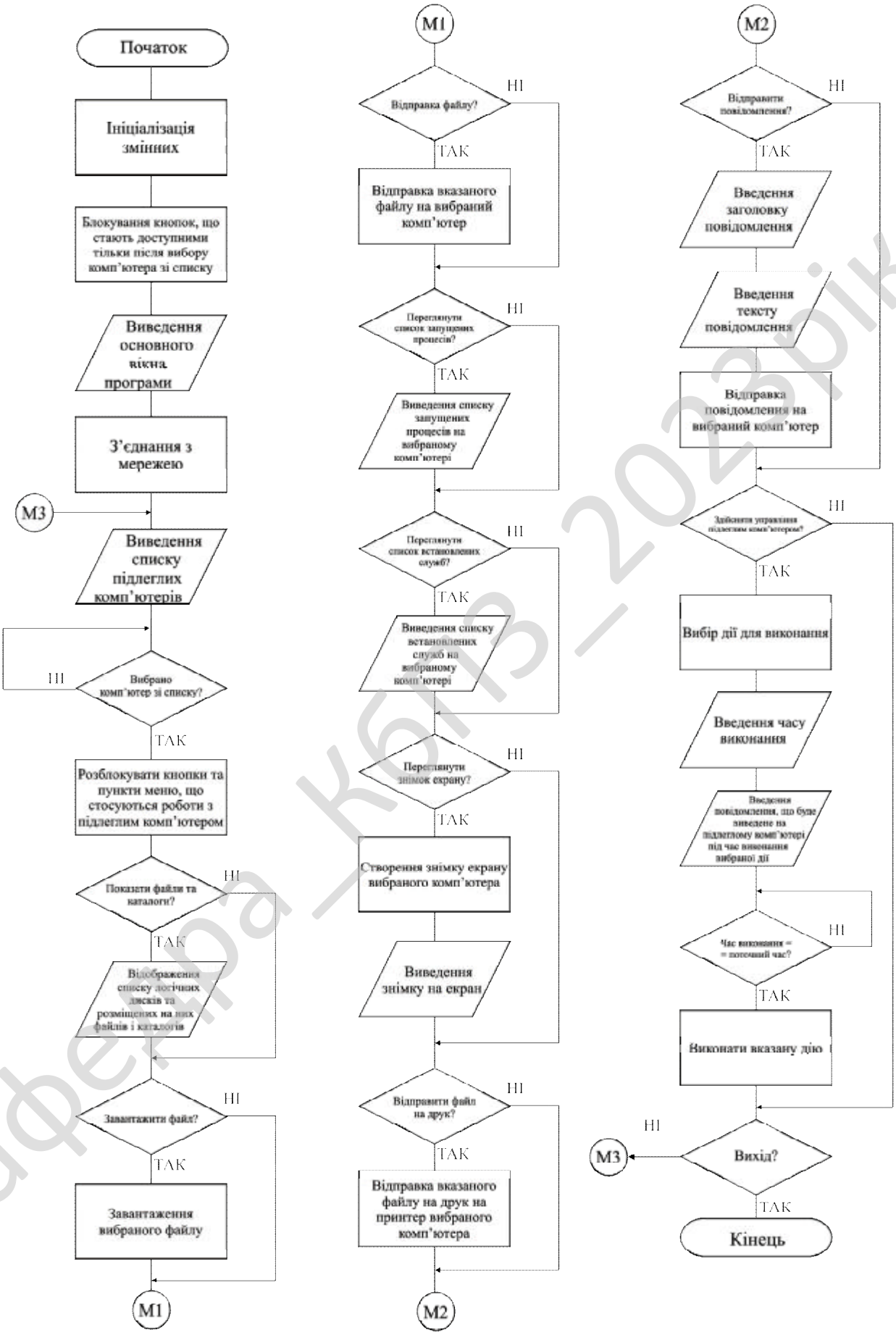

Рисунок 4.1 – Блок-схема роботи основної програми

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.

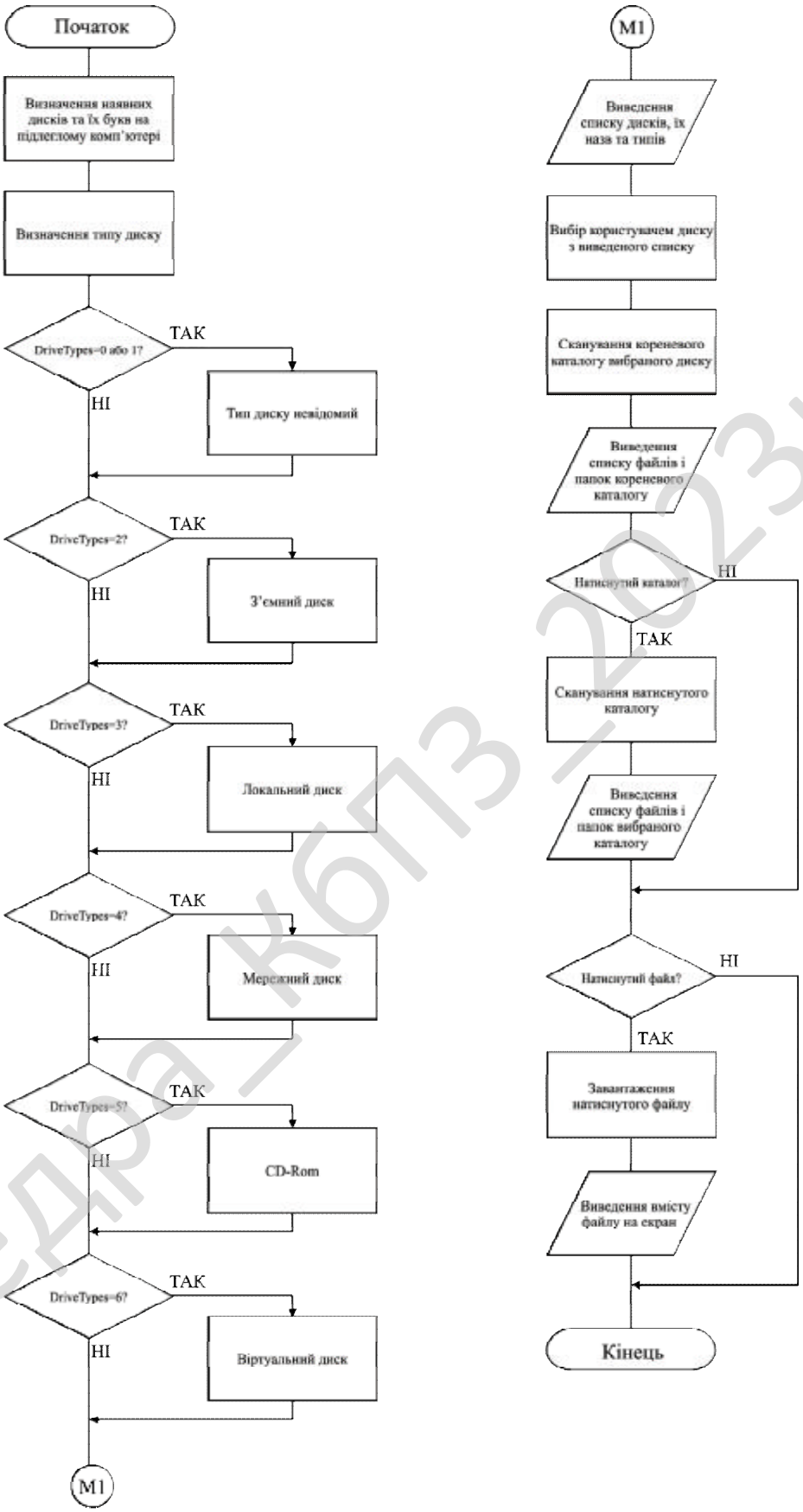

Рисунок 4.2 – Блок-схема роботи підпрограми перегляду файлів та папок підлеглого комп'ютера

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 = 3 то локальний диск.

Якщо DriveTypes = 4 то мережний 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.

Розглянемо методику написання програмного забезпечення адміністрування віддаленим комп’ютером по протоколу TCP/IP.

Тут я буду використовувати стандартний компонент TServerSocket, небагато перероблений для того, щоб він не використовував TForms. Я для простоти використовував TServerSocket, через цього моя програма збільшилася

до 89 кбайт, тому що Unit ScktComp використовує SysUtils і Classes. Але якщо створити свій Сервер, використовуючи Unit Winsock, те можна дуже здорово зменшити свою програму. Замість TTimer потрібно в методі Create використовувати tim:=SetTimer(0,0,600000,@TRojan.mail);

Де:

```
tim: integer
```

@TRojan.mail покажчик на процедуру обробки.

Убивається в Destroy.

```
KillTimer(0,tim);
```

Почнемо. Створимо новий проект. Видалимо всі стандартні Unit, які входять у новий проект. Зробимо так, щоб файл Project1.dpr виглядав так:

```
program Project1;
uses windows, Unit1, messages ;
var
msg: TMsg;
sock: TRojan;
begin
  sock:= TRojan.Create;
       while GetMessage( Msg, HInstance, 0, 0) do
       begin
         TranslateMessage(msg);
         DispatchMessage(msg);
       end;
end.
```

Тут іде зациклення програми, і всі повідомлення передані програмі передаються в клас.

а unit1.pas щоб виглядав так.

```
unit Unit1;
interface
uses
  Windows, Messages;
type
     TRojan= class
       private
          { Private declarations }
       public
          { Public declarations }
```

```
        end;
implementation
end.
```

Тепер збережемо в якій-небудь директорії наш проект. Закриємо його й відкриємо заново. Якщо відкомпілювати, то одержимо програму обсягом 15 кілобайт. Додамо в директорію проекту файл ScktComp.pas з папки Source Delphi. Додамо в Unit1 Unit ScktComp. Видалимо з нього посилання на Forms у другому uses. Спробуємо скомпілювати. Тут використовуються деякі функції, що перебувають у Unite Forms. При компіляції одержали неправильні ідентифікатори DeallocateHWnd, AllocateHwnd і Application.HandleException(Self). Заремарим Application.HandleException(Self) а процедури DeallocateHWnd і AllocateHwnd додамо в ScktComp.pas з Unit Forms. Додамо оголошення цих функцій:

```
TSocketErrorProc
SetErrorProc(ErrorProc: TSocketErrorProc):
implementation
```

видалимо:

```
var
WSAData: TWSAData;
```

додамо сюди це:

```
var
WSAData: TWSAData; // це було
InstBlockList: PInstanceBlock; // це й далі додали
InstFreeList: PObjectInstance;
UtilWindowClass: TWndClass = (
style: 0;
lpfnWndProc: @DefWindowProc;
cbClsExtra: 0;
cbWndExtra: 0;
hInstance: 0;
hIcon: 0;
hCursor: 0;
hbrBackground: 0;
lpszMenuName: nil;
lpszClassName: 'TPUtilWindow');
procedure FreeObjectInstance(ObjectInstance: Pointer);
begin
```

```
if ObjectInstance <> nil then
begin
PObjectInstance(ObjectInstance)^.Next := InstFreeList;
InstFreeList := ObjectInstance;
end;
end;

function StdWndProc(Window: HWND; Message, WParam: Longint;
LParam: Longint): Longint; stdcall; assembler;
asm
XOR EAX,EAX
PUSH EAX
PUSH LParam
PUSH WParam
PUSH Message
MOV EDX,ESP
MOV EAX,[ECX].Longint[4]
CALL [ECX].Pointer
ADD ESP,12
POP EAX
end;

function CalcJmpOffset(Src, Dest: Pointer): Longint;
begin
Result := Longint(Dest) - (Longint(Src) + 5);
end;
function MakeObjectInstance(Method: TWndMethod): Pointer;
const
BlockCode: array[1..2] of Byte = (
$59, { POP ECX }
$E9); { JMP StdWndProc }
PageSize = 4096;
var
Block: PInstanceBlock;
Instance: PObjectInstance;
begin
if InstFreeList = nil then
begin
Block := VirtualAlloc(nil, PageSize, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
Block^.Next := InstBlockList;
Move(BlockCode, Block^.Code, SizeOf(BlockCode));
Block^.WndProcPtr := Pointer(CalcJmpOffset(@Block^.Code[2], @StdWndProc));
```

```
Instance := @Block^.Instances;
repeat
Instance^.Code := $E8; { CALL NEAR PTR Offset }
Instance^.Offset := CalcJmpOffset(Instance, @Block^.Code);
Instance^.Next := InstFreeList;
InstFreeList := Instance;
Inc(Longint(Instance), SizeOf(TObjectInstance));
until Longint(Instance) - Longint(Block) >= SizeOf(TInstanceBlock);
InstBlockList := Block;
end;
Result := InstFreeList;
Instance := InstFreeList;
InstFreeList := Instance^.Next;
Instance^.Method := Method;
end;

function AllocateHWnd(Method: TWndMethod): HWND;
var
TempClass: TWndClass;
ClassRegistered: Boolean;
begin
UtilWindowClass.hInstance := HInstance;
ClassRegistered := GetClassInfo(HInstance, UtilWindowClass.lpszClassName,
TempClass);
if not ClassRegistered or (TempClass.lpfnWndProc <> @DefWindowProc) then
begin
if ClassRegistered then
Windows.UnregisterClass(UtilWindowClass.lpszClassName, HInstance);
Windows.RegisterClass(UtilWindowClass);
end;
Result := CreateWindowEx(WS_EX_TOOLWINDOW, UtilWindowClass.lpszClassName,
'', WS_POPUP {!0}, 0, 0, 0, 0, 0, 0, HInstance, nil);
if Assigned(Method) then
SetWindowLong(Result, GWL_WNDPROC, Longint(MakeObjectInstance(Method)));
end;

procedure DeallocateHWnd(Wnd: HWND);
var
Instance: Pointer;
begin
Instance := Pointer(GetWindowLong(Wnd, GWL_WNDPROC));
DestroyWindow(Wnd);
```

```
if Instance <> @DefWindowProc then FreeObjectInstance(Instance);
end;
```

Додамо після:

```
const
CM_SOCKETMESSAGE = WM_USER + $0001;
CM_DEFERFREE = WM_USER + $0002;
```

оголошення додаткових типів:

```
type
TWndMethod = procedure(var Message: TMessage) of object;

PObjectInstance = ^TObjectInstance;
TObjectInstance = packed record
Code: Byte;
Offset: Integer;
case Integer of
0: (Next: PObjectInstance);
1: (Method: TWndMethod);
end;
PInstanceBlock = ^TInstanceBlock;
TInstanceBlock = packed record
Next: PInstanceBlock;
Code: array[1..2] of Byte;
WndProcPtr: Pointer;
Instances: array[0..313] of TObjectInstance;
end;
```

Заремарим ще один рядок Application.ShowException(FException) і скомпілюємо. Обсяг програми склав 57 кілобайт.

Оголосимо в нашім класі процедури Create і Destroy і компонент TServerSocket:

```
TRojan= class
     Server:TServerSocket;
     constructor Create;
     destructor Destroy; override;
```

Опишемо ці функції:

```
constructor TRojan.Create;
begin
     Server:=TServerSocket.Create(nil);
     Server.OnClientRead:=ClientRead;
     Server.OnClientError:=ServerSocketClientError;
```

```
        Server.Port:=23;
        Server.Active:=true;
    end;
    destructor TRojan.Destroy;
    begin
    Server.Free;
    inherited Destroy;
    end;
```

Додамо процедури для роботи із сервером у наш клас.

```
    procedure ClientRead(Sender: TObject; Socket: TCustomWinSocket);
    procedure      ServerSocketClientError(Sender:      TObject;      Socket:
TCustomWinSocket; ErrorEvent: TErrorEvent;
    var ErrorCode: Integer);
```

На ці процедури сервер буде передавати керування. Опишемо дії при виникненні помилки

```
    procedure    TRojan.ServerSocketClientError(Sender:    TObject;    Socket:
TCustomWinSocket; ErrorEvent: TErrorEvent;
    var ErrorCode: Integer);
    begin
    if errorcode= 10054 then errorcode := 0;// або можна щоб взагалі не було
ніяких реакцій на помилки
    //   errorcode := 0;
    end;
```

І при одержанні рядка від клієнта.

```
    procedure TRojan.ClientRead(Sender: TObject; Socket: TCustomWinSocket);
    var
      s:string;
    begin
      s:=Socket.ReceiveText;
      if Uppercase(s) = 'HELP' then
        begin
          Socket.SendText('I'm remote server version 1.0 designed HaWord
(c)2001'#13#10);
          Socket.SendText('HELP – help :)'#13#10);
          Socket.SendText(#13#10);
        end;
    if Uppercase(clientdata) = 'STOP' then
        begin
          Socket.SendText(                    '#13#10);
```

```
        PostQuitMessage(WM_QUIT);
        end;
end;
```

Додамо вітання. Оголосимо в класі процедуру.

```
procedure ClientConnect(Sender: TObject; Socket: TCustomWinSocket);
```

Опишемо оброблювач.

```
procedure TRojan.ClientConnect(Sender: TObject; Socket: TCustomWinSocket);
begin
  Socket.SendText('Привіт, я програма Віддаленого адміністрування'+#13#10);
end;
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм MMB, в основі якого лежить змішування операцій різних алгебраїчних груп. MMB – ітеративний алгоритм, що складається з лінійних дій (XOR і використання ключа) і паралельного застосування чотирьох великих оборотних нелінійних підстановок. Ці підстановки визначаються за допомогою множення по модулю 2^{32}-1 з постійними множниками. У підсумку з'являється алгоритм, що використовує 128-бітовий ключ і 128-бітовий блок.

Алгоритм MMB оперує 32-бітовими підблоками тексту (x_0, x_1, x_2, x_3) і 32-бітовими підблоками ключу (k_0, k_1, k_2, k_3). Це спрощує реалізацію алгоритму на сучасних 64-бітових процесорах. Чергуючись із операцією XOR, шість разів використовується нелінійна функція *f*. Запишемо операції алгоритму (всі операції з індексами виконуються по модулю 4):

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_i \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+1} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

$$x_i = x_i \oplus k_{i+2} \text{ для } i = 0..3$$

$$f(x_0, x_1, x_2, x_3)$$

Функція *f* виконується в три кроки:

1. $x_i = c_i * x_i$ для $i = 0..3$ (Якщо на вході множення одні одиниці, то на виході – теж одні одиниці).

2. Якщо молодший значущий біт $x_0 = 1$, то $x_0 = x_0 \oplus C$. Якщо молодший значущий байт $x_3 = 0$, то $x_3 = x_3 \oplus C$.

3. $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$ для $i = 0..3$.

Всі операції з індексами виконуються по модулю 4. Операція множення на кроці 1 виконується по модулі 2^{32}-1. Спеціальний випадок для даного алгоритму: якщо другий операнд дорівнює 2^{32}-1, результат теж дорівнює 2^{32}-1. В алгоритмі використовуються наступні константи:

C = 2aaaaaaa, c_0 = 025f1cdb, $c_1 = 2*c_0$, $c_2 = 2^3 * c_0$, $c_3 = 2^7 * c_0$.

Константа C – «найпростіша» константа без кругової симетрії, високою трійковою вагою й нульовим молодшим значущим бітом. У константи c_0 є інші особливі характеристики. Константи c_1, c_2 і c_3 – зрушені версії c_0, і служать для запобігання атак, заснованих на симетрії.

Розшифрування виконується у зворотному порядку, Етапи 2 і 3 інверсні їм самим. На етапі 1 замість c_i використовується c_i^{-1}. Значення c_0^{-1} = 0dad4694.

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 з мережею->Підключити**.

Для перегляду файлів та каталогів підлеглого комп'ютер, треба натиснути на потрібному комп'ютері у списку та вибрати пункт меню користувача **Управління комп'ютерами->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).

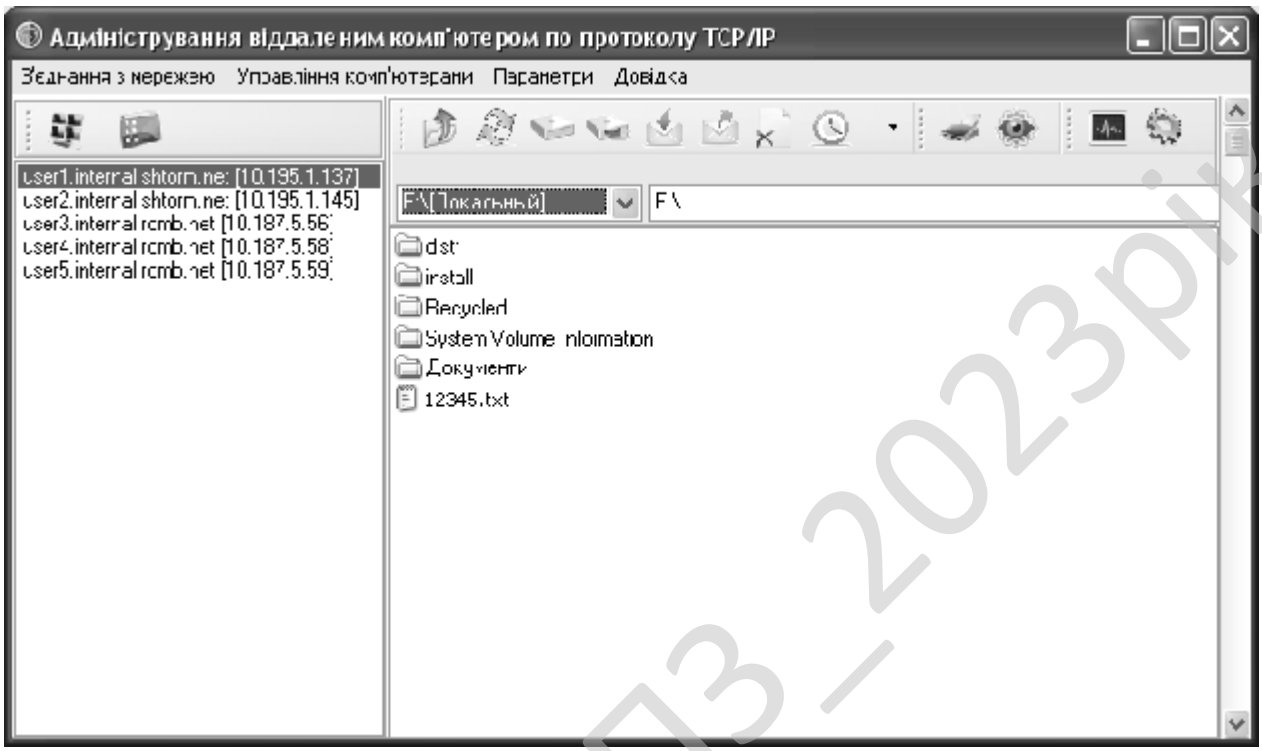

Рисунок 5.1 – Головне вікно програми (відображення файлів та каталогів підлеглого комп'ютера)

Для перегляду списоку служб підлеглого комп'ютера, треба вибрати пункт меню користувача **Управління комп’ютерами ->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.

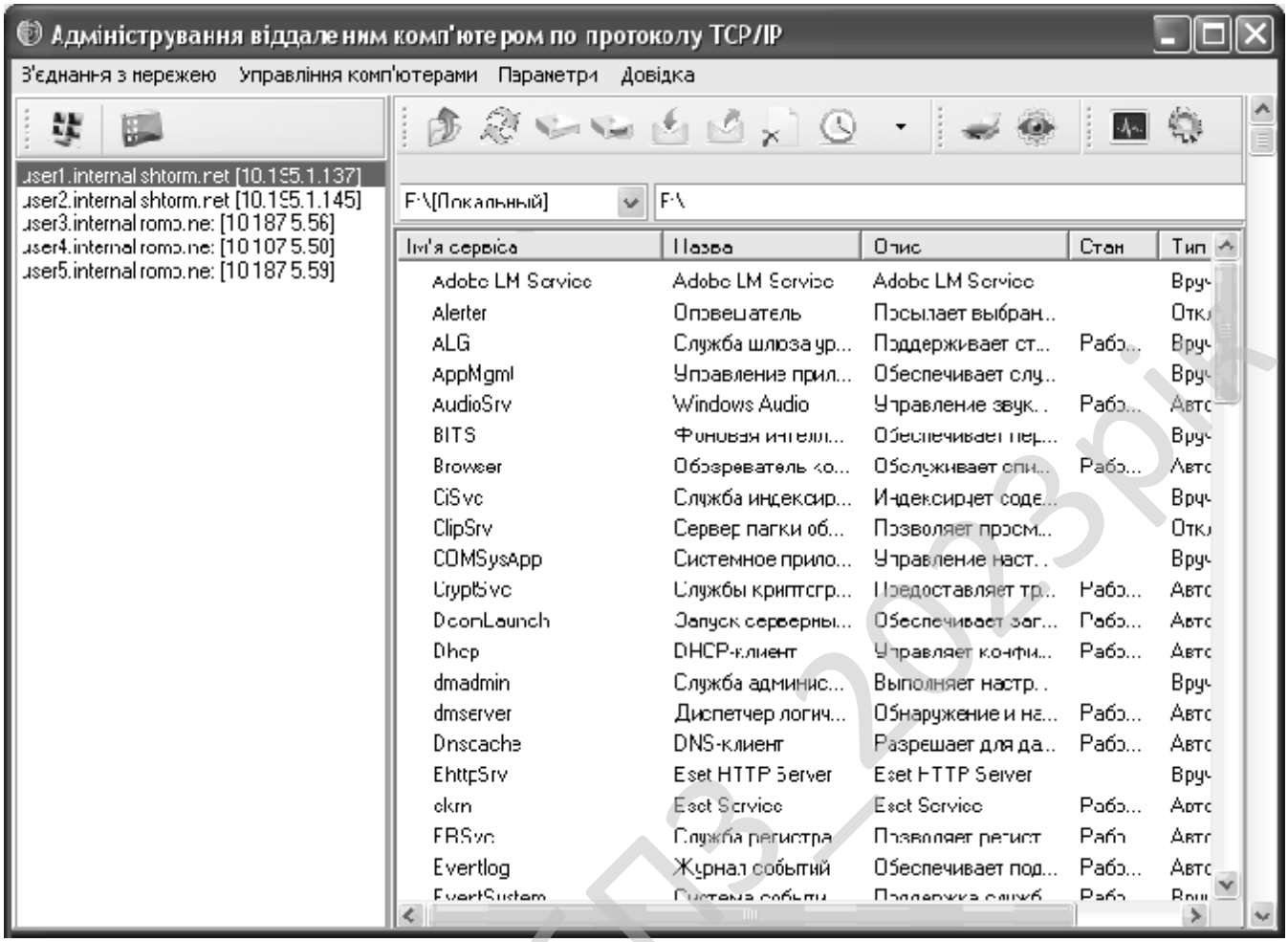

Рисунок 5.2 – Головне вікно програми (список служб підлеглого комп'ютера)

Для перегляду списоку запущених процесів на підлеглому комп'ютері, треба вибрати пункт меню користувача **Управління комп'ютерами->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.

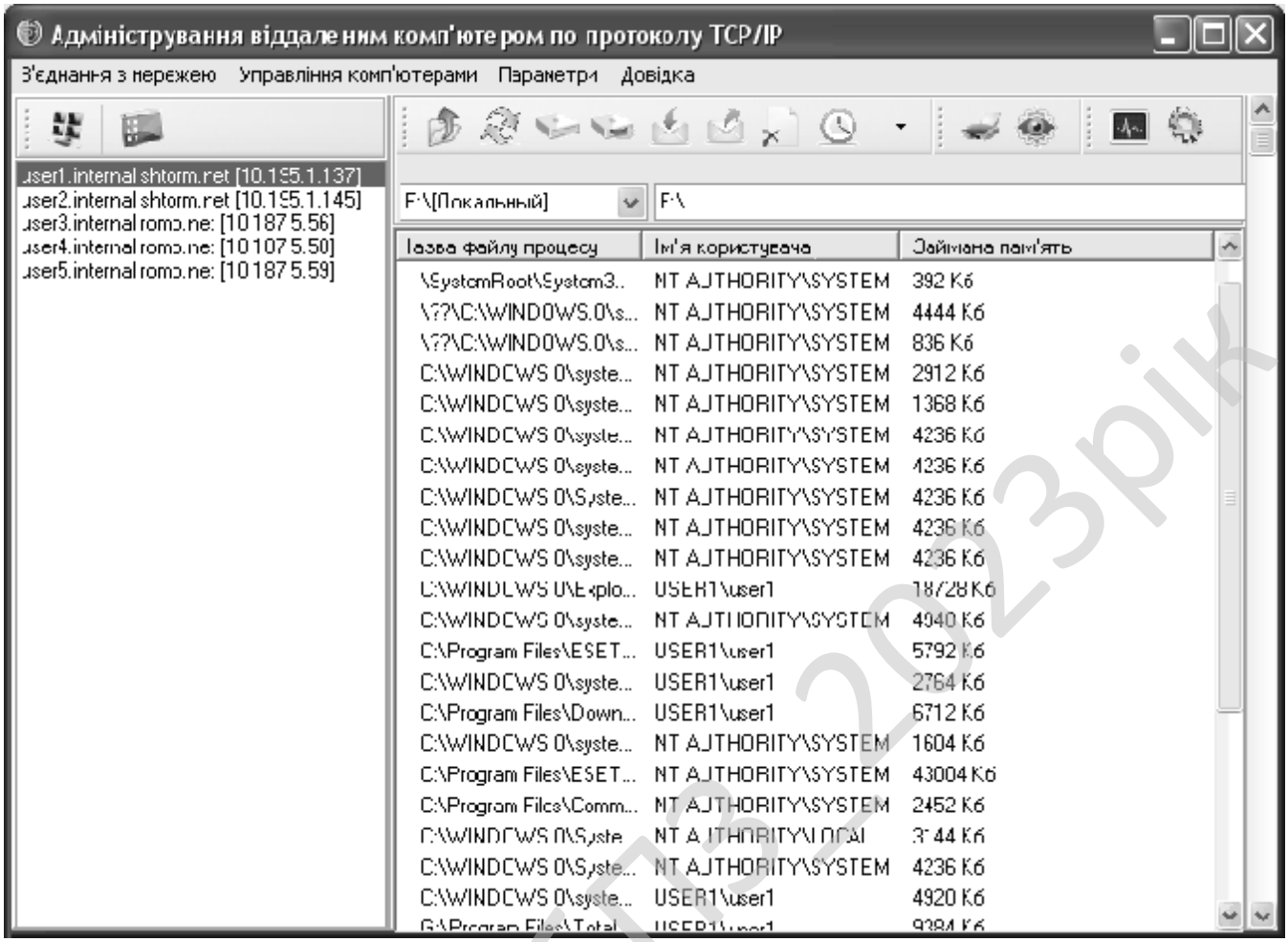

Рисунок 5.3 – Головне вікно програми (список запущених процесів на підлеглому комп'ютері)

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 **Управління комп'ютерами->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.

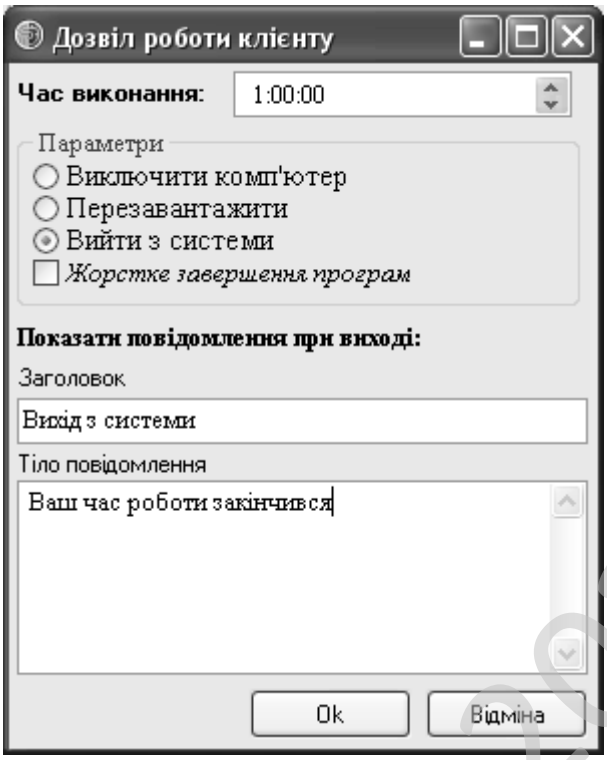

Рисунок 5.4 – Дозвіл роботи клієнту

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

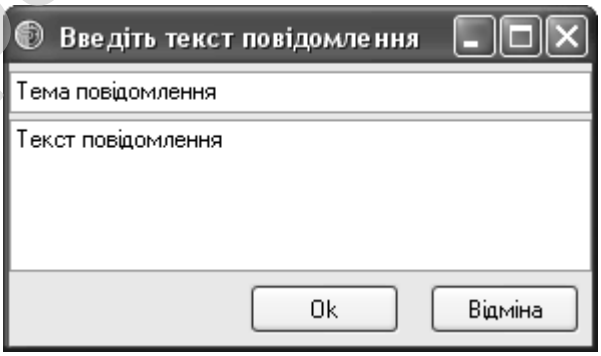

Рисунок 5.5 – Відправка повідомлення

Для зміни параметрів програми слід вибрати пункт меню користувача

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

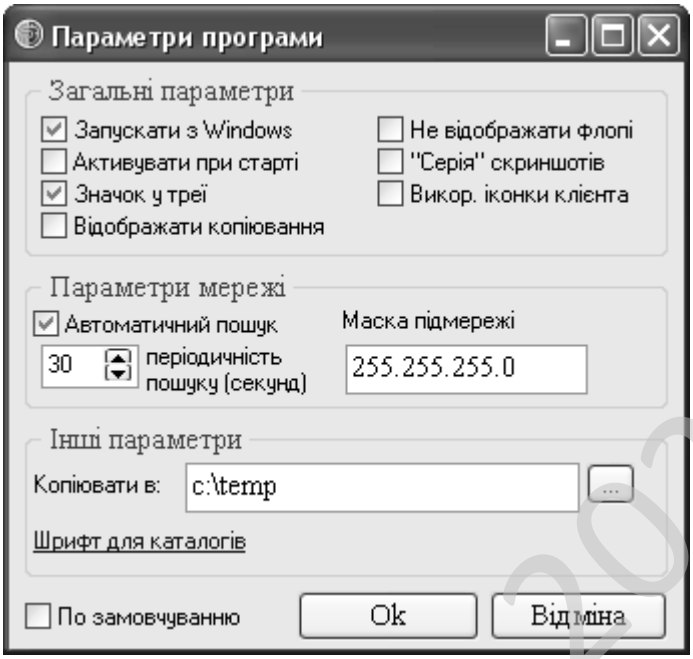

Рисунок 5.6 – Параметри програми

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.

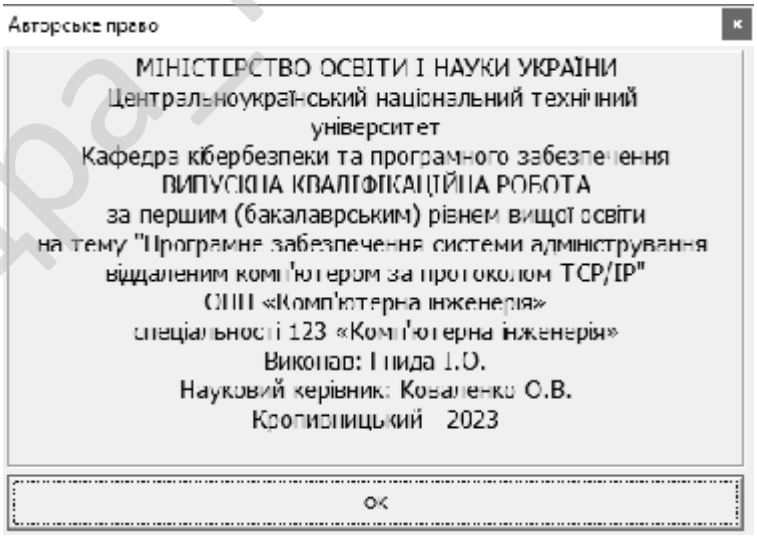

Рисунок 5.7 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи адміністрування віддаленим комп'ютером за протоколом TCP/IP.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем адміністрування віддаленим комп'ютером за протоколом TCP/IP.

– Досліджена система адміністрування віддаленим комп'ютером за протоколом TCP/IP.

– На основі отриманих результатів досліджень створена програмна реалізація системи адміністрування віддаленим комп'ютером за протоколом TCP/IP.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання адміністрування віддаленим комп'ютером за протоколом TCP/IP.

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 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи адміністрування віддаленим комп'ютером за протоколом TCP/IP. 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 ММВ.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

2. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов, О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

3. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

4. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

5. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

6. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

7. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и

сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

8. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

9. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А. Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

10. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

11. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

12. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

13. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам

Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

14. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

15. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

16. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

17. Смирнов С. А. Комплекс gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

18. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

19. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

20. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы /

В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

21. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

22. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем / В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

23. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

24. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

25. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

26. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару

«Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

27. Смирнов С. А. технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

28. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

29. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

30. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

31. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

32. Смирнов С. А. Разработка комплекса gert-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

33. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

34. Смирнов С. А. gert-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

35. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

36. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

37. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы /

А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

38. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

39. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

40. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

41. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

42. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку

ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

43. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез ІІІ міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

44. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая - 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

45. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. - техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

46. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

47. Столлингс В. Современные компьютерные сети / Вильям Столлингс. –СПб.: Питер, 2003. – 778 с.

48. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

49. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

50. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи адміністрування віддаленим комп’ютером за протоколом TCP/IP.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 8-02 від 5.01.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи адміністрування віддаленим комп’ютером за протоколом TCP/IP.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи адміністрування віддаленим комп’ютером за протоколом TCP/IP;

– 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 10.4.

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 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 76 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2023 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 12.06.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко О.В.

Програмне забезпечення системи адміністрування віддаленим комп’ютером за протоколом TCP/IP

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 81

Літера: РП

Кропивницький – 2023 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:='БАКАЛАВРСЬКИЙ ПРОЕКТ';
Label2.Caption:='на тему:';
Label3.Caption:='Програмне забезпечення системи адміністрування';
Label4.Caption:='віддаленим комп''ютером за протоколом TCP/IP';
Label5.Caption:='Розробив: Гнида Іван Олегович';
Label6.Caption:='гр. КІ-20-3СК';
Label7.Caption:='Керівник:  Коваленко О.В..';
Label8.Caption:='м. Кропивницький 2023';
end;
end.
```